Transforming Human-Computer Interaction with Real-Time Gaze-Tracking Technology

An undergraduate project progress report submitted to the

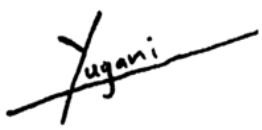
Department of Electrical and Information Engineering
Faculty of Engineering
University of Ruhuna
Sri Lanka

in partial fulfillment of the requirements for the

Degree of the Bachelor of the Science of Engineering Honours

by

N. Hariharasakthy - EG/2020/3956
M.H.A. Minhaj - EG/2020/4076
M.N.S. Morais - EG/2020/4077
J. Praveenan - EG/2020/4119



.....
Ms. O.G.Y.N. Gamlath
(Supervisor)

Abstract

Human Computer interface (HCI) is a multidisciplinary field focusing on enhancing the interface between users and computers. Gaze tracking is one of the developing technologies in HCI that has received a lot of interest due to its potential to improve user experience. In HCI systems, gaze tracking is an essential method for spotting usability problems and real-time gaze tracking improves interactions by observing users' eye movements.

gaze tracking in HCI provides an alternative to typical input techniques like the mouse and keyboard. The system provided by this method is more user friendly, accessible and simple to use allowing for quick task completion without the need for traditional equipment. Particularly, this innovation holds great potential for those with disabilities, as it allows hands free contact with digital interfaces. It serves as a benchmark for expanding accessibility and improving the quality of life for individuals with physical disabilities.

This study presents an innovative approach to HCI by constructing a deep learning model trained to detect gaze movements in real time and operate computer programs accordingly. The fundamental purpose is to build a software solution capable of properly recognizing eye movements and activating predetermined actions. The suggested solution outperforms existing applications by providing dependability, accuracy, cost effectiveness and accessibility. Specifically, the initiative aims to develop computer control for assistive communication, web browsing, digital engagement and gaming applications. Unlike many existing solutions that focus on narrow activities, this system is designed to automate a wide range of common computer operations.

Additionally, a user friendly interface is developed to properly understand and respond to eye movements. The ultimate purpose of this project is to contribute significantly to the evolution of HCI technologies by creating a powerful, adaptive tool that applies computer vision and deep learning to promote accessibility and accelerate user interaction with computers.

Contents

List of Figures

List of Tables

Acronyms

3D	-	Three-Dimensional
API	-	Application Programming Interface
CNN	-	Convolutional Neural Network
DTW	-	Dynamic Time Warping
EAR	-	Eye Aspect Ratio
FPS	-	Frames Per Second
GPU	-	Graphics Processing Unit
HCI	-	Human Computer Interaction
IR	-	Infrared
JSON	-	JavaScript Object Notation
LSTM	-	Long Short Term Memory
MAD	-	Mean Absolute Deviation
MRL	-	MRL Eye Dataset
MSE	-	Mean Squared Error
NLL	-	Negative Log Likelihood
RGB	-	Red Green Blue
RGB-D	-	Red Green Blue with Depth
RMSE	-	Root Mean Squared Error
UCD	-	User Centered Design
UI	-	User Interface

Chapter 1

Introduction

The primary goal of Human Computer Interaction (HCI) has constantly been to produce more natural, intuitive and efficient ways for users to engage with digital systems. Among the new technologies within the HCI domain, eye tracking has established a unique and promising place. Eye tracking technology is gradually used across various domains, including medical diagnosis, marketing, computer vision and human computer interface. It offers a beneficial alternative to traditional input devices such as keyboards and a mouse, which often pose accessibility challenges particularly for those with physical disabilities.

When it comes to this, gaze tracking is an outstanding technique that allows people to communicate with computer systems using just their eye movements. This hands free interaction not only improves accessibility but also enhances the entire user experience. However, current eye tracking technologies have serious drawbacks despite their potential. Many of these technologies lack the accuracy needed for effective communication and are unreasonably expensive. These challenges limit research opportunities for academics and developers interested in eye tracking technology as well as its wide use.

To address solution for these limitations, an increasing number of academics and researchers are working toward developing eye tracking devices that are both affordable and accessible for everyone. Recent innovations in deep learning and computer vision have played an important role in this evolution, significantly increasing the precision and responsiveness of gaze tracking systems. By identifying users' gaze movements these devices may allow seamless involvement with computer applications without the need for physical contact. This innovation not only offers efficiency but also helps those with physical disabilities to independently use computer devices. However, there are still issues with accuracy, usability and adaptability to changing environmental circumstances. The purpose of the study is to address these issues by designing a real-time gaze tracking system that is precise, reliable and cost effective.

1.1 Background

1.1.1 Human Computer Interaction

Human Computer Interaction (HCI) is a multidisciplinary field that tries to enhance and optimize the interaction between humans and computers. The primary objective of HCI is to build systems that promote ease of use, accessibility, efficiency and responsiveness. Traditionally, interaction relied mainly on physical input devices like as keyboards, mouse and touchscreens. However, with developments in technology and a shift toward more natural user interfaces, HCI has expanded to embrace additional modalities such as gesture recognition, speech commands, facial expression analysis and especially, eye tracking technologies[3].

HCI plays a crucial role in the development of new technologies including wearable computing, augmented and virtual reality and context aware systems [3]. The integration of processing power and artificial intelligence has further enabled the real-time deployment of systems like gaze tracking, which significantly reshapes digital interactions and enables adaptive technology.

1.1.2 Gaze Tracking Technology Using Deep Learning and Computer Vision

Gaze tracking has developed as a breakthrough solution within HCI, providing an easy and hands free technique for human computer interaction. Computers can determine attention, intention and interaction context by analyzing the direction and fixation of a user's gaze. This allows for more natural user interfaces. This method is very useful for applications like gaming, immersive virtual worlds, assistive communication and accessibility tools [4].

Convolutional Neural Networks (CNNs), a type of deep learning model, have improved gaze estimating systems' capabilities. CNNs are employed for gaze vector estimation, pupil center identification, iris contour mapping and feature extraction, all of which increase the accuracy of gaze recognition in a variety of head positions and lighting scenarios [3]. Transfer learning implementing pre trained models like VGG16 and AlexNet significantly increases performance in instances with noise and lighting variability [3].

Computer vision techniques improve deep learning by enabling robust recognition and tracking of facial landmarks. Appearance based approaches, which assess entire eye regions rather than individual features, reduce rely on ideal lighting and allow real-time processing without the need for expensive hardware systems [4].

1.1.3 Image Processing and Preprocessing

Image preprocessing is a fundamental stage in gaze estimation systems. It boosts image consistency and reduces background noise to improve the accuracy of feature extraction. Common procedures include illumination normalization, histogram equalization, grayscale conversion and morphological processing [3]. For instance, Zhang et al. [5] designed an appearance based preprocessing pipeline that includes illumination normalization and Haar cascade detection for eye region segmentation. Fischer et al. [6] employed YCbCr color space segmentation, median filtering and background removal to provide resilience and minimal computing cost.

It results in the eye area being isolated by eliminating unnecessary background components. Moreover, grayscale conversion helps to simplify the image,reducing computing effort. It also aids in faster gaze estimation. In order to remove minor artifacts without negatively impacting the image quality, noise reduction is carried out using median filters.

The preprocessing pipeline ensures the robust eye detection and tracking. By ensuring that the gaze tracking system performs well in a range of areas and lighting circumstances, these preprocessing techniques enhance the precision and dependability of eye movement analysis. The system can precisely and intuitively handle computer interfaces by accurately reading human gaze by minimizing background noise and increasing the characteristics of the eye region.

1.1.4 Challenges in Eye Tracking Systems

Despite huge advances, many existing eye tracking systems suffer from restrictions such as high cost, lack of portability and calibration complexity. Commercial systems frequently require infrared light sources and head stabled systems, making them unavailable to the general public or researchers with low resources [7]. Wearable alternatives give greater freedom of movement but impose analytical complications and may have more serious precision issues [8]. Moreover, usability difficulties such as adaptation to lighting conditions, user lack of sleep and limited field of view keep affecting performance in real world applications [3]. These problems emphasize the need for cost effective, real-time gaze tracking frameworks using commonly available hardware like conventional cameras, enhanced through software based solutions leveraging deep learning and computer vision.

1.1.5 User Centered Design and Interaction Enhancements

To maximize usability, modern HCI systems must embrace user centered design (UCD) principles that prioritize user needs, preferences and behaviours during the development process [4]. Gaze based interaction models offer a high degree of personalization and can improve user experience by enabling seamless navigation, command triggering and application control all without physical input.

Additionally, heatmaps, fixation metrics and gaze based interaction patterns have become essential tools for evaluating user attention and interface usability [3]. These tools allow researchers and developers to assess interface design, identify usability flaws and refine interactive systems for better performance and accessibility.

1.2 Problem Statement

Individuals with physical disabilities and movement disabilities experience significant challenges when interacting with computers through traditional input techniques such as keyboards, mouse and touchscreens. These traditional interfaces generally require precise movement, which could limit accessibility, independence and active involvement in digital settings. While assistive technologies such as switch based controls or specialized hardware offer partial relief, they remain expensive, unintuitive and generally hard to implement and maintain [7] [8]. Moreover, existing gaze tracking systems face their own set of restrictions in terms of cost, accuracy, adaptability and real-time response. Many of these systems depend on high end infrared hardware or wearable eye tracking glasses that are expensive and impossible for ordinary people [7] [8]. Others, while aiming for real-time interaction at frame rates such as 30 FPS, frequently require computationally expensive setups with GPU acceleration or dedicated hardware, which prevents their mainstream use and scalability [7] [8].

Accessibility difficulties carry as well. Validation difficulty, lighting sensitivity and variable gaze detection limits the reliability of present systems, especially in uncontrolled conditions [3] [8]. Furthermore, systems lacking strong computer vision pipelines suffer from high noise levels in eye detection, making them unsuitable for users with involuntary head movements or irregular eye patterns [7] [3]. This research aims to solve these restrictions by constructing a real-time, webcam based gaze tracking system that leverages deep learning and computer vision to provide a hands free, simple interface for computer interaction. The suggested solution aims to remove the requirement for specialized hardware, making gaze controlled interaction cost effective, accurate and flexible across a variety of lighting and environment conditions [7] [4]

By including techniques such as convolutional neural networks for gaze prediction and lightweight preprocessing pipelines for noise reduction [7] [3], the system is designed to perform efficiently even on consumer grade hardware [8]. This will offer a cost effective and accessible alternative for providing systems while allowing important computing tasks such as web browsing, application control and assistive communication [7] [3]. Ultimately, this research aims to contribute to the development of Human Computer Interaction (HCI) by enabling users particularly those with physical disabilities to engage with digital systems more independently, efficiently and automatically [3] [4]. By reducing the physical barriers associated with typical input devices, the system seeks to enhance accessibility, increase usability and extend the range of digital accessibility [3].

1.3 Objectives and Scope

1.3.1 Aim

The primary goal of this project is to develop a real-time, cost effective gaze tracking system using deep learning and computer vision to enhance Human Computer Interaction (HCI). The system will enable users, particularly individuals with physical disabilities to interact with and control computer applications hands free through intuitive, gaze based commands. By leveraging a trained Convolutional Neural Network (CNN), the project seeks to provide accurate, responsive eye movement detection integrated into a user friendly software application. The ultimate goal is to improve accessibility, usability and independence in digital environments while eliminating the need for specialized hardware.

1.3.2 Objectives

The main objective of this project is to develop a real-time gaze tracking system using deep learning techniques and transform Human computer Interaction through a software application. The below mentioned objectives can be highlighted.

- Find a dataset to train, design a suitable architecture and develop the model.
- Train and optimize a CNN based model to get real-time response with high accuracy.
- Develop a gaze tracking prototype that can detect eye movements and lead for the control for computer applications.
- Enhance gaze based interactions for actions like selecting options, navigating interfaces and control applications.
- By addressing the existing applications and research gaps introduce a novel approach to system control using gaze tracking technology.
- Compare the gaze based interactions with traditional inputs.
- Ensure the project is cost effective and compatible so it is affordable for daily use.
- Identify the technical challenges and propose solutions specially when having precision errors, calibration issues and other conditions like lighting conditions, user diversity.
- Implement a user friendly application for gaze tracking by utilizing the CNN model and computer vision and software development technologies.

1.3.3 Scope

- Development of a real-time gaze tracking system, that identifies and interprets eye movement of users to have an enhanced interaction with computer applications.
- Integrating developed deep learning model with a software application which allows user to control computer and interact with softwares using the gaze based commands.
- Using the CNN architecture to detect different eye movements with a high precision and accuracy.
- Enhance the HCI by exploring how gaze based controls improve usability and accessibility for both general users and individuals with physical disabilities.
- Develop the gaze tracking system with real-time response with high accuracy.
- Address the problems related to existing systems, seeking the gaps and suggest a novel contribution to Human Computer Interaction (HCI).
- Design an intuitive, hands free control system which can be used by disabled people to interact with computers effectively and efficiently.
- Consider the various conditions like lighting conditions, head positioning, age groups and including users who wear glasses to ensure robustness.
- Introduce a cost effective solution which is affordable and that can be used on ordinary laptops without depending on complex, external hardware requirements.

Chapter 2

Literature Review

2.1 Previous Work

2.1.1 Human Computer Interaction

HCI can be considered as a multidisciplinary field. It includes various contributions. Computer science, cognitive psychology, design and engineering are on the rise. According to Kumar et al., the core objectives of HCI design include usability, safety, utility and efficiency[3].The major focus is only to provide user friendly and effective computer human interactions. Whenever the natural powerful processor which is human try to communicate with another powerful information processor which are computers through a limited bandwidth medium, it's called as Human Computer Interaction. General Human Interactions involve natural communication methods including language, emotions, expressions and gestures, whereas computers may vary from desktops to embedded devices. As people depend more and more on digital systems, mobile apps and smart gadgets, HCI has emerged as a crucial component of daily life [4].In some research papers Human Computer Interaction (HCI) is referred as Man Machine Interaction (MMI), Computer Human Interaction (CHI) and Human Machine Interaction (HMI) [3].

Early systems were built around user command inputs via keyboards and mouce, but these were often rigid, unintuitive and more specifically inaccessible to users with disabilities. The move towards user centered design (UCD) has reshaped HCI by emphasizing user satisfaction, emotional engagement and system adaptability [4].The use of multimodal interfaces such as voice commands, touch input, gestures and eye tracking expands the possibilities for seamless interaction. Kheder et al.further made arguments of the integration of emotional and experiential design into HCI. Especially through systems that adapt to the user's mental model [4].

In early HCI the foundation was WIMP (Windows, Icons, Menus and Pointers) based interfaces. The reason was they allowed the interaction in a hierarchical and structured way [3].But as technology has advanced, interaction paradigms have

changed. Improvements include developing virtual environments, context aware systems and wearable computing, which improves usability and functionality.A user friendly HCI design will be produced by following the well balanced approach between usability and functionality [3].

To get the better understanding of the user intent and enhance the interaction, image based HCI utilizes the facial analysis, gesture recognition and gaze detection. One such advancement in HCI is the integration of real-time gaze tracking technology, which allows users to interact with systems using their eyes. With the help of gesture based Interfaces, users can engage with their digital material in a natural way [9][1].

2.1.2 Eye Tracking as a Modality in HCI

Eye tracking has emerged as a powerful technique in HCI for both control and analysis.Eye tracking technology can be considered as a valuable technique in assistive technology, allowing users with physical impairments to navigate digital environments using only their eye movements. Moreover, it is also used in marketing to track consumer attention, in gaming for adaptive interfaces, healthcare and in education for understanding student engagement. Poole et al.distinguish between the use of eye tracking as an input method (e.g., for gaze based control) and as an analytical tool for usability testing and cognitive workload assessment. Accordingly, metrics such as fixation duration, saccades, scan paths and blink rate can provide deep insights into user behavior and attention[10].

Wearable trackers such as those discussed by MacInnes et al. and Chang et al. They extend the usability of eye tracking into mobile and real world contexts. These systems allow for dynamic mapping of gaze coordinates even during movement. This makes eye tracking applicable in fields like psychology, sports training and rehabilitation [11][12].

Several studies have been conducted using traditional technologies to capture gaze moments with high precision, utilizing Infrared (IR) Cameras and Pupil Center Corneal Reflection (PCCR) technique. However, considering the cost of monitoring and controlled environments required for traditional methods, recent advancements have shifted towards the deep learning and computer vision field for eye movement tracking. Especially, 7 the Convolutional Neural Networks (CNNs) has been widely utilized as the focal point in many research papers.Other than the basic CNN model, some researches have been adopted by YOLOv3 tiny architecture of CNN with 23 layers to execute without the need of GPU accelerator[1][2][13].

Existing Gaze Tracking Approaches

In the classical systems, gaze tracking research is mainly focused on the IR camera to detect the corneal reflections and iris glints. Zhu and Ji proposed a gaze tracking system in which is capable of handling dynamic and natural head movements

which eliminated the need for a static head position while using the eye tracking technologies[14].

In order to monitor and arrange the treatment facilities to the post traumatic stress disorder (PTSD) and eye movement desensitization and reprocessing therapy (EMDR), a device has been developed by Chang et al. using two sensors which are significantly cost effective compared to other sensors to capture the both left and right eye movements. Eventually, a high correlation between the precise measurements of left and right pupils movements was achieved through this proposed approach which led to the innovations of wearable and cost effective tracking applications in the medical field[12].

A notable study by Mohamed et al. focused on reducing road accidents caused by driver fatigue. Unlike traditional systems that rely on infrared sensors, this research proposed a smartphone based solution using an Android device (Samsung Galaxy S4). The purpose was to monitor signs of drowsiness and yawning by analyzing facial expressions and blinking patterns. When the system detects unusually frequent eye blinks which is an indicator of fatigue, after it issues alerts to the driver. The implementation relies on conventional image processing techniques, which combines MATLAB and Java based algorithms for eye tracking. This approach offers a cost effective alternative to earlier systems that required specialized hardware or controlled environments. Furthermore, it suggests expanding this method to support features like automated braking and parking assistance, all through smartphone based eye tracking technology[15].

Deep Learning Based Gaze Tracking Approaches with HCI

The emergence of deep learning, particularly convolutional neural networks (CNNs), has significantly advanced the field of gaze tracking in Human Computer Interaction (HCI). Traditional gaze tracking methods often depend on infrared (IR) illumination, pupil corneal reflection, or geometric modeling of the eye, which necessitates expensive and complex hardware setups. These systems while effective under controlled conditions, struggle with adaptability in dynamic environments and are sensitive to lighting changes and head movements, thus limiting their applicability in real world scenarios [16] [1]. Deep learning, especially CNN based approaches has addressed many of these limitations. This has been done by enabling real-time gaze estimation using standard hardware such as webcams with improved robustness, affordability and user accessibility.

Recent studies have showed the effectiveness of deep learning models in learning complex spatial and contextual features from eye images to estimate gaze direction. For instance Jigang et al. proposed a monocular 3D gaze tracking model that incorporated geometric constraints to improve accuracy under natural head movements. Their system was also extended to mobile platforms. This marked a significant milestone in enabling robust and hardware light gaze interaction solutions [17]. Similarly, Nandini et al. developed a CNN based system trained on thousands of annotated eye images to estimate gaze coordinates with over 85% accuracy using a standard laptop webcam. Their work highlighted the

impact of incorporating efficient preprocessing, data augmentation and carefully tuned optimization techniques in improving real-time gaze estimation performance without the need for specialized equipment [1]. In a parallel effort, Vidhya et al. introduced a shallow CNN model trained on eye region images extracted using MediaPipe and OpenCV, demonstrating high accuracy with reduced computational cost, thus further proving the feasibility of real-time deployment on consumer grade devices [2].

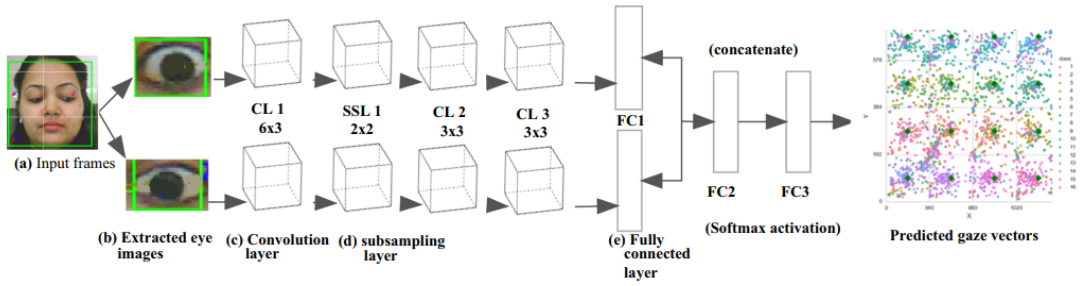


Figure 2.1: CNN architecture for eye gaze estimation[1]

Figure 2.1 illustrates the architecture of a CNN-based model for real-time gaze estimation. The system extracts eye regions from input frames and processes them through convolutional and subsampling layers, which are followed by fully connected layers. The final output layer generates gaze vector predictions, enabling accurate mapping of the user's point of focus[2].

Beyond general interaction, CNN based gaze tracking has been particularly impactful in assistive technologies. Gaze based control offers an intuitive interface for individuals with physical disabilities, enabling hands free interaction with digital systems. Louedec et al., for instance, applied deep learning techniques to study chess players gaze patterns. Furthermore, they used saliency maps to predict user intent based on visual attention. Their work depicts how gaze tracking can assist in cognitive load analysis and adaptive system responses in real time [18]. Similarly, MacInnes et al. developed a wearable gaze tracking system that combines deep learning with facial landmark detection and object mapping, allowing real world interaction scenarios such as gaze object alignment in mobile environments. Despite the advancements, they noted challenges in calibration and robustness under naturalistic conditions [11].

Applications of deep learning based gaze tracking also expand into fields like marketing, gaming and user experience design. Modi et al. leveraged CNNs to analyze user fixation patterns on social media advertisements. They only used standard webcams. Their study on the Pepsi brand Facebook page provided actionable insights for UI layout and marketing optimization through gaze heatmaps and fixation density analysis [1]. In the domain of gaming, gaze tracking is employed to enrich player immersion by allowing the game environment to adapt based on user attention. Louedec et al. further showed this by predicting chess moves based on where players focused on the board, thus linking gaze behavior to strategic cognition [18].

One notable trend in recent literature is the increased focus on using low cost, non intrusive hardware such as integrated laptop webcams and mobile phone cameras for gaze tracking. Figueroa et al. showed that gaze estimation using CNNs could achieve reasonable accuracy without depending on external hardware, making the technology more accessible for mainstream adoption [19]. Their findings are in alignment with those of MacInnes et al., who employed geometric modeling and data fusion to address issues arising from head movement and inconsistent lighting. These approaches are paving the way for more scalable, accurate and deployable gaze tracking systems [11].

In addition to static gaze detection, modern CNN based models are being featured with multimodal capabilities. Vidhya et al. integrated blink detection into their gaze tracking system using a lightweight CNN model. This enabled additional user input methods such as triggering actions by blinking. This interface achieved a blink detection accuracy of over 92% and was successfully used in tasks such as navigating between applications and playing media content hands free, thereby enhancing system interactivity and responsiveness [2]. Real-time responsiveness is further supported by optimized CNN architectures. Optimized CNN architectures are increasingly shallow and efficient. Use of Adam optimizer, dropout and batch normalization allows these models to perform well even on machines without having dedicated GPUs [1], [2].

Evaluation of gaze tracking systems is critical to establish their effectiveness and applicability. Nandini et al. employed a 4×4 grid based gaze calibration technique and evaluated the model performance using accuracy metrics, softmax based classification and Euclidean distance. Their CNN model achieved a training accuracy of 84% and testing accuracy of 85.6%, with heatmaps used to visualize fixation consistency across different users [1]. Vidhya et al. introduced a trajectory based evaluation framework. In there, users followed moving stimuli such as a ball across a screen and gaze trajectories were matched against reference paths using metrics like Mean Absolute Deviation (MAD) and Root Mean Squared Error (RMSE). These techniques offered more dynamic insights into gaze estimation performance in real-time contexts [2].

Despite these advancements, several limitations remain in the deployment of CNN based gaze tracking. Many models require large diverse datasets for effective training. Additionally, performance can degrade significantly when applied to users with varying facial features eye shapes or under varying lighting conditions. Jigang et al. noted that even state of the art 3D models may struggle with generalization in uncontrolled environments specially when users wear glasses or deviate from expected head positions [17]. Additionally,the limited resolution of webcam input poses challenges for tasks requiring fine grained control such as virtual keyboards or precise cursor manipulation [2].

Looking forward, further research is needed to enhance the robustness and scalability of gaze tracking systems. Promising directions include the integration of temporal deep learning models such as Long Short Term Memory (LSTM) networks and Transformer architectures to capture eye movement trends over the time, potentially improving prediction continuity and context awareness [20]. The

incorporation of synthetic data generation, domain adaptation and unsupervised learning may also help address dataset limitations and improve generalization across different user groups. More specially, for users having physical impairments.

2.1.3 Model Training, Validation and Testing Methods

The effectiveness of deep learning based gaze tracking systems in Human Computer Interaction (HCI) is largely dependent on the design and execution of the model training, validation and testing phases. These stages are important in ensuring that the model can generalize to unseen users, handle real-world variability and deliver responsive interaction in real time. With the introduction of Convolutional Neural Networks (CNNs), numerous studies have explored their application in gaze tracking using diverse datasets, architectural variations and performance evaluation strategies.

CNN architectures form the backbone of most real-time gaze estimation systems. This is due to their capability to automatically learn hierarchical spatial features from raw image inputs. Vidhya et al. developed a CNN model that consisted of three convolutional layers with increasing filter sizes (32, 64 and 128), followed by max pooling to reduce dimensionality and prevent overfitting. The feature maps were then passed through two fully connected layers with 128 and 64 neurons respectively, activated with ReLU. The final output layer, trained to predict continuous gaze coordinates (x,y), used the Mean Squared Error (MSE) as the loss function and was optimized using the Adam optimizer with a learning rate of 0.001. The dataset was split with 80% allocated for training and 20% for validation. Performance across epochs was analyzed using loss and accuracy curves, with the training loss stabilizing after approximately 20 epochs. The model achieved a final MSE of 0.0112 and an R-squared value of 0.9593. This indicates strong predictive accuracy and alignment with actual gaze positions [2].Figure 2.2 illustrates the model's performance across training epochs using three key metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2). As shown, both MSE and MAE decrease consistently, indicating improved prediction accuracy, while R^2 values increase.

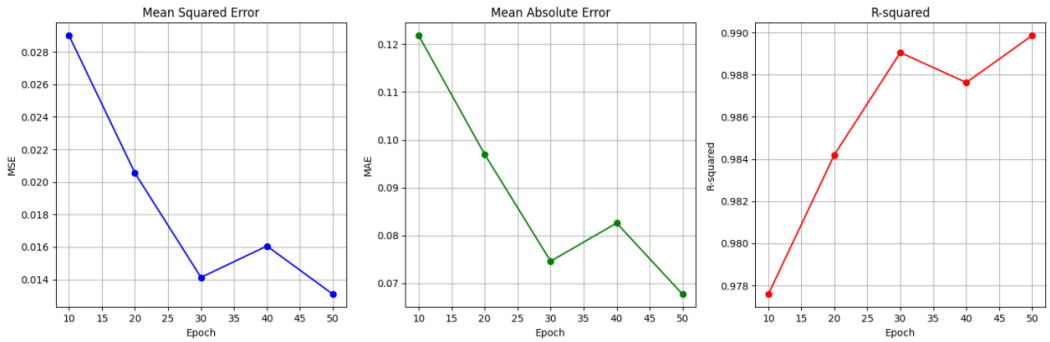


Figure 2.2: MSE (left), MAE (middle), and R-squared (right) values across epochs.[2]

In another prominent study, MF Anzari et al. proposed three separate CNN architectures tailored for different input perspectives. They are for one eye, both eyes and full face. Each model was trained on a dataset that included thousands of images under varying conditions, such namely users wearing glasses. The architectures employed ReLU activation, Xavier initialization for weight distribution and used Negative Log Likelihood (NLL) as the loss function, combined with Softmax activation at the output to classify gaze into 20 discrete zones. The both eyes model performed best among the three, utilizing parallel convolutional stacks for each eye before concatenating the outputs into a shared fully connected layer. This architecture allowed the system to capture symmetrical gaze features and improve classification accuracy under diverse viewing conditions [21].

Nandini et al. developed a CNN trained on a dataset containing 13,940 labeled samples. It was collected from 41 participants. This was done by individuals focusing on predefined gaze targets across a 4×4 screen grid. Their model architecture included three convolutional layers a max pooling layer and three fully connected layers. Images were processed and divided into a $70 : 30$ training to testing ratio.Gradient descent was used to optimize the model. They achieved a training accuracy of 84% and a testing accuracy of 85.6%. The results were further validated through loss analysis, accuracy graphs and gaze heatmaps, demonstrating consistent and reliable performance under test conditions [1].

Preprocessing plays a vital role in the effectiveness of all models. In the work of Vidhya et al., facial landmarks were detected using MediaPipe and the eye region was extracted using OpenCV. Each eye image was resized to 256×256 pixels and converted to grayscale, providing consistent input dimensions for training. In contrast, Nandini et al. implemented a frame by frame preprocessing pipeline using RGB video input. Each frame was first converted to grayscale, followed by histogram equalization to enhance contrast. Haar cascade classifiers were used for facial and eye detection and Gaussian filtering helped suppress noise. The pupil region was further refined using Canny edge detection and a gradient based algorithm for estimating the center of the pupil, enhancing accuracy in gaze direction prediction [1].

Model validation and testing are essential for evaluating generalization and robustness. Vidhya et al. implemented both static and dynamic validation. In the static phase, users focused on points displayed on a 4×4 screen grid and predicted gaze points were compared to true values using Euclidean distance and Root Mean Squared Error (RMSE). Gaze heatmaps and scatter plots visually confirmed prediction accuracy. In the dynamic validation phase, participants were asked to follow a moving object across the screen along reference trajectories such as circles and zigzags. The predicted gaze trajectory was compared with the reference using metrics including Mean Absolute Deviation (MAD), Dynamic Time Warping (DTW) and RMSE. A threshold based accuracy metric was also introduced to measure the proportion of predictions within 150 pixels of the reference trajectory, providing a flexible framework for real-time system evaluation [2].

Similarly, Nandini et al. validated their model using saliency visualizations and fixation density heatmaps. Predicted gaze points were overlaid on screen regions

of interest, allowing for qualitative validation in addition to numerical metrics. These visual assessments showed high consistency between predicted and actual gaze areas, especially in regions of high user attention [1]. To improve generalization and convergence during training, several optimization techniques were applied across the studies. Adam optimizer with low learning rates (0.001 or 0.00015) was widely used for stable and adaptive gradient updates [2], [21]. Dropout layers, typically set at a rate of 0.5, were employed to mitigate overfitting [1]. Weight initialization techniques like Xavier and He initialization helped maintain signal flow across deep networks while batch normalization was applied to stabilize learning by keeping input distributions consistent across layers [2]. Evaluation metrics used to assess model performance included Mean Squared Error (MSE) to quantify prediction error, Euclidean distance for measuring spatial accuracy and classification accuracy. Vidhya et al. also reported an R-squared value of 0.9593, indicating the model's ability to explain variance in gaze behavior. DTW and MAD were used for analyzing dynamic trajectories, while gaze heatmaps served as a qualitative tool to visualize user attention and track focus over time [2], [1].

2.1.4 Image Processing Techniques

Image processing plays a foundational role in determining the accuracy, reliability and real-time performance of gaze tracking systems. In deep learning based gaze estimation the clarity, consistency and quality of input images directly affect how well the system can learn gaze relevant features and generalize to varied real world conditions. The image processing pipeline from raw acquisition to final input significantly influences the ability of the system to detect eye movements, pupil positions and ultimately predict gaze coordinates with precision, particularly in Human Computer Interaction (HCI) applications.

The effectiveness of Convolutional Neural Networks (CNNs) in gaze tracking is tightly coupled with the quality of eye region inputs. Poor contrast noise inconsistent image resolutions and changing lighting conditions can introduce noise into the training process and degrade inference performance. Preprocessing is essential to reduce background clutter emphasize critical features such as eye contours and pupils and standardize inputs for the CNN model. In their work, Vidhya et al. applied OpenCV and MediaPipe for facial landmark detection, extracting 468 landmarks and isolating those around the right eye. The extracted eye regions were resized to 256 $\times$ 256 pixels and converted to grayscale, providing consistent image dimensions and reducing unnecessary color information. This preprocessing approach reduced training complexity and memory usage while increasing accuracy and reliability in real-time settings [2].Figure 2.3 illustrates the preprocessing pipeline used by Vidhya et al.

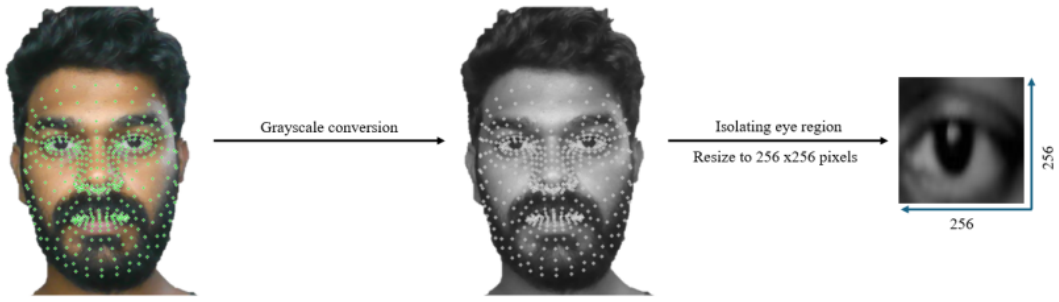


Figure 2.3: Image Pre-processing: Landmark detection and eye-region segmentation [2]

In video based systems, preprocessing must be done on each frame to ensure consistency. Nandini et al. implemented a real-time preprocessing pipeline that processed video streams captured via a low cost webcam. Each RGB frame was first converted to grayscale, followed by histogram equalization to enhance contrast and bring out features like the iris and pupil. Eye detection was performed using Haar cascade classifiers, which scan the image and detect the sharp contrast differences between the dark pupil and the surrounding sclera. After detecting the eye region, Gaussian filtering was applied to smooth the image and remove high frequency noise. This was followed by Canny edge detection, which helped extract the contours of the pupil and surrounding eye features by identifying high gradient edges. This edge based representation is particularly effective for locating the circular boundaries of the pupil even in noisy or poorly illuminated environments [1].

Pupil center detection plays a vital role in accurate gaze vector estimation. In the same study, Nandini et al. employed a gradient based algorithm to locate the pupil center by identifying points of maximum gradient convergence. To reduce computational load detection was performed on every alternate frame. Which balanced performance and accuracy, reducing processing complexity to logarithmic time. Further refinement was achieved using a joint probability model that incorporated data from both eyes allowing the system to statistically constrain the region of interest and improve robustness under partial occlusions or when one eye was temporarily undetected [1].

Early gaze tracking systems often relied on the Viola Jones object detection framework, introduced by Paul Viola and Michael Jones, which uses Haar like features and cascading classifiers to detect faces and eyes in varying image scales. Although this method remains effective in structured environments, it is sensitive to lighting changes and struggles in naturalistic conditions. Nevertheless, it continues to be used as a base step in many systems before finer eye region extraction techniques are applied [22]. When combined with grayscale conversion and contrast enhancement, Viola Jones can form a solid foundation for eye localization in CNN based pipelines.

Edge detection and gradient based mapping are fundamental to locating and isolating the pupil. Canny edge detection filters out short and low gradient edges,

retaining only sharp transitions, which are useful for identifying the dark circular boundary of the pupil. Once these edges are extracted, gradient directions are computed and used to determine convergence points, allowing precise estimation of pupil center. These methods prove especially beneficial in low light conditions and reduce false detections. Additionally, thresholding techniques applied to gradients ensure that only meaningful edge vectors contribute to gaze estimation, thereby improving prediction robustness [1].

The final preprocessing step often involves normalization of pixel values. Scaling all input image intensities between 0 and 1 removes contrast based bias and ensures that the CNN model receives stable inputs during training. This normalization improves convergence rates and helps avoid gradient explosion or vanishing issues during backpropagation. Vidhya et al. further employed data augmentation techniques such as brightness adjustment rotation and zooming to simulate natural head movements and lighting variability. This increased the dataset size artificially and enhanced model generalization to unseen conditions [2].

Efficient preprocessing is particularly critical for real-time gaze tracking systems running on low resource devices. Various computational shortcuts have been adopted, such as skipping frames for pupil detection, processing only regions of interest and resizing images to smaller dimensions. MF Anzari et al. demonstrated how optimized CNN architectures paired with lightweight preprocessing techniques could maintain performance while reducing inference time. Their models were trained on datasets that included subjects with and without eyeglasses, emphasizing the robustness of preprocessing under reflective glare and occlusion conditions [21].

2.1.5 Gaps in Literature

Despite the notable advancements in gaze tracking systems for Human Computer Interaction (HCI), significant research gaps remain particularly in the areas of hardware dependency, robustness under natural conditions dataset diversity software usability, application scope and real-time multitasking. While deep learning particularly .CNNs has enhanced accuracy and responsiveness many approaches still face constraints in delivering scalable, affordable and user friendly solutions for broader adoption.

A persistent issue in current systems is the reliance on specialized hardware. Many gaze trackers use infrared (IR) cameras, RGB-D sensors, or proprietary devices like Tobii EyeX, which, though accurate, are expensive and complex to set up. For example Shehu et al. proposed a remote gaze tracking system using RGB-D and IR cameras that significantly improved precision, but its high hardware demands make it unsuitable for general use [16]. Similarly, Hasegawa et al. developed an event based system using a neuromorphic camera to capture pixel level changes for gaze and blink tracking, offering superior low light performance but suffering from limited accessibility and scalability due to specialized equipment [23]. In contrast, studies like those by Vidhya et al. and Figueroa et al. explored CNN powered gaze tracking using only standard webcams offering

affordable alternatives. However webcam based systems face challenges related to low resolution, sensitivity to lighting changes and decreased tracking reliability during head movement or partial facial occlusion [2], [19].

Another critical limitation is the decline in model accuracy under uncontrolled or dynamic conditions. Jigang et al. proposed a monocular 3D gaze tracking system that employed geometric constraints to account for natural head movements. Nevertheless the system accuracy degraded when exposed to reflective surfaces, glasses, or variable lighting common factors in real world settings [17]. Moreover, CNN based models tend to be sensitive to variations in user specific features like eye shape, skin tone and camera alignment, all of which can distort feature extraction and reduce gaze prediction consistency. This presents significant obstacles for applications requiring high precision, such as eye typing or adaptive attention tracking.

(a) Navigation to new webpage

(b) Zoomed image in new window

Figure 2.4: Blink-based user interactions using gaze fixations to navigate webpages, zoom logos, and open video content [1].

Figure 2.4 demonstrates a gaze-controlled interface where user actions are triggered by eye fixation combined with double blinks. Users can navigate to new webpages, zoom into brand logos, or play videos simply by looking and blinking. Heatmaps indicate areas of maximum gaze concentration. This interaction model enhances hands-free web navigation using only visual attention and blinking cues.

Dataset quality also presents a prominent bottleneck in advancing gaze tracking performance. Many gaze tracking systems are trained on limited datasets collected in controlled environments with minimal variation in age, ethnicity and visual conditions. This leads to dataset bias and poor generalization. MF Anzari

et al. emphasized this challenge in their study by comparing model performance on datasets with single and multiple participants. The models trained on homogenous data performed well within that domain but showed reduced accuracy in new or uncontrolled contexts [21]. Jigang et al. similarly acknowledged that limited sample diversity in training data restricted their model adaptability and real world applicability [17]. Moreover, the scarcity of publicly available, standardized and annotated gaze datasets hampers reproducibility and progress in this field. There is a pressing need for inclusive datasets that encompass a broader spectrum of lighting environments camera angles, eye morphologies and user demographics.

Usability is another under addressed area in current research. Many gaze tracking studies focus solely on evaluating technical performance metrics such as classification accuracy precision, recall or latency, without translating their models into deployable software solutions. As a result there is a gap between research prototypes and systems usable by end users, particularly individuals with physical disabilities. Khaleel et al. suggested extending gaze tracking functionality to support diverse computing environments including mobile platforms. Yet most systems still operate as single use applications such as cursor control or visual attention logging without integrating into multifunctional and user friendly software interfaces [24]. This lack of holistic system design restricts gaze tracking's potential as a general purpose HCI modality.

The narrow focus of existing applications is another limitation. For instance, Modi et al. developed a CNN based system to analyze gaze fixations on a branded Facebook page for marketing insights. Although effective in identifying visual attention patterns this system was tailored to a single use case and lacked broader integration capabilities [25]. Likewise Louedec et al. used gaze tracking in a chess gameplay context to analyze focus regions and predict player moves using saliency maps. While innovative, the model was designed exclusively for chess applications and did not generalize to broader HCI needs [18]. These examples underscore a pattern of specialized models with limited applicability, which undermines the goal of creating universal hands free control systems.

Real-time performance and multitasking capabilities also remain underdeveloped in many CNN based gaze tracking systems. Although CNNs have made real-time inference possible on consumer grade hardware, latency and computational demands remain challenges particularly for systems requiring high resolution input or complex preprocessing pipelines. On devices lacking dedicated GPUs, this often results in lag or frame skipping, reducing usability. Moreover, only a few existing systems support multitasking, such as controlling multiple applications or functions simultaneously using gaze alone. This is especially critical for users with severe motor impairments, who depend on gaze interaction for full computer access and navigation.

To address these gaps current project proposes the development of a real-time, CNN based gaze tracking system using only built in webcams. The system aims to eliminate hardware dependency while ensuring affordability and accessibility. It will be trained on a diverse dataset representing different user demographics, lighting environments and head poses to improve generalization. The image pro-

cessing pipeline will be optimized to reduce latency and enhance performance under variable conditions, including users wearing glasses. Furthermore, the project emphasizes the development of a user friendly software interface that integrates gaze tracking with various functionalities such as system navigation, media control and communication tools. By supporting multitasking and gaze triggered interactions, the proposed system offers a scalable and inclusive solution tailored to enhance everyday human computer interaction.

Chapter 3

Methodology

This section provides detailed information about the systematic process of developing a robust Human-Computer Interaction (HCI) system using gaze and voice control. The system is designed and developed to detect the gaze direction by combining Convolutional Neural Networks (CNNs) with iris and eye landmarks, allowing real-time application control. Additionally, voice commands enhance usability. Figure 3.1 shows the overview of the research methodology. The development process is divided into the following core phases:

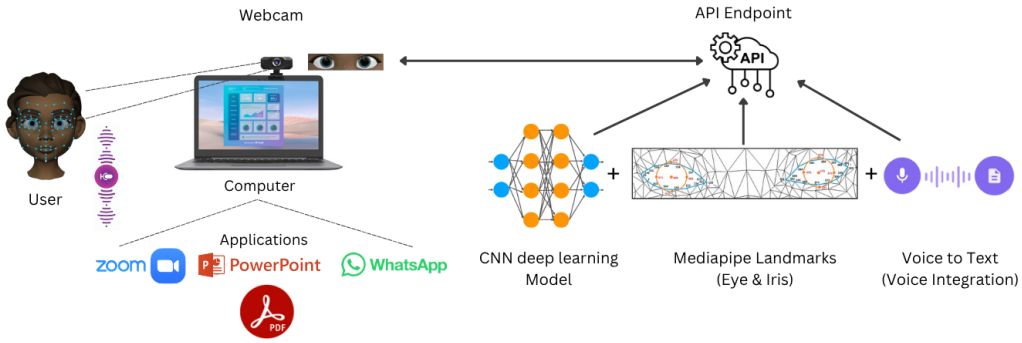


Figure 3.1: Overview of the Research Methodology.

3.1 Research Background and Model Selection

Before beginning the development process, a comprehensive review of previous literature and research papers was conducted to identify robust and real-time capable techniques of gaze-based HCI systems for accurately classifying gaze direction using webcam input. Among these deep learning models researched, Convolutional

Neural Networks have been widely used in gaze direction detection tasks due to their strong performance in image classification.

Research by Zhang et al.[5] proposed a CNN-based appearance-based gaze estimation, which achieved significant accuracy even with low-resolution webcam images. A recent study by Khaleel et al.[26] proposed a hybrid gaze tracking system that combines both CNN-based classification and iris-landmark geometric processing for real-time accuracy. Their model uses facial landmarks to segment eye regions and CNN to predict gaze direction. Based on the findings from these previous researches the CNN model was chosen as the core architecture for gaze direction estimation in our project due to its adaptability, high performance and real-time capabilities.

3.2 Data Collection

A combination of publicly available and custom-collected datasets was used to develop a reliable gaze classification CNN model. Initially, a collection of eye images representing the left, right, center, up, and down gaze directions was collected from various internet sources to establish a baseline for model training. However, in order to achieve real-world performance and generalizability, a custom dataset was created by capturing images from multiple individuals using a laptop webcam under different lighting conditions from users with diverse facial features to ensure robustness across environments.

3.3 CNN Model Building

The core component of the gaze detection system is a CNN model trained to classify eye images into five gaze directions: left, right, center, up, and down. To ensure consistency throughout the dataset, eye images from both publicly available and custom datasets were preprocessed by detecting the eye region, converting it to grayscale, and resizing it to a uniform input size of 64×56 pixels. The CNN model architecture was developed using the Keras deep learning framework, which was designed to extract meaningful spatial features from the eye region, automatically learning the variations in iris positioning.

The labeled dataset was divided into training and testing subsets to evaluate performance objectively. During the training process, the model was optimized using categorical cross-entropy loss and monitored for accuracy and generalization through validation metrics. Upon completion, the trained model could classify new eye inputs in real time, outputting both the predicted gaze direction and the associated confidence score. This CNN model forms the backbone of the real-time gaze tracking pipeline integrated into the HCI system.

3.4 Real-Time Gaze Tracking System

Following the CNN model training, a real-time gaze tracking system was developed using OpenCV, MediaPipe, and the trained CNN model. The webcam continuously captures live video frames. Each frame is processed using MediaPipe FaceMesh, which detects and maps 468 facial landmarks. From these landmarks, the eye regions and iris coordinates are extracted, and then the identified eye regions are cropped and preprocessed to ensure a consistent format for the input requirements of the CNN model. These images are passed through the trained CNN model for gaze prediction. The system provides high performance with real-time interaction without the need for specialized hardware.

3.5 Gaze Calibration Process

To ensure robust and reliable gaze detection estimation across different lighting, facial conditions, and head orientations, a personalized calibration setup was implemented within the system. During the calibration process, the user is instructed to look at five predefined points (left, right, up, down, and center) on the screen, each displayed for 5 seconds. While the user focuses on each point, the system captures a series of frames using a webcam and uses MediaPipe FaceMesh[27] to extract precise eye and iris landmarks. From these landmarks, the horizontal and vertical iris offsets are computed and averaged for each frame, and then the system derives direction-specific thresholds. These thresholds are saved in a JSON configuration file and later used in real-time gaze direction prediction by combining landmark-based estimation with CNN predictions. This calibration process significantly enhances system accuracy by considering the individual variations such as eye shape, iris size, and screen distance.

3.6 Blink Detection Using Eye Aspect Ratio

In addition to gaze direction, the system also uses blinks as a form of input, which are detected using the Eye Aspect Ratio (EAR). EAR is calculated using the distance between points on the upper and lower eyelids. When a user blinks, the eye closes, and this distance becomes much smaller. The system continuously checks this distance in real time through the webcam. If it goes below a certain threshold value, the system confirms it as a blink. Once a blink is detected, it can be used as a command to control the computer applications. This feature makes the system more interactive and user-friendly.

3.7 Voice Integration for Application Interaction

To enhance user interaction beyond gaze direction and blinking, a voice-to-text converting model was implemented using OpenAI's Whisper and integrated into the gaze tracker application, which is used to perform tasks such as typing messages or searching content. The system records voice input using the microphone and passes it through the Whisper speech recognition model, which transcribes spoken words into text. This feature enables users to open desktop applications such as media player, zoom, powerpoint, and perform app actions such as typing messages in Zoom chat or Whatsapp using their voice alone.

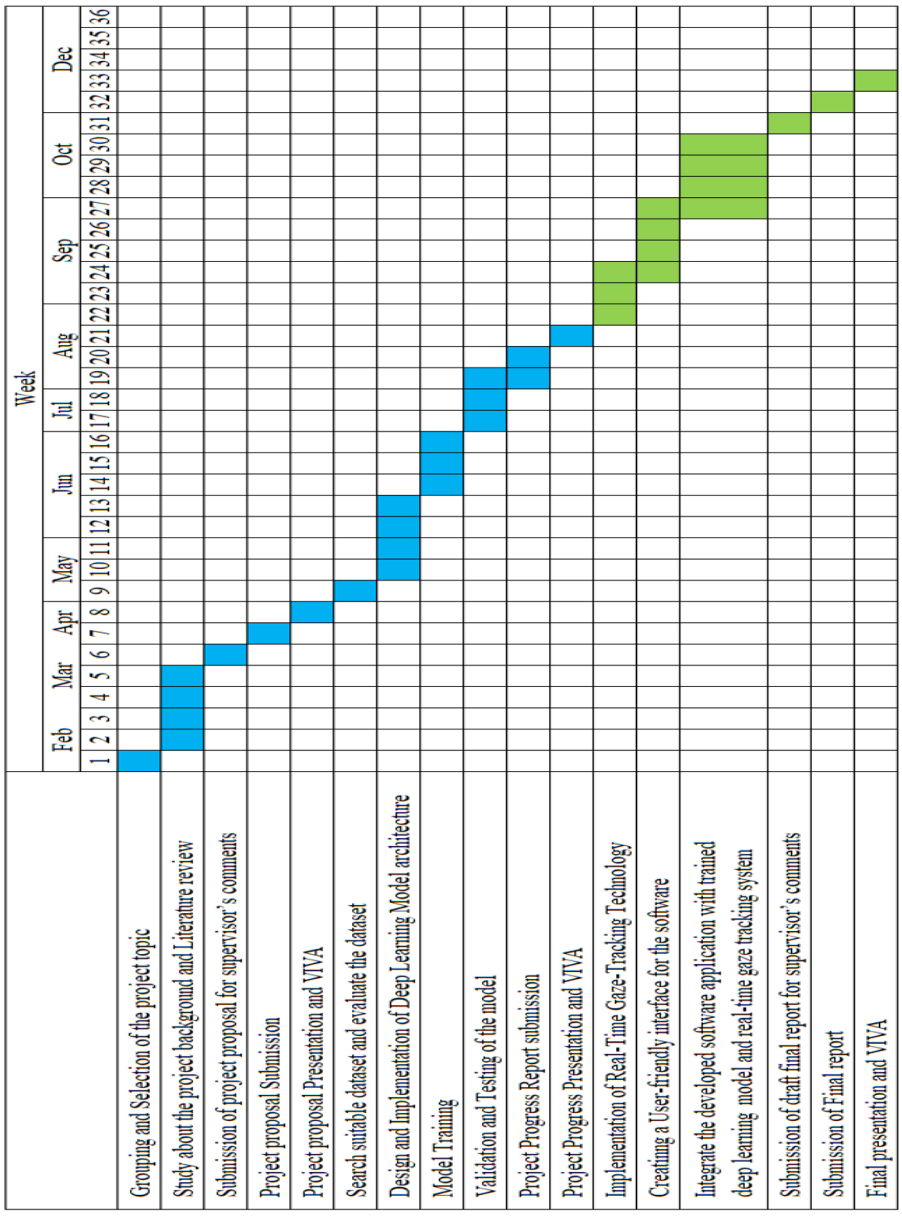
3.8 Application Control Framework

A Python-based computer application control framework was developed using the pyautogui and win32gui libraries to enable gaze-based interaction with active desktop applications. The system keeps monitoring the currently active application window using win32gui and, based on the detected gaze direction or blink input, triggers specific keyboard actions through pyautogui. This allows users to control computer applications without hands, only using the gaze direction or blinking. The framework supports several commonly used desktop applications such as PowerPoint, Media Player, Whatsapp, Zoom and etc. This framework converts gaze inputs into real-time commands, providing an accessible and intuitive way to interact with computers without using input devices such as a mouse and keyboard.

Chapter 4

Work Plan

Table 4.1: On going work plan



Chapter 5

Discussion

The goal of this project is to build a smart and user-friendly gaze tracking system that allows people with physical disabilities to control computer applications using only their eye movements and voice commands. The system was implemented using a hybrid approach that combines a CNN-based gaze classification model with real-time eye landmark analysis.

5.1 Dataset Collection

A diverse and labeled dataset of eye images was used to build the CNN gaze classification model. For this task, both publicly available datasets and a custom-collected dataset were used. The publicly accessible datasets were downloaded from various internet sources based on their quality and requirements.

The Mendeley Gaze Dataset [28] contains a total of 7,500 eye gaze images collected from 12 people. All images are labeled with one of five gaze directions: left, right, up, down, and center. This dataset is evenly distributed, with 1,500 images per class. The SBVPI Sclera Dataset [29] provides 1632 labeled eye images suitable for training directional classifiers. The Kayvan Shah Eye Dataset [30] includes 14.5k regularized and augmented eye images under four classes: close, forward, left, and right, while the MRL Eye Dataset [31] provides eye images captured under varying lighting conditions.

In addition to downloaded datasets, a custom dataset of a total of 1,012 eye images was created, which was collected from 12 different individuals using a standard laptop webcam. Each participant was instructed to look at five directions in various lighting conditions, while full-face images were captured. These were then processed by cropping the left and right eyes separately to a size of 64×56, followed by conversion to grayscale, which ensures the robust and reliable gaze prediction of the CNN model.Figure 5.1 and Figure 5.2 show sample images from the download and custom datasets, respectively. Each figure includes one representative image from each gaze direction class.

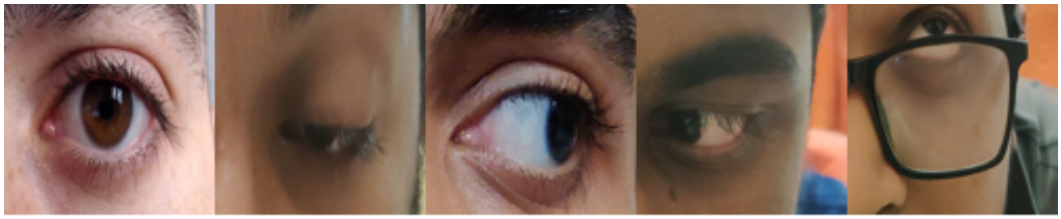


Figure 5.1: Sample Images From Downloaded Dataset.

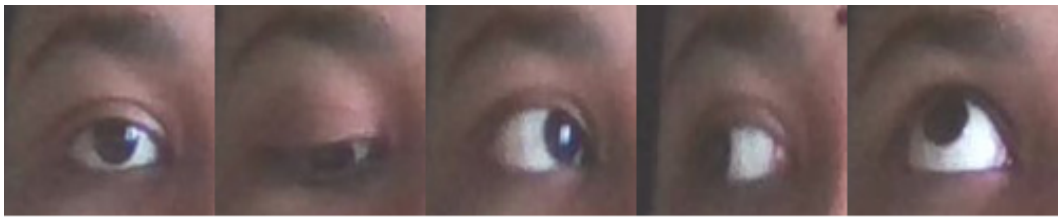


Figure 5.2: Sample Images From Custom Dataset.

5.2 CNN Model Architecture

A CNN deep learning model was designed and implemented using the Keras framework for more accurate classification of gaze direction. This process involved three main stages: data preprocessing, model architecture design and training, and performance evaluation.

5.2.1 Data Pre-Processing

The collected eye images were converted to grayscale and then resized to 64×56 pixels, maintaining a consistent input shape compatible with the CNN model. Additionally, to increase the diversity of the training images, data augmentation techniques were applied using Keras ImageDataGenerator shown in Figure 5.3.

```
traindata_gen = ImageDataGenerator(
    rotation_range=10,      # Randomly rotate images by ±10 degrees
    rescale=1/255.,         # Normalize pixel values from [0,255] to [0,1]
    width_shift_range=0.1,  # Randomly shift image horizontally by up to 10%
    height_shift_range=0.1, # Randomly shift image vertically by up to 10%
    shear_range=0.1,        # Apply shear transformation (slanting)
    zoom_range=0.1,         # Random zoom in/out by ±10%
    fill_mode='nearest'     # Fill empty pixels after rotation/shifts with nearest pixel value
)
```

Figure 5.3: Code Implementation of Data Augmentation.

This involved several transformations, including image rotations (up to ±10°) to simulate natural head tilts, rescaling image pixel values from range [0, 255] to

[0, 1] for normalization, horizontal and vertical shifts up to 10% to handle slight changes in eye position, shearing transformations to introduce slanting distortions, zooming in or out the images by ±10% to recognize features at various scales, Additionally, any empty pixel resulting from these transformations were filled with nearby pixel values to preserve image structure. These data augmentation techniques allowed the deep learning model to learn various scale features from the eye images, reduced overfitting, and improved model performance across different users and lighting conditions.

5.2.2 Model Architecture and Training Details

The model uses a sequential CNN of three convolutional layers with filter sizes 32, 64, and 128, respectively. Each convolutional layer is followed by a MaxPooling layer to reduce spatial dimensions. After the final convolution, a Flatten layer converts the feature maps into a 1D vector array of size 7168. This is followed by a Dense layer with 512 neurons, using ReLu activation and dropout for regularization. The final output layer with 5 neurons and a softmax activation function, representing the 5-gaze direction (left, right, center, up, and down) classes. Figure 5.4 shows the full structure of the sequential CNN architecture used in the model.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 56, 64, 32)	320
max_pooling2d (MaxPooling2D)	(None, 28, 32, 32)	0
conv2d_1 (Conv2D)	(None, 28, 32, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 16, 64)	0
conv2d_2 (Conv2D)	(None, 14, 16, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 7, 8, 128)	0
flatten (Flatten)	(None, 7168)	0
dense (Dense)	(None, 512)	3,670,528
activation (Activation)	(None, 512)	0
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 5)	2,565
activation_1 (Activation)	(None, 5)	0

Figure 5.4: Code Implementation of Data Augmentation.

The model was compiled using the categorical cross-entropy loss function and optimized with the Adam optimizer. The model was trained for 15 epochs on the training dataset, which contained over 16,000 labeled images.

Table 5.1: Training and Testing Dataset Image Count Per Each Class

Direction	Training Dataset	Testing Dataset
Center	3,083 images	797 images
Down	3,119 images	873 images
Left	3,260 images	800 images
Right	3,555 images	901 images
Up	3,264 images	816 images

The full dataset was split into training and testing sets, and all images were organized into five gaze direction classes: left, right, up, down, and center. The distribution details are shown in Table 6.1.

5.2.3 Model Evaluation

The model training had continuous improvement in both training and validation performance over the epochs. The model achieved an accuracy of approximately 94% on the training set and a validation accuracy of 96%, indicating strong generalization and low overfitting. The final epoch validation loss was approximately 0.12, showing that the model effectively minimized error during training. The model learned rapidly in the early stages, and later training helped improve its predictions even more.

5.3 Gaze Calibration Process

To enhance the accuracy and performance of the system across different laptop screen sizes, the gaze calibration process was implemented. During the calibration process, the user is instructed to look at four fixed reference points displayed on the screen: Left, Right, Up, and Down. While the user focuses on each point, the system uses MediaPipe FaceMesh to extract corresponding eye and iris landmarks shown in Figure 5.5. For each direction, specific iris-to-eye offsets are calculated to determine how much the iris shifts relative to the eye boundary.

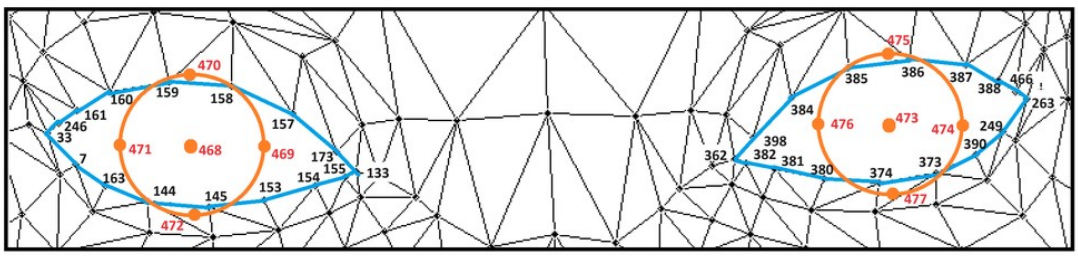


Figure 5.5: Code Implementation of Data Augmentation.

Left/Right, Up, and Down threshold values are calculated using Eq.(5.1), Eq.(5.2), and Eq.(5.3), respectively.

- x: X-Coordinate of the eye and iris landmarks
- y: Y-Coordinate of the eye and iris landmarks

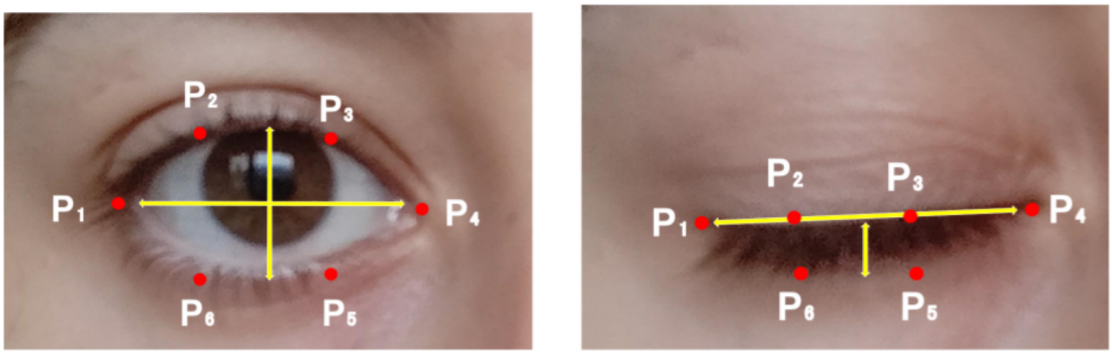
$$\frac{1}{2}\left[\left(\frac{x_{469}+x_{471}}{2}-\frac{x_{33}+x_{133}}{2}\right)+\left(\frac{x_{474}+x_{476}}{2}-\frac{x_{362}+x_{263}}{2}\right)\right] \tag{5.1}$$

$$\frac{1}{2}\left[\left(\frac{y_{470}+y_{472}}{2}-\frac{y_{159}+y_{145}}{2}\right)+\left(\frac{y_{475}+y_{477}}{2}-\frac{y_{386}+y_{374}}{2}\right)\right] \tag{5.2}$$

$$\frac{1}{2}\left[(y_{374}-y_{386})+(y_{145}-y_{159})\right] \tag{5.3}$$

5.4 Blink Detection Using Eye Aspect Ratio

In addition to gaze direction, we included the blink as another input method. The blink is detected based on the Eye Aspect Ratio (EAR) calculation using specific facial landmarks around the eye shown in Figure 5.6. The EAR is calculated using Eq.(5.4).



(a) Open Eye (b) Closed Eye

Figure 5.6: Eye landmarks position for the EAR calculation with open eye and closed eye.

$$EAR = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2 \times \|P_1 - P_4\|} \tag{5.4}$$

Here, P1 to P6 represent six main points around the eye. When the eye is open, the vertical distance between the eyelids is large compared to the horizontal width of the eye, which gives a higher EAR value. When the user blinks, the eyelids come closer together, reducing the vertical distance and causing the EAR to drop. below a defined threshold. If this drop continues for a few frames, it is detected as a blink. These blinks are then used as inputs to perform specific actions on the computer applications.

5.5 Application Control Framework Development

A Computer application control framework is developed using input as gaze direction and blink detection. The framework uses the win32gui Python library to detect the currently active window on the computer and pyautogui to execute the corresponding keyboard inputs. File Explorer, Adobe PDF Reader, Windows Media Player, Microsoft PowerPoint, Zoom, WhatsApp, and Snake vs Fruit Game are the applications that were integrated with gaze control. Each application is controlled with a specific logic block where gaze directions such as left, right, up, down, and blink are mapped to appropriate keyboard shortcuts. This approach makes the gaze tracking system flexible, scalable, and easy to extend to additional desktop applications.

The File Explorer allows horizontal or vertical file navigation based on the left and right gaze detection, while a blink opens the selected file. In Adobe PDF Reader, left and right gaze directions are used to scroll up and scroll down, respectively, and up and down directions are mapped to zoom in and out, while a blink closes the PDF reader. In Media Player, gaze direction allows for playing or pausing the music and videos and enables navigation between the songs and videos. Microsoft PowerPoint supports file browsing and selection; additionally, it allows slideshow mode, left and right slide navigation. Zoom integrates gaze commands for features like mute or unmute, video on or off, and chat opening. In WhatsApp, gaze detection is used to navigate between users' chats, select a user's chat, and send messages.

5.6 Voice-to-Text Conversion Model

To enhance the usability and support multimodal interaction, a voice-to-text conversion model was integrated with a gaze tracking system. This additional feature allows users to perform tasks like opening desktop applications, typing messages, and searching online platforms using spoken commands. The voice module was implemented using OpenAI's Whisper model, which is currently loaded locally for offline use to ensure faster processing.

This Automatic Speech Recognition pipeline is built using the transformers.pipeline API, and providing high-quality transcription of spoken input in real time. First, the audio input is recorded using the sounddevice library, then passed to the Whisper model pipeline, which transcribes the audio into text. The transcribed text is used for multiple purposes in the desktop applications. In Zoom, the user can gaze up to open the Zoom chat and then type a message using their voice, which is automatically typed into the chatbox. In the WhatsApp desktop application, users can search for a friend by speaking their name and then typing the message and sending. The ability to perform both command-based operations and content typing makes the voice model a powerful part of the gaze-based control system.

5.7 Challenges

During the development of the gaze tracking system for human-computer interaction, several technical and practical challenges were addressed across many stages of the project. One of the earliest challenges encountered in this project was collecting a suitable dataset for training the deep learning model. Publicly available datasets helped to provide a baseline of the model, but they were often limited in terms of lighting variation, gaze directions, and user diversity. To overcome this limitation, a custom dataset was collected using a webcam from different participants. However, this process introduced additional challenges. Participants had to be carefully guided to look in specific directions to capture the full facial images. Then it was necessary to crop the eye region to maintain model consistency.

Another major challenge in this project was achieving real-time performance while handling multiple computational tasks simultaneously. The system needed to continuously capture live webcam video frames, extract facial landmarks using MediaPipe, preprocess eye images, and run gaze classification through the CNN model. Another challenge the blink detection module had to distinguish between natural blinks and intentional ones. Since natural blinks happen frequently, the system implemented a time-based confirmation mechanism to detect the intentional blinks.

Another important challenge faced during the development was the variation in laptop screen sizes and viewing angles, which significantly affected the accuracy of gaze-based application control. Since users may use the system on different computers, the relative positioning of the eyes to the screen and gaze direction can vary widely. To overcome this issue, a calibration module was implemented using MediaPipe eye and iris landmarks. During calibration, users are guided to look at predefined points (left, right, up, and down) on their own computer screen, while the system captures iris and eye landmark offsets. These values are stored as gaze thresholds, allowing the system to adapt dynamically to each device and user.

It is not possible to type text on a computer using gaze detection only; this became a major limitation in making the system fully hands-free. To overcome this issue, we implemented a voice-to-text conversion model using OpenAI Whisper. This allows users to speak their message, which is then converted into text and typed automatically into applications like Zoom chat and WhatsApp Desktop. This voice input feature made the system more practical and accessible.

Chapter 6

Results

This chapter represents the results obtained from the model training, calibration and testing of gaze tracking system. The gaze prediction system built using the CNN architecture and calibrated using the facial landmarks to improve the gaze direction estimation. In the software application development phase, the simple interfaces were developed using the figma UI design and the built model was tested with the power point application. In addition to these, to improve the usability of this application the voice integration also added to open and close the apps and voice to text conversion also integrated to type any messages using this gaze tracking HCI based software application in the apps kind of zoom chat. Each step of the implementation process was evaluated in many ways which are qualitative and quantitative.

6.1 Results of CNN architectural gaze classifications

Datasets collected from various e-sources and gaze images obtained from the local people were re-labelled under the category of left, right, up, down and center and utilized for the CNN model training. In order to get the critical points in the gaze, the captured and collected images were converted into grey scale, cropped as left and right eye images separately and resized into 64x56 pixels and adopted during the training. After all the pre-processing and training of 15 epochs, the trained model was evaluated using the separate test dataset and achieved the accuracy level as shown in the Figure 6.1.

Based on the Figure 6.1, it is proved that the built in CNN model performed very well across all the gaze directions especially achieving the F1-scores more than 0.92. Highest was captured during the down and right directions which are 0.9802 and 0.9726 respectively. Slightly low performances were observed during the center which was confused with the left and right directions and it was more challenging to this 0.8871 precision and 09561 recall. Eventually all the high recall values in the all 5 directions highlight that the trained model misses the correct

```
Overall Accuracy: 0.9594

Detailed Evaluation Metrics:

   Label  Precision   Recall  F1-Score
  center   0.887078 0.956085  0.920290
    down   0.995277 0.965636  0.980233
    left   0.963384 0.953750  0.958543
   right   0.980813 0.964484  0.972580
      up   0.971357 0.955882  0.963558
Average   0.959582 0.959167  0.959041
```

Figure 6.1: CNN model evaluation metrics

gaze directions very rarely and high precision values indicates that false positives are very minimum, so that classification between the similar gaze directions is highly reliable through this model.

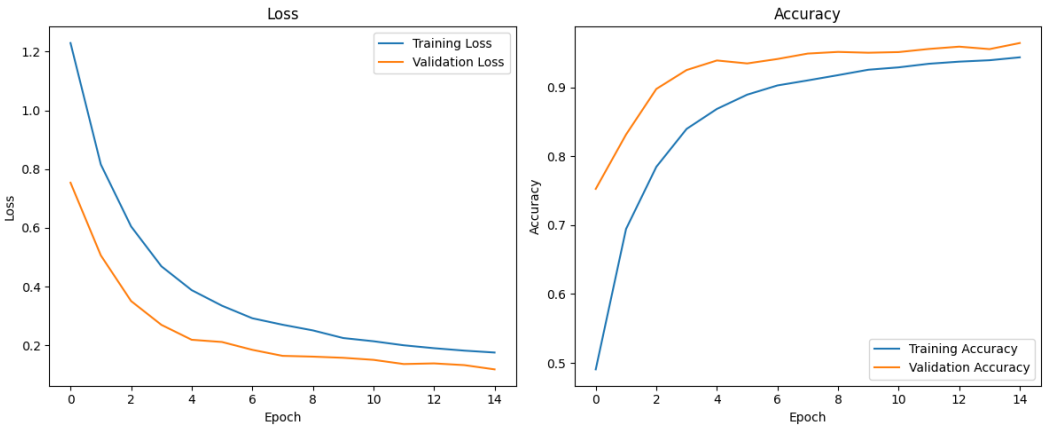


Figure 6.2: Loss and accuracy analysis plots

The Figure 6.2 illustrates the Loss and Accuracy analysis plots where the training loss seems gradually dropped from about 1.25 to about 0.18, demonstrating successful optimization over 15 epochs. There is no indication of overfitting and early generalization as the loss decreased more quickly before slowing down at about 0.10. The small gap between two curves indicate that there were strong regularization and good generalization to unknown data. The accuracy analysis plot shows that training accuracy had risen from about 50% to about 94.5% at the end of training. The validation loss had risen above 90% and peaked at about 96.5% by 4th epoch, demonstrating the model's exceptional consistency and generalization. This implies that the CNN model learned discriminative features that effectively generalized in addition to fitting the training data well.

The better understanding on the class-wise prediction behaviour can be obtained from the confusion matrix shown in Figure 6.3 for the final CNN model tested on the test dataset. The diagonal dominance in the matrix shows that the model correctly predicted the majority of labels and a small amount of confusion was noted between center and left/up, as would be predicted given how similar their eye images appear. At the same time, there were very few samples from any

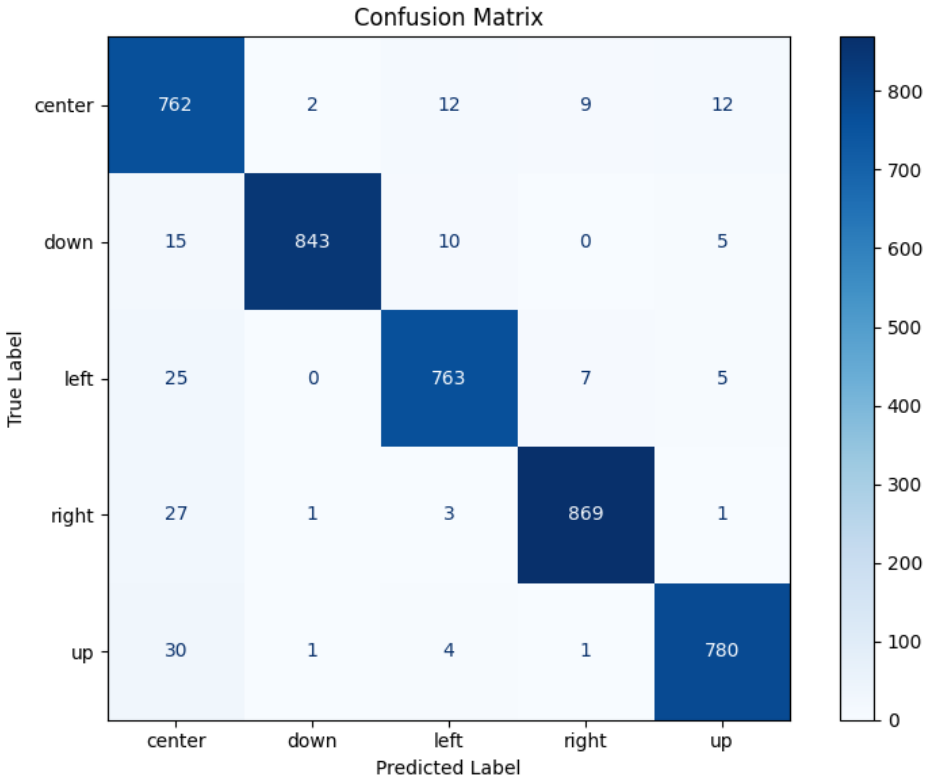
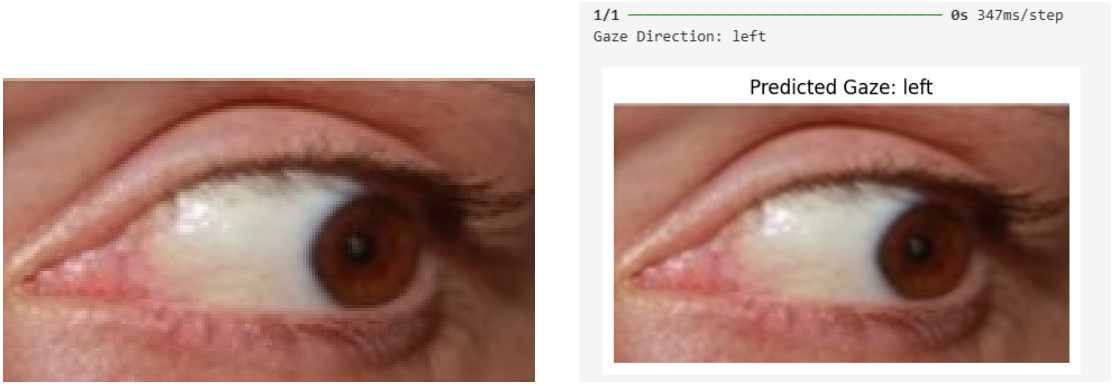


Figure 6.3: Confusion matrix of the trained CNN model

class that were incorrectly classified as belonging to the right class, suggesting that it was especially unique. The previously proven high recall and precision values were re-confirmed by the confusion matrix as well.

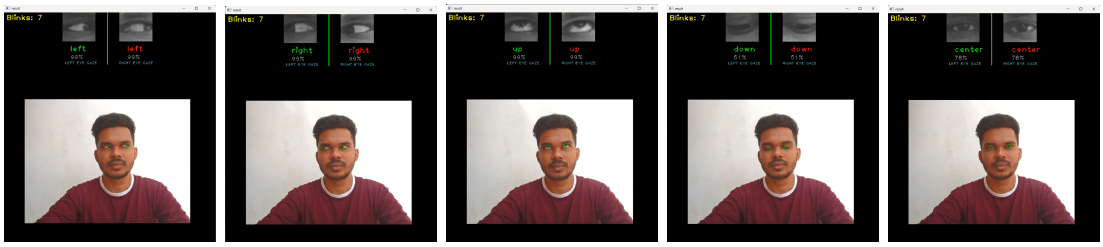
6.2 Real-time prediction results with the trained CNN model

After the training process the model was initially tested with some image input data as shown in the Figure 6.4. Then it was tested with real time video of the user using the laptop web camera as shown in Figure 6.5.



(a) Input image (b) Predicted output

Figure 6.4: Given input test image and the predicted output from the CNN model.



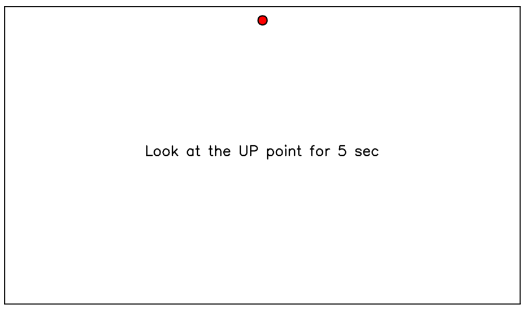
(a) Left (b) Right (c) Up (d) Down (e) Center

Figure 6.5: Model predictions for five different gaze directions: left, right, up, down, and center.

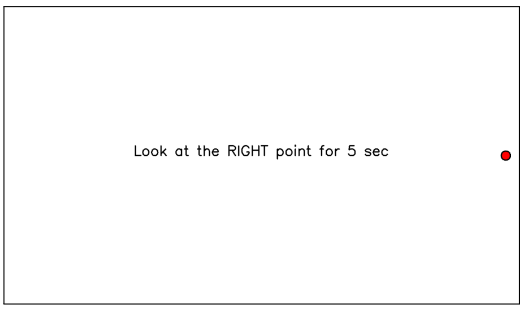
But out of the 5 directions, the more challenges were observed during the prediction if center direction. Because as the requirement of the application says, the center direction should be captured through all over the screen area and out of the screen area only the other 4 gaze directions should be predicted. But in the standalone gaze prediction CNN model, small gaze movements towards any direction, considered as that particular direction gaze. So to overcome this issue the iris landmark based calibration also was implemented after the model training.

6.3 Gaze calibration results using facial landmarks

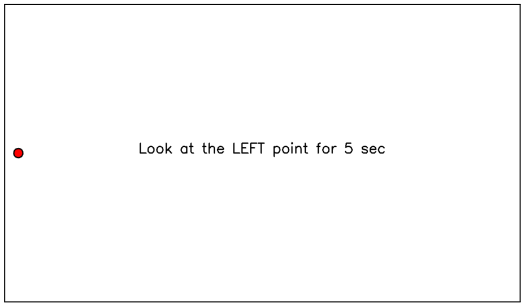
To enhance the prediction results robustness, the facial landmarks based gaze calibration system was developed using media-pipe python library. In this calibration process, at the beginning the user was guided through main four calibration target points on the screen which are left, right, up and down as shown in Figure 6.6.After the calibration points were measured, it was stored into the project for the gaze direction accurate prediction purpose as shown in the Figure 6.7.



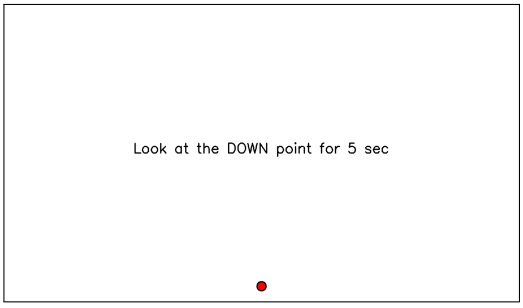
(a) Up



(b) Right



(c) Left



(d) Down

Figure 6.6: Gaze calibration prompts instructing the user to look in four directions - up, right, left, and down. Each image includes a red dot indicating the fixation point.

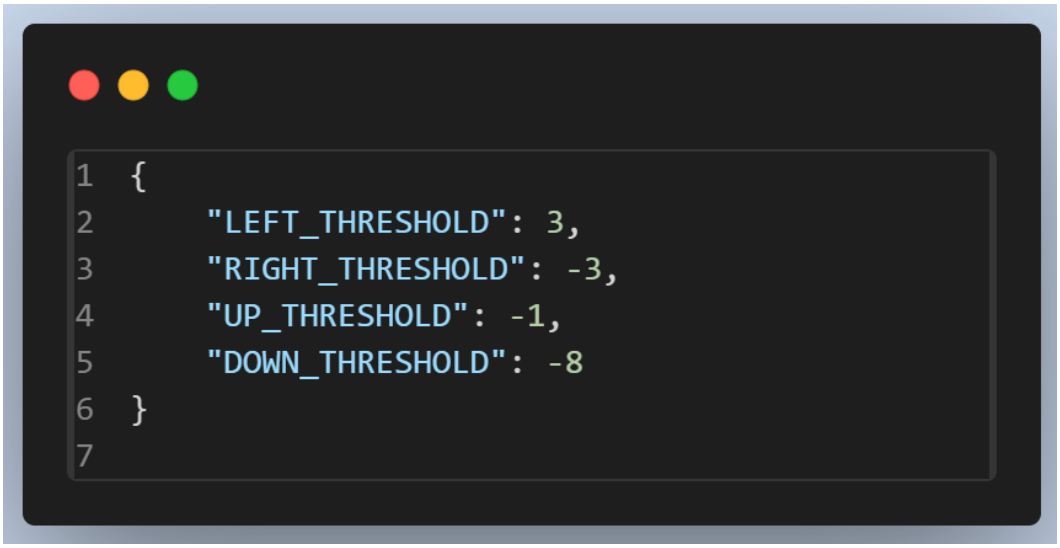


Figure 6.7: Stored calibration threshold values from a user's Dell laptop screen

The threshold values got from the user's screen is then processed and compared with the predicted gaze direction from the CNN trained model and used for accurately predict the gaze directions as described earlier in previous chapters.

6.4 Hybrid Gaze Direction Prediction : CNN + Landmarks offset

Both the CNN and facial landmarks offset were used together to confirm the gaze direction. To find the direction using the landmarks, the stored threshold values were processed and matched with the computed offsets and if that computed offset is equal or less than that stored threshold value, it resulted the gaze direction in that particular direction. Then the predicted gaze direction using CNN model confirmed with the landmark offset result to represent the strong gaze direction as shown in Figure 6.8, 6.9, 6.10, 6.11 and 6.12.

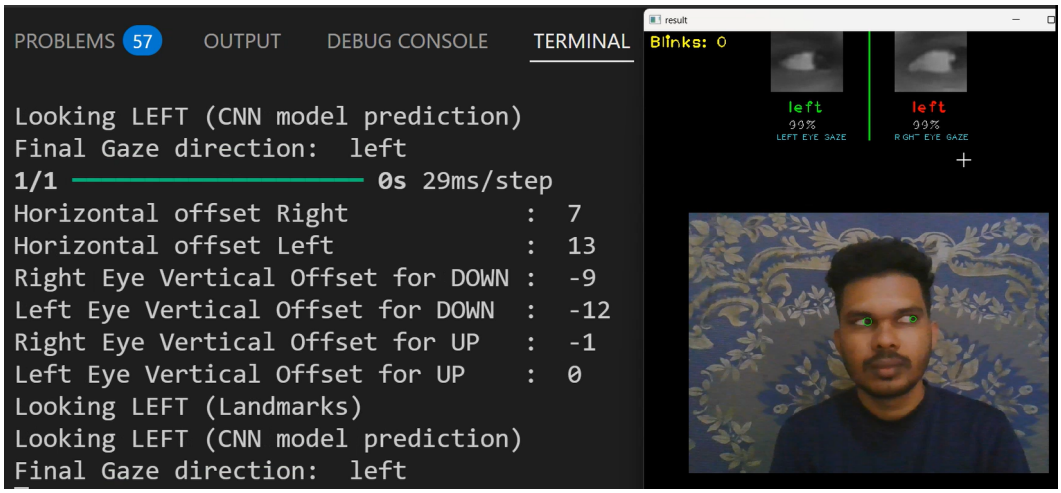


Figure 6.8: Calculated offset in each direction while the user looking in the direction of left

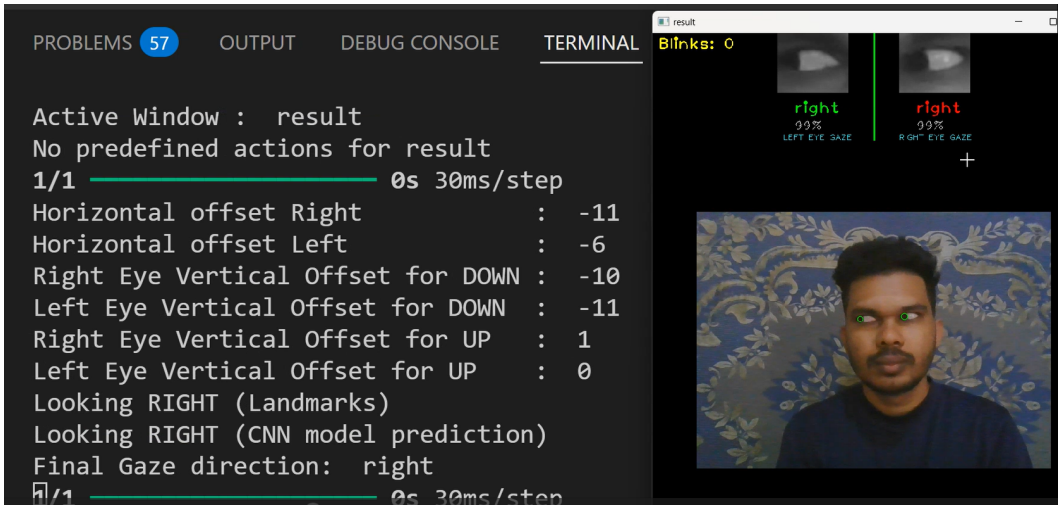


Figure 6.9: Calculated offset in each direction while the user looking in the direction of right

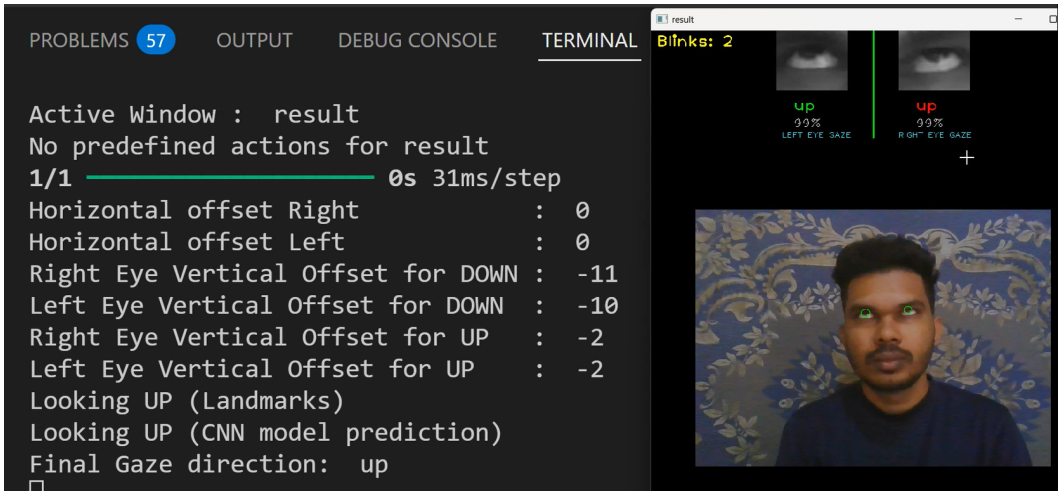


Figure 6.10: Calculated offset in each direction while the user looking in the direction of up

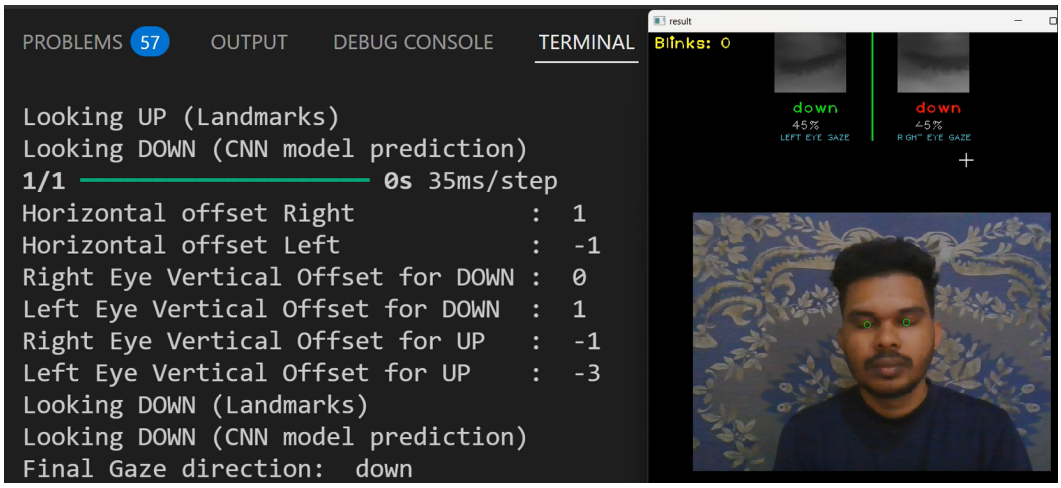


Figure 6.11: Calculated offset in each direction while the user looking in the direction of down

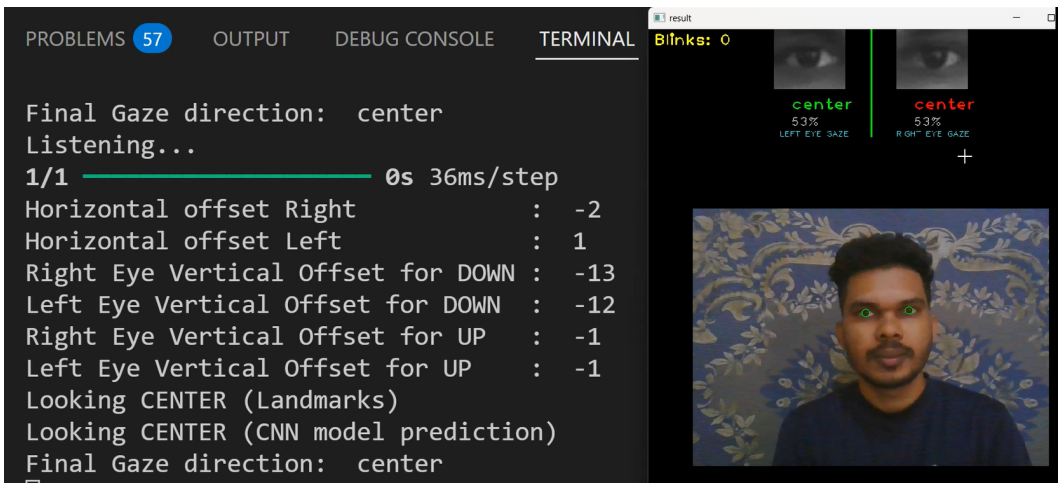


Figure 6.12: Calculated offset in each direction while the user looking in the direction of center

In addition to these gaze direction predictions, blink detection also implemented using Eye Aspect Ratio (EAR) computed from MediaPipe landmarks. These blinks were detected based on the condition that is EAR falling below 0.22 for greater than 2 seconds and confirmed by the audio alert using winsound.Beep. A running total blinks detected was displayed in the interface as seen in the Figure 6.12.

6.5 Real-time app controlling with predicted gaze directions and voice commands

The calibrated gaze prediction model was integrated with multiple applications and controlled the feature actions using those gaze directions. As the software application controlling framework, it was utilized the voice commands also to open the apps like zoom, power-point, you tube and media player and then controlled with gaze directions and speech to text recognition was also included to search the videos in you tube as shown in Figure 6.13.

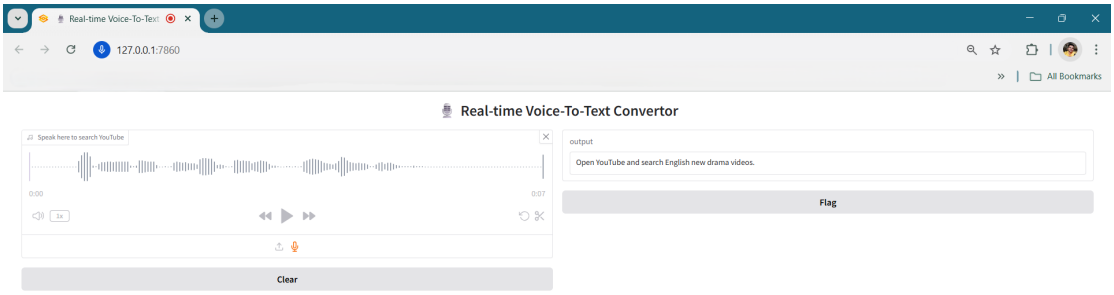


Figure 6.13: Real-time Voice-to-Text Converter Interface Using Gradio and Whisper

In addition to that for the usability testing purpose, the gaze directions and blink actions were mapped with some frequent applications and triggered their main functionalities as listed down in the Table 6.1. This will be considered as the basement for next phases of software development and integration with the centralized remote control based control panel.

Table 6.1: Gaze-Based Control Mappings Across Applications

Application	Gaze-Based Actions
PowerPoint	*Home Screen:* Move selection up/down (Up/Down Gaze), switch file focus (Left/Right Gaze), open selected file (Blink). *Inside Presentation:* Previous/next slide (Left/Right Gaze), start slideshow (Up Gaze), exit slideshow (Down Gaze), close PowerPoint (Blink).
Zoom	Mute/unmute microphone (Left Gaze), start/stop video (Right Gaze), open chat and type message using Whisper (Up Gaze), leave meeting (Down Gaze).
PDF Viewer	Scroll up (Left Gaze), scroll down (Right Gaze), zoom in (Up Gaze), zoom out (Down Gaze), close PDF viewer (Blink).
Media Player	Navigate libraries using gaze: up/down (Up/Down Gaze), left/right (Left/Right Gaze), play/pause (Up Gaze), open selected item (Blink), access music/video library from home screen (Left/Right Gaze).
Snake Game	Move left/right/up/down (Left/Right/Up/Down Gaze), and start or pause the game (Blink).

Chapter 7

Conclusion and Future work

7.1 Conclusion

At this stage of the project, the combinational frame work of deep learning and facial landmark-based methods for hands-free human-computer interaction has been developed which proves the significant progress towards a real time gaze tracking system using HCI. The developed CNN model has been trained not only using publicly available datasets but also custom datasets collected from different age group local people, to improve the accuracy of the model by increasing the variation of datasets. Moreover, through the augmentation process, the dataset is enhanced further to accurately detect the five gaze directions: left, right, up, down, and center.

In the model training and evaluation, there were significant confusions between the center, left and right directions. So to avoid this type of confusions, the personalized gaze calibration system has been developed utilizing the facial landmarks to enhance the robustness of the gaze direction predictions across different laptop screens and positions. Even though the model has been trained across different lightning conditions, as this model is developed with the intent to aid users who have physical disabilities, who will still need to ensure that their environment has sufficient lightning in order to capture the gaze movements through their web cameras. So based on these conditions, a framework model of real-time gaze prediction has been developed and integrated with the webcam input with the CNN inferences. In addition to these features, it has been extended to detect the eye blinks using the Eye Aspect Ratio (EAR) with the predicted gaze directions.

Based on the developed gaze tracking model, it has been tested with the real-world applications such as PowerPoint, Adobe PDF Reader, Zoom and Media Player to confirm their capabilities to handle all the main features in that applications. Although those target applications have many features, the currently developed framework has only 5 directions and eye blinks to control the functions. To over come these limitations, a remote control based user interface design also planned to improve the control flexibility and system's usability by using the limited gaze directions. Furthermore, to open the apps and typing purposes in the

applications voice commands also integrated with the developed framework which confirming its potential to support intuitive, hands-free control.However, several components remain in progress to complete the software application with usability and reliability. Likewise, there are more integration processes also still under development.

7.2 Future Work

While the current gaze prediction framework shows the promising results, several areas need the development and integration process which are listed below.

1. **Voice Integration**
   Although the basic commands have been explored up to now, the full voice control module should be developed with command parsing and application mapping. Because even though through the gaze directions, the application functionalities will be fully controlled, some of the text based functions can not be achieved using only through the gaze directions (e.g: Zoom chat, google search, you tube video search etc). So in order to enable more flexible hands-free interaction with the applications, this voice integration module takes pivotal role in this project.

2. **Software application development**
   Based on the designed mock-ups of software interfaces as illustrated in Figure 7.1, a dedicated remote control based interface will be developed to provide a centralized control panel for all the gaze and voice-triggered actions in desktop application. This software application will act as a bridge between the user especially with the disabilities and system level application control.

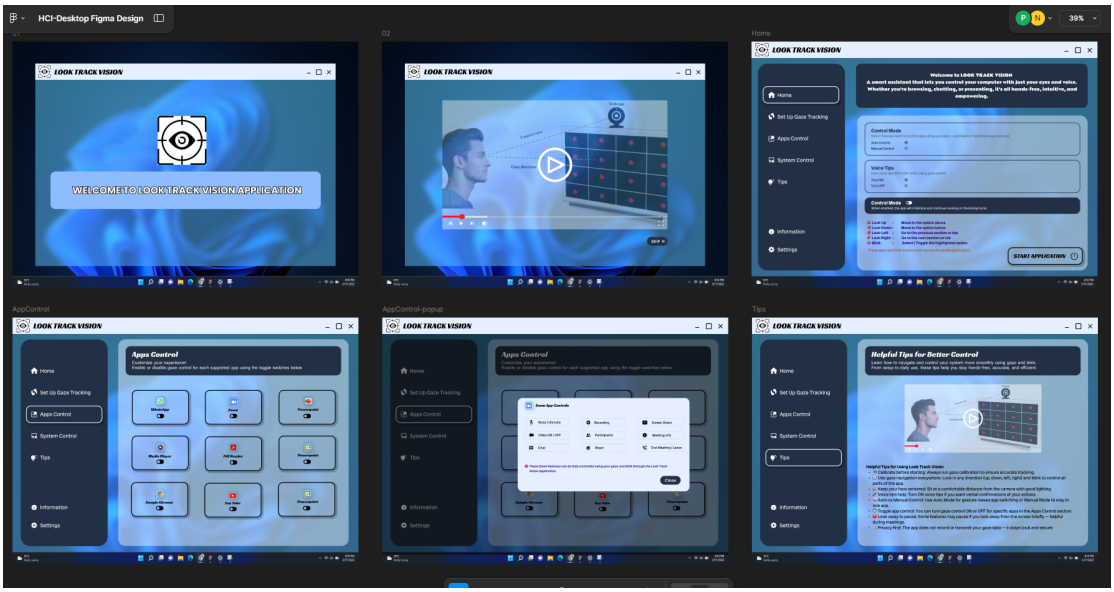


Figure 7.1: Figma mock-ups of some UI designs for the desktop application

3. **Integration of calibrated framework with application interface**
   The current calibrated framework needs to be integrated seamlessly with the customized software application. This will ensure the precision and responsiveness by directly trigger the specific UI elements (e.g: mute, unmute, chat) using the directional gaze and voice commands.

4. **Multi-Directional and context-aware interaction**
   Future interactions will expand the control options beyond those five gaze directions and blinking, through the planned centralized remote control based control panel. This includes the proper guidelines to the special need users through the software application to support more complex interactions and avoid misfiring the action.

5. **Usability and accessibility testing**
   Users with physical disabilities will take part in an extensive usability assessment to confirm the system's comfort, responsiveness, and accessibility after development and integration. Refinements to command mappings, calibration processes, and user interface layout will be guided by input from actual use cases.

6. **System optimization and development**
   The final setup will work smoothly with low-end hardware and be compatible with common operating systems and webcams. The deployment techniques like packaging into a desktop application (e.g., via PyInstaller or Electron) also will be looked up in coming phases.

References

[1] N. Modi and J. Singh, "Real-time camera-based eye gaze tracking using convolutional neural network: A case study on social media website," *Virtual Reality*, vol. 26, pp. 1489–1506, Dec 2022.

[2] V. Vidhya and D. R. Faria, "Real-time gaze estimation using webcam-based cnn models for human–computer interactions," *Computers*, vol. 14, no. 2, p. 57, 2025.

[3] R. Kumar *et al.*, "Human-computer interaction," *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, no. 2, 2021.

[4] H. A. Kheder, "Human–computer interaction: Enhancing user experience in interactive systems," *Kufa Journal of Engineering*, vol. 14, pp. 23–41, Oct 2023.

[5] X. Zhang, Y. Sugano, and A. Bulling, "Evaluation of appearance-based methods and implications for gaze-based applications," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, (New York, NY, USA), p. 1–13, Association for Computing Machinery, 2019.

[6] T. Fischer, H. Chang, and Y. Demiris, "RT-GENE: Real-Time Eye Gaze Estimation in Natural Environments," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), Lecture Notes in Computer Science, pp. 339–357, Springer, Oct. 2018. The European Conference on Computer Vision (ECCV), 08–14 Sept. 2018.

[7] D. Cazzato, M. Leo, C. Distante, and H. Voos, "When i look into your eyes: A survey on computer vision contributions for human gaze estimation and tracking," *Sensors*, vol. 20, no. 13, 2020.

[8] A. Haria, A. Subramanian, N. Asokkumar, S. Poddar, and J. S. Nayak, "Hand gesture recognition for human computer interaction," in *Procedia Computer Science*, vol. 115, pp. 367–374, Elsevier, 2017. 7th International Conference on Advances in Computing Communications (ICACC-2017), 22–24 August 2017, Cochin, India.

[9] Jyoti and G. Kaur, "Research paper on human computer interaction (hci)," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 5, pp. 1–9, Mar–Apr 2023.

[10] A. Poole and L. J. Ball, "Eye tracking in human-computer interaction and usability research: Current status and future prospects," in *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research* (C. Ghaoui, ed.), pp. 211–219, Hershey, PA, USA: Idea Group Inc./IGI Global, 2006.

[11] J. J. MacInnes, S. Iqbal, J. Pearson, and E. N. Johnson, "Wearable eye-tracking for research: Automated dynamic gaze mapping and accuracy/precision comparisons across devices," *bioRxiv*, 2018.

[12] S. J. Chang, J. Lee, S. Kim, S. Lee, and D. W. Lee, "Simple wearable eye tracker with mini-infrared point sensors," *IEEE Access*, vol. 12, pp. 41019–41026, 2024.

[13] W. Kurdthongmee, P. Kurdthongmee, K. Suwannarat, and J. Kiplagat, "A yolo detector providing fast and accurate pupil center estimation using regions surrounding a pupil," *Emerging Science Journal*, vol. 6, pp. 985–997, Oct 2022.

[14] Z. Zhu and Q. Ji, "Eye gaze tracking under natural head movements," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 918–923, IEEE Computer Society, 2005.

[15] M. A. Mohamed, A. I. Abdel-Fatah, and B. M. El-Den, "Fighting accident using eye detection for smartphones," *International Journal of Engineering Research and Applications (IJERA)*, vol. 4, pp. 67–73, Aug 2014.

[16] I. S. S. Shehu, Y. Wang, A. M. Athuman, and X. Fu, "Paradigm shift in remote eye gaze tracking research: Highlights on past and recent progress," *Electronics*, vol. 10, no. 24, p. 3165, 2021.

[17] H. Deng and W. Zhu, "Monocular free-head 3d gaze tracking with deep learning and geometry constraints," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 341–350, 2017.

[18] K. Louedec *et al.*, "Gaze tracking in gaming: A saliency map study for chess players," *Journal of Game Analytics*, 2021.

[19] A. Figueroa *et al.*, "Gaze estimation using consumer-grade webcams," in *International Conference on Human Factors*, 2020.

[20] M. Kanade *et al.*, "DINN: A Deep Integrated Neural Network for Driver Fatigue Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 580–590, 2019.

[21] M. F. Ansari, P. Kasprowski, and M. Obetkal, "Gaze tracking using an unmodified web camera and convolutional neural network," *Applied Sciences*, vol. 11, no. 19, p. 9068, 2021.

[22] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 511–518, IEEE, 2001.

[23] K. Hasegawa *et al.*, "Event-based human gaze tracking with blink detection," *Sensors*, vol. 21, no. 11, pp. 3642–3657, 2021.

[24] A. Khaleel *et al.*, "Expanding gaze-based interfaces to mobile applications," *Applied Artificial Intelligence*, vol. 36, no. 2, pp. 141–155, 2022.

[25] M. Modi *et al.*, "Gaze tracking for marketing: A cnn case study on social media," *Virtual Reality*, 202x. To be completed: volume, issue, pages, and year.

[26] A. H. Khaleel, T. H. Abbas, and A.-W. S. Ibrahim, "Best low-cost methods for real-time detection of the eye and gaze tracking," *i-com*, vol. 23, no. 1, pp. 79–94, 2024.

[27] Aiphile, "Iris segmentation: Mediapipe + python." https://medium.com/@aiphile/iris-segmentation-mediapipe-python-a4deb711aae3, 2023. Accessed: 2025-07-28.

[28] M. Data, "Gaze direction dataset." https://data.mendeley.com/datasets/vy4n28334m/1, 2023. Accessed: 2025-07-29.

[29] H. O. Bhatt, "Sbvpi sclera dataset." https://www.kaggle.com/datasets/hariohmbhatt/sbvpi-sclera, 2021. Accessed: 2025-07-29.

[30] K. Shah, "Eye dataset." https://www.kaggle.com/datasets/kayvanshah/eye-dataset, 2021. Accessed: 2025-07-29.

[31] A. Shingha, "Mrl eye dataset." https://www.kaggle.com/datasets/akashshingha850/mrl-eye-dataset, 2022. Accessed: 2025-07-29.