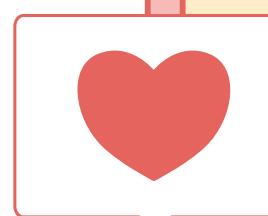
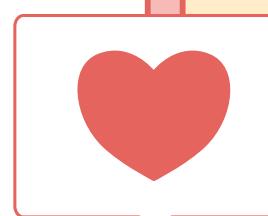


Prepared by group 5

AI PROJECT

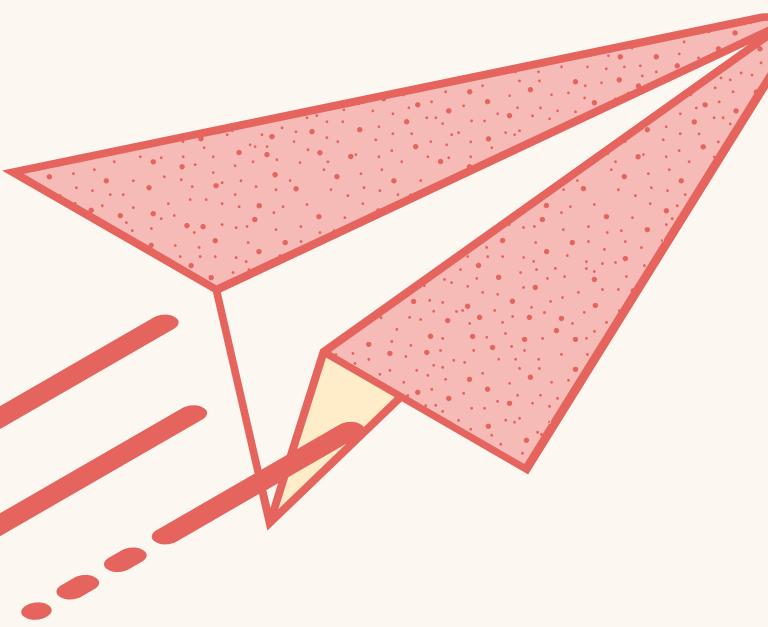
Thiết kế ứng dụng hỗ trợ người bệnh ALS



NHÓM 5



STT	THÀNH VIÊN	CÔNG VIỆC
1	PHẠM HUỲNH BẢO TRÂN	Leader, Backend AI, Project Manager
2	NGUYỄN THÙY AN	Frontend AI, Design, Research Assistant
3	HOÀNG THIÊN THƯ	Backend AI, Design, Research Assistant)
4	NGUYỄN KHÁNH LINH	Frontend AI, Design, Research Assistant
5	NGUYỄN THỊ MỸ DUYÊN	Backend AI, Design, Research Assistant)



Mục lục

01. Giới thiệu

02. Tổng quan
nghiên cứu

03. Phương
pháp đề xuất

04. Kết quả
thí nghiệm

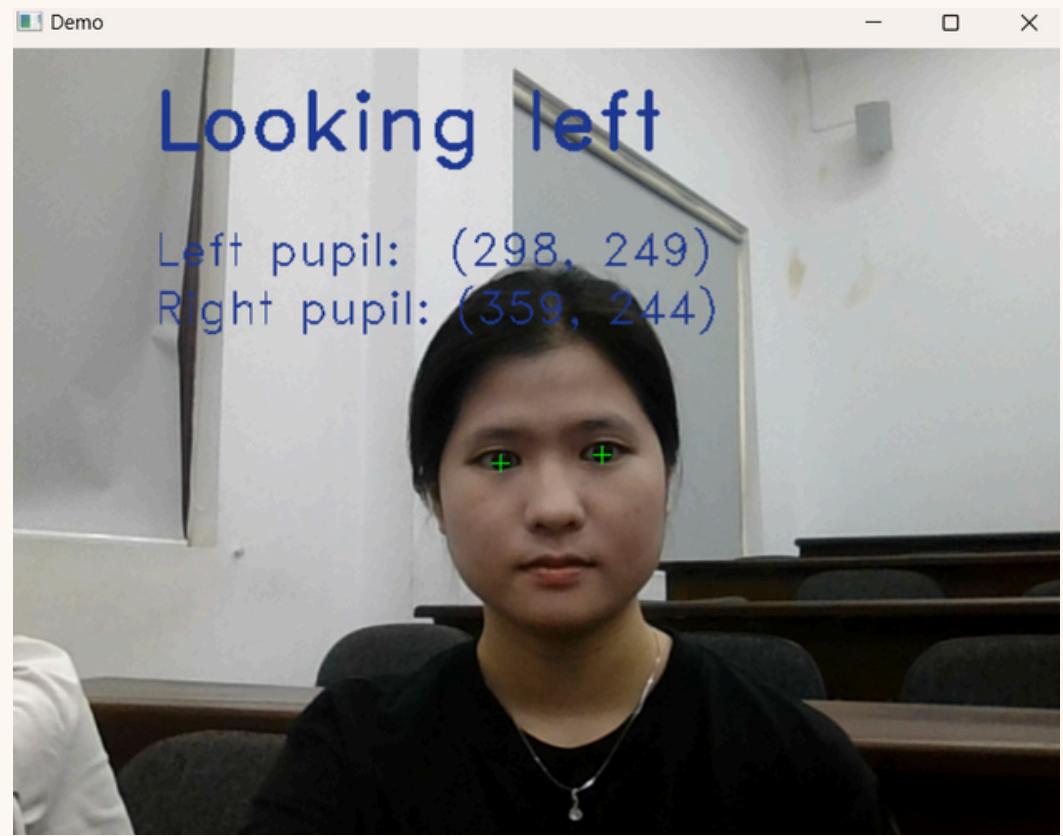
05.
Testing
demo

06. Kết luận



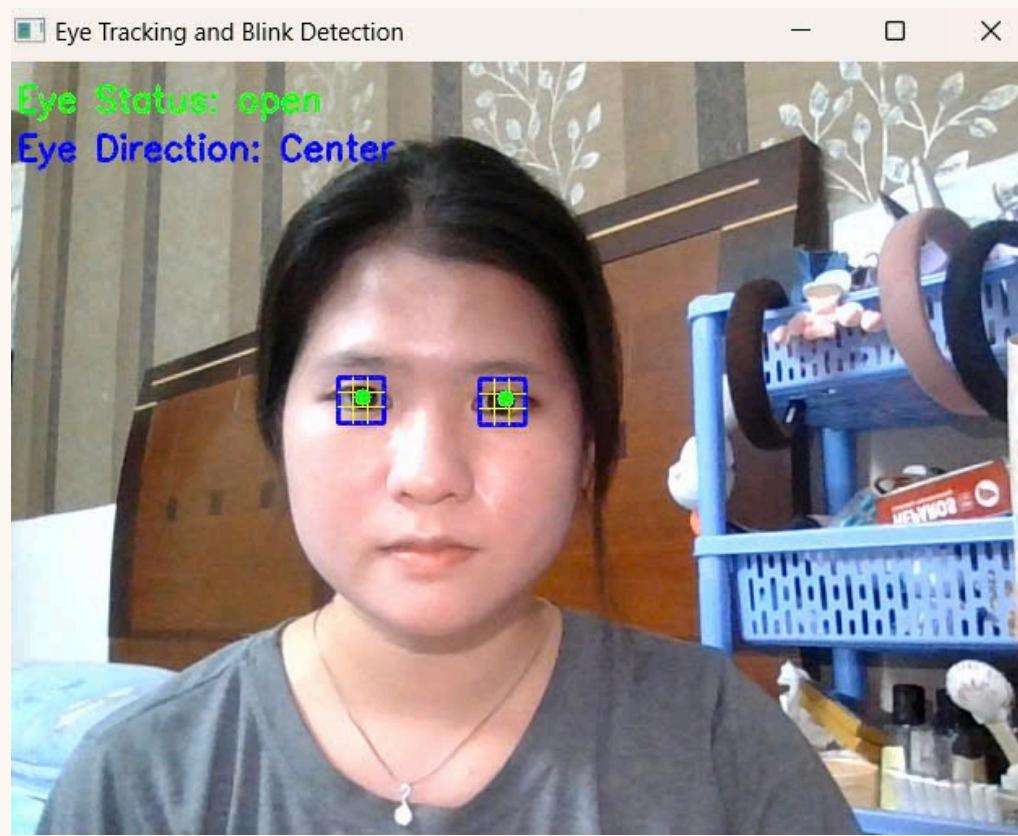
[Quay lại](#) [Trang Mục lục](#)

TẠO RA NHẬN DẶNG MẮT (TRÊN, DƯỚI, TRÁI, PHẢI, GIỮA)

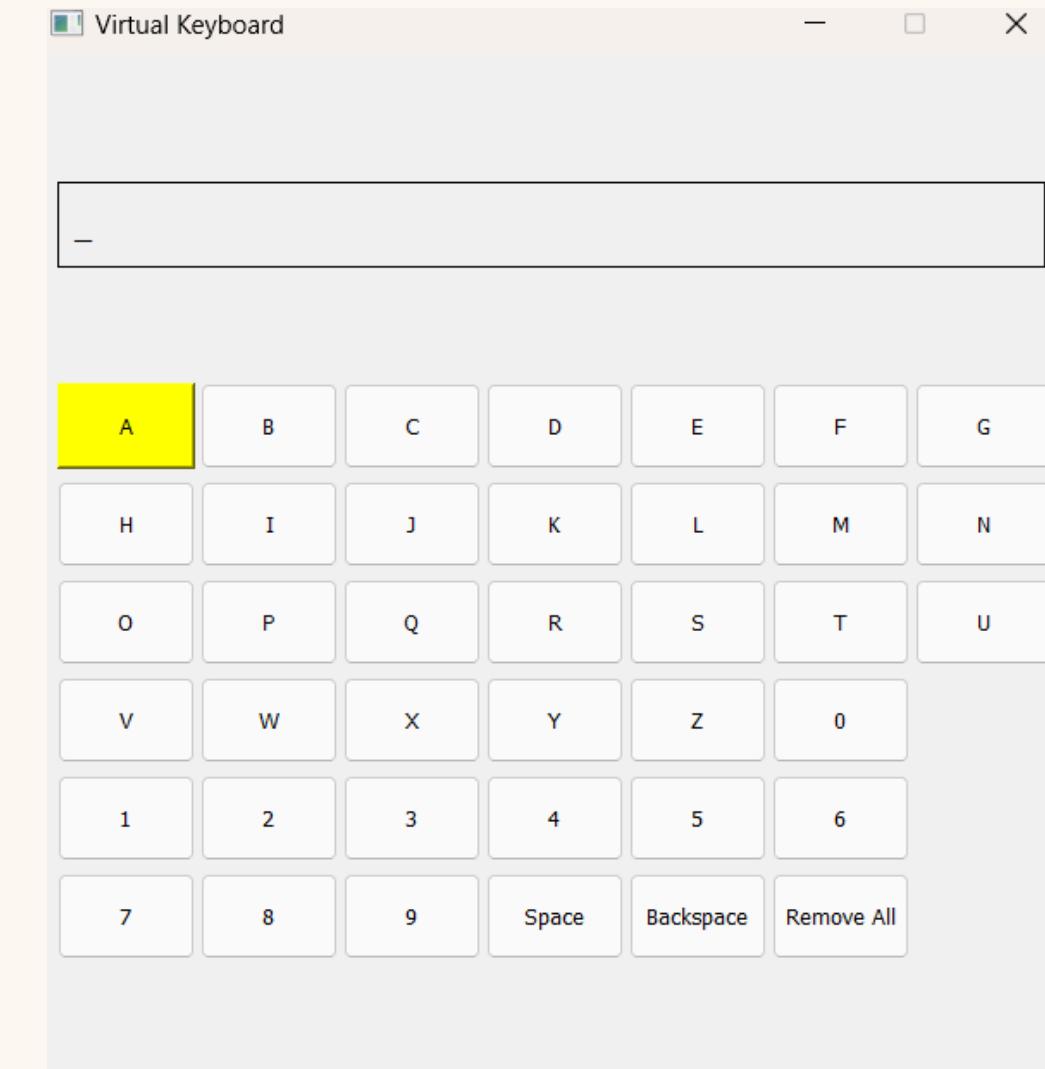


UP LÊN WEB

ĐỊNH HƯỚNG MẮT RÕ HƠN NHỜ Ô VUÔNG



TẠO BÀN PHÍM _ MÀU KHI CHỌN



To address this issue, the team decided to design the interface as two separate windows, allowing flexibility in adjusting the position of components to meet individual user needs. This approach not only improves the accuracy of gaze direction detection but also maximizes comfort for patients, better meeting their communication and device control needs.

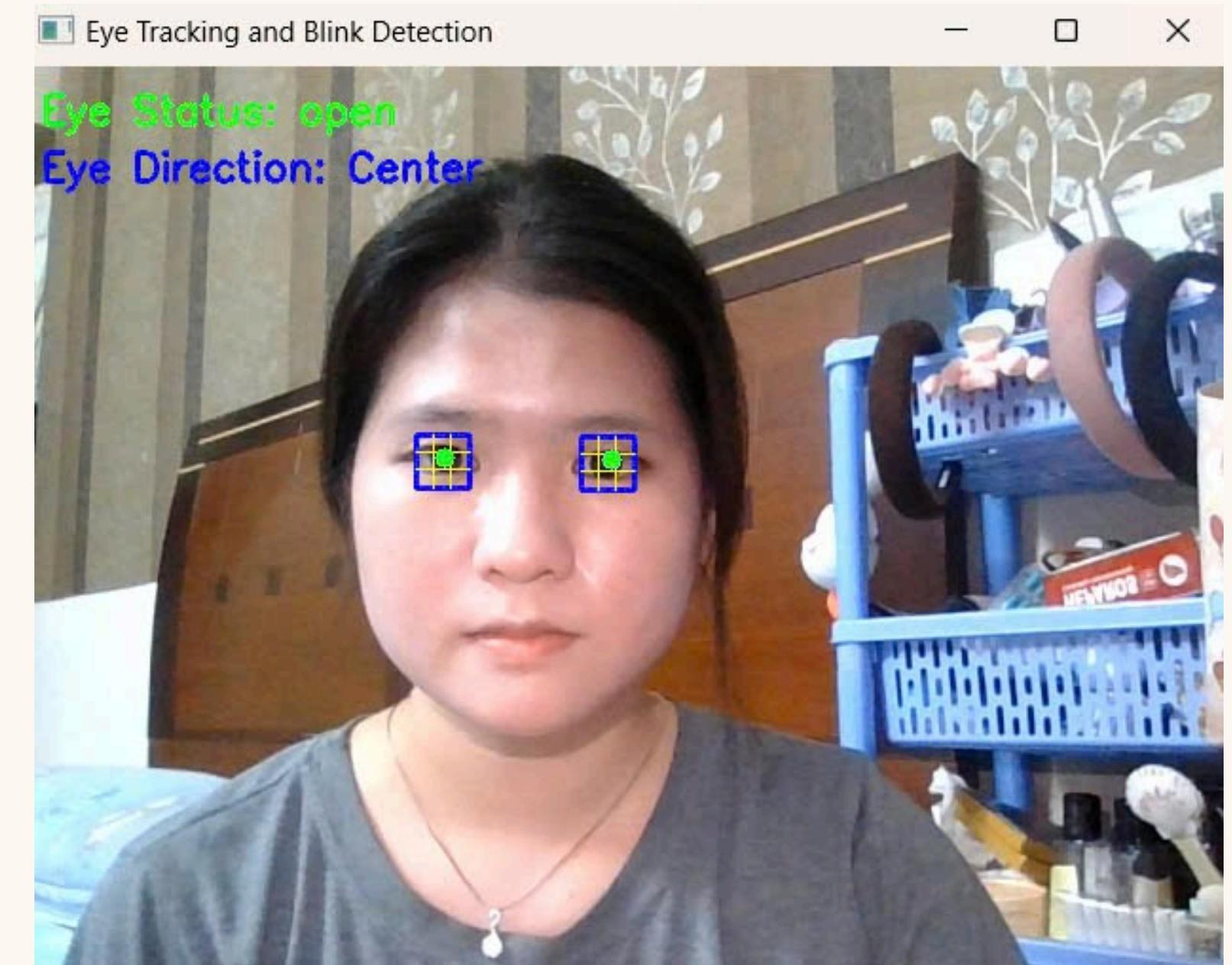
4.3.1 Virtual Keyboard Interface Design

The virtual keyboard interface in the application is designed using PvOtS components

Thầy đề xuất



Face ID



Ô vuông theo dõi trạng thái mắt

LÝ DO CHỌN ĐỀ TÀI

Cảm Hứng từ Stephen Hawking Cuộc Đời Phi Thường

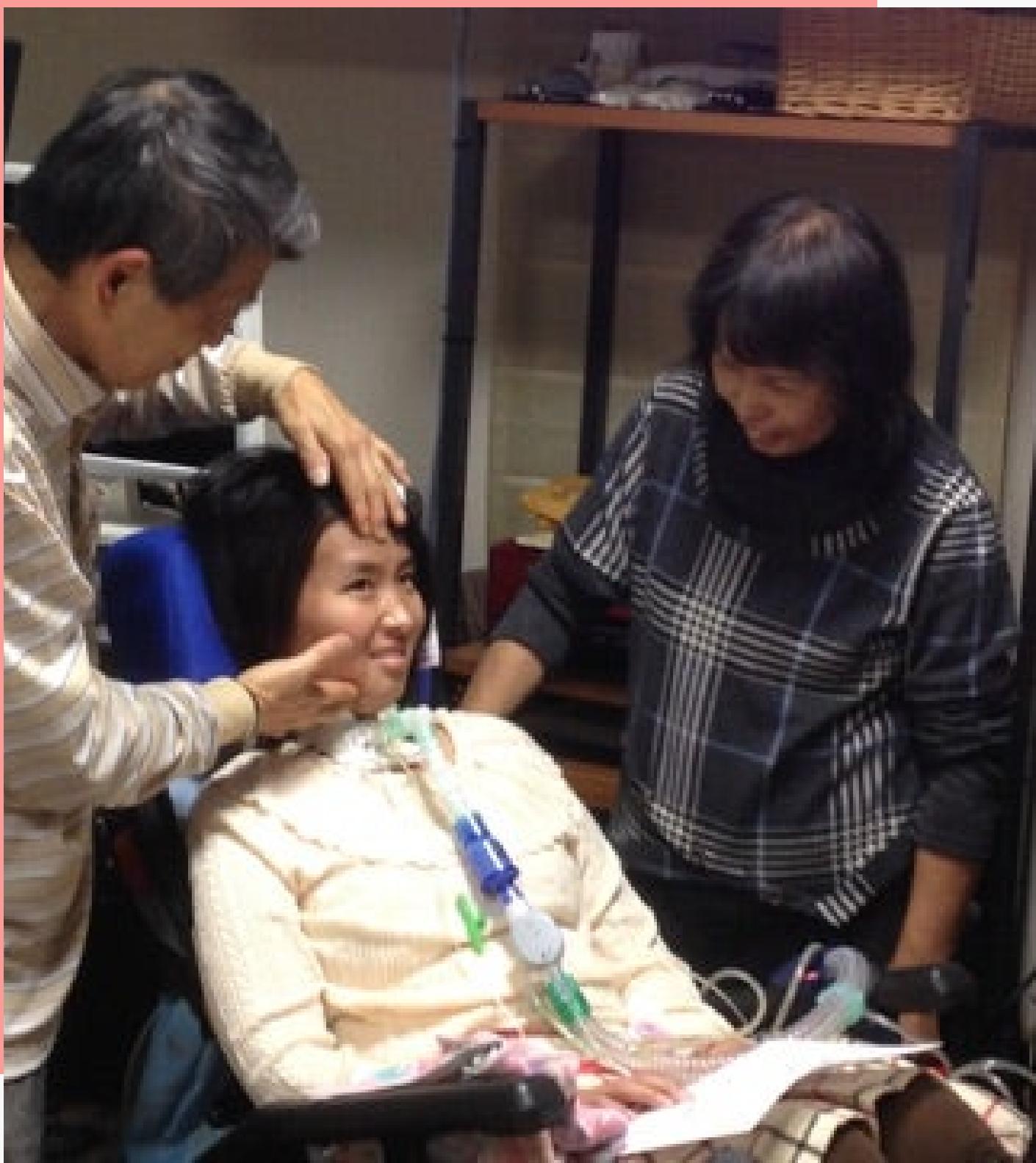
Được chẩn đoán mắc ALS ở tuổi 21, bác sĩ dự đoán chỉ còn 2-3 năm sống. Chiến đấu và đạt được những thành tựu vĩ đại trong lĩnh vực vật lý trong suốt 55 năm. Truyền tải kiến thức về vũ trụ, trở thành biểu tượng của nghị lực.

AI đóng vai trò ngày càng quan trọng trong Y Tế

Công nghệ AI đang cải thiện chẩn đoán và điều trị trong y tế. Nó giúp phát hiện sớm các bệnh lý như ung thư và rối loạn thần kinh qua phân tích hình ảnh, cung cấp chẩn đoán nhanh chóng từ dữ liệu y tế, và phát triển phương pháp điều trị cá nhân hóa. AI cũng hỗ trợ chăm sóc từ xa, giúp bệnh nhân dễ dàng theo dõi sức khỏe.



..... Giới thiệu về ALS



- ALS:
 - Là bệnh thần kinh hiếm gặp, gây tổn thương tăng dần tế bào thần kinh vận động ở não và khung sống dẫn đến liệt toàn thân và mất khả năng giao tiếp.
- Tác động đến cuộc sống:
 - Phụ thuộc vào gia đình và nhân viên chăm sóc sức khỏe
 - Mất khả năng giao tiếp bằng lời nói khiến việc kết nối với gia đình và xã hội trở nên khó khăn.
 - Cần giải pháp công nghệ để giúp bệnh nhân duy trì kết nối với người thân và xã hội.

Để giúp bệnh nhân duy trì sự kết nối với người thân và cộng đồng, cần phát triển các giải pháp công nghệ hỗ trợ, như công nghệ theo dõi mắt, nhằm phục hồi khả năng giao tiếp và cải thiện chất lượng cuộc sống của họ.

Phạm vi nghiên cứu



Đối tượng Nghiên cứu:

Hỗ trợ người bệnh ALS, đặc biệt những người mất khả năng cử động và giao tiếp, giúp duy trì kết nối xã hội.

Công nghệ Nhận diện Chuyển động Mắt:

Sử dụng Python và các thư viện như OpenCV, dlib, MediaPipe, và numpy để nhận diện chuyển động và trạng thái nhắm/mở mắt, điều khiển con trỏ trên bàn phím.



Hệ thống Vận hành:

Ứng dụng phát triển và kiểm thử trên Windows, không bao gồm khả năng tương thích với MacOS hoặc Linux.

Thiết kế Giao diện với PyQt5:

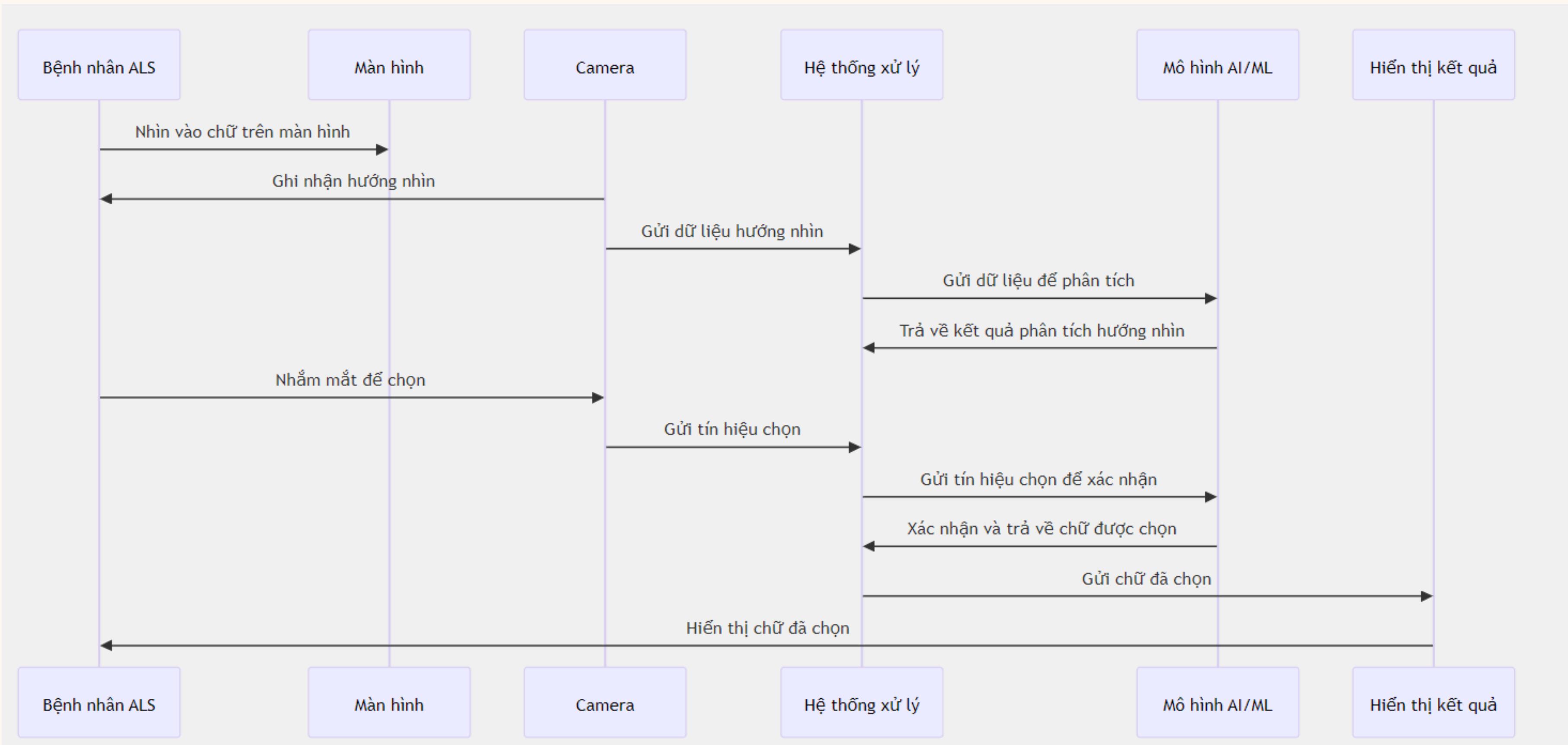
Giao diện đơn giản, thân thiện, tối ưu hóa cho người bệnh ALS, giúp dễ sử dụng mà không cần thao tác phức tạp.

Thử nghiệm và Đánh giá:

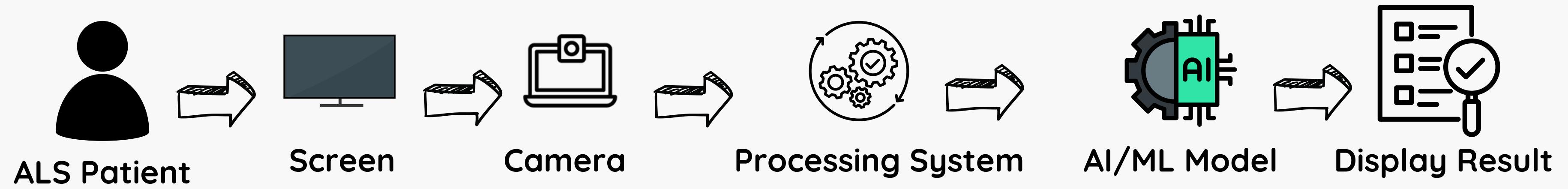
Thực hiện thử nghiệm trên người bệnh hoặc mô phỏng để kiểm tra độ chính xác của nhận diện chuyển động mắt và tính tiện lợi của giao diện.



Sequence Diagram



Sequence Diagram

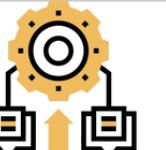


Mục tiêu dự án



- Phát triển ứng dụng theo dõi chuyển động mắt hỗ trợ bệnh nhân ALS trong việc giao tiếp và điều khiển.
- Người dùng tương tác thông qua theo dõi ánh mắt, tăng cường khả năng tiếp cận cho những người bị suy giảm vận động.

Model AI Training



- Machine Learning Models: Sử dụng CNN, SVM, và KNN để huấn luyện nhận diện hướng nhìn và trạng thái mắt.
- Model : dataset để train Kaggle và các tập train sẵn shape_predictor_68_face_landmarks,...
- Đánh giá: Sử dụng Mean Error, Standard Deviation, FPS để đo độ chính xác và hiệu suất.

Front-End



- PyQt5: Dùng để xây dựng giao diện người dùng, tạo bàn phím ảo và các nút chức năng, cho phép người dùng điều khiển qua ánh mắt.

Back-End



- Dlib: Xác định các điểm đặc trưng trên khuôn mặt và mắt.
- OpenCV: Phát hiện khuôn mặt, mắt, và xử lý hình ảnh theo thời gian thực.
- MediaPipe: Hỗ trợ phát hiện và theo dõi mắt chính xác trong thời gian thực.
- NumPy: Xử lý dữ liệu và thực hiện các tính toán như Eye Aspect Ratio (EAR).



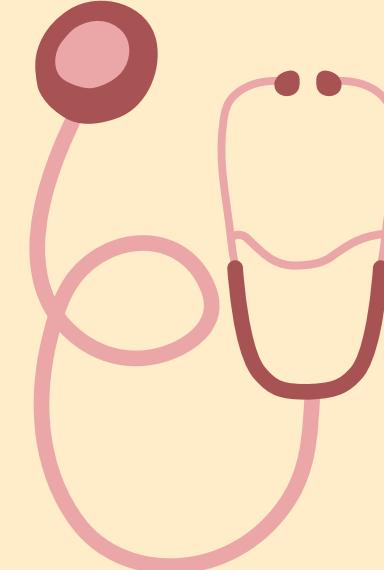
Hạn chế

- Kiến thức hạn chế về lập trình.
- Độ chính xác và phản hồi nhanh của hệ thống theo dõi ánh mắt trong môi trường thực tế.
- Mất thời gian để thử nghiệm tập train và test



Kết quả mong đợi

- Xây dựng hệ thống có khả năng nhận diện mắt và cho phép người dùng gõ chữ bằng cách di chuyển mắt đến các ký tự ở bàn phím trên màn hình.
- Giao diện đơn giản



Ý nghĩa

- Cải thiện khả năng giao tiếp và tự chủ cho người bệnh.
- Tăng cường sự nhận thức về ALS trong xã hội.
- Đóng góp cho sức khỏe cộng đồng và phát triển công nghệ hỗ trợ.



2. Related work/Literature review

2.1 Giới thiệu về Công Nghệ Theo Dõi Mắt



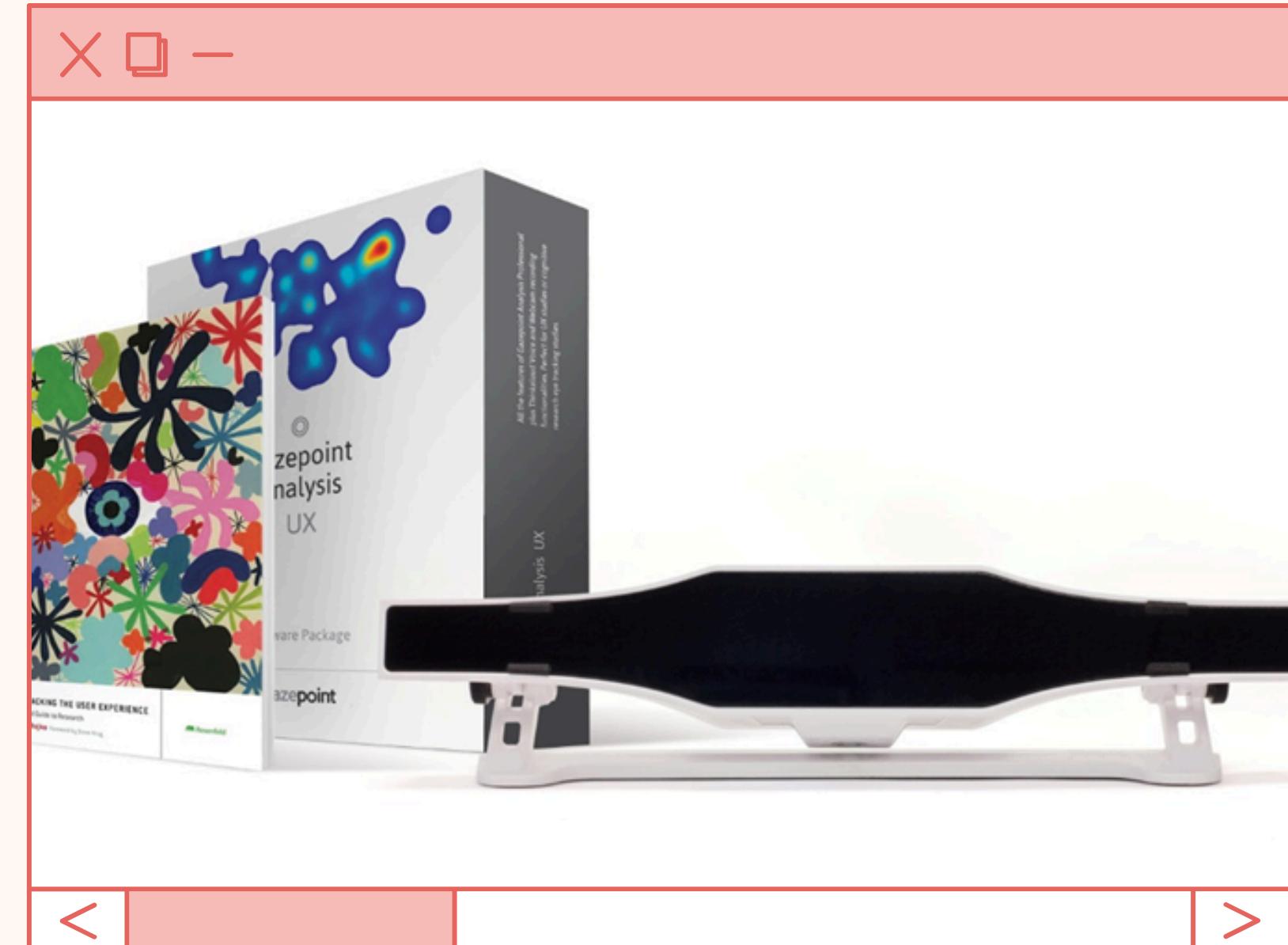
Công nghệ theo dõi điểm nhìn và chuyển động mắt

Bảng So Sánh Các Hệ Thống Theo Dõi Mắt

Technology	AI Model	Data Type	Evaluation Metrics	Summary of Results	Limitations
Tobii Eye-Tracker	CNN, SVM	ALS patients	Pixel deviation, Stability	High accuracy, reliable but sensitive to head movement	High cost, sensitive to head movement
PyGaze	Python open-source	General public	Calibration error rate	Customizable, effective in lab settings	Requires specialized hardware, limited to lab usage
Gazepoint GP3	Low-cost device	Cognitive awareness	Time stability	Affordable, useful for basic research	Limited to controlled environments
OpenGazer	Face recognition	ALS patients	Detection accuracy	Supports real-time eye tracking, AAC integration	Requires regular calibration, fixed setup



Tobii Eye Tracker 5



Gazepoint GP3 Tracking Device

Một số công nghệ hỗ trợ phổ biến

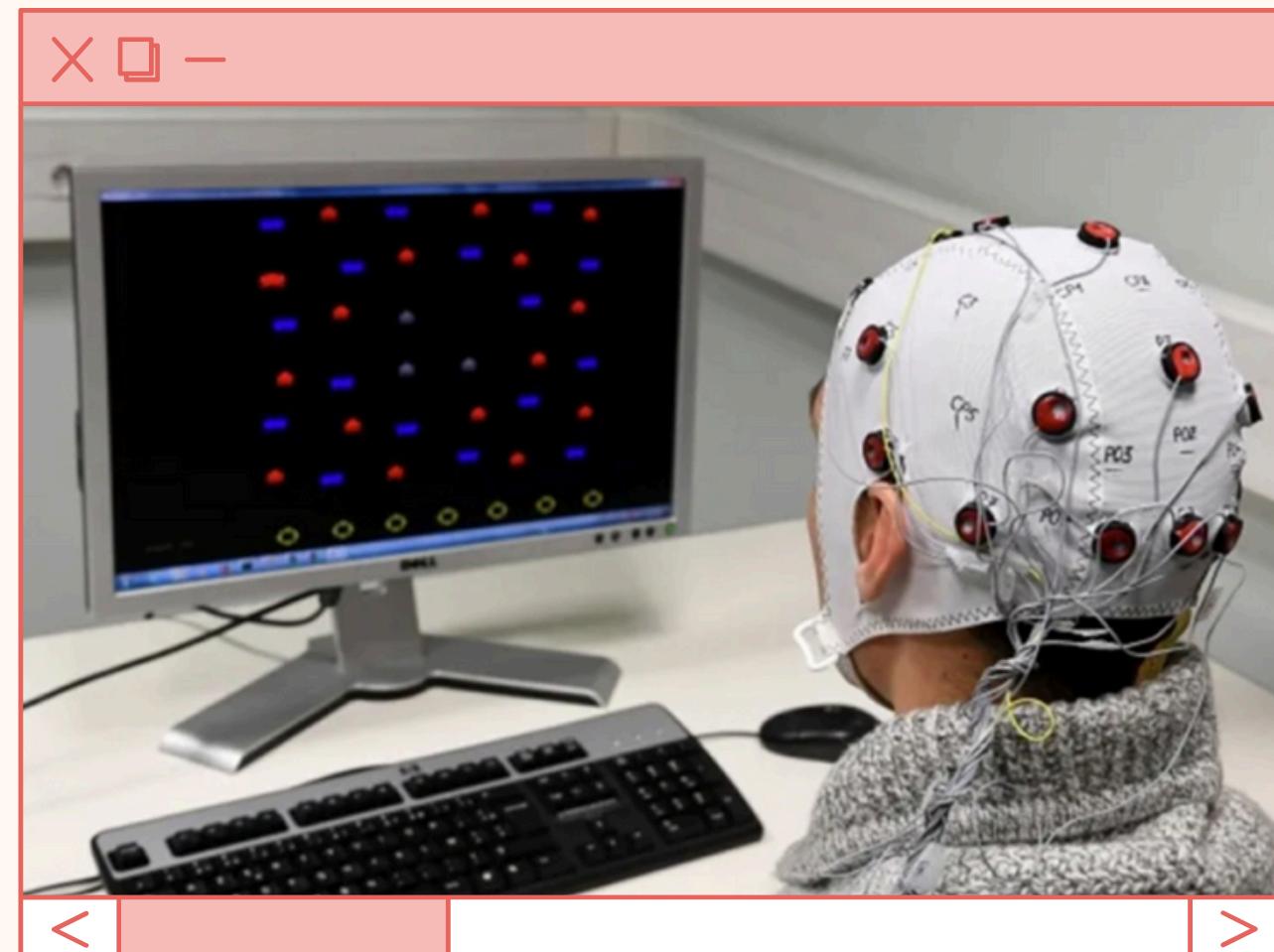


Thiết bị giao tiếp thay thế (AAC):

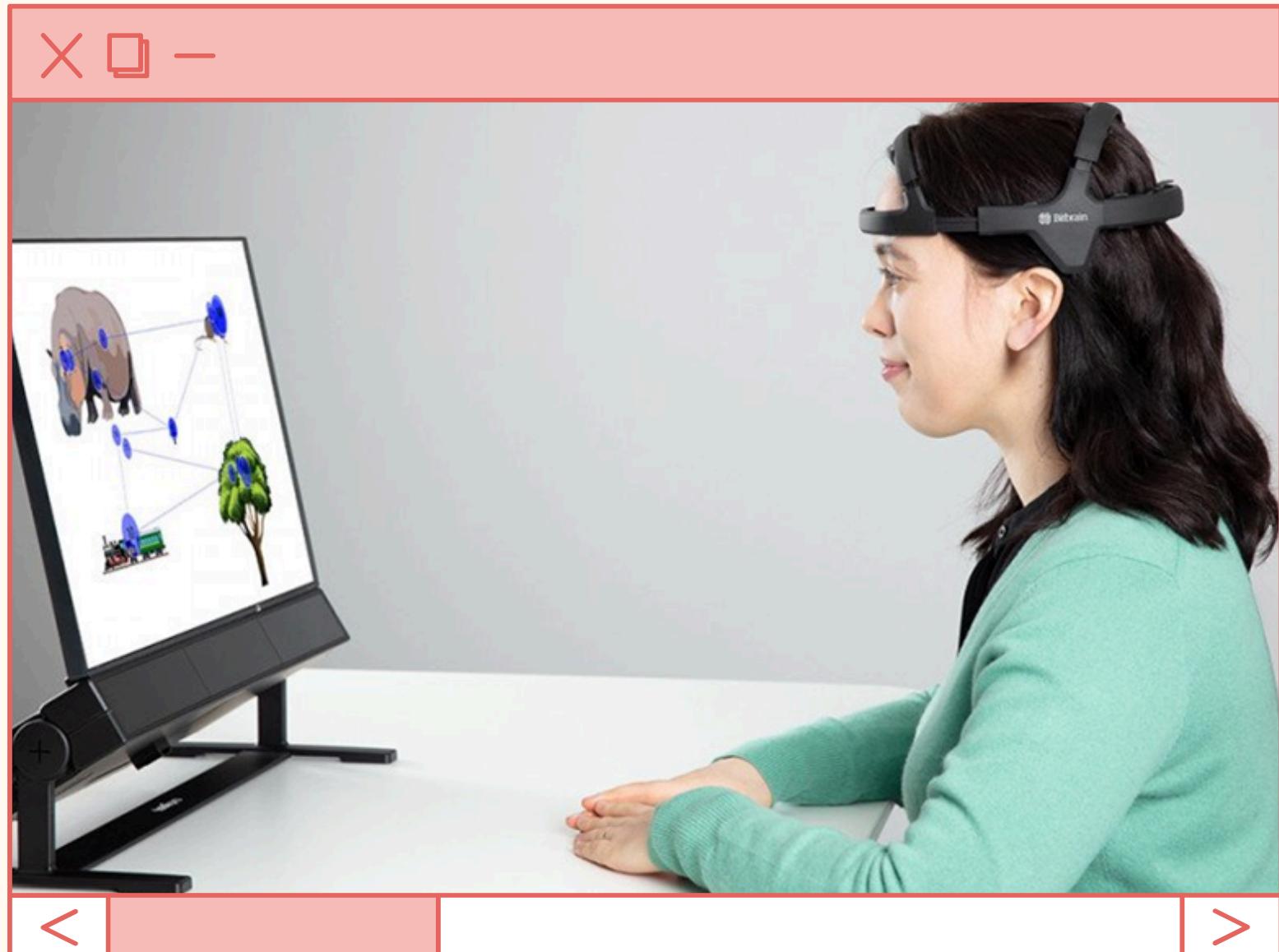
- Công nghệ giúp bệnh nhân ALS giao tiếp không cần giọng nói/cử động
- Sử dụng bảng chữ cái điện tử, máy tính với phần mềm hỗ trợ
- Nhập liệu bằng nút bấm hoặc chuyển động mắt

Giao diện điều khiển bằng tín hiệu não (BCI):

- Công nghệ cho phép điều khiển thiết bị qua tín hiệu não
- Sử dụng cảm biến đeo trên đầu
- Chức năng: di chuyển chuột, gõ chữ, điều khiển thiết bị thông minh
- Đang phát triển, tiềm năng lớn cho bệnh nhân ALS



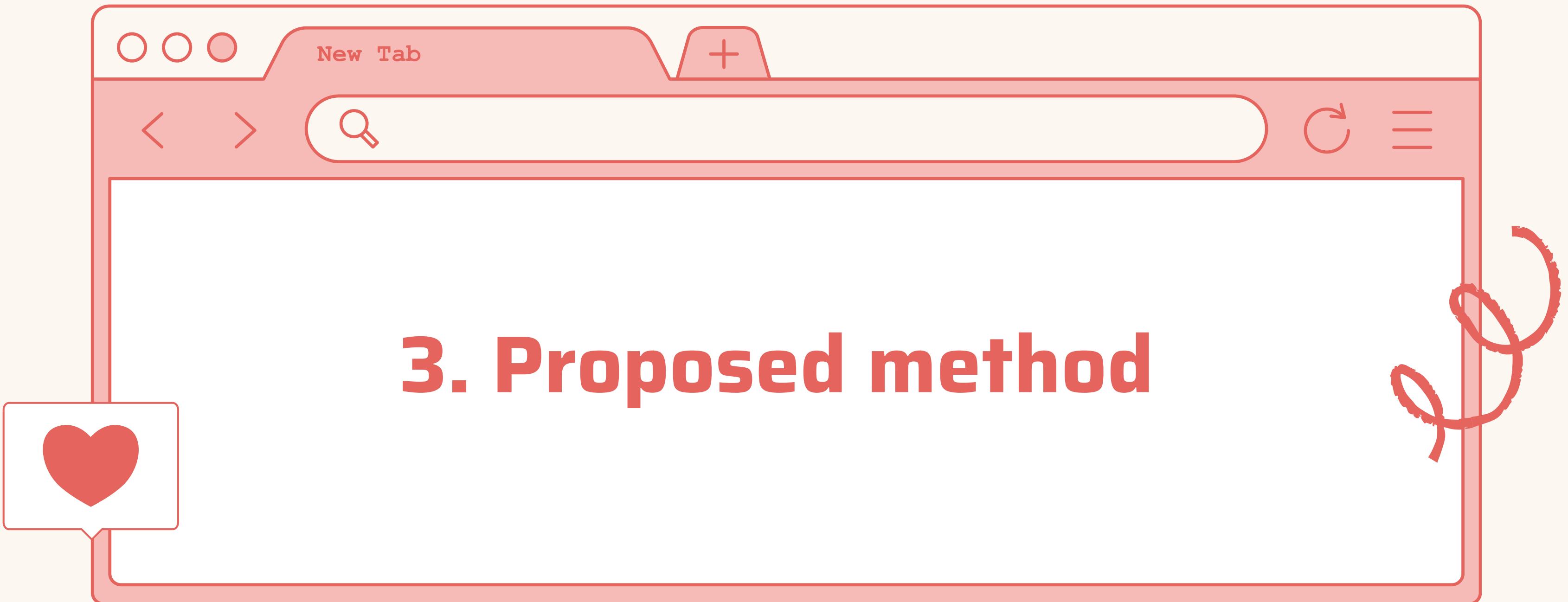
Một số công nghệ hỗ trợ phổ biến



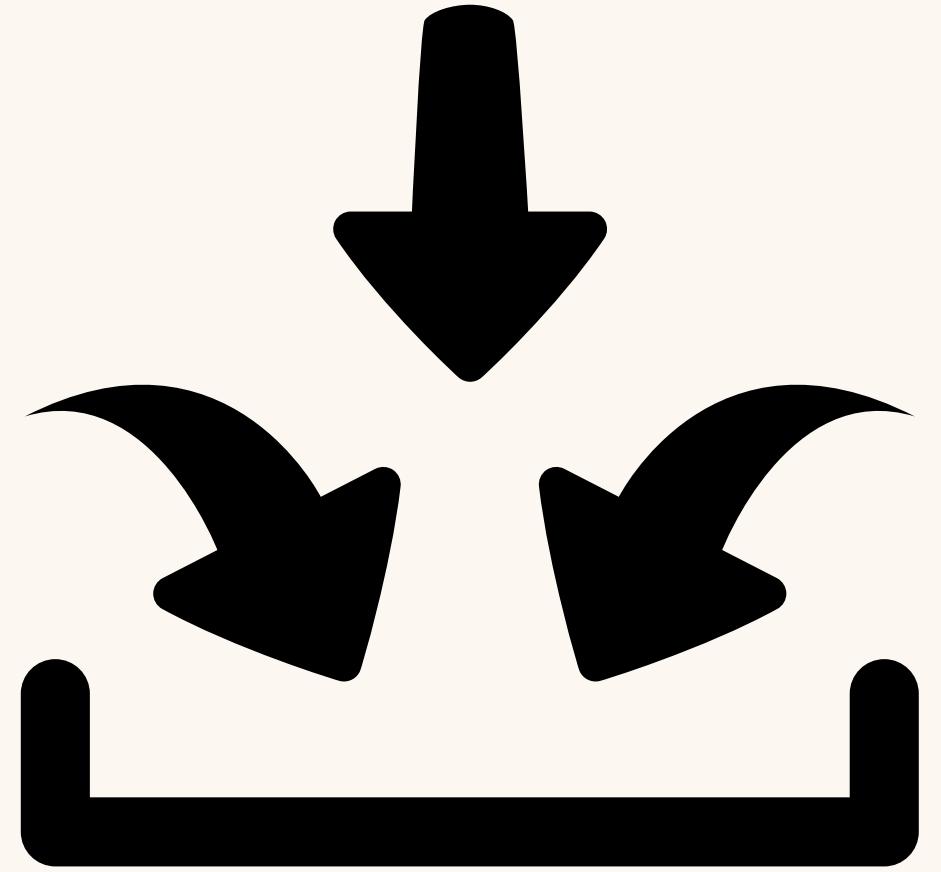
Hệ thống nhận diện chuyển động mắt:

- Điều khiển thiết bị qua chuyển động mắt
- Chức năng: điều khiển con trỏ, nhập văn bản, điều khiển thiết bị
- Tăng khả năng tự chủ và giao tiếp

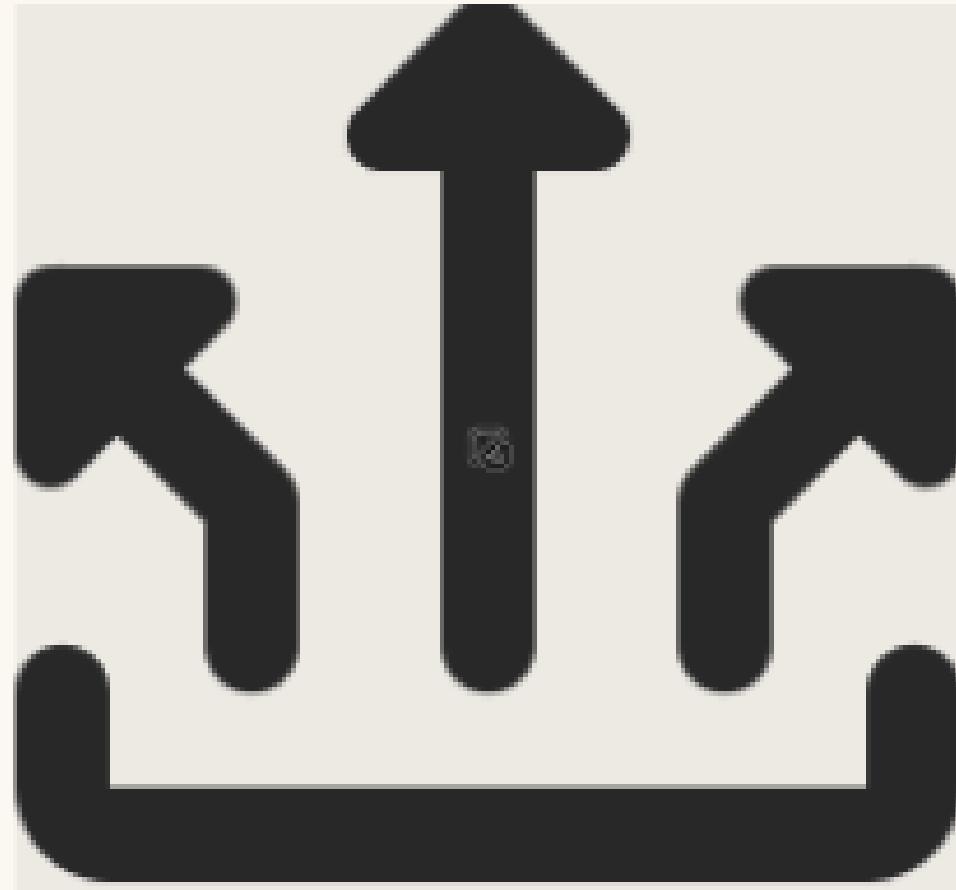
3. Proposed method



INPUT



OUTPUT



- **Chuyển động mắt**
- **Tương tác người dùng**
- **Cài đặt hệ thống**

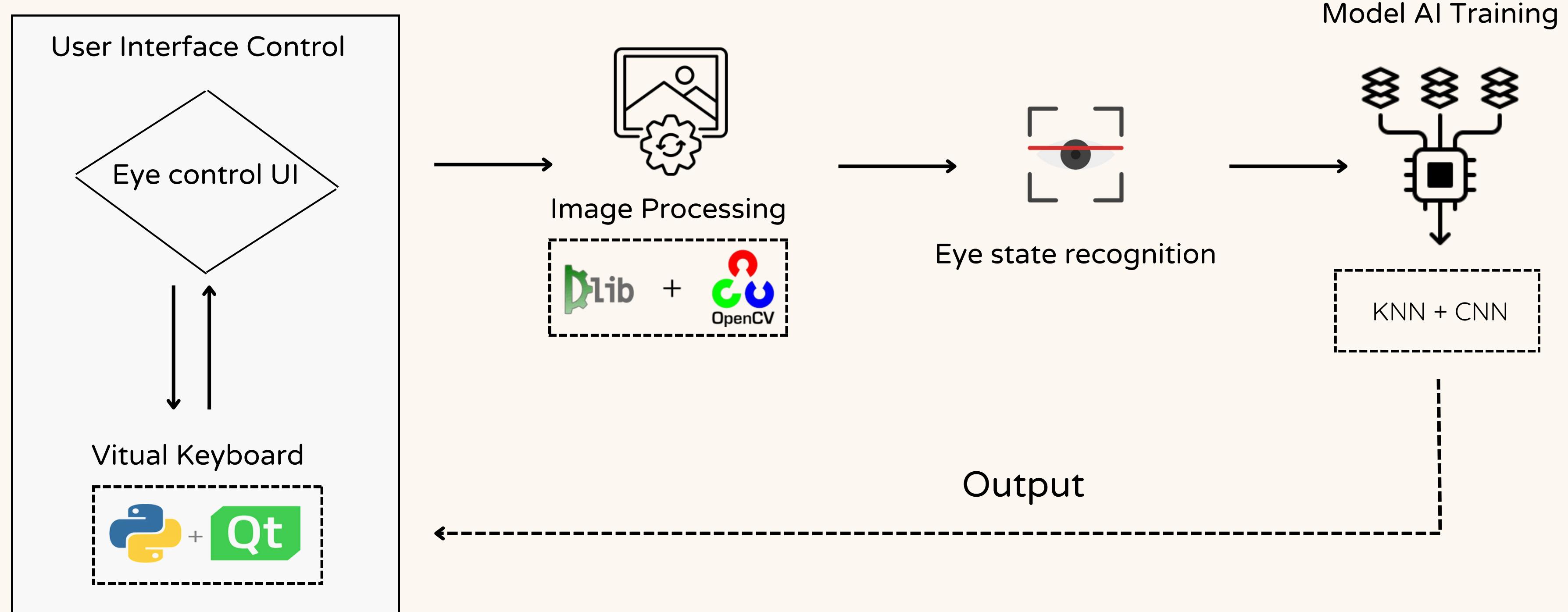
Ví dụ:

Đầu vào: Nhìn vào nút "A".

Đầu ra: Ký tự "A" hiển thị

- **Phản hồi giao tiếp**
- **Giao diện người dùng**

System Architecture Diagram



3.1 Thiết kế hệ thống nhận diện chuyển động mắt

3.1.1 Mô hình nhận diện khuôn mặt và mắt

- **Phát hiện khuôn mặt:** Thư viện Dlib cung cấp công cụ get_frontal_face_detector()
- **Xác định điểm đặc trưng:** Dlib cung cấp mô hình shape_predictor
- **Cắt và xử lý vùng mắt:** tính toán chỉ số EAR.

Một số hàm chính trong file eye_tracking.py liên quan đến nhận diện khuôn mặt và mắt:

```
# Initialize Mediapipe Face Mesh and Dlib
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(refine_landmarks=True)
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

- **get_frontal_face_detector():** Hàm của Dlib dùng để phát hiện khuôn mặt trong video.
- **shape_predictor:** Hàm dùng để xác định các điểm đặc trưng trên khuôn mặt. Hàm này nhận diện 68 điểm đặc trưng, bao gồm các điểm quanh mắt, giúp xác định vị trí và trạng thái mắt.

3.1 Thiết kế hệ thống nhận diện chuyển động mắt

3.1.2 Xác định điểm đặc trưng trên mắt

Một số hàm và biến quan trọng trong eye_tracking.py liên quan đến xác định điểm đặc trưng của mắt:

- **shape.parts()**: Trích xuất các điểm đặc trưng của khuôn mặt từ mô hình dự đoán. Các điểm đặc trưng này được sử dụng để xác định vùng mắt.
- **eye_landmarks**: Danh sách các điểm đặc trưng xung quanh mắt, giúp xác định vùng mắt và trạng thái của mắt.

3.1 Thiết kế hệ thống nhận diện chuyển động mắt

3.1.3 Tính toán chỉ số EAR để nhận diện trạng thái mắt

Một số hàm và biến quan trọng trong eye_tracking.py

liên quan đến tính toán chỉ số EAR:

```
# Calculate EAR for blink detection
ear_left = self.calculate_ear(left_eye)
ear_right = self.calculate_ear(right_eye)

# Determine eye status based on EAR values
if ear_left < 0.2 or ear_right < 0.2:
    eye_status = "closed"
else:
    eye_status = "open"

# Emit eye status
self.eye_status_signal.emit(eye_status)
```

- **calculate_EAR(eye_points):** Hàm nhận vào các điểm đặc trưng của mắt và trả về giá trị EAR. Giá trị này được so sánh với một ngưỡng để xác định trạng thái nhắm hoặc mở của mắt.
- **EAR_threshold:** Giá trị ngưỡng cho EAR để phân biệt trạng thái nhắm và mở.

3.2 Phát triển thuật toán phát hiện hướng nhìn

3.2.1 Phân tích chuyển động đồng tử

- Xác định vùng đồng tử
- Phát hiện đồng tử bằng cách phân tích cường độ màu
- Theo dõi vị trí đồng tử theo trục X và Y

Hàm quan trọng trong eye_tracking.py liên quan đến phân tích chuyển động đồng tử:

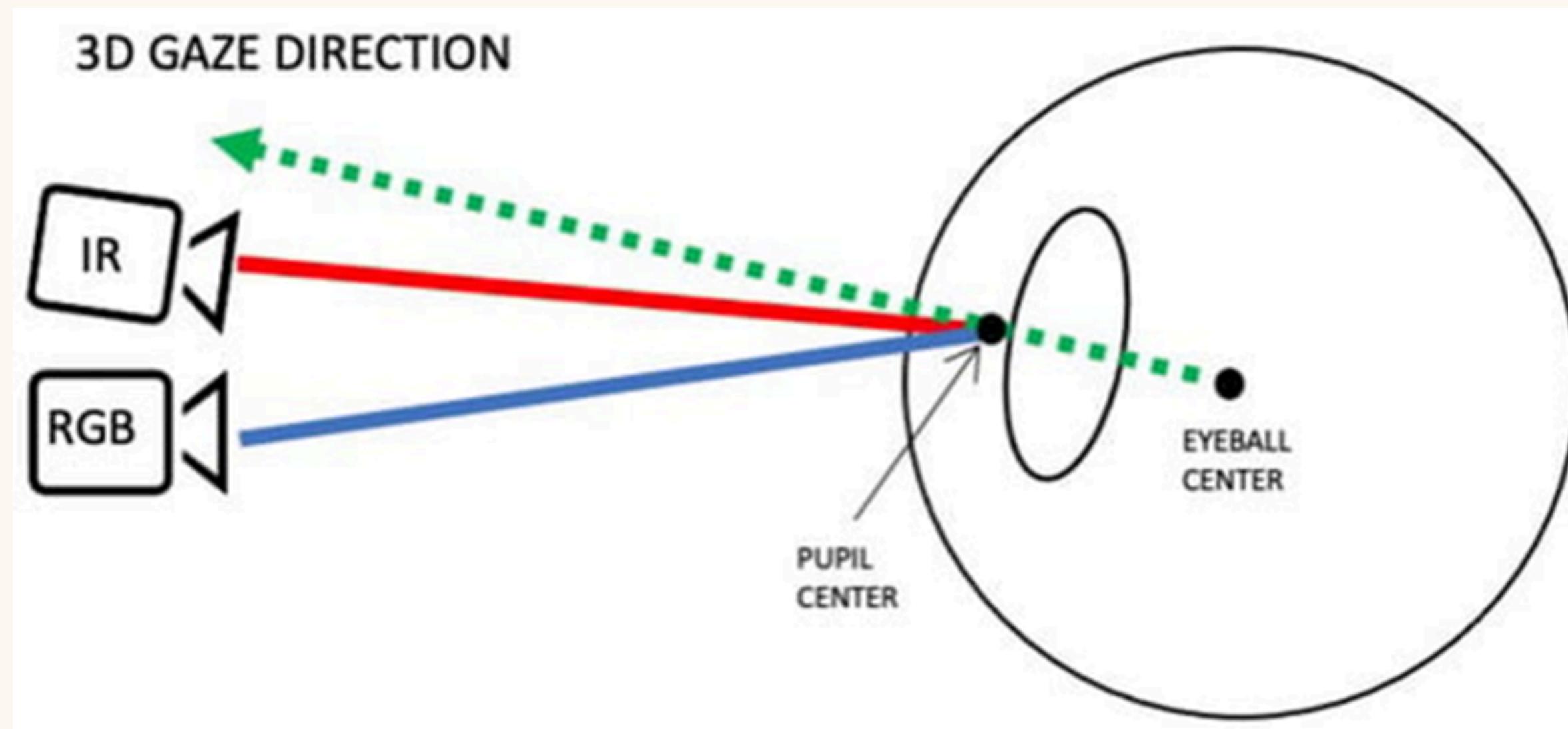
```
def detect_pupil_position(self, gray_frame, eye_region):  
    # Cắt vùng mắt  
    x_min, y_min = np.min(eye_region, axis=0)  
    x_max, y_max = np.max(eye_region, axis=0)  
    eye_roi = gray_frame[y_min:y_max, x_min:x_max]  
    # Áp dụng cân bằng histogram để tăng cường độ tương phản  
    eye_roi = cv2.equalizeHist(eye_roi)  
    # Áp dụng GaussianBlur để giảm nhiễu  
    eye_roi = cv2.GaussianBlur(eye_roi, (5, 5), 0)  
    # Áp dụng ngưỡng nhị phân với một giá trị ngưỡng cố định  
    threshold_value = 50 # Điều chỉnh giá trị này dựa trên điều kiện ánh sáng và độ tương phản  
    _, thresholded_eye = cv2.threshold(eye_roi, threshold_value, 255, cv2.THRESH_BINARY_INV)  
    # Tìm các đường viền trong ảnh đã áp dụng ngưỡng  
    contours, _ = cv2.findContours(thresholded_eye, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)  
    if contours:  
        # Lọc các đường viền theo diện tích để loại bỏ nhiễu (giả định rằng đồng tử nằm trong một phạm vi kích thước hợp lý).  
        min_area = 5  
        max_area = 500 # Điều chỉnh dựa trên kích thước đồng tử điển hình trong thiết lập của bạn.  
        valid_contours = [cnt for cnt in contours if min_area < cv2.contourArea(cnt) < max_area]  
        if valid_contours:  
            # Tìm đường viền hợp lệ lớn nhất, giả định đó là đồng tử.  
            largest_contour = max(valid_contours, key=cv2.contourArea)  
            (x, y), radius = cv2.minEnclosingCircle(largest_contour)  
            if radius > 1:  
                # Trả về tọa độ tâm đồng tử trong khung hình gốc.  
                return np.array([int(x + x_min), int(y + y_min)])  
    return None # Trả về None nếu không phát hiện được đồng tử hợp lệ.
```

- **detect_pupil_position():** Hàm này xác định vị trí đồng tử dựa trên phân ngưỡng và kỹ thuật xử lý ảnh để tìm vùng tối trong mắt.

3.2 Phát triển thuật toán phát hiện hướng nhìn

3.2.2 Xác định hướng nhìn theo phương pháp ánh sáng hồng ngoại

- Phát ra ánh sáng hồng ngoại
- Xác định vị trí của điểm phản xạ
- Tính toán góc nhìn dựa trên vị trí đồng tử và điểm phản xạ



3.2 Phát triển thuật toán phát hiện hướng nhìn

3.2.3 Ứng dụng học máy để dự đoán hướng nhìn

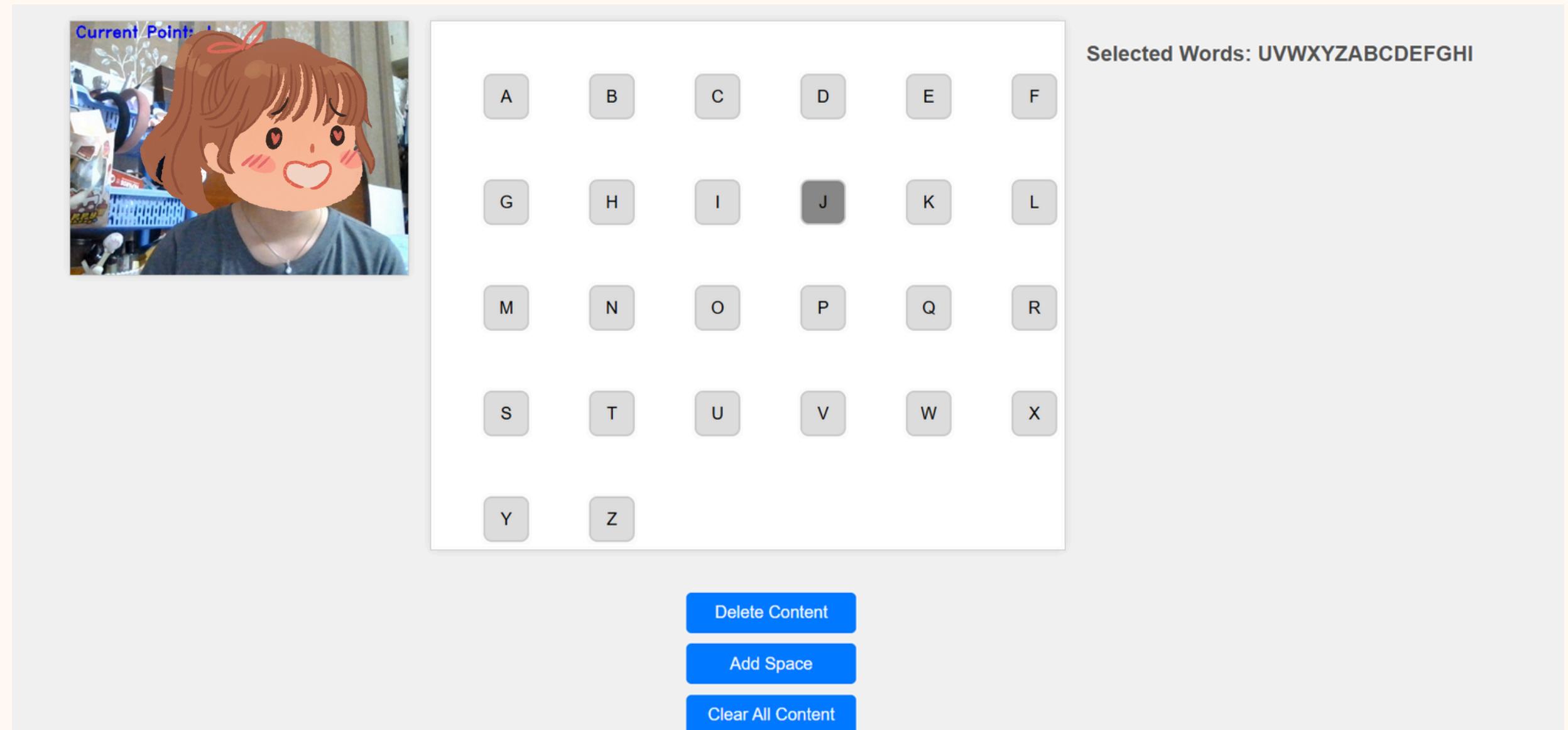
Quá trình xây dựng mô hình học máy:

- Thu thập dữ liệu huấn luyện
- Xây dựng mô hình học máy: KNN, CNN
- Dự đoán hướng nhìn trong thời gian thực

Một số hàm quan trọng liên quan đến học máy mà có thể được triển khai trong eye_tracking.py:

- **train_model()**: Hàm này dùng để huấn luyện mô hình học máy với dữ liệu thu thập được về vị trí đồng tử và hướng nhìn.
- **predict_gaze_direction()**: Hàm này sử dụng mô hình đã huấn luyện để dự đoán hướng nhìn dựa trên vị trí đồng tử trong thời gian thực.

3.3 Phát triển giao diện người dùng

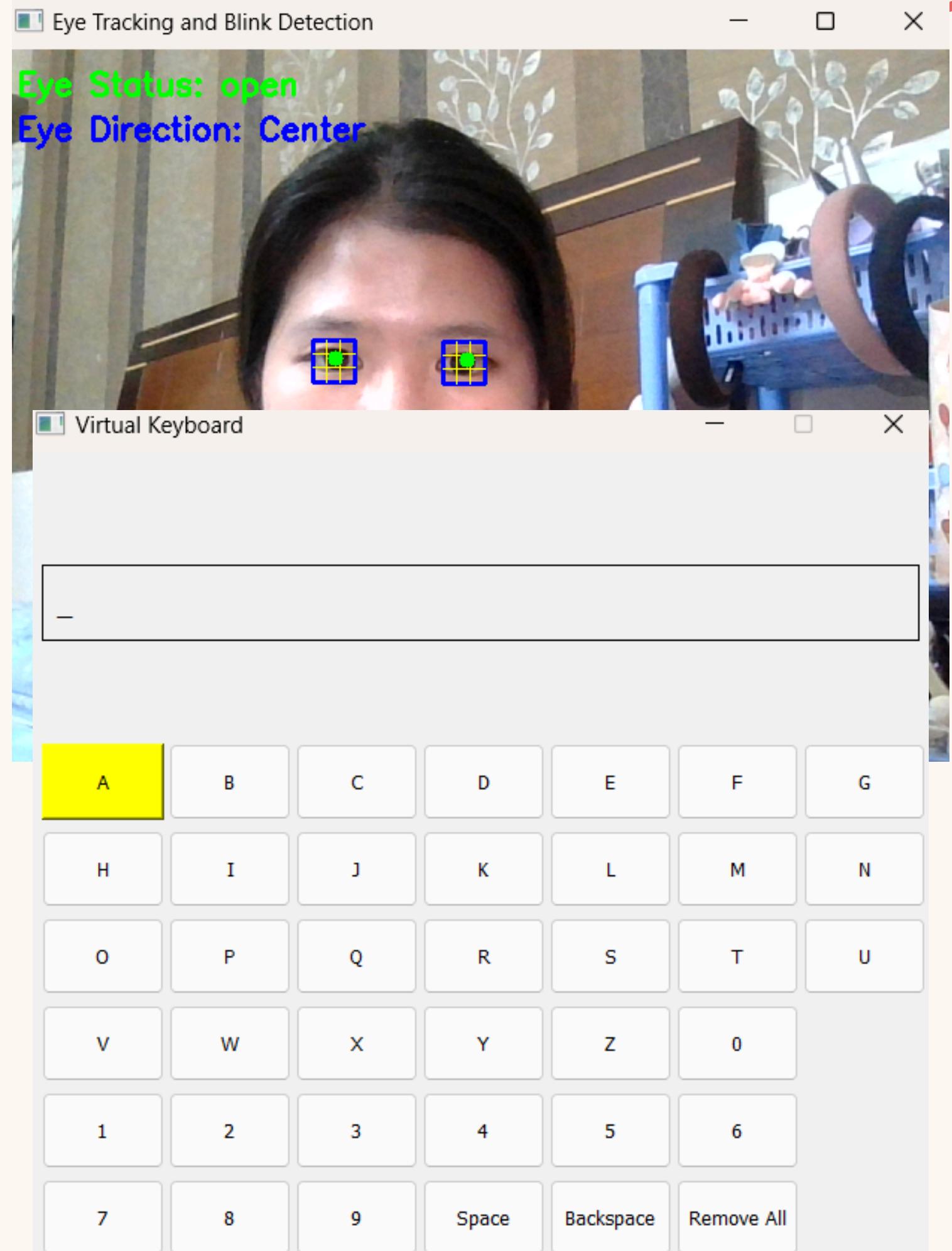


- Khi đặt **camera (trái)** và **bàn phím ảo (phải)**, người bệnh liên tục nhìn bên phải để chọn phím -> hệ thống nhận diện sai hướng ánh mắt và gây ra các lỗi nhập liệu.
- Khi chuyển camera lên trên và bàn phím xuống dưới -> không đủ không gian hiển thị trên một trang, phải cuộn lên xuống, gây bất tiện cho người bệnh.

3.3 Phát triển giao diện người dùng

3.3.1 Bàn phím ảo

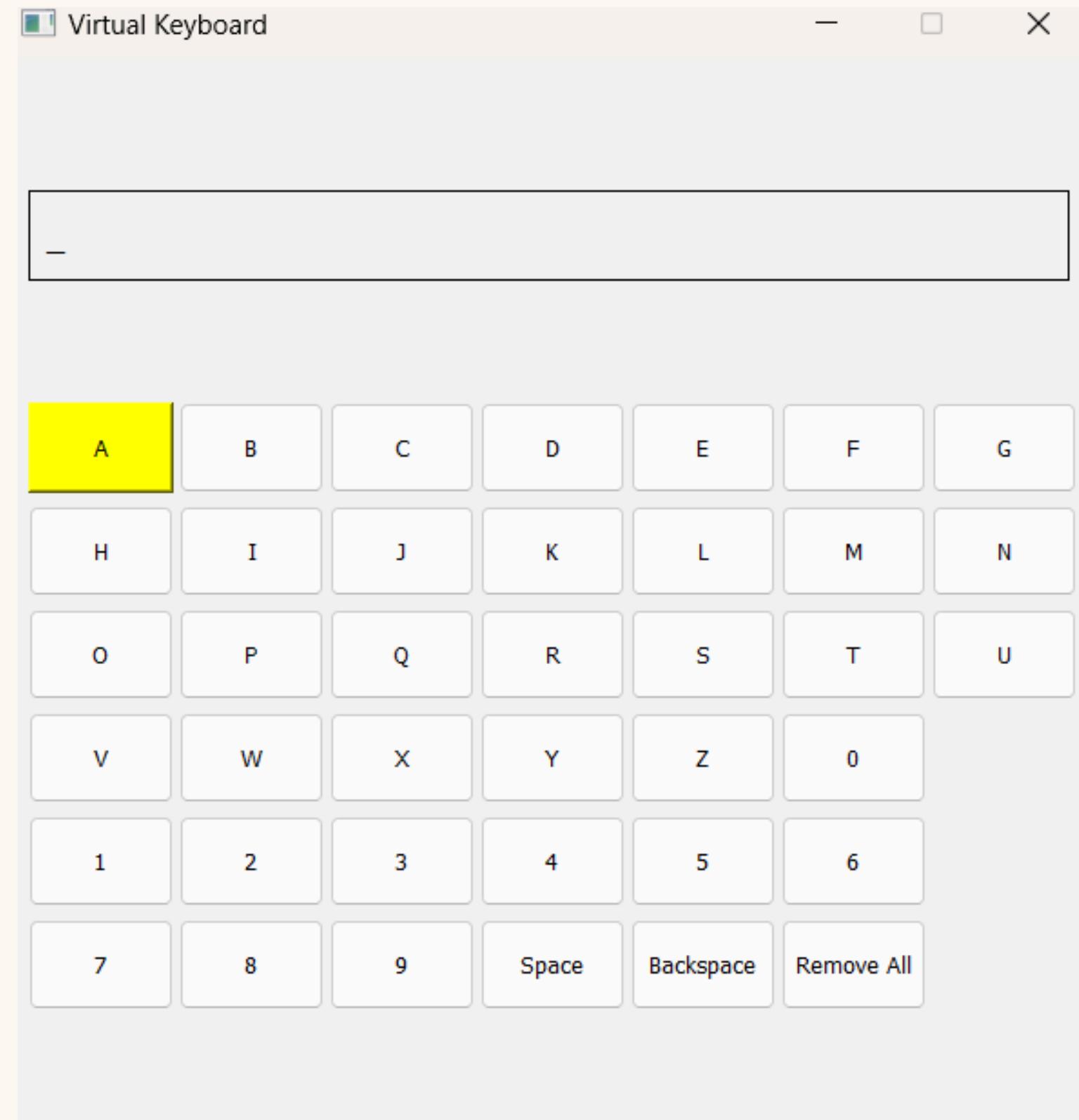
Thiết kế giao diện mới thành 2 cửa sổ độc lập bao gồm một bàn phím ảo và các chức năng điều khiển cơ bản để người dùng có thể nhập văn bản, xóa và chọn các phím mà không cần sử dụng tay.



3.3 Phát triển giao diện người dùng

Được thiết kế bằng cách sử dụng các thành phần của PyQt5, gồm các **nút bấm (QPushButton)** đại diện cho các ký tự và các phím chức năng như "**Space**", "**Backspace**", và "**Remove All**".

- **Cấu Trúc Bảng Chữ Cái:** Ký tự được sắp xếp theo thứ tự bảng chữ cái, giúp người dùng dễ nhớ và chọn.
- **Nút Chức Năng:** Bao gồm "Space" (thêm khoảng trắng), "Backspace" (xóa ký tự cuối) và "Remove All" (xóa toàn bộ văn bản).
- **Hiển Thị Văn Bản:** Vùng hiển thị (QLabel) để người dùng theo dõi văn bản đã nhập.



3.3 Phát triển giao diện người dùng

3.3.2 Tích hợp chức năng điều khiển bằng ánh mắt

Cho phép người dùng chọn các nút trên bàn phím ảo chỉ bằng hướng nhìn, không cần chạm tay.

EyeTracker: Nhận diện hướng và trạng thái mắt, gửi tín hiệu điều khiển giao diện.

Dựa trên hướng mắt (trái, phải, lên, xuống), hệ thống sẽ di chuyển con trỏ và chọn nút tương ứng:

- Kết nối tín hiệu từ EyeTracker
- Điều hướng và chọn nút

Các Thành phần chính:

- **self.eye_direction:** Lưu hướng nhìn, cập nhật khi EyeTracker phát hiện thay đổi.
- **update_button_selection():** Cập nhật nút được chọn dựa trên hướng nhìn.
- **eye_status:** Theo dõi trạng thái mở/nhắm mắt, xác nhận chọn nút khi mắt nhắm.

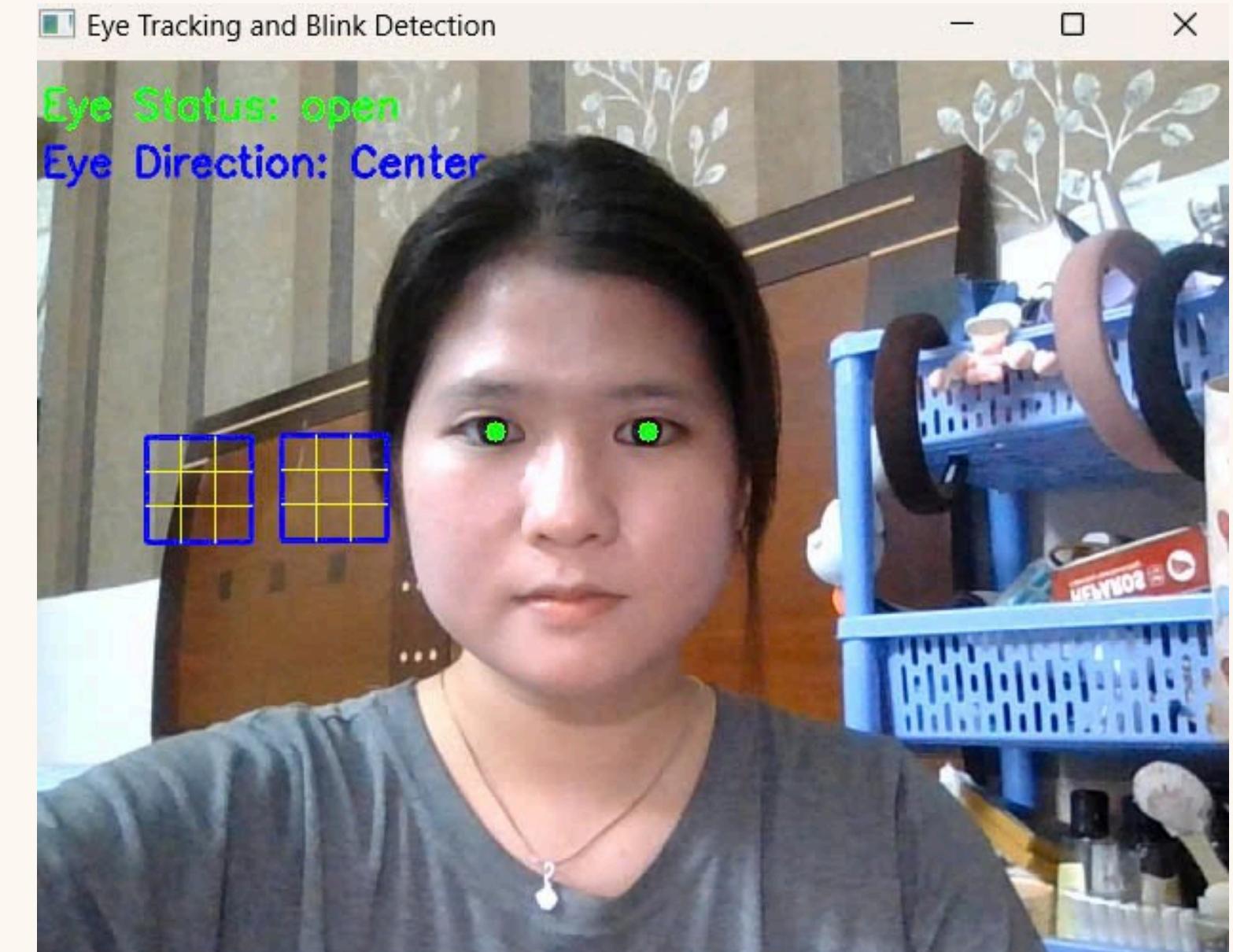
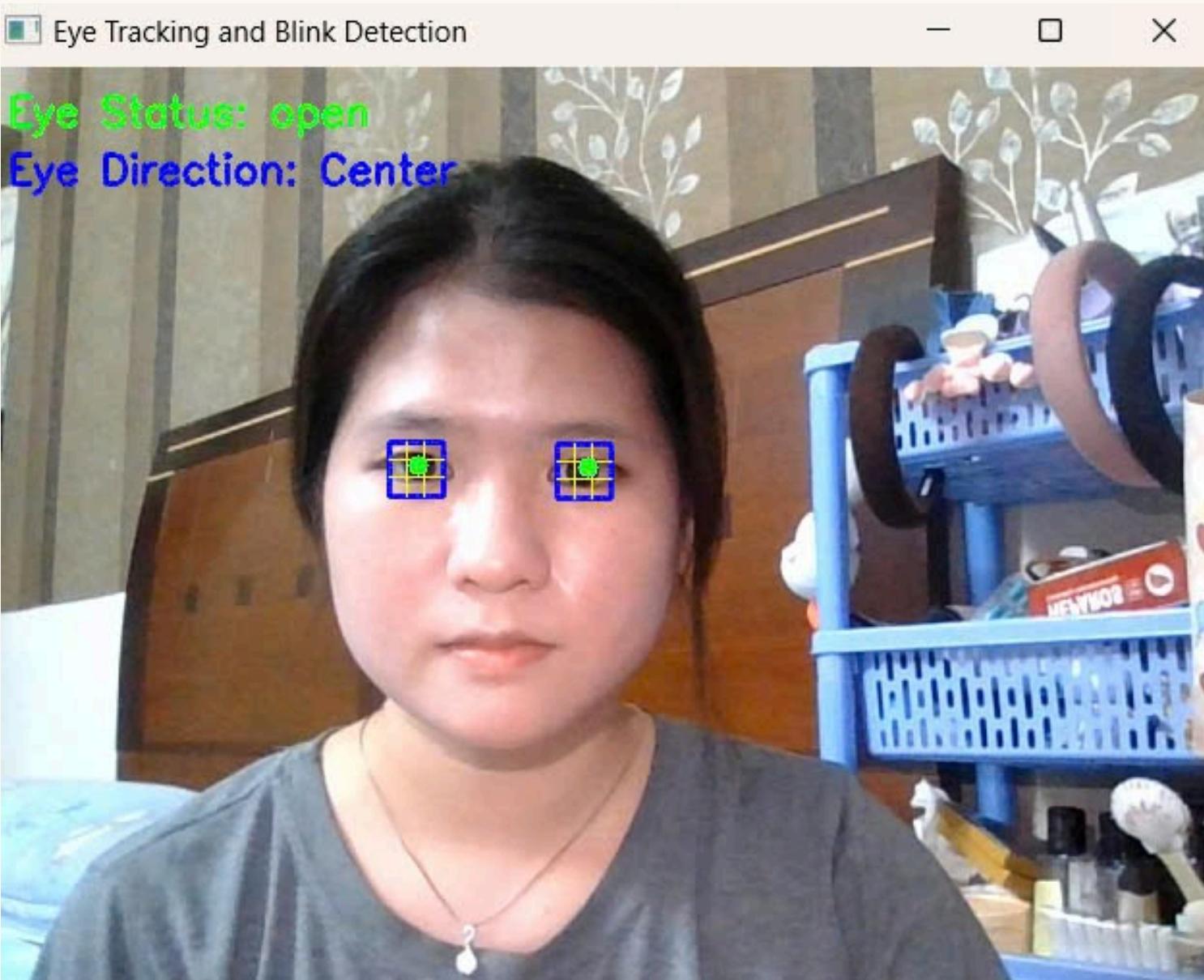
```
# Khai báo biến  
self.eye_direction = "Center"  
self.eye_status = "open"
```

```
self.update_button_selection()
```

3.3 Phát triển giao diện người dùng

Or

3.3.3 Ô vuông theo dõi trạng thái mắt

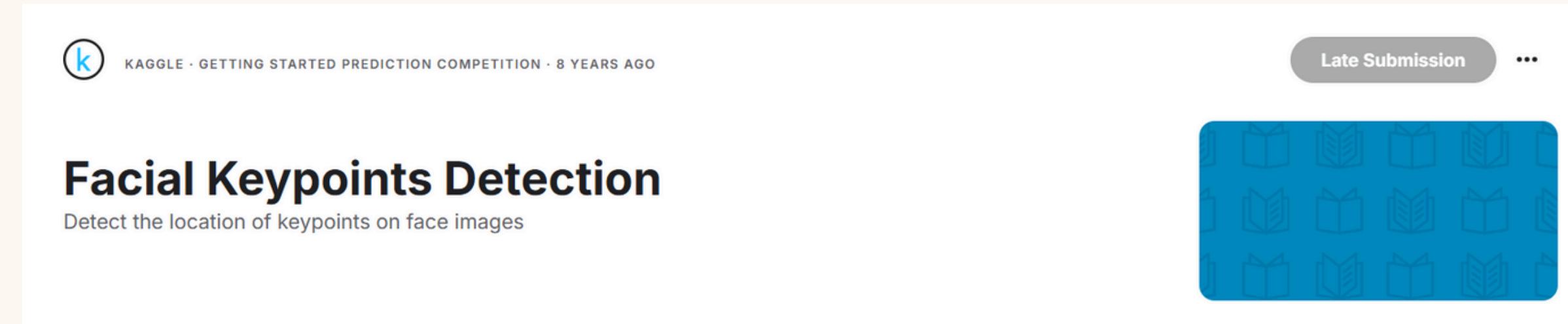


- Theo dõi và hiển thị trạng thái mắt, giúp hệ thống xác định chính xác hướng nhìn và tình trạng mở/nhắm mắt.
- Tự động điều chỉnh kích thước, vị trí theo khuôn mặt để tăng độ chính xác của hệ thống theo dõi mắt.

Dữ liệu

Nhóm nghiên cứu đã triển khai hai phương pháp để sử dụng mô hình:

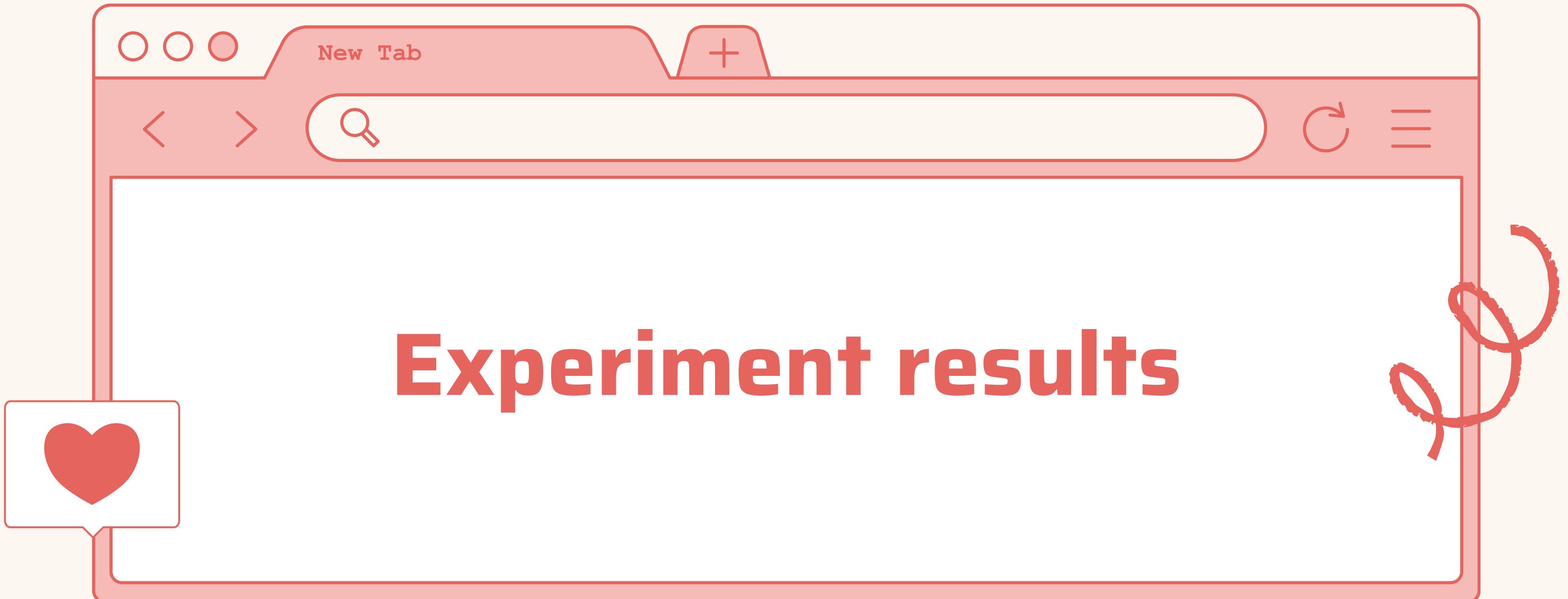
- Tự Train (Self-training): Phương pháp này liên quan đến việc sử dụng tập dữ liệu có sẵn tại Kaggle



- Sử dụng các tập dữ liệu đã được train sẵn (Pre-trained datasets):
shape_predictor_68_face_landmarks
haarcascade_eye: Python Programming
haarcascade_frontalface_default

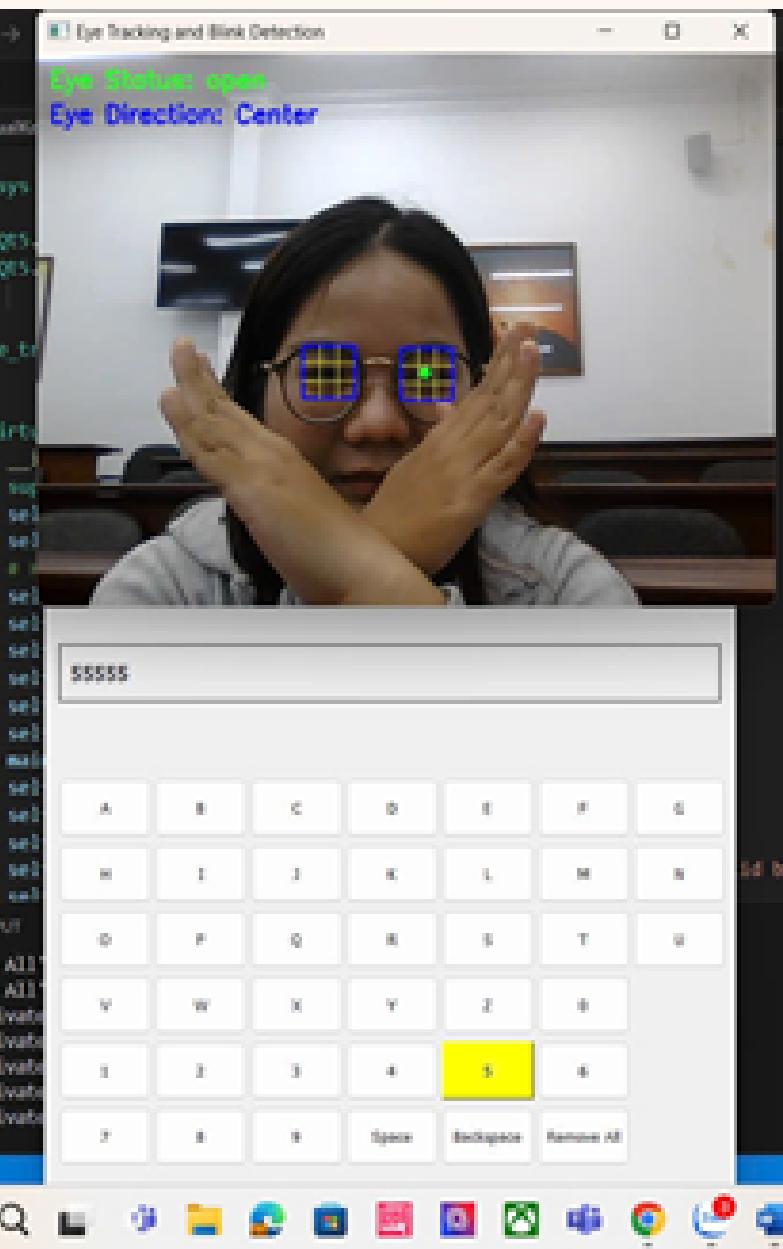
3.4 Kiểm tra mô hình

Model	<code>shape_predictor_68_face_landmarks (1)</code>	<code>haarcascade_eye (2)</code>	<code>haarcascade_frontalface_default (3)</code>	<code>facial keypoints model (4)</code>
Nguồn	Dlib	OpenCV	OpenCV	Kaggle
Đặc điểm	68 điểm mốc khuôn mặt	Phát hiện vùng mắt	Phát hiện khuôn mặt	Phát hiện các điểm mốc khuôn mặt
Độ chính xác	Cao (nhận diện chi tiết khuôn mặt)	Trung bình (nhạy cảm với ánh sáng)	Trung bình (nhạy cảm với góc quay)	Phụ thuộc vào dữ liệu huấn luyện
Tốc độ xử lý	Trung bình	Nhanh	Nhanh	Chưa biết
Độ phức tạp	Cao (có sẵn trong Dlib)	Thấp (có sẵn trong OpenCV)	Thấp (có sẵn trong OpenCV)	Cao (cần mô hình huấn luyện)
Ứng dụng	Theo dõi chính xác chuyển động mắt	Phát hiện nháy mắt đơn giản	Phát hiện khuôn mặt tổng quan	Theo dõi chi tiết các điểm mốc khuôn mặt
Ưu điểm	Độ chính xác cao, theo dõi chi tiết	Dễ sử dụng, nhanh chóng	Dễ sử dụng, hiệu quả cho nhận diện khuôn mặt	Chưa biết
Nhược điểm	Cần tài nguyên tính toán lớn	Không chính xác bằng khi có nhiều nhiễu	Không phù hợp cho theo dõi chi tiết	Cần tài nguyên tính toán lớn và dữ liệu huấn luyện

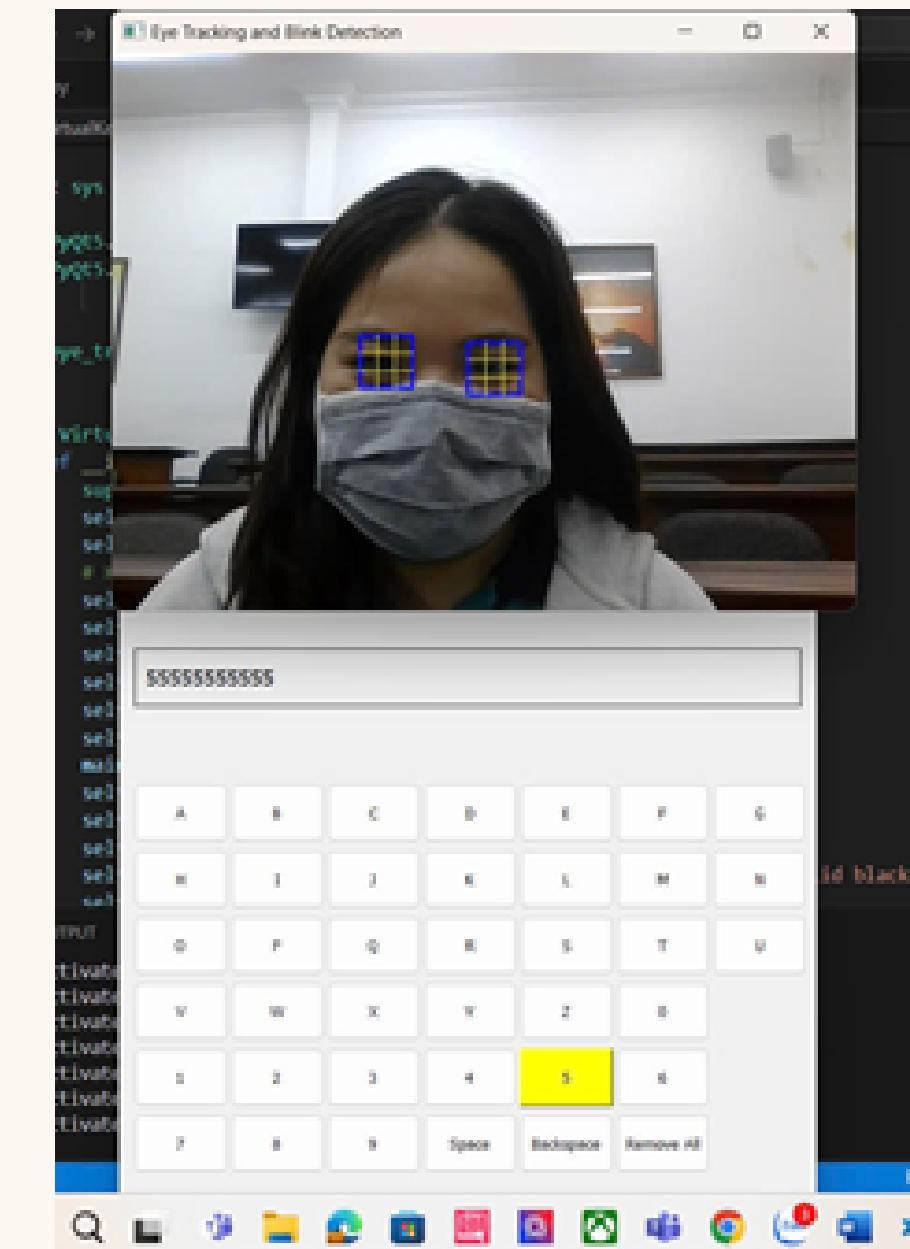


[Quay lại](#) [Trang Mục lục](#)

⚠️ Lưu ý:



Không nhận
diện được khi
đeo kính



Không nhận
diện được khi
đeo khẩu trang

Metric

Các chỉ số độ chính xác

Chỉ số	Mô tả	Ý nghĩa
Sai số trung bình (ME)	Sai số trung bình (pixel)	Đánh giá độ chính xác của việc định vị
Độ lệch chuẩn (STD)	Độ lệch chuẩn của các dự đoán	Đánh giá độ ổn định của hệ thống
Tỷ lệ phát hiện	Tỷ lệ phát hiện thành công	Đánh giá độ tin cậy của hệ thống

Các chỉ số hiệu suất

Chỉ số	Mô tả	Ý nghĩa
FPS	Số khung hình xử lý mỗi giây	Đánh giá khả năng xử lý theo thời gian thực
Sử dụng CPU	Mức độ sử dụng CPU	Đánh giá hiệu quả tài nguyên
Sử dụng bộ nhớ	Mức độ sử dụng bộ nhớ	Đánh giá yêu cầu hệ thống

Kết quả tập train

Kết quả nhận diện mắt

Metric	Value
Training Time	Approximately 26-71 seconds per epoch
Accuracy	0.8024
Loss	83.5561
Val_accuracy	0.9950
Val_loss	68.8546

Kết quả của Nhận diện mắt với ô vuông theo dõi trạng thái

1. Metric Results Of Eye Landmark Tracking

Eye Tracking Condition	Accuracy	Latency (ms)	Error Rate
Center gaze	78.2%	85-100	21.8%
Right gaze	72.5%	90-120	27.5%
Left gaze	71.8%	90-120	28.2%
Up gaze	65.5%	100-130	34.5%
Down gaze	63.8%	100-130	36.2%
Blinking	75.5%	80-100	24.5%
Eyes closed	77.8%	75-95	22.2%

2. Tracking Performance By Environmental Conditions

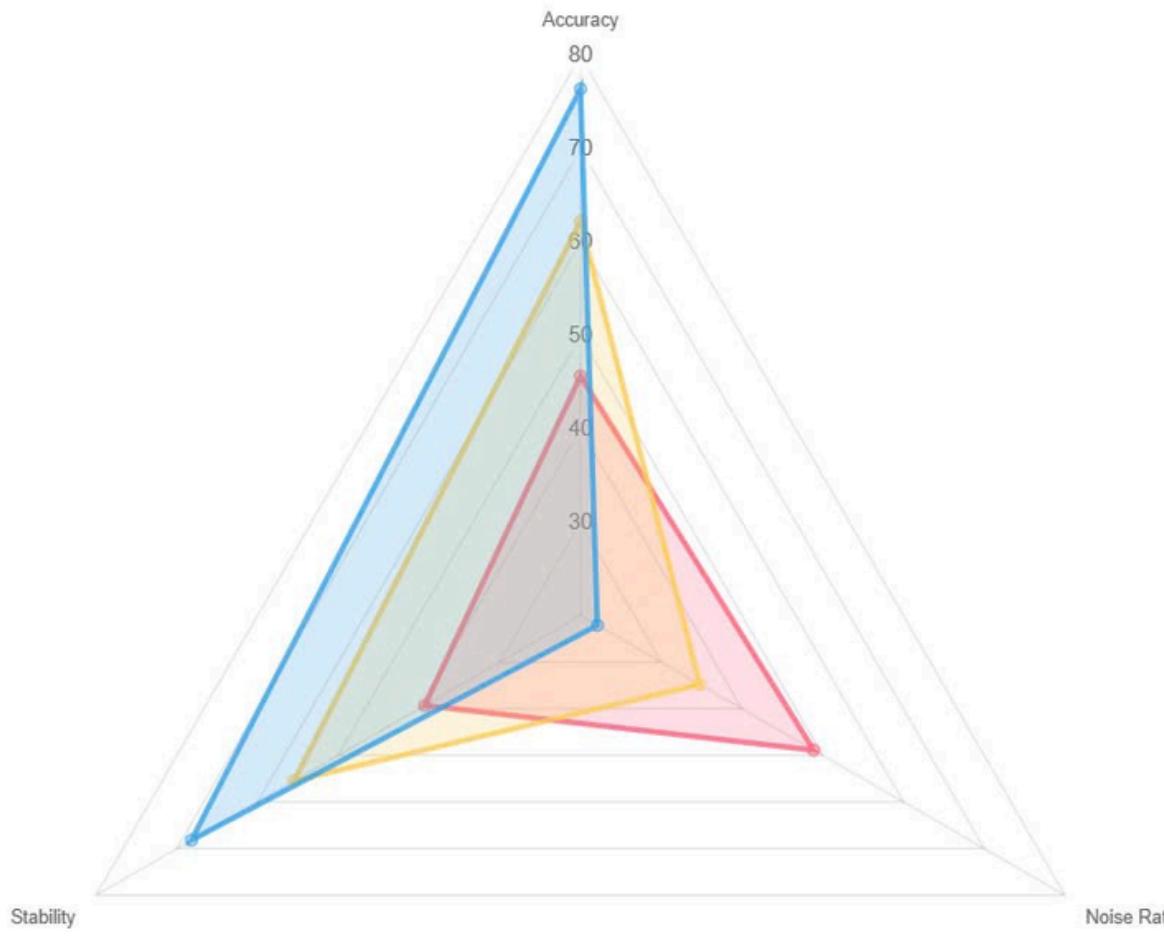
Lighting Condition	Accuracy	Noise Rate	Stability
Good light (>500 lux)	76.5%	22.1%	68.2%
Medium light	62.3%	34.8%	55.5%
Low light (<200 lux)	45.7%	48.9%	39.3%

3. Model Training Results (15 Epochs)

Epoch	Accuracy	Loss	Val_accuracy		Val_loss
5/15	0.5125	125.428	0.6850		112.334
10/15	0.5845	117.234	0.7456		103.562
15/15	0.6024	103.556	0.7950		98.8546

Performance by Environmental Conditions

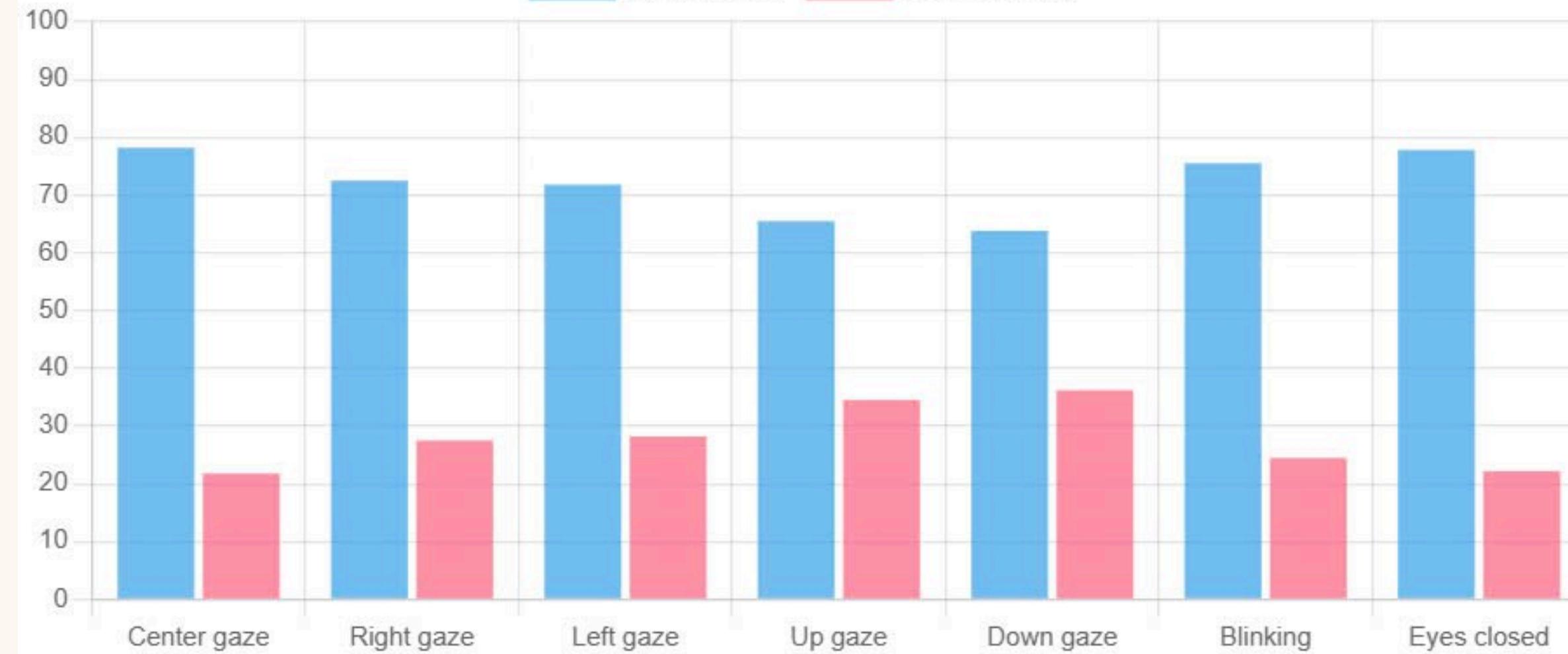
Good light Medium light Low light



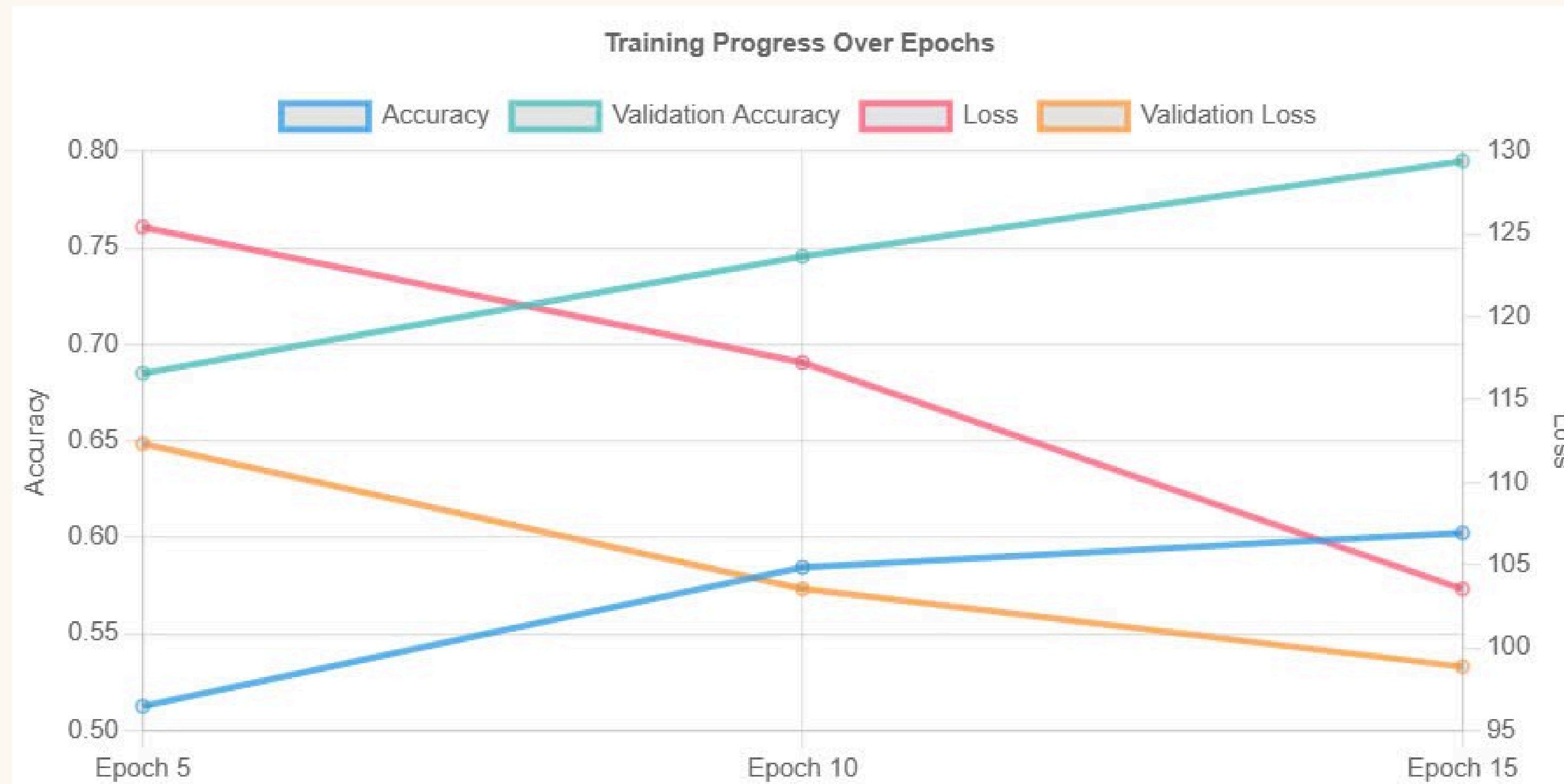
Kết quả của Nhận diện mắt với ô vuông theo dõi trạng thái

Eye Tracking Performance by Condition

Accuracy (%) Error Rate (%)



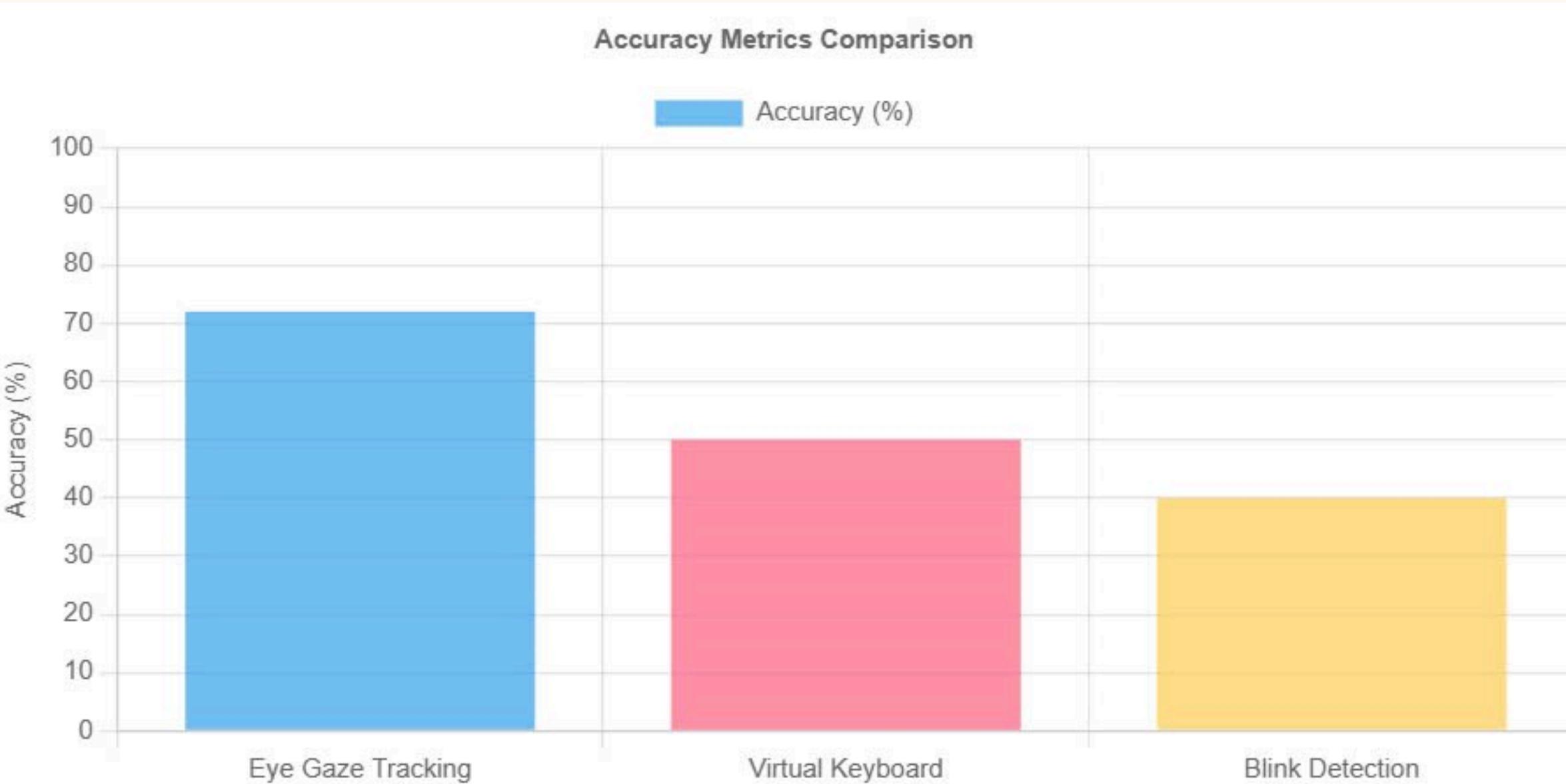
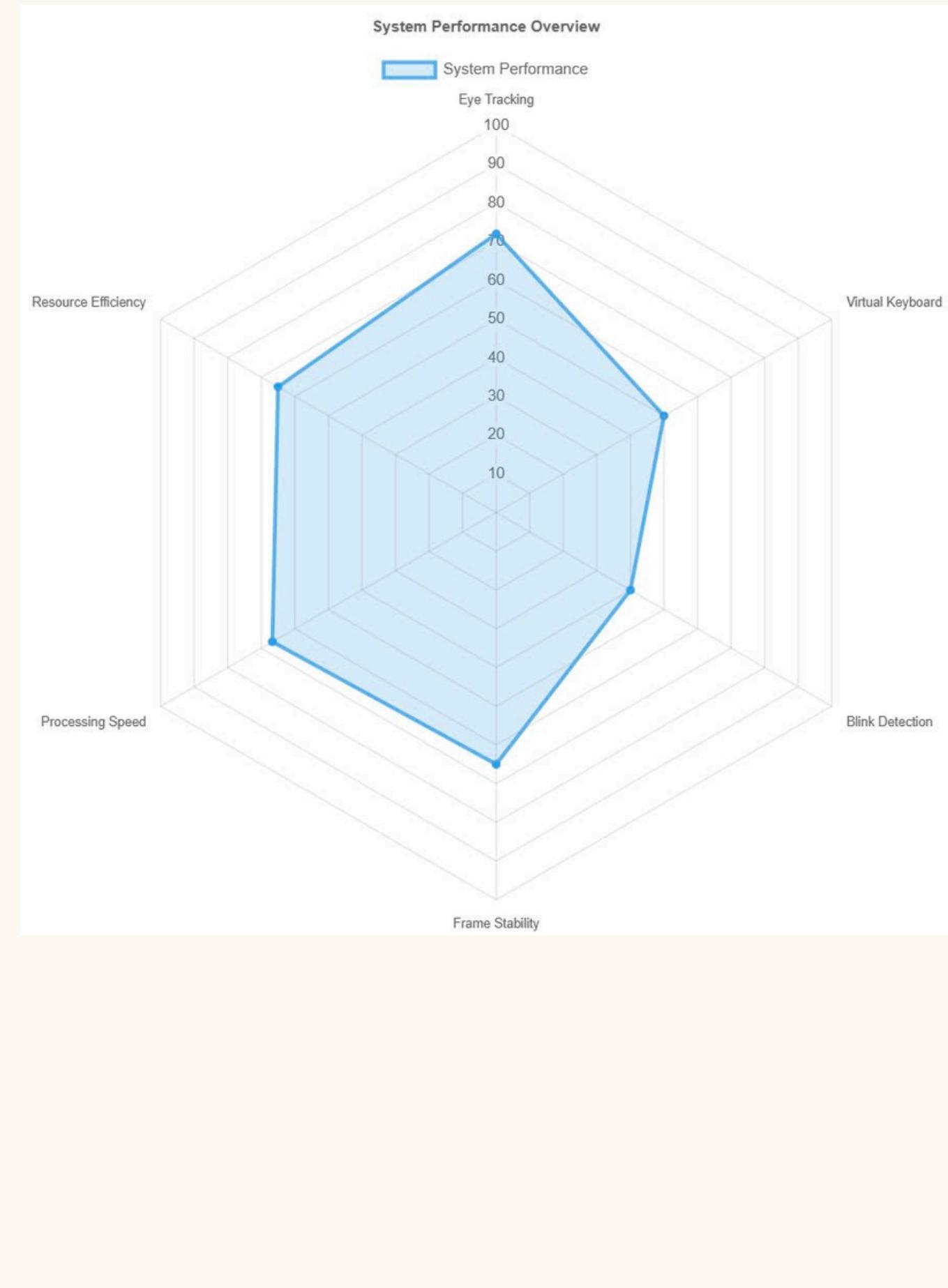
Kết quả của Nhận diện mắt với ô vuông theo dõi trạng thái



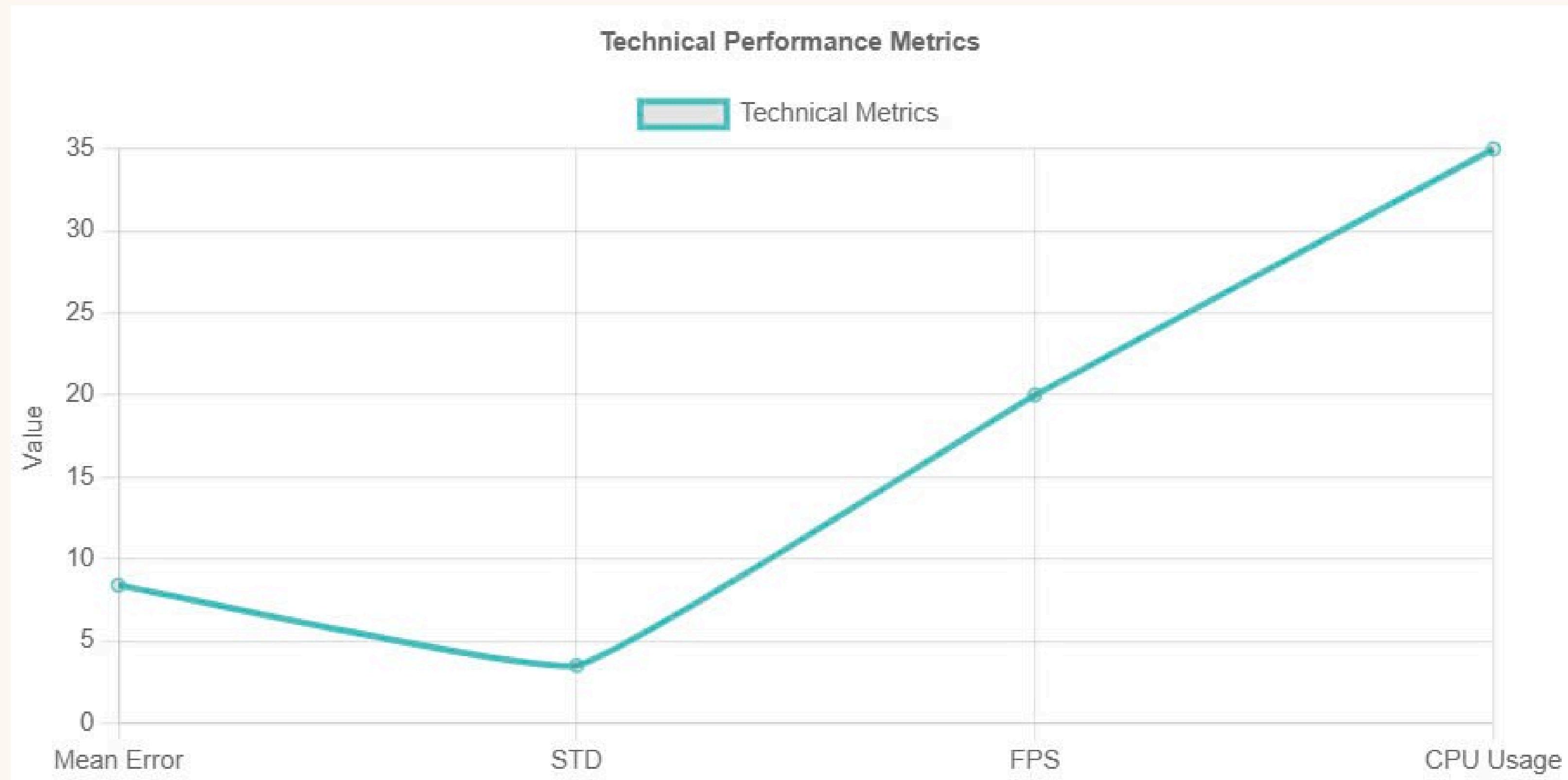
Kết quả của mắt với bàn phím

Metric	Description	Value	Remarks
Eye Gaze Tracking Accuracy	Accuracy of eye direction tracking	72%	Often deviates when the user moves quickly or changes gaze rapidly
Virtual Keyboard Accuracy	Accuracy in selecting characters via eye gaze	50%	Difficulty accurately detecting characters, prone to errors when lag or lighting interference occurs
Mean Error (ME)	Average distance error (in pixels)	8.4 px	High error rate affecting input accuracy
Standard Deviation (STD)	Variability in tracking accuracy	3.5 px	Unstable tracking, making interaction difficult
Frames Per Second (FPS)	Processing speed for real-time video	20 FPS	Slow processing, causing noticeable delay in eye movement tracking
Blink Detection	Accuracy in detecting blinks	40%	Inconsistent, prone to errors when eyes move rapidly
Display Frame Stability	Stability of display frame when tracking gaze	Frequently jumps	The display frame often jumps or becomes unstable when the user changes gaze direction quickly
Resource Usage (CPU)	CPU usage during operation	35%	High CPU load, potentially causing overload during prolonged use

Kết quả của mắt với bàn phím



Kết quả của mắt với bàn phím



Kết quả tập test

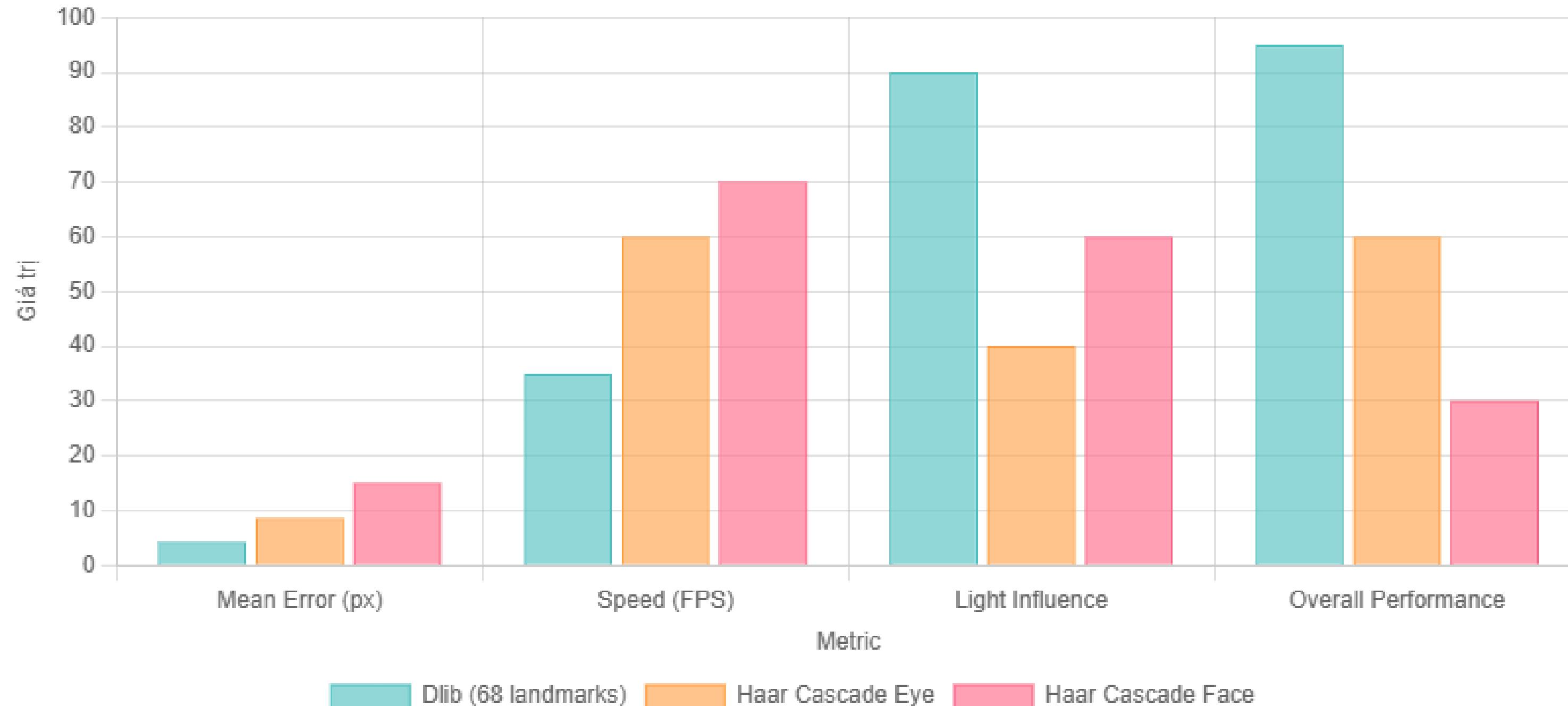
1. Kết quả nhận diện mắt

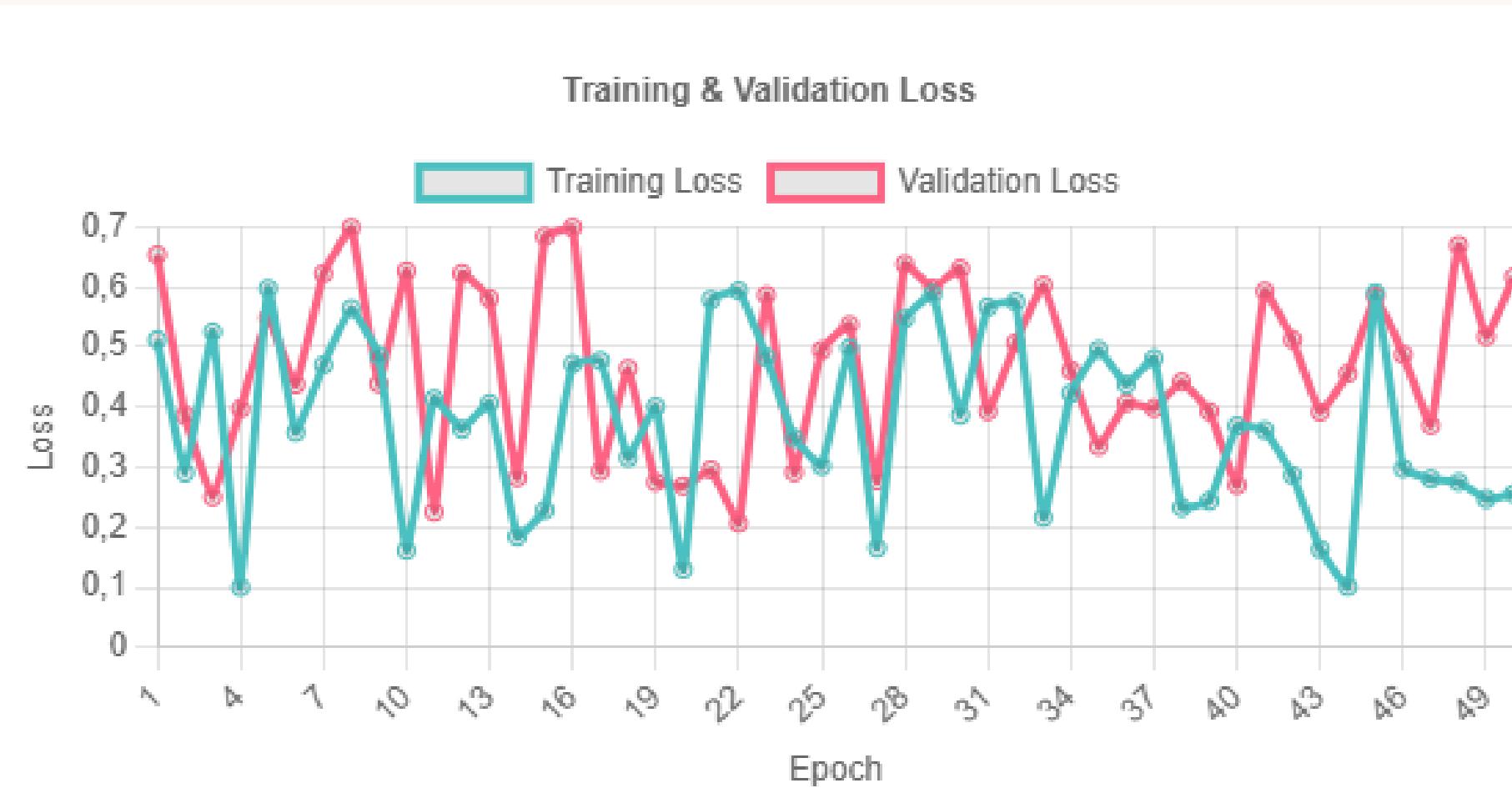
Model	Dlib (68 landmarks)	Haar Cascade Eye	Haar Cascade Face	Observation Notes
Mean Error (ME)	4.2 pixels	8.5 pixels	Not suitable	<ul style="list-style-type: none">- Dlib: Stable even when wearing glasses- Haar Eye: Frequently jumps points when the user moves quickly- Haar Face: Tracking points often drift
Standard Deviation (STD)	1.2 pixels	Unstable	Not suitable	<ul style="list-style-type: none">- Dlib: Tracking points are almost fixed- Haar Eye: Jumps points when blinking- Haar Face: Not accurate enough for eye tracking
Real-time Speed	30-40 FPS	~60 FPS	~70 FPS	<ul style="list-style-type: none">- Dlib: Slight lag on low-end machines- Haar: Smooth on most devices
Light Influence	Low	High	Medium	<ul style="list-style-type: none">- Dlib: Works well under varying lighting conditions- Haar Eye: Easily loses track in low light- Haar Face: Requires good contrast
Performance	Best	Average	Not suitable	<ul style="list-style-type: none">- Dlib: CPU usage about 20% higher- Haar: Low CPU usage but poor accuracy- Face: Does not meet eye tracking requirements

Kết quả của tập test

1.Kết quả Nhận diện mắt

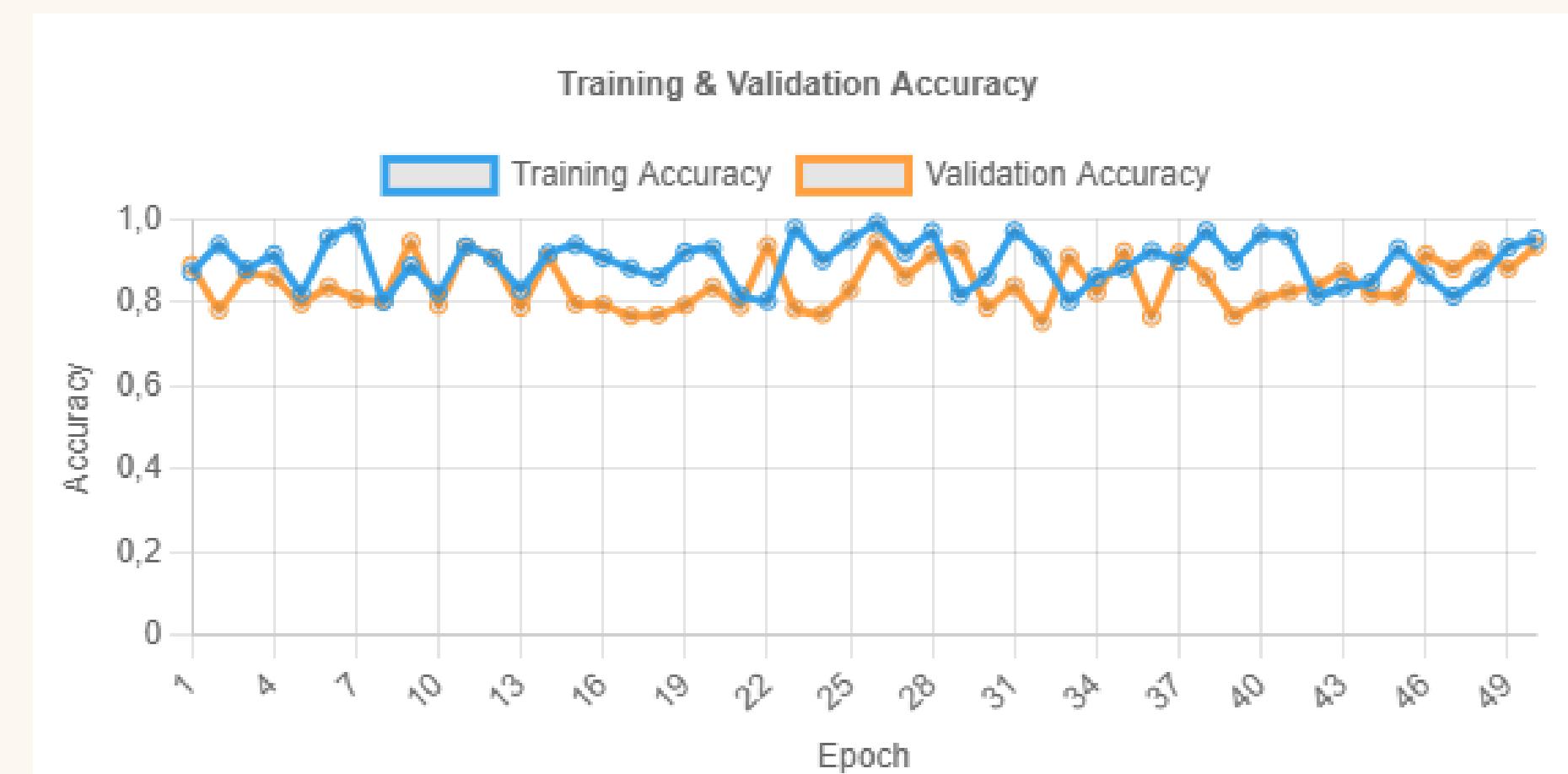
So sánh các metric giữa các model Eye Detection





Kết quả của tập test

1. Kết quả Nhận diện mắt



Kết quả nhận diện với ô vuông theo dõi trạng thái mắt

Position-Specific Accuracy

Position	Dlib	Haar Cascade	MediaPipe
Center gaze	99.5%	85.2%	97.8%
Right gaze	98.8%	82.5%	96.5%
Left gaze	98.5%	81.8%	96.2%
Blinking	99.2%	88.5%	97.5%
Eyes closed	99.8%	90.2%	98.0%

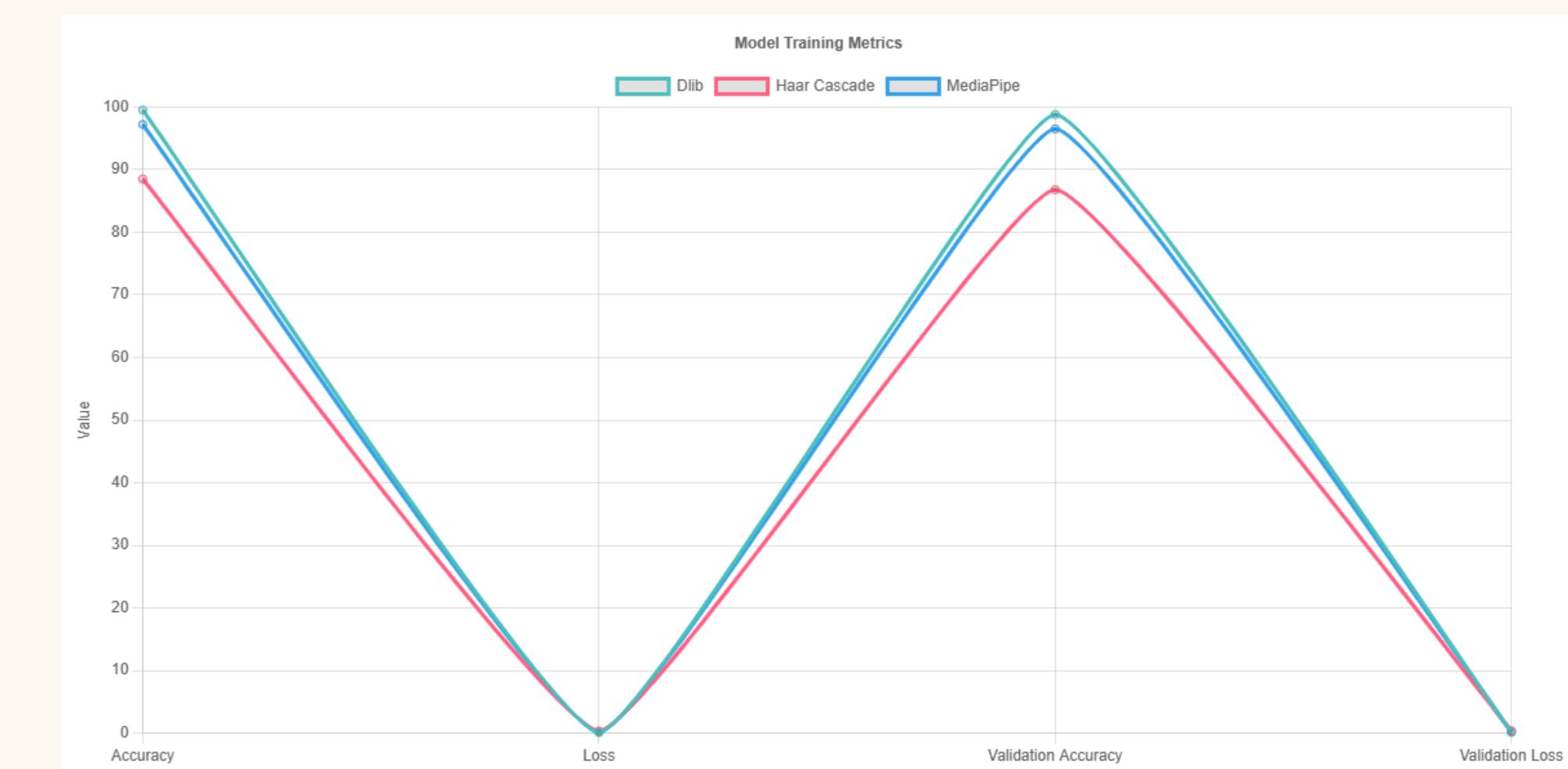
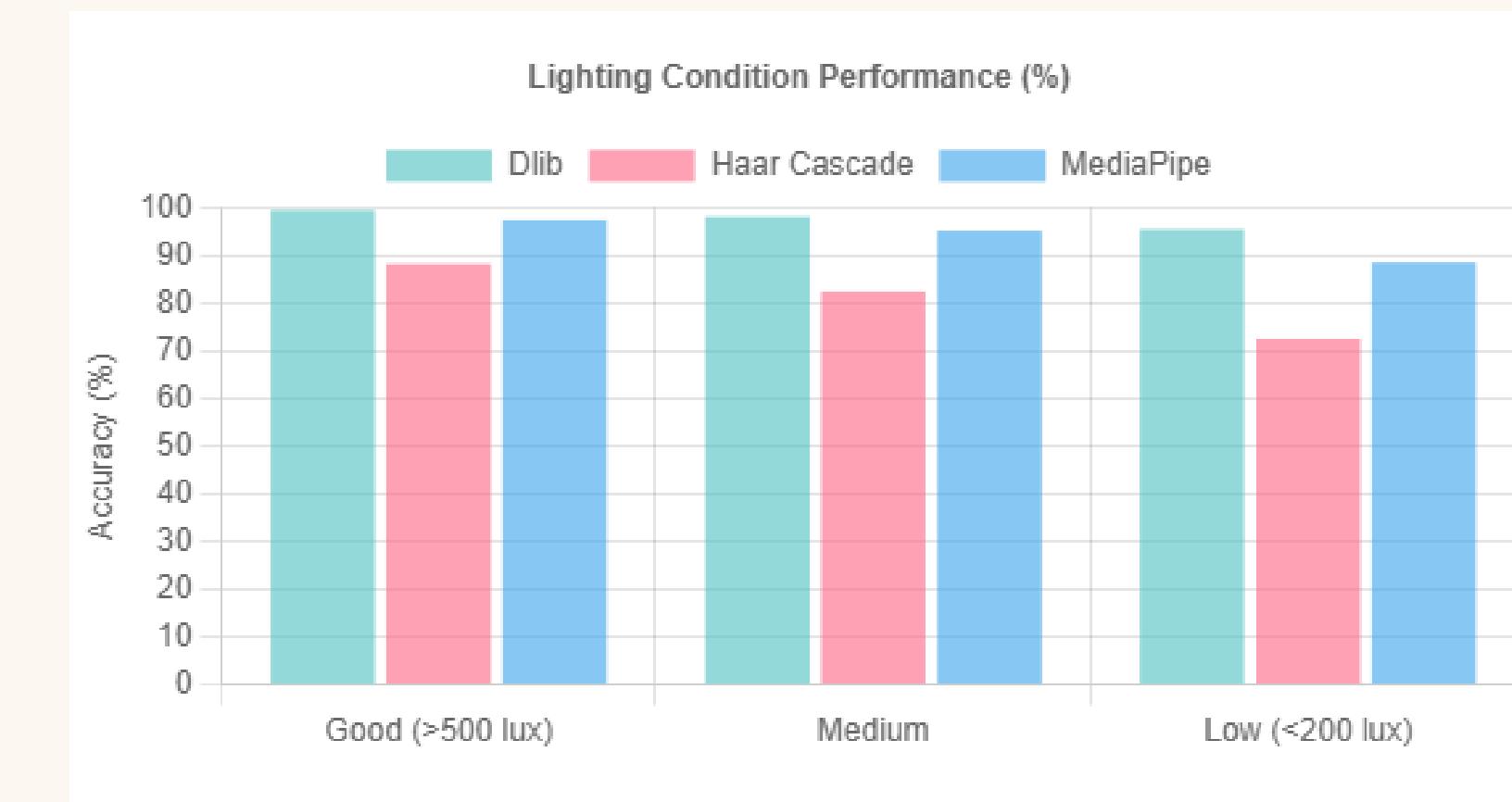
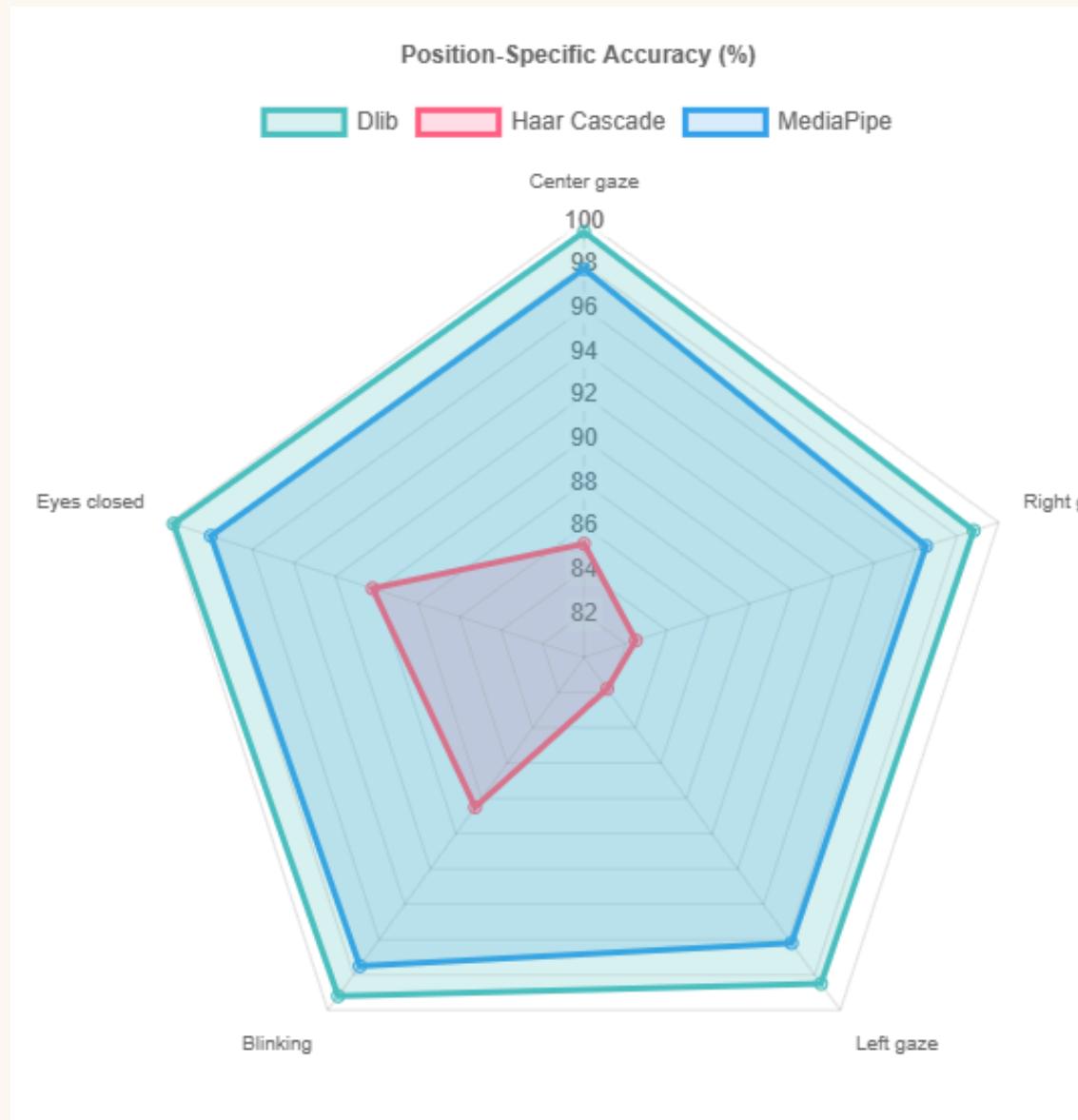
Lighting Condition Performance

Light Level	Dlib	Haar Cascade	MediaPipe
Good (>500 lux)	99.5%	88.2%	97.5%
Medium	98.2%	82.5%	95.2%
Low (<200 lux)	95.5%	72.5%	88.5%

Model Training Metrics

Model	Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
Dlib	50	99.5%	0.12	98.8%	0.15
Haar Cascade	50	88.5%	0.35	86.8%	0.38
MediaPipe	50	97.2%	0.18	96.5%	0.20

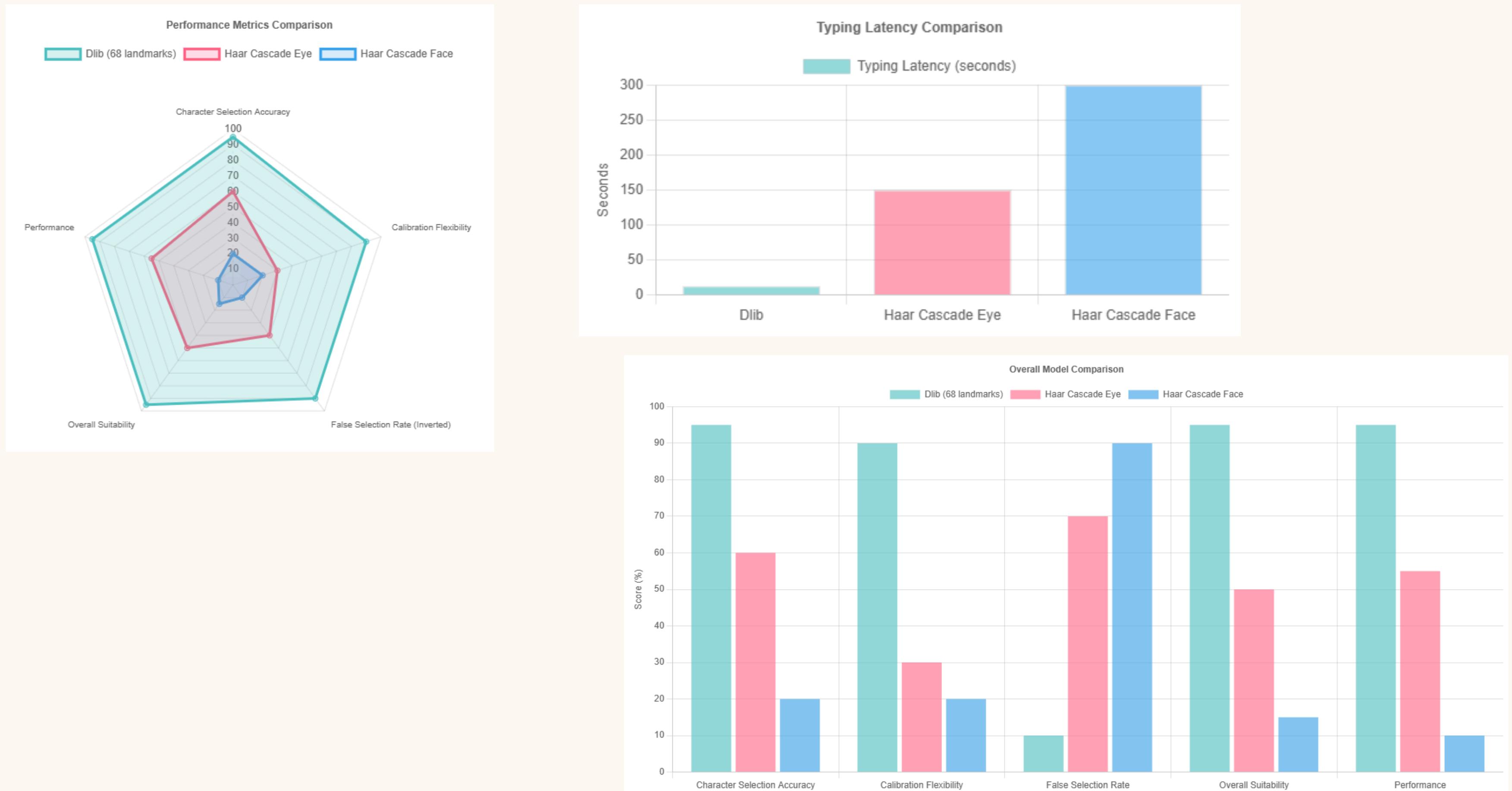
Kết quả nhận diện với ô vuông theo dõi trạng thái mắt



Kết quả của mắt đối với bàn phím

Model	Dlib (68 landmarks)	Haar Cascade Eye	Haar Cascade Face	Observation Notes
Accuracy for Character Selection	High (Stable, few errors)	Average (Prone to false selections)	Not suitable for accurate input	<ul style="list-style-type: none"> - Dlib: Best choice for eye interaction with virtual keyboard - Haar Eye: Struggles to accurately select characters, especially with fast eye movement
Calibration Flexibility	High (Can adapt to different users)	Low (Difficult to calibrate)	Low (Difficult to adapt to different faces)	<ul style="list-style-type: none"> - Dlib: Highly customizable for different users - Haar Eye: Difficult to adjust and calibrate for different users
Typing Latency	10-15 seconds	2-3 minutes	Not inputting	<ul style="list-style-type: none"> - Dlib: Typing latency around 10-15 seconds, can be felt as a slight delay on low-end machines - Haar Eye: Very high latency, 2-3 minutes, causing user frustration - Haar Face: Cannot input, unsuitable for eye tracking applications
False Selection Rate	Low	High	Very High	<ul style="list-style-type: none"> - Dlib: Low error rate, selects correct character - Haar Eye: Prone to false selections, especially with lighting changes - Haar Face: High false selection rate, not suitable for virtual keyboard
Overall Suitability	Ideal for eye tracking applications	Suitable for basic use in good lighting	Not suitable for eye tracking	
Performance	Best	Average	Not suitable	

Kết quả của mắt đối với bàn phím



Kết quả cuối cùng

Metric	Dlib (68 landmarks) landmarks
Mean Error (ME)	4.2 pixels
Standard Deviation (STD)	1.2 pixels
Real-time Speed (FPS)	30-40 FPS
Light Influence	Low, performs well under various lighting conditions
Accuracy for Character Selection	High, stable, few errors
Calibration Flexibility	High, easily adaptable for individual users
Typing Latency	10-15 seconds, slight delay on low-end devices
False Selection Rate	Low
User Experience	Excellent, accurate and stable
Overall Performance	Best
Overall Suitability	Ideal for precise eye-tracking applications

Dlib (68 điểm mốc) là lựa chọn tối ưu với độ chính xác cao, tỷ lệ lỗi thấp và hiệu suất ổn định. Cụ thể, độ trễ gõ của nó chỉ từ 10-15 giây, phù hợp cho các ứng dụng chính xác và hiệu quả.

Độ chính xác và
ổn định



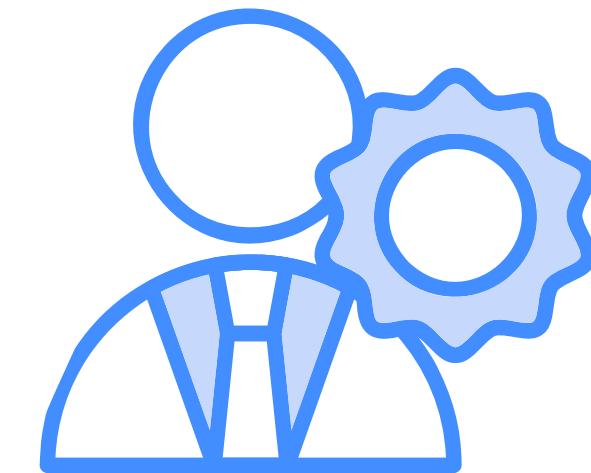
Hiệu suất



Sử dụng tài
nguyên



Kháng nhiễu

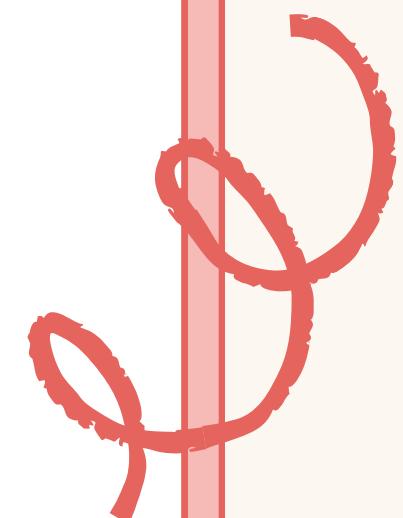


`shape_predictor_68_face_landmarks` là model tốt nhất



[Quay lại](#) [Trang Mục lục](#)

Kết luận & Hướng phát triển





Kết quả đạt được

Hạn chế của đề tài

Hướng phát triển

Kết quả đạt được

- Ứng dụng bàn phím ảo điều khiển bằng mắt
- Phản hồi trung bình và chính xác
- Giao diện thân thiện, dễ sử dụng

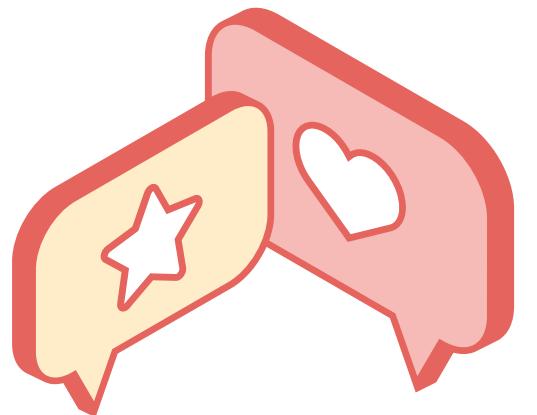
Hạn chế

- Độ chính xác phụ thuộc điều kiện ánh sáng và thiết bị
- Chưa thực hiện được trên web và ứng dụng

Hướng phát triển

- Nghiên cứu thêm để triển khai trên web và ứng dụng máy tính
- Nghiên cứu giải pháp giảm chi phí sản xuất
- Phát triển tương thích đa nền tảng
- Mở rộng thử nghiệm và cải tiến theo phản hồi người dùng





Thank You!

Cảm ơn thầy và các bạn đã theo dõi!

