TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

SAGARMATHA ENGINEERING COLLEGE

A

PROJECT REPORT

ON

EYE CONTROLLED ELECTRIC WHEEL-CHAIR

BY

ARJUN KOIRALA 33053

GOKUL SUBEDI 33054

SUSHMIT PAUDEL 33070

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR IN ELECTRONICS, COMMUNICATION AND INFORMATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

SANEPA, LALITPUR, NEPAL

June, 2024

ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to Er. Bharat Bhatta,Head of Department(HOD) and Er.Baikuntha Acharya,Deputy Head of Department(DHOD) of Department of Electronics and Computer Engineering, Sagarmatha Engineering college for providing us an opportunity to work on our minor project and providing proper suggestions and allowing us to use the facilities available.

In addition to our mentor, we would like to thank Er. Bipin Thapa Magar for his guidance, suggestions and for directly supporting us in tackling various difficulties.

We are also indebted to all the faculty members of the Department of Electronics and Computer engineering for support, advice,help, and suggestion during this project. Our thanks and appreciation also go to all the teachers, class friends as well as our well wishers who have willingly helped us out with their knowledge and support.

ABSTRACT

This project aims to enable wheelchair control using eye gestures captured by a laptop camera, providing an intuitive and accessible interface for individuals with mobility impairments. Data collection is done gathering a diverse dataset of image frames of eye movement representing control commands (e.g., forward, left, right, stop). Training a deep learning-based model with tensor flow and using image processing techniques to recognize and classify pupil movement from the captured data. Deploy the trained model on the mobile device to perform real-time operation on incoming camera frames. Interpret recognized movement to generate specific control signals for the wheelchair. Mapping is done for the recognized gestures or objects to corresponding control of wheelchair movement directions and generate precise control signals to direct the wheelchair's actions. Outcome: The expected output is a functional system capable of interpreting visual clues captured by the camera, accurately recognizing pupil movement, and translating them into precise control signals for the wheelchair, ultimately empowering users with improved mobility and independence.

KeywordS: Deep Learning, Image processing, improved mobility

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

The global population of individuals reliant on wheelchairs for daily mobility stands at a staggering 75 million, constituting approximately 1 percent of the world's populace, as reported by the World Health Organization (WHO). Among these individuals are those grappling with the complex challenges posed by Locked-in syndrome, a rare neurological condition characterized by near-complete paralysis of voluntary muscles, save for eye movement. Locked-in syndrome stems from various causes, including traumatic brain injury, stroke, or brainstem disorders, rendering individuals conscious yet incapable of verbal communication or bodily movement. Consequently, those afflicted face profound obstacles in their daily lives, heavily relying on assistive technologies for communication and interaction with their environment.

In this landscape, the development of tailored solutions and technologies to meet the unique needs of individuals with Locked-in syndrome is imperative. These innovations encompass advanced wheelchair designs and innovative communication devices, essential for enhancing mobility, independence, and overall quality of life for affected individuals. By addressing the distinct challenges encountered by this population, we strive to promote inclusivity and facilitate their meaningful participation in society.

1.2 Problem Definition

The pressing issue at hand lies in the profound challenges faced by individuals afflicted with Locked-in syndrome, a debilitating neurological condition characterized by severe paralysis, except for eye movement. These individuals, constituting a subset of the 75 million global wheelchair users identified by WHO, grapple with immense hurdles in daily communication and mobility. Traditional assistive technologies often fail to adequately address their unique needs, leaving them isolated and reliant on caregivers for basic tasks. Consequently, there exists a critical need for tailored solutions and technologies specifically designed to empower individuals with Locked-in syndrome, facilitating enhanced communication, mobility, and independence. Addressing this need not only improves the quality of life for affected individuals but also promotes inclusivity and societal participation for a population often marginalized by their condition.

Due to this, people began to curse themselves regarding their problem and start getting depressed and their care taker also feels like they are the burden for their day to day livelihood. In order to make their life bit easier, this project is put forward.

1.3 Objectives

The objectives of this project is

- To develop a system that allows individuals with mobility impairments to control a wheelchair through eye movement and mobile devices.

1.4 Scope

Making the project feasible to be implemented to the real world we get the following scopes analysed below:

Social inclusion: Eye-controlled wheelchairs contribute to social inclusion by enabling users to participate in social events, gatherings, and activities. The ability to move freely and interact with others fosters a sense of belonging and reduces potential feelings of isolation. The scope of eye-controlled wheelchairs extends beyond the individual user to impact their immediate social circles and the broader community. Continued research and development in this field aim to further enhance the capabilities and accessibility of these technologies, ultimately improving the lives of individuals with severe motor disabilities.

1.5 Features

The features of our project are as follows:

- Take the real time eye movement snapshots in order to move the wheel chair.
- Use of the eye movement of the people with disability to make their mobility easier.
- Implementation of CNN model and deep learning models to process camera input for eye gesture recognition and classification.

1.6 Feasibility

1.6.1 Technical Feasibility

Availability of Technology: Existence of suitable CNN algorithm and deep learning techniques for eye sight recognition, which are feasible for implementation on electric wheel chair. Computational Requirements: Assessing whether the processing power and capabilities of mobile devices are sufficient to handle real-time gesture recognition and wheelchair control.

1.6.2 Operational Feasibility

User-Friendly Interface: The system's interface will be intuitive and easy to use, ensuring that individuals with locked in syndrome also other people with different mobility impairments can comfortably interact with it. Assessing the ease with which users will adapt to controlling the wheelchair using eye gestures make it universally accessible. In both indoors, outdoors, different lighting conditions, etc. the device can be operated. Evaluating the practicality and effectiveness of using gestures or objects for controlling the wheelchair in everyday scenarios and activities, this system can effectively implemented on the commonly available wheelchair that users can afford.

1.7 System Requirements

The system requirements of our project are:

1.7.1 Software Requirements

The software requirements for our project are as follows:

(a) Libraries of Python: Pyplot, numpy, torchvision, drive, dataloader, open CV for image procesing, Matplotlib,pyttsx3,

(b) Libraries of Deep Learning : Tensorflow,torch

(c) Module used for ML: CNN(convolutional neural network)

1.7.2 Hardware Requirements

The hardware requirements for our project are as follows:

a) Laptop

b) Arduino

c) Wheelchair

d) Bluetooth Module

e) Battery

CHAPTER 2

LITERATURE REVIEW

Recent Advancements in eye-controlled electric wheelchairs for paralyzed users integrate sophisticated gaze estimation techniques, including Adaboost classifiers and pupil analysis. Seamless IR camera and spectacles integration ensure precise tracking, while assuming a static eye location streamlines the process.Further research is needed to optimize real-world performance.[1]

The proposed eye-controlled wheelchair system leverages real-time CNN-based eye gaze classification, ensuring swift responsiveness at 30 FPS. It features lightweight, low-cost controller subsystems for enhanced user mobility and comfort, facilitated by an IR camera headset. Integration of ultrasonic sensors mitigates collision risks.Used OpenMP API for efficient and accessible wheelchair control through eye movement.[2]

The gesture-based control systems, emphasizing usability, user experience, and real-world feasibility in assistive technologies and mobility aids. Research studies investigating gesture recognition for accessibility purposes are examined, showcasing advancements in enhancing user interaction. Integration of mobile devices such as smartphones and tablets into assistive technologies is explored, highlighting their pivotal role in improving accessibility. [3]

Considering the research paper the project was validated using Electrooculography (EOG) and video-based approaches in eye-gaze controlled wheelchair systems. It discusses modules like the Control Subsystem, facilitating real-time navigation, and Communication Subsystem, enabling social interaction. The Entertainment Subsystem offers accessible entertainment options, while Actuators execute physical actions. Sensors capture user and environmental data for decision-making. Challenges include continuous gaze requirements and calibration efforts, with advancements focusing on accuracy and adaptability, particularly for individuals with severe disabilities like ALS. Commercially available eye-tracking technologies are compared for suitability in enhancing user autonomy and daily living activities.[4]

CHAPTER 3

RELATED THEORY

The concept of an eye-controlled wheelchair revolves around utilizing advanced technologies, primarily eye-tracking systems, to enable individuals with limited mobility to navigate their environment more effectively.By capturing and analyzing the user's eye movements,the system translates these movements into actionable commands that control the movement and direction of a motorized wheelchair.This innovative approach aims to enhance mobility,independence and quality of life for individuals with severe physical disabilities.

Eye-Tracking Technology:Eye-tracking technology involves capturing and analyzing the movements and positions of a user's eyes using specialized cameras or sensors.These systems detect various eye movements to determine the user's commands. Advanced algorithms process the collected data,identifying specific patterns or gestures that correspond to predefined actions,such as moving forward,turning,or stopping the wheelchair.

Computer Vision and Image Processing: The theory behind extracting meaningful information from images or video frames,including edge detection,corner detection and keypoint extraction.

Control Systems: Fundamental principles of control systems,including feedback control,and algorithms for translating recognized gestures/objects into wheelchair control signals by understanding theoretical concepts related to navigation algorithms used in wheelchairs,such as path planning and obstacle avoidance methods.

Real-Time Processing: Theoretical aspects of ensuring responsive and real-time processing of captured gestures/objects to generate immediate control signals for wheelchair maneuvering.

Haarcascade-frontalface-default model: refers to a pre-trained Haar Cascade classifier model used for face detection in computer vision applications. Haar Cascade classifiers are machine learning-based algorithms used to identify objects in images or video streams. The ‘frontalface-default‘ variant specifically focuses on detecting frontal faces in images or video frames. This classifier is trained on a dataset of positive and negative examples of frontal face images to recognize patterns and features indicative of a human face. It is commonly used in applications such as face detection, facial recognition, and emotion recognition.

3.0.1 Machine Learning

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are like the detectives of the digital world. They’re specialized in understanding images, kind of like how our brains recognize patterns and shapes. These networks have transformed the way computers see and understand pictures, all by learning from examples.

How They Work:

1. Convolutional Layers:These are layers of neurons of model that move over the image, looking for specific features or patterns like edges or textures.

2. Pooling Layers:They help focus on the most important parts of the image, making things simpler to understand.

3. Activation Functions: They help the network recognize complex patterns.

4. Fully Connected Layers: These layers are like group discussions, where every member shares what they’ve observed, helping the network piece together the bigger picture.

Training Journey:

1 Forward Propagation: It’s like taking a journey through the network. You show it an image, and it predicts what it sees. If it’s wrong, it learns from its mistakes.

2. Back propagation: This is the network’s way of learning from its errors. It adjusts its value based on how wrong or right its predictions were.

3.Optimization: Just like honing a skill, the network fine-tunes its 'brush strokes' to become better at recognizing patterns, all to minimize its mistakes.

3.0.2 Transfer Learning

Let's go through an example, Imagine you're learning to bake cookies. You start with a recipe book, follow the steps, and get delicious cookies. Now, imagine using that same knowledge to bake a cake. That's transfer learning! We reuse what we've learned in one task to excel at another.

Why Transfer Learning?

1. Quick Learning: It's like having a head start. We begin with a solid foundation (pre-trained models) and adapt it to our specific task, saving time and effort.

2. Better Results: Just like a seasoned chef's expertise, pre-trained models bring a wealth of knowledge from vast datasets, enhancing performance even with limited data.

3. Efficiency Boost: Instead of starting from scratch every time, we build upon existing knowledge, making the learning process smoother and more resource-efficient.

3.0.3 Residual Networks (ResNets)

Residual Networks (ResNets) are the problem solvers of deep learning. They were created to tackle a pesky issue known as the vanishing gradient problem, which made it hard for deep networks to learn effectively.

Why this?

1. Skip Connections: These are like secret passages in a maze. They help the network bypass tricky areas, ensuring a smoother flow of information and preventing learning obstacles.

2. Identity Mapping: Just like keeping track of where you've been, ResNets

maintain a clear path from input to output, ensuring information isn't lost or distorted along the way.

3. Deep Dive: ResNets are like deep-sea explorers, capable of delving into the depths of data with hundreds of layers, uncovering hidden insights for improved performance.

3.0.4 Image Processing

mage processing is the art and science of manipulating digital images to extract meaningful information or enhance their visual appearance. It encompasses various techniques and algorithms to analyze, modify, or interpret images.

Adopted Processing

1. Grayscale Conversion: Grayscale conversion is the process of converting color images (which typically have three channels: Red, Green, and Blue) into single-channel images representing intensity levels. This conversion simplifies image processing tasks by focusing only on the brightness information, neglecting color details. OpenCV, a popular library for computer vision tasks, provides the function cv2.cvtColor() for converting BGR images to grayscale.

gray = 0.21*R + 0.72*G + 0.07*B

2. Image Transformation: Image transformation involves modifying the size, shape, orientation, or appearance of an image. Common transformations include resizing, rotation, translation, and scaling. These transformations are essential for preprocessing images before feeding them into machine learning models, ensuring uniformity and compatibility.

3. Filtering and Convolution: Filtering techniques, such as blurring, sharpening, and edge detection, are used to enhance or suppress certain features in images. These techniques often involve applying convolution operations, where a kernel matrix is passed over the image to perform local operations. Convolution helps extract features and patterns from images, facilitating tasks like object detection and image segmentation.

4. Feature Extraction: Feature extraction is the process of identifying and representing relevant information from images in a compact and discriminative manner. Features can include edges, corners, textures, or more complex structures. Extracted features serve as input to machine learning algorithms for tasks like classification, object recognition, and image retrieval.

Applications

1. Edge Detection: Grayscale images are commonly used in edge detection algorithms like Sobel, Canny, and Prewitt to identify changes in intensity and outline object boundaries.

2. Feature Extraction: Many feature extraction techniques, such as corner detection and blob analysis, are performed more efficiently on grayscale images due to their reduced complexity.

3. Pattern Recognition: Grayscale images serve as the basis for pattern recognition tasks, including facial recognition, object detection, and optical character recognition (OCR).

3.0.5 Hardware Description

laptop as a source of capturing images of eye movement and data processing.

Figure 3.1: laptop

Arduino as a controller of the wheelchair which operates through the command generated from the data processing unit.

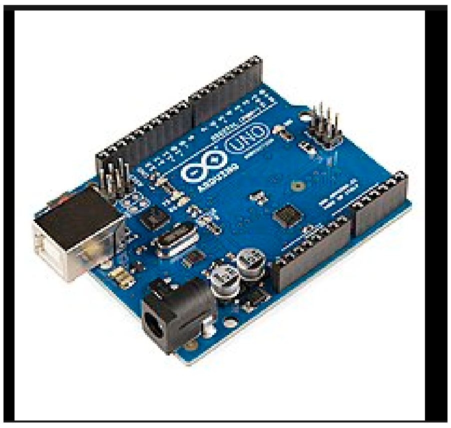


Figure 3.2: Arduino UNO

Wheel chair as the main equipment which will be used for the mobility of the users.

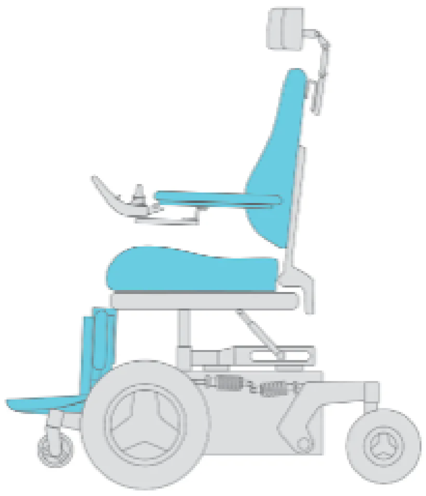


Figure 3.3: electric wheel chair

Bluetooth module as the data transmission unit which receives the control signal from the processing unit and transfer same to the wheelchair control unit.

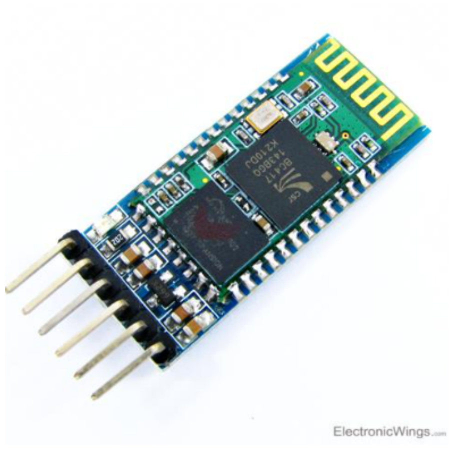


Figure 3.4: Bluetooth module

Lipo battery as the power supply to the overall system.

Figure 3.5: Lipo battery

CHAPTER 4

METHODOLOGY

4.1 Block Diagram

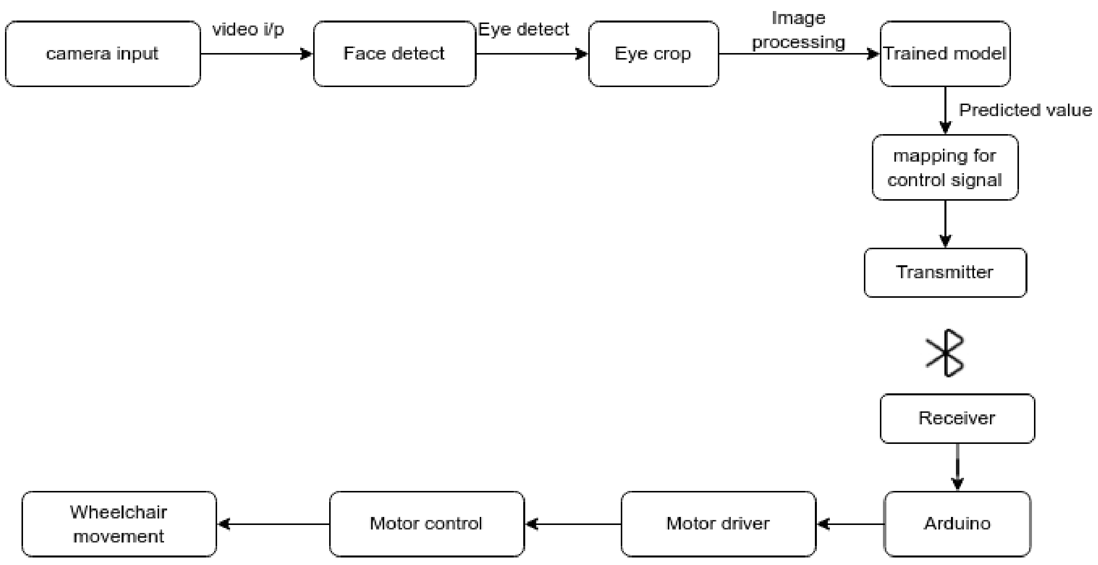


Figure 4.1: System block diagram

4.1.1 Description

The system described in section 4.1 integrates various components to facilitate wheelchair control based on user eye movements. Initially, the camera captures live video of the user, which is then processed for face detection using OpenCV and the haarcascade-frontalface-default model. Subsequently, the system identifies the user's eyes and crops the respective regions using the haarcascade-eye.xml model. An AI model, pre-trained in machine learning, tracks the user's eye movements to derive control signals for the wheelchair. These signals, representing the position of the user's eyes, are converted into a usable format (-1, 0, 1) by the control method. The transmitter component employs laptop Bluetooth technology to transmit these control signals to the wheelchair. Upon reception, the Bluetooth model HC-05 captures the transmitted signals and forwards them to the Arduino, acting as the main controller of the wheelchair. The Arduino interprets these signals and provides appropriate instructions to the motor driver, which in turn regulates the motor's speed and direction to drive the wheelchair accordingly. Finally, the motor serves as the primary driving element of the wheelchair, executing the desired movements based on the interpreted signals from the Arduino. Through this integrated system, users can effectively control the wheelchair using their eye movements, enhancing accessibility and mobility for individuals with disabilities.

4.2 Flow chart

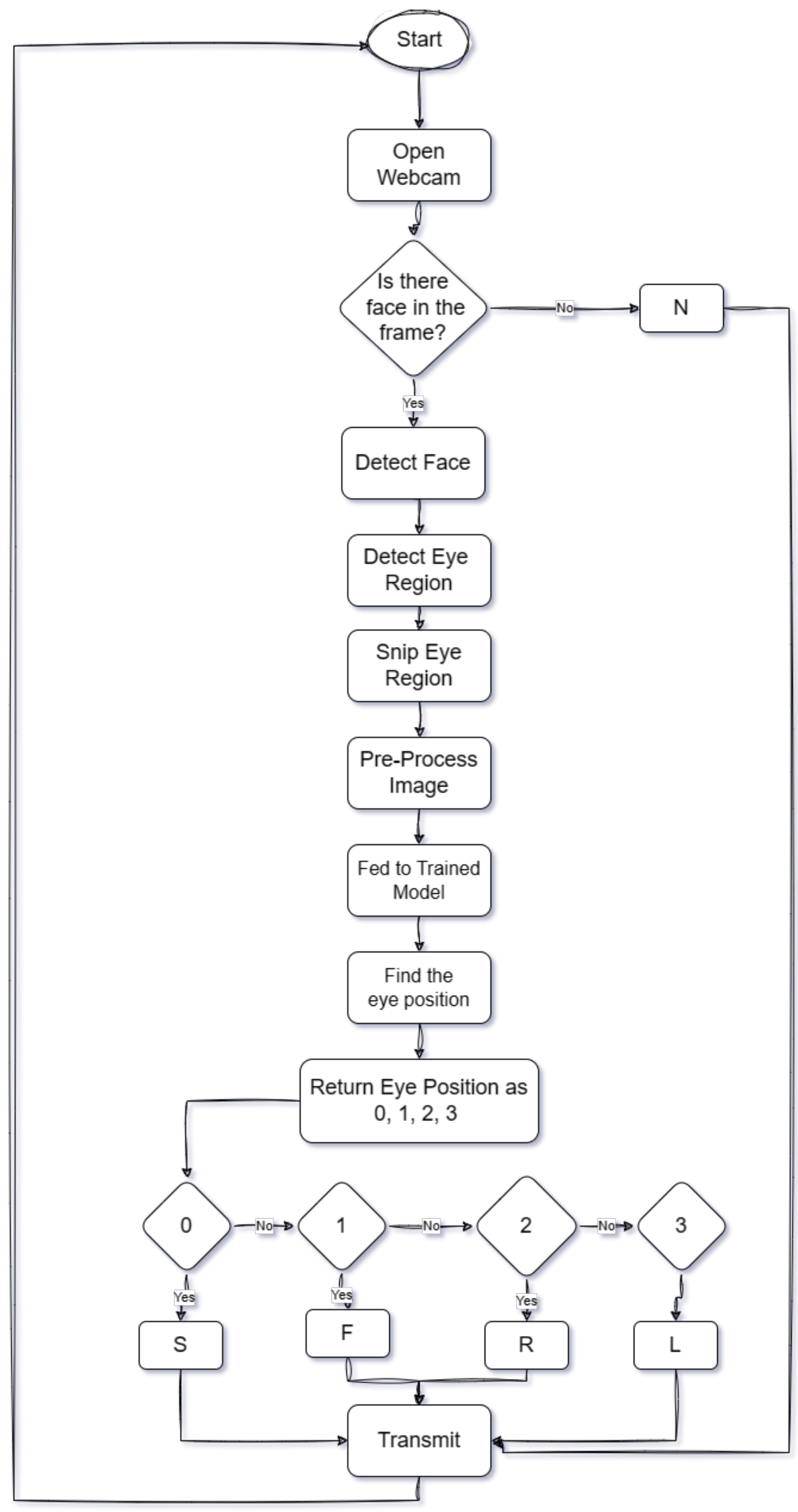


Figure 4.2: Flowchart transmitter side

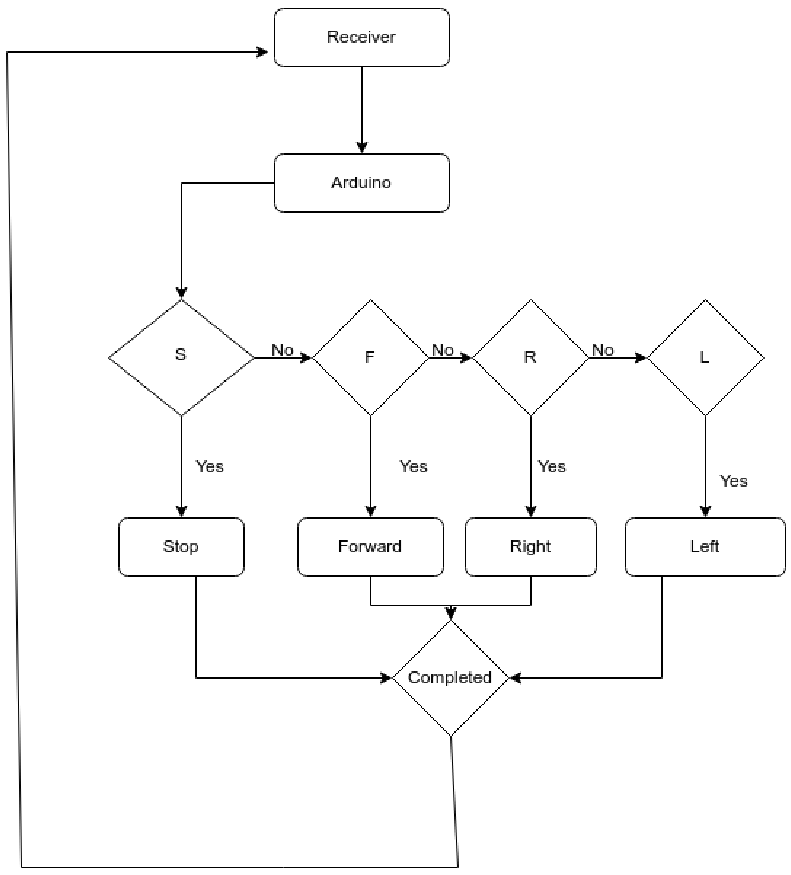


Figure 4.3: Flowchart receiver side

4.3 Model and Dataset

4.3.1 Dataset

The dataset used for training and evaluation comprises images of human eyes, sourced from both Kaggle and custom data collection efforts. The dataset is categorized into four classes based on eye position: closed, centered, right, and left. Each class contains images representing the respective eye positions.

Kaggle Dataset

Source: The initial portion of the dataset was obtained from Kaggle, a popular platform for sharing datasets and data science projects. - Quantity:The Kaggle dataset consists of approximately:

Image Size:

The images from Kaggle are predominantly sized at 118x118 pixels.

Dataset Count:

S.No.	Data Label	data count
1	closed eye data	5484
2	forward looking data	5511
3	right looking data	5374
4	left looking data	5341

Table 4.1: Table showing kaggle dataset count

Custom Dataset

Source: In addition to the Kaggle dataset, a custom dataset was compiled through manual data collection efforts.

Image Size:

The images from the custom dataset are standardized to 120x120 pixels. Dataset Count:

S.No.	Data Label	data count
1	closed eye data	2580
2	forward looking data	2029
3	right looking data	2048
4	left looking data	2936

Table 4.2: Table showing custom dataset count

Image Processing:

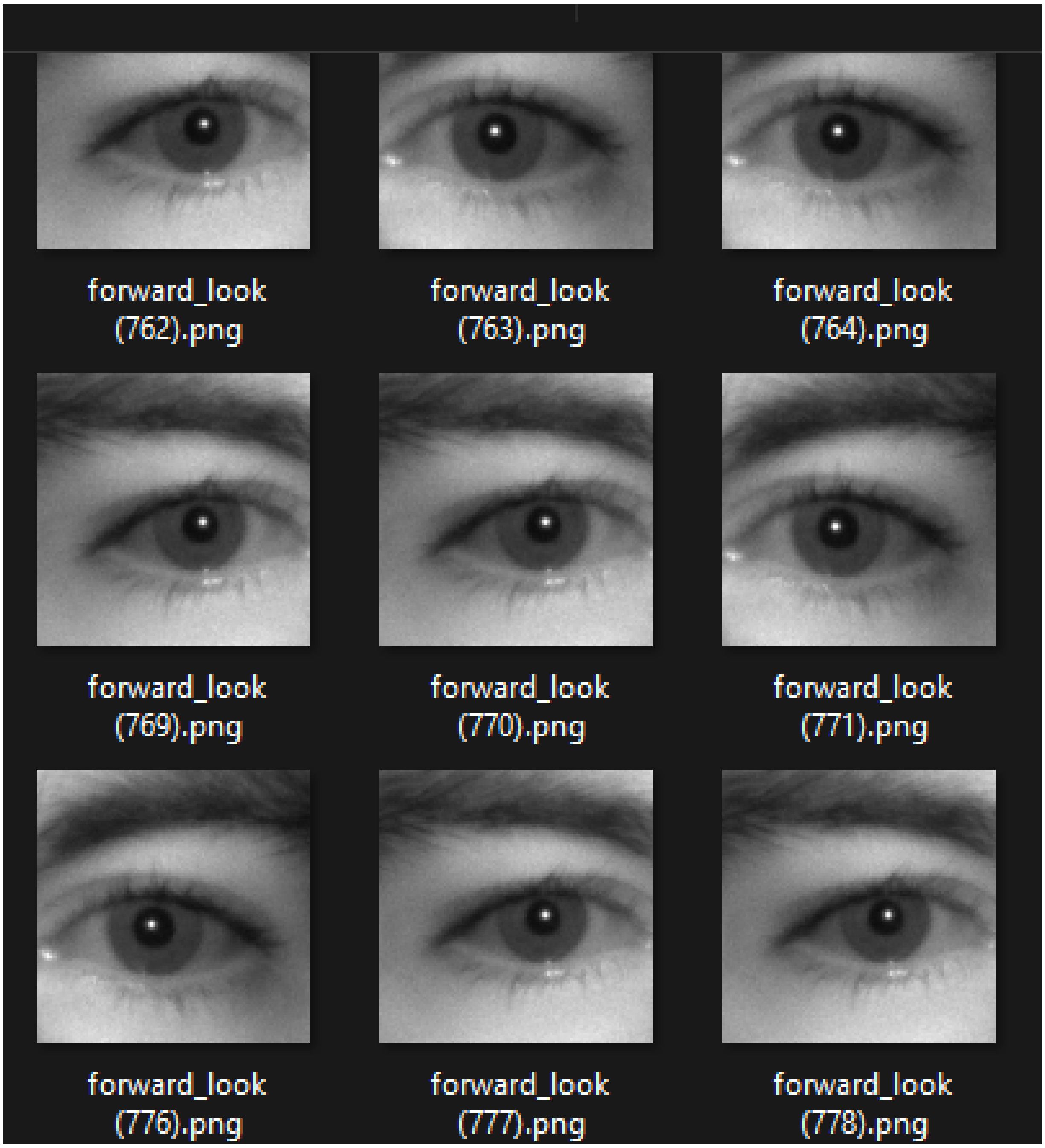


Figure 4.4: Image processing sample

Three different forms of images were(gray scale,edges canny and the original image) created from the available data set in order to train the RNN model for predicting the live user eye position and controlling the wheelchair movement.

Normalization: Before being fed into the model, all images undergo pre-processing steps to ensure uniformity and facilitate model training. This includes resizing all images to a fixed size of 224x224 pixels.

Color Space: All the original images are converted into grayscale versions of the custom dataset images are included, effectively doubling the number of images sourced from the custom dataset.

By integrating data from both Kaggle and custom sources, the dataset offers a diverse array of images capturing various eye positions. This comprehensive dataset enables the model to learn robust features and generalize well to unseen data, ultimately enhancing the model's accuracy and effectiveness in classifying human eye positions.

4.3.2 Model

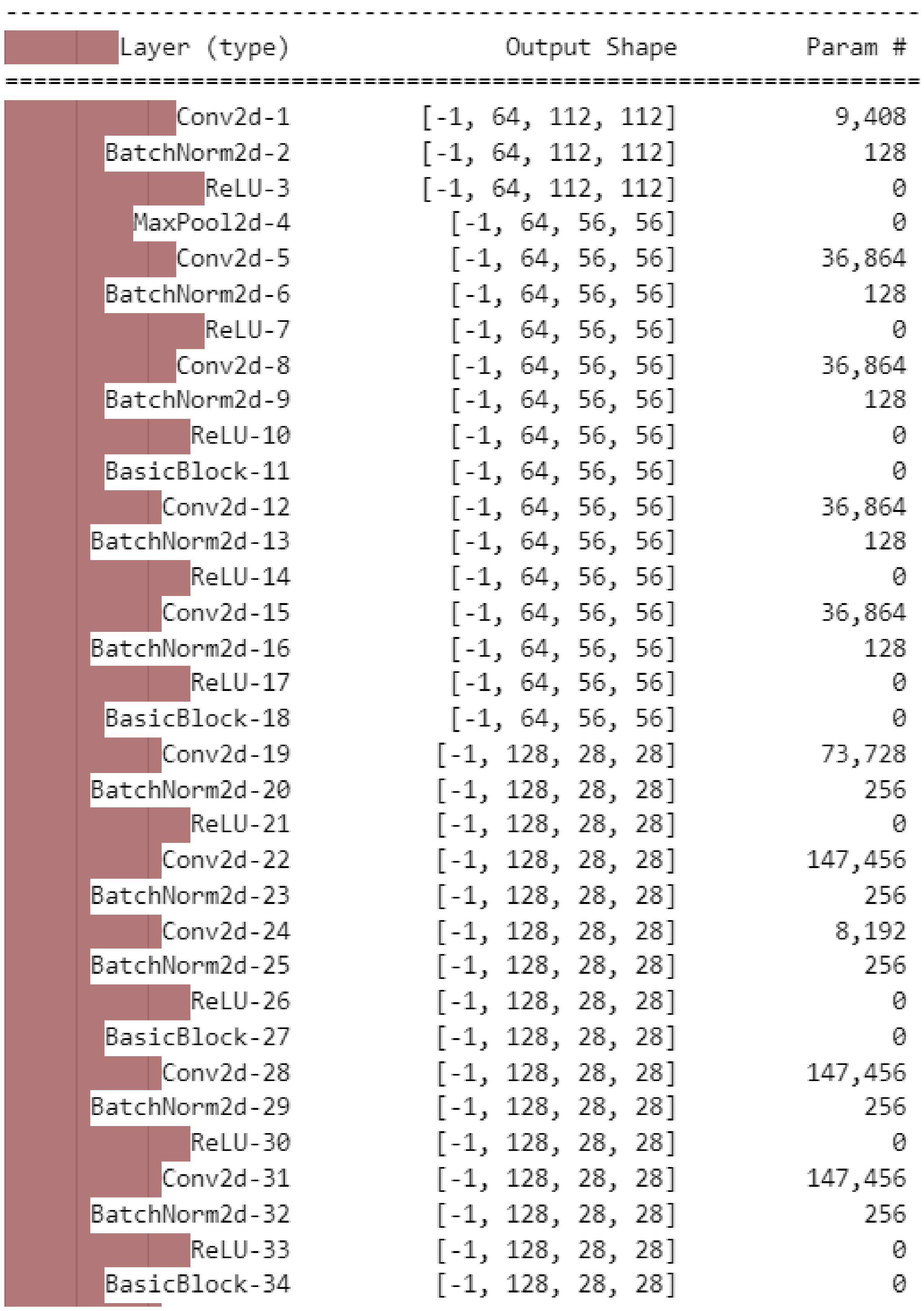
The model architecture is rooted in the ResNet18 framework, a deep convolutional neural network (CNN) renowned for its ability to effectively learn hierarchical features from images. Leveraging the ResNet18 architecture provides a strong foundation for the classification task at hand due to its proven performance in image recognition tasks.

Specifically, the model consists of several convolutional layers followed by residual blocks. These residual blocks enable the model to learn intricate features while mitigating the vanishing gradient problem, which can impede training in deeper networks. The ResNet18 architecture also incorporates max-pooling layers to downsample feature maps, reducing computational complexity and increasing the receptive field.

For fine-tuning the model to classify human eye positions, the final fully connected layer of the ResNet18 architecture is replaced with a new linear layer. This adaptation tailors the model's output to match the desired classification scheme, comprising four distinct categories: left eye position, right eye position, closed eyes, and centered eyes.

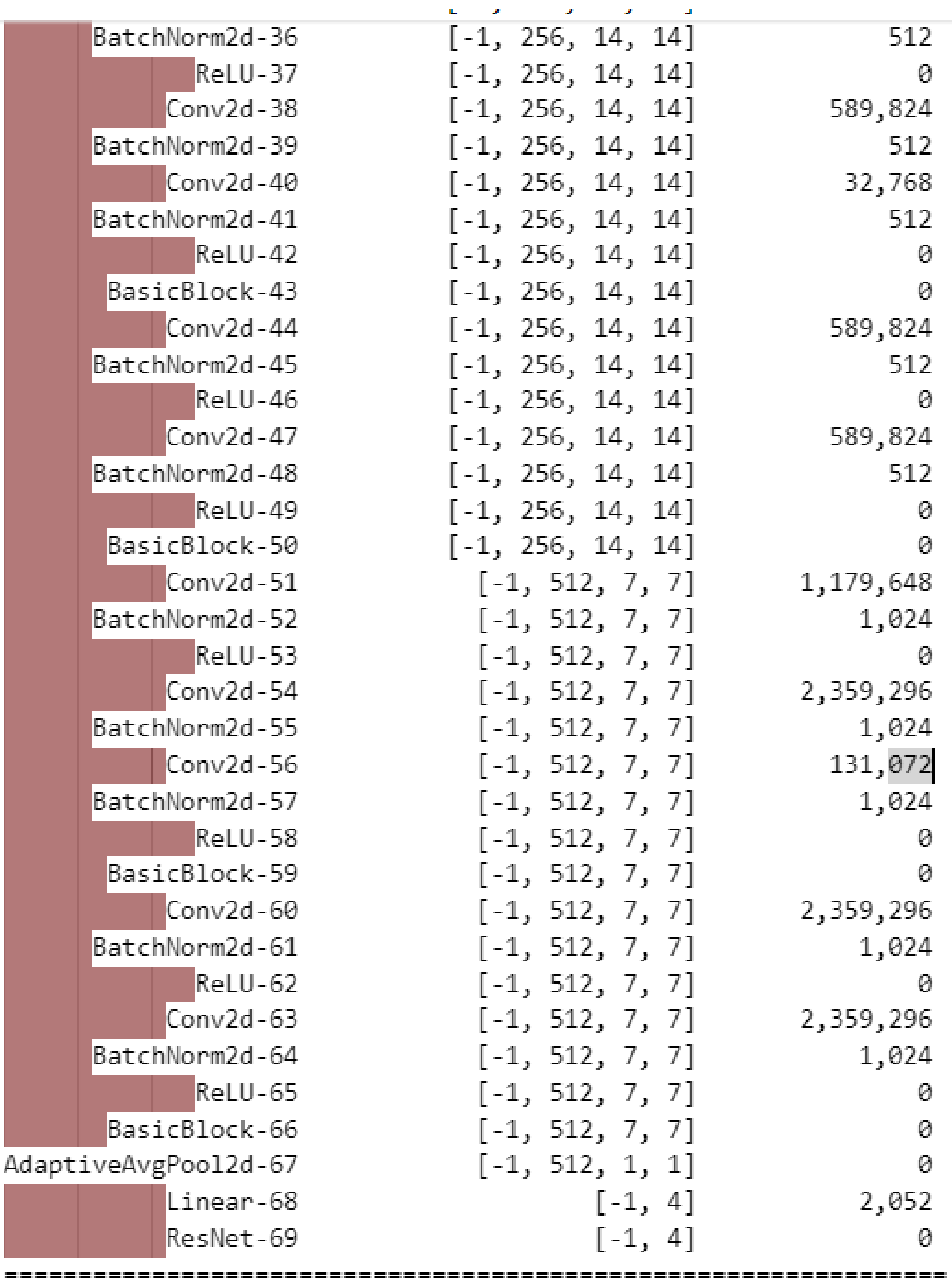
By leveraging the pre-trained ResNet18 model and customizing the output layer, the model can efficiently extract meaningful features from input images and make accurate predictions regarding the position of human eyes. This design choice not only capitalizes on the wealth of knowledge encoded in the ResNet18 architecture but also ensures that the model is appropriately tailored to the specific classification task of interest.

4.3.3 Summary Table



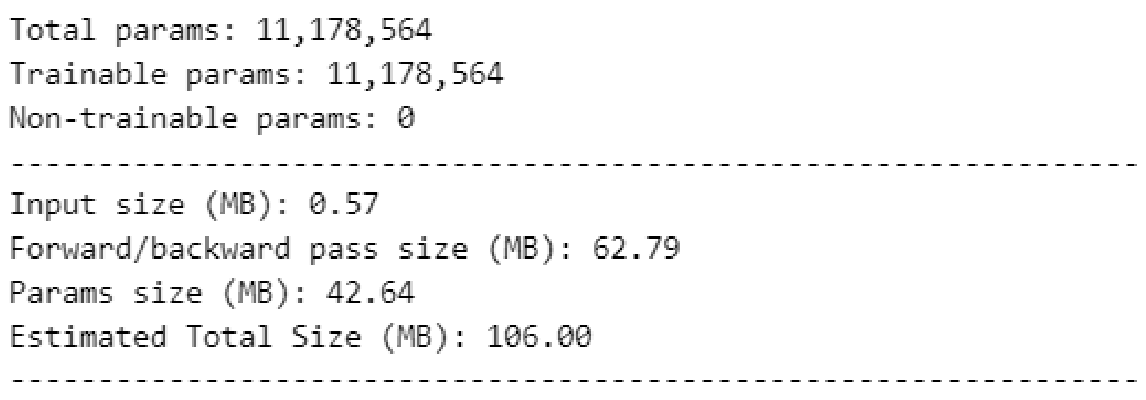
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]          36,864
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]          36,864
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
       BasicBlock-11           [-1, 64, 56, 56]               0
           Conv2d-12           [-1, 64, 56, 56]          36,864
      BatchNorm2d-13           [-1, 64, 56, 56]             128
             ReLU-14           [-1, 64, 56, 56]               0
           Conv2d-15           [-1, 64, 56, 56]          36,864
      BatchNorm2d-16           [-1, 64, 56, 56]             128
             ReLU-17           [-1, 64, 56, 56]               0
       BasicBlock-18           [-1, 64, 56, 56]               0
           Conv2d-19          [-1, 128, 28, 28]          73,728
      BatchNorm2d-20          [-1, 128, 28, 28]             256
             ReLU-21          [-1, 128, 28, 28]               0
           Conv2d-22          [-1, 128, 28, 28]         147,456
      BatchNorm2d-23          [-1, 128, 28, 28]             256
           Conv2d-24          [-1, 128, 28, 28]           8,192
      BatchNorm2d-25          [-1, 128, 28, 28]             256
             ReLU-26          [-1, 128, 28, 28]               0
       BasicBlock-27          [-1, 128, 28, 28]               0
           Conv2d-28          [-1, 128, 28, 28]         147,456
      BatchNorm2d-29          [-1, 128, 28, 28]             256
             ReLU-30          [-1, 128, 28, 28]               0
           Conv2d-31          [-1, 128, 28, 28]         147,456
      BatchNorm2d-32          [-1, 128, 28, 28]             256
             ReLU-33          [-1, 128, 28, 28]               0
       BasicBlock-34          [-1, 128, 28, 28]               0
```

Figure 4.5: Model training summary



```
      BatchNorm2d-36         [-1, 256, 14, 14]             512
             ReLU-37         [-1, 256, 14, 14]               0
           Conv2d-38         [-1, 256, 14, 14]         589,824
      BatchNorm2d-39         [-1, 256, 14, 14]             512
           Conv2d-40         [-1, 256, 14, 14]          32,768
      BatchNorm2d-41         [-1, 256, 14, 14]             512
             ReLU-42         [-1, 256, 14, 14]               0
       BasicBlock-43         [-1, 256, 14, 14]               0
           Conv2d-44         [-1, 256, 14, 14]         589,824
      BatchNorm2d-45         [-1, 256, 14, 14]             512
             ReLU-46         [-1, 256, 14, 14]               0
           Conv2d-47         [-1, 256, 14, 14]         589,824
      BatchNorm2d-48         [-1, 256, 14, 14]             512
             ReLU-49         [-1, 256, 14, 14]               0
       BasicBlock-50         [-1, 256, 14, 14]               0
           Conv2d-51           [-1, 512, 7, 7]       1,179,648
      BatchNorm2d-52           [-1, 512, 7, 7]           1,024
             ReLU-53           [-1, 512, 7, 7]               0
           Conv2d-54           [-1, 512, 7, 7]       2,359,296
      BatchNorm2d-55           [-1, 512, 7, 7]           1,024
           Conv2d-56           [-1, 512, 7, 7]         131,072
      BatchNorm2d-57           [-1, 512, 7, 7]           1,024
             ReLU-58           [-1, 512, 7, 7]               0
       BasicBlock-59           [-1, 512, 7, 7]               0
           Conv2d-60           [-1, 512, 7, 7]       2,359,296
      BatchNorm2d-61           [-1, 512, 7, 7]           1,024
             ReLU-62           [-1, 512, 7, 7]               0
           Conv2d-63           [-1, 512, 7, 7]       2,359,296
      BatchNorm2d-64           [-1, 512, 7, 7]           1,024
             ReLU-65           [-1, 512, 7, 7]               0
       BasicBlock-66           [-1, 512, 7, 7]               0
AdaptiveAvgPool2d-67           [-1, 512, 1, 1]               0
           Linear-68                   [-1, 4]           2,052
           ResNet-69                   [-1, 4]               0
================================================================
```

Figure 4.6: Model training summary



```
Total params: 11,178,564
Trainable params: 11,178,564
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 62.79
Params size (MB): 42.64
Estimated Total Size (MB): 106.00
----------------------------------------------------------------
```

Figure 4.7: Used parameter summary

4.4 Hardware Block Diagram

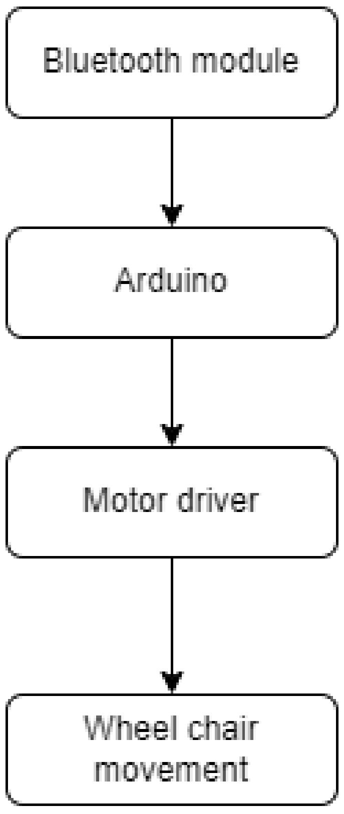


Figure 4.8: Hardware block diagram

4.4.1 Bluetooth Module:

Bluetooth module is used for the serial communication or to pass the predicted value to the arduino.

4.4.2 Arduino:

Arduino is used as controller which controls the motor driver pulse as per the mapped data obtained from the predicted value.When the mapped value 'F' is obtained from bluetooth module the front movement is processed and is sends the signal to motor driver to make both motors to spin the wheel at same ratio. similarly, if 's' then makes the both wheel to stop. like as, if 'r' then makes right wheel to move slower in compare to left wheel by which the right movement is obtained. and if 'l' is obtained then vice-versa to the right movement and finally the left turn is made possible.

4.4.3 Motor Driver:

Motor Driver controls the motors movement as per the control signal transmitted by arduino and finally the wheel chair moves as per the predicted value given by the trained module.

4.5 Communication

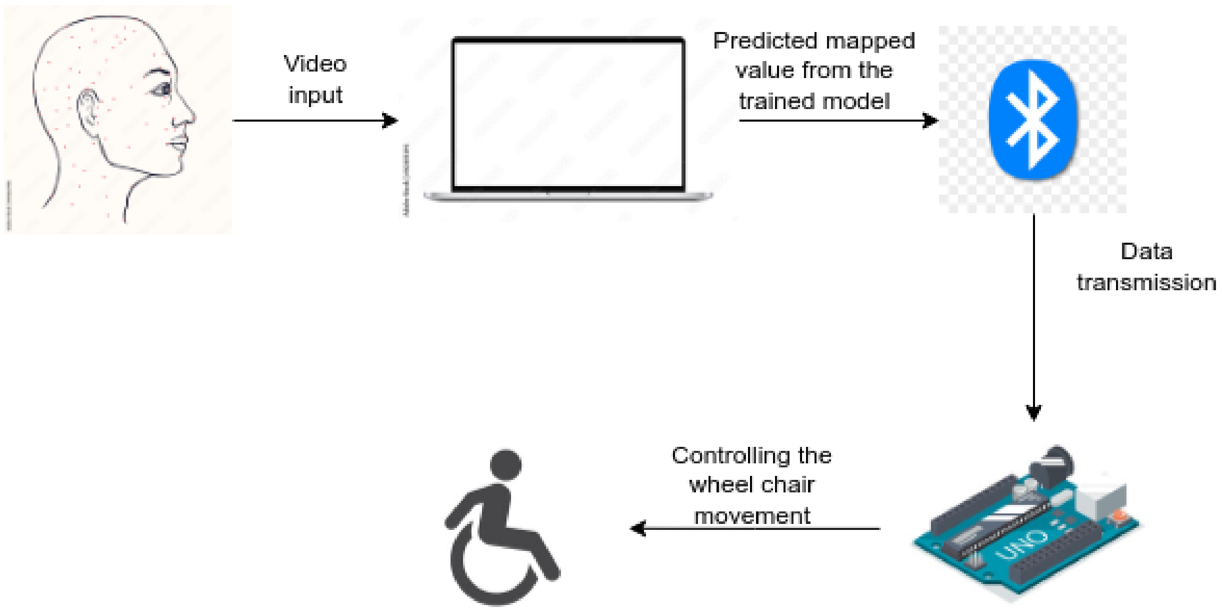

Video input
Predicted mapped value from the trained model
Data transmission
Controlling the wheel chair movement
UNO

Figure 4.9: Communication process

eye position	predicted value	mapped value	received value	action
closed	0	S	S	
forward look	1	F	F	
right look	2	R	R	
left look	3	L	L	

Table 4.3: Communication table along with mapping

In our project, establishing seamless communication between the machine learning model (ML) running on the host computer and the Arduino microcontroller embedded in the vehicle is crucial for real-time decision-making and control. We achieve this communication through a serial connection facilitated by the pyserial library in Python.

4.5.1 Functions and Their Roles:

Establish-connection(): This function initializes the serial connection between the host computer and the Arduino board. It iterates through connection attempts to ensure robustness against transient errors and returns a serial connection object upon successful establishment.

Speak-in-background(text): This function utilizes threading to allow for non-blocking speech synthesis. It enables the system to provide vocal instructions or alerts without disrupting the main execution flow.

Send-to-arduino(value, ser-connection): This function serves as the interface for sending commands or instructions from the ML model to the Arduino board. 25

Parameters:

Value: Represents the command or instruction to be sent to the Arduino, which could be 'S' (Stop), 'R' (Right), 'L' (Left), or 'F' (Forward), among others.

Ser-connection: Represents the serial connection object established earlier. It also manages the logic for handling different scenarios: Incrementing counters for stop, right, and left actions. Triggering specific actions based on the received command, such as vehicle engagement or disengagement. Handling special cases like stopping the vehicle or engaging reverse gear. Encoding and transmitting commands to the Arduino board over the serial connection.

Close-connection(ser-connection): This function gracefully closes the serial connection when it's no longer needed, ensuring proper resource management and system stability. Overall Communication

Flow: Upon initialization, the system establishes a serial connection with the Arduino board, ensuring reliable communication channels. As the ML model processes video input from the webcam, it predicts the eye position and sends corresponding commands to the Arduino. The Arduino board interprets these commands and controls the vehicle's behavior accordingly, such as stopping, moving forward, turning left, or turning right. Additionally, the system utilizes speech synthesis to provide real-time feedback or instructions to the user, enhancing the user experience and safety aspects of the vehicle operation.

CHAPTER 5

RESULT AND ANALYSIS

5.1 Training vs Valida Result

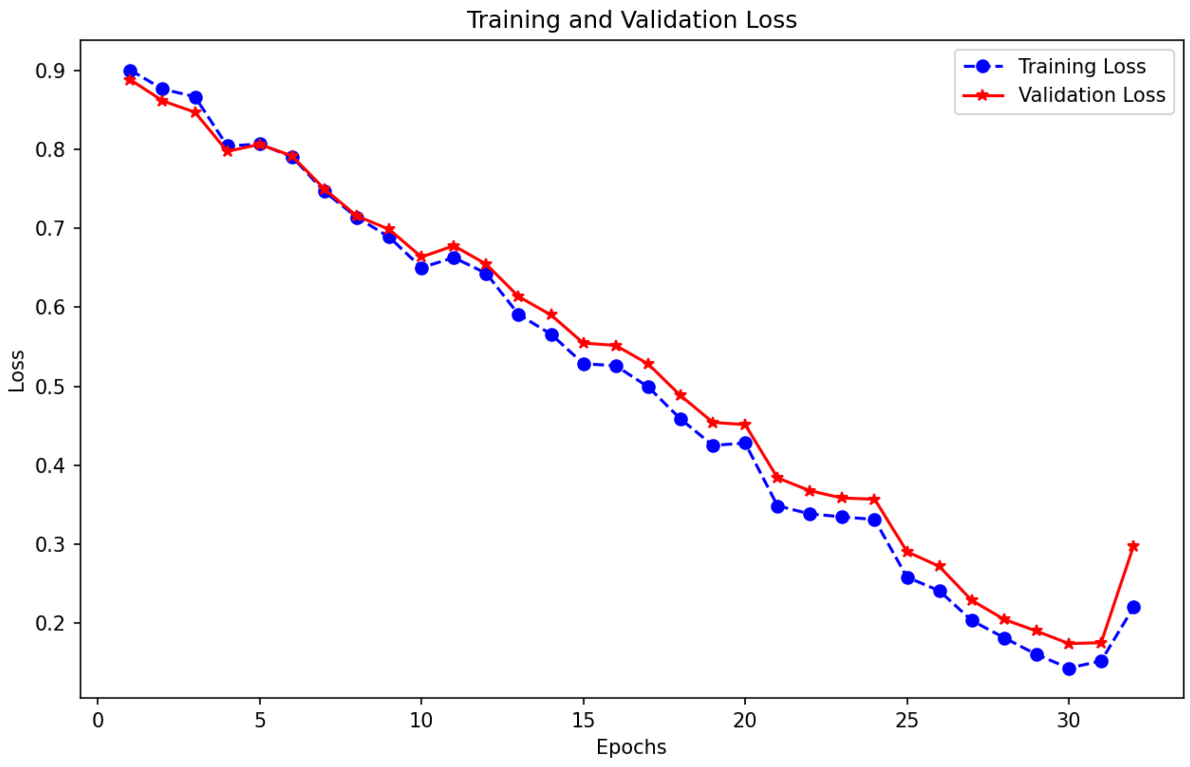


Figure 5.1: Training and validation loss

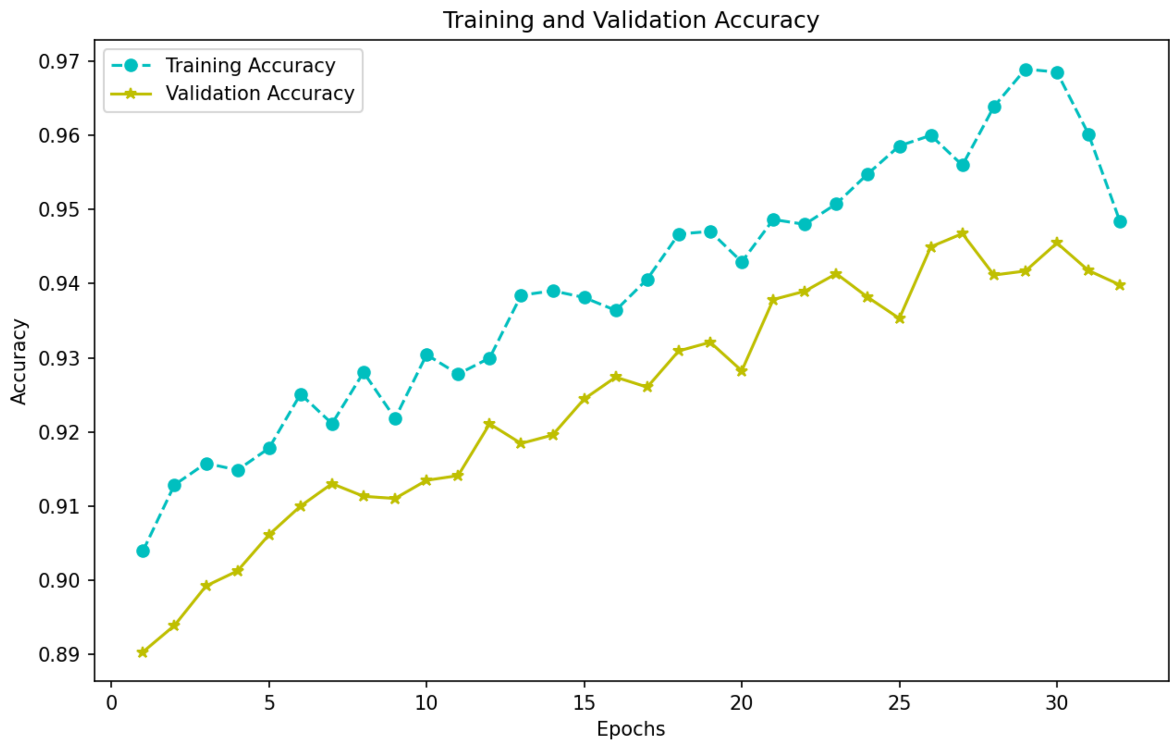


Figure 5.2: Training and validation accuracy

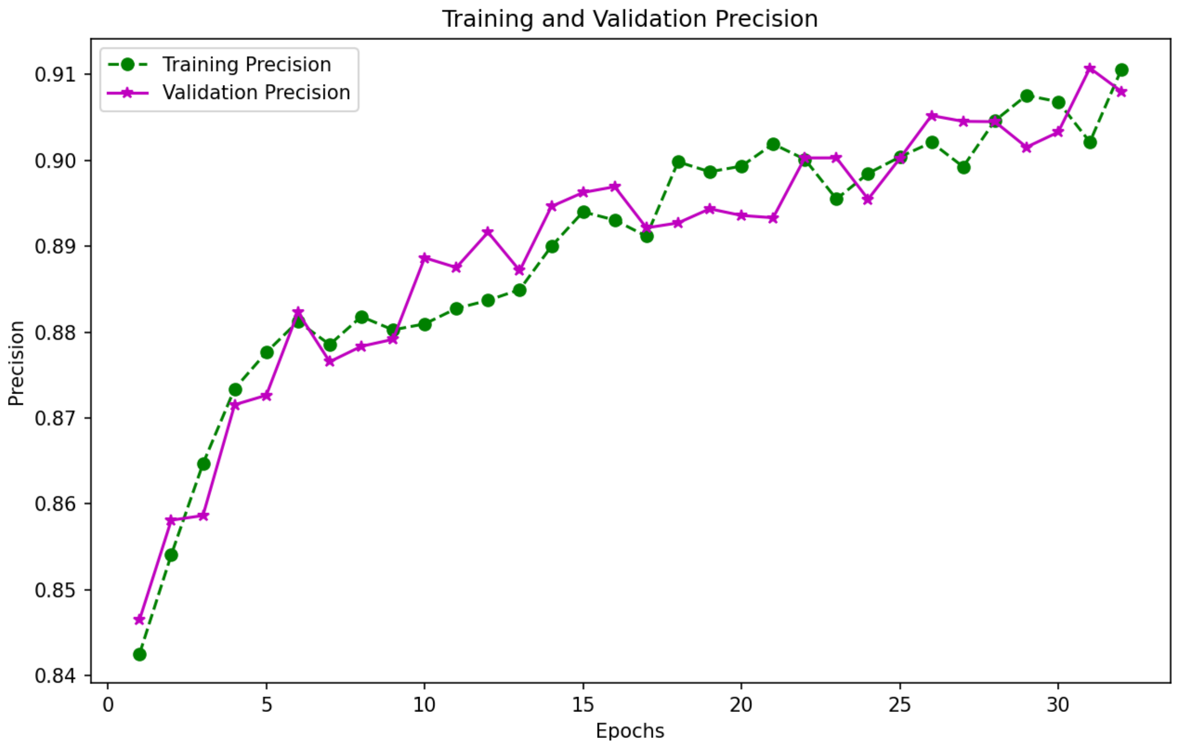


Figure 5.3: Training and validation graph of precision

5.2 Graph Explanation

The graph illustrates the performance of our model during the training phase for predicting human eye positions (left, right, forward, and closed). Let's break it down

1. Training Loss and Validation Loss:

Title and Context:

The graph is labeled "Training and Validation Loss." It represents the performance of a machine learning model during training. The x-axis shows the number of epochs (training iterations), and the y-axis represents loss.

Lines on the Graph:

Blue Line (Training Loss): The blue line with circle markers represents the training loss of the model. As the model trains over successive epochs, the training loss generally decreases. This reduction indicates that the model is learning from the training data.

Red Line (Validation Loss): The red line with star markers represents the validation loss. Validation loss measures how well the model performs on unseen data (validation set). Initially, validation loss decreases in sync with training loss. However, around epoch 25, there is a noticeable increase in validation loss before it continues to decrease. This behavior suggests that the model might be overfitting—becoming too specialized for the training data and losing generalization ability.

Interpretation:

The decreasing training loss indicates that the model is improving its fit to the training data. The fluctuation in validation loss around epoch 25 warrants attention. Overfitting could be a concern if the gap between training and validation loss widens significantly.

Recommendations:

Monitor the model's performance closely beyond epoch 25. Consider techniques like early stopping or regularization to prevent overfitting. Further analysis, such as examining learning curves and confusion matrices, can provide deeper insights.

2. Training Accuracy and Validation Accuracy:

Title and Context:

The graph is titled "Training and Validation Accuracy." It represents the performance of a machine learning model during training. The x-axis shows the number of epochs (training iterations), and the y-axis represents accuracy.

Lines on the Graph:

Blue Line (Training Accuracy): The blue line with circle markers represents the training accuracy of the model. As the model trains over successive epochs, the training accuracy generally increases. However, there are some fluctuations, indicating that the model might be adjusting to the training data.

Yellow Line (Validation Accuracy): The yellow line with star markers represents the validation accuracy. Validation accuracy measures how well the model performs on unseen data (validation set). Initially, validation accuracy increases along with training accuracy. After around 25 epochs, it starts to fluctuate and slightly decline. This behavior suggests that the model might be overfitting—becoming too specialized for the training data and losing generalization ability.

Interpretation:

The increasing training accuracy indicates that the model is learning from the training data. The validation accuracy's fluctuations suggest that the model's performance on unseen data is not consistently improving. Overfitting could be a concern, especially if the gap between training and validation accuracy widens.

Recommendations:

Consider monitoring the model's performance closely beyond epoch 25. Techniques like early stopping or regularization may help prevent overfitting. Further analysis, such as examining learning curves and confusion matrices, can provide deeper insights.

1. Training Loss and Validation Precision:

Title and Context:

The graph is labeled "Training and Validation Precision." It represents the performance of a machine learning model during training. The x-axis shows the number of epochs (training iterations), and the y-axis represents precision.

Lines on the Graph:

Green Line (Training Precision): The green line with circle markers represents the training precision of the model. As the model trains over successive epochs, the training precision generally increases. This upward trend indicates that the model is learning from the training data and becoming more precise.

Purple Line (Validation Precision): The purple line with triangle markers represents the validation precision. Validation precision measures how well the model performs on unseen data (validation set). Initially, validation precision increases in tandem with training precision. However, around epoch 25, there is a noticeable fluctuation in validation precision before it continues to increase. This behavior suggests that the model might be fine-tuning its precision on the validation data.

Interpretation:

The increasing training precision indicates that the model is improving its precision on the training data. The fluctuations in validation precision around epoch 25 warrant attention. Fine-tuning and avoiding overfitting are crucial considerations.

Recommendations:

Monitor the model's performance closely beyond epoch 25. Investigate the cause of the fluctuation in validation precision. Consider techniques like early stopping or hyperparameter tuning to optimize precision.

These minor fluctuation shown in the graph is found to be normal and was tickled by further optimization during processing after prediction and proper image processing before prediction.

5.3 Actual Output

The eye-controlled wheelchair system is designed to enable individuals with limited mobility to operate a wheelchair using their eye movements. By tracking the person's gaze, the system translates specific eye behaviors into wheelchair commands.

5.3.1 Key Features:

Eye Closed (Stop):

When the person's eye is closed, the wheelchair immediately stops. This safety feature ensures that the wheelchair halts whenever the user blinks or closes their eyes.

Looking Straight (Move Forward):

If the person is looking straight ahead, the wheelchair moves forward. The system detects the neutral gaze direction and initiates forward motion.

Looking Left (Turn Left):

When the person looks to the left, the wheelchair turns left. The system interprets the leftward gaze as a command to change direction.

Looking Right (Turn Right):

Similarly, if the person looks to the right, the wheelchair turns right. Rightward gaze signals the system to adjust the wheelchair's heading.

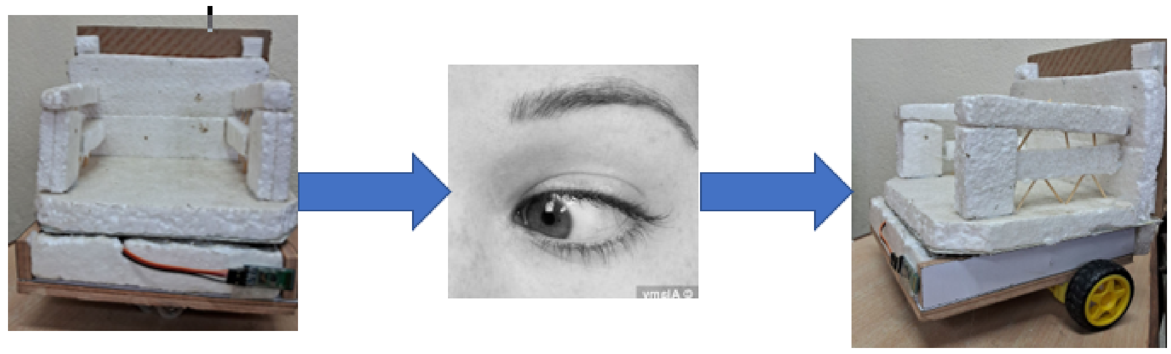


Figure 5.4: Actual output

5.3.2 From Figure:

Straight Chair (Image 1):

The first image shows the wheelchair in its default position, aligned straight ahead. No specific eye movement commands are active at this point.

Person Looking Left (Image 2):

In the second image, the person's gaze is directed to the left. The system interprets this eye movement and initiates a left turn for the wheelchair.

Chair Slightly Left (Image 3):

The third image depicts the wheelchair after executing the left turn. It is now slightly angled to the left, following the person's gaze.

REFERENCES

[1] Kohei Arai and Ronny Mardiyanto. Eyes based eletric wheel chair control system–i (eye) can control electric wheel chair. *International journal of advanced computer science and applications*, 2(12), 2011.

[2] Mahmoud Dahmani, Muhammad EH Chowdhury, Amith Khandakar, Tawsifur Rahman, Khaled Al-Jayyousi, Abdalla Hefny, and Serkan Kiranyaz. An intelligent and low-cost eye-tracking system for motorized wheelchair control. *Sensors*, 20(14):3936, 2020.

[3] Kohei Arai and Ronny Mardiyanto. A prototype of electric wheelchair controlled by eye-only for paralyzed user. *Journal of Robotics and Mechatronics*, 23(1):66, 2011.

[4] Mohamad A Eid, Nikolas Giakoumidis, and Abdulmotaleb El Saddik. A novel eye-gaze-controlled wheelchair system for navigating unknown environments: case study with a person with als. *IEEE Access*, 4:558–573, 2016.

CHAPTER 6

APPENDIX

In the future, our aim is to integrate this prototype onto a real wheelchair, enhancing accessibility for individuals with mobility impairments. Furthermore, we envision expanding its functionality to address the needs of visually impaired individuals. By incorporating additional sensors and advanced AI algorithms, we aspire to develop a comprehensive solution that not only assists in navigating the wheelchair but also provides real-time feedback and alerts to users with visual impairments, thereby promoting independence and safety. This holistic approach reflects our commitment to leveraging technology to improve the quality of life for individuals with disabilities.