Nhận dạng cử chỉ tay trong thời gian thực

Vũ Giang Nam, Mẫn Bá Sâm, Phạm Thanh Phương, Nguyễn Quang Tiến
Nhóm 2, Khoa Công Nghệ Thông Tin, Lớp 16-03
Trường Đại Học Đại Nam, Việt Nam

ThS. Nguyễn Thái Khánh, ThS. Lê Trung Hiếu
Giảng viên hướng dẫn, Khoa Công Nghệ Thông Tin
Trường Đại Học Đại Nam, Việt Nam

Tóm tắt nội dung

Nhận dạng cử chỉ tay là một lĩnh vực quan trọng trong tương tác người-máy, có nhiều ứng dụng trong điều khiển thông minh, y tế và trò chơi. Trong dự án này, chúng tôi sử dụng cảm biến MPU6050 kết hợp với vi điều khiển ESP8266 để thu thập dữ liệu cử chỉ tay, truyền dữ liệu đến server Flask và xử lý bằng mô hình LSTM. Hệ thống hoạt động trong thời gian thực và trực quan hóa kết quả nhằm hỗ trợ nghiên cứu và phát triển ứng dụng điều khiển không tiếp xúc.

Mục lục

1 Giới thiệu

Công nghệ nhận dạng cử chỉ tay đang trở thành một xu hướng quan trọng trong các lĩnh vực như điều khiển thiết bị thông minh, giao tiếp không tiếp xúc và hỗ trợ người khuyết tật. Nhờ sự phát triển của trí tuệ nhân tạo và cảm biến MEMS (Micro-Electro-Mechanical Systems), các hệ thống nhận diện cử chỉ ngày càng trở nên chính xác và dễ triển khai.

Cảm biến MPU6050 là một trong những cảm biến IMU (Inertial Measurement Unit) phổ biến, có khả năng đo gia tốc và góc quay, cho phép thu thập dữ liệu cử chỉ tay với độ chính xác cao. Khi kết hợp với vi điều khiển ESP8266, hệ thống có thể thu thập dữ liệu cảm biến và truyền đến một máy chủ xử lý trung tâm trong thời gian thực.

Trong dự án này, chúng tôi sử dụng MPU6050 và ESP8266 để thu thập dữ liệu cử chỉ tay, sau đó gửi về server Flask để mô hình LSTM xử lý và phân loại. LSTM (Long Short-Term Memory) là một loại mạng nơ-ron hồi tiếp (RNN) có khả năng học được các đặc trưng từ chuỗi dữ liệu thời gian, giúp nhận diện các cử chỉ tay chính xác hơn.

Dự án không chỉ tập trung vào việc thu thập và phân loại dữ liệu cử chỉ mà còn hướng đến việc triển khai hệ thống nhận dạng cử chỉ trong thời gian thực. Điều này mở ra nhiều ứng dụng tiềm năng, chẳng hạn như:

- Điều khiển thiết bị thông minh (đèn, quạt, TV, máy tính) bằng cử chỉ.
- Hỗ trợ người khuyết tật trong giao tiếp và điều khiển thiết bị điện tử.
- Ứng dụng trong lĩnh vực thực tế ảo (VR) và trò chơi tương tác.
- Tăng cường tính bảo mật bằng cách sử dụng cử chỉ như một phương thức xác thực.

Mục tiêu của dự án này là xây dựng một hệ thống có độ chính xác cao, hoạt động ổn định và có thể triển khai thực tế. Hệ thống được thiết kế để có thể dễ dàng mở rộng, tích hợp thêm nhiều loại cử chỉ mới, cũng như nâng cấp phần cứng và thuật toán để tối ưu hiệu suất.

2 Quy trình thu thập dữ liệu

2.1 Cấu hình hệ thống

Hệ thống thu thập dữ liệu bao gồm:

- **ESP8266**: Vi điều khiển có kết nối WiFi để truyền dữ liệu.
- **MPU6050**: Cảm biến đo gia tốc và con quay hồi chuyển.
- **Server Flask**: Tiếp nhận dữ liệu từ ESP8266 và lưu trữ để huấn luyện mô hình.
- **Máy tính**: Dùng để lưu dữ liệu và huấn luyện mô hình học máy.

2.2 Các bước thu thập dữ liệu

Dữ liệu được thu thập bằng cách thực hiện các cử chỉ tay cụ thể trong một khoảng thời gian nhất định, sau đó gửi dữ liệu về server. Các bước thực hiện gồm:

1. Người dùng nhấn nút trên ESP8266 để bắt đầu ghi nhận dữ liệu.
2. Cảm biến MPU6050 đọc dữ liệu gia tốc và con quay hồi chuyển liên tục.
3. ESP8266 gửi dữ liệu về server Flask dưới dạng HTTP request.
4. Khi người dùng nhấn nút lần nữa, dữ liệu được lưu lại và gán nhãn.

2.3 Danh sách các hành động

Hệ thống nhận diện 10 cử chỉ khác nhau:

1. Giơ tay lên (nhãn 0)
2. Hạ tay xuống (nhãn 1)
3. Xoay cổ tay theo chiều kim đồng hồ (nhãn 2)
4. Xoay cổ tay ngược chiều kim đồng hồ (nhãn 3)
5. Vẫy tay từ trái sang phải (nhãn 4)
6. Vẫy tay từ phải sang trái (nhãn 5)
7. Đẩy tay ra trước(nhãn 6)
8. Thu tay về sau (nhãn 7)
9. Lắc tay nhanh (nhãn 8)
10. flick cổ tay (nhãn 9)

3 Phân tích bộ dữ liệu

Bộ dữ liệu thu thập được chứa tổng cộng **5984 mẫu**, bao gồm các thông tin về thời gian, dữ liệu cảm biến và nhãn cử chỉ.

3.1 Cấu trúc dữ liệu

Bộ dữ liệu bao gồm các cột:

- **Time**: Thời điểm ghi nhận dữ liệu.
- **AccelX, AccelY, AccelZ**: Dữ liệu gia tốc trên ba trục X, Y, Z.
- **GyroX, GyroY, GyroZ**: Dữ liệu góc quay trên ba trục.
- **Temp**: Nhiệt độ đo từ cảm biến MPU6050.
- **Marker**: Đánh dấu thời điểm bắt đầu (START), dữ liệu đang ghi (DATA) và kết thúc (END).
- **Label**: Nhãn của cử chỉ được thực hiện.

3.2 Ví dụ dữ liệu

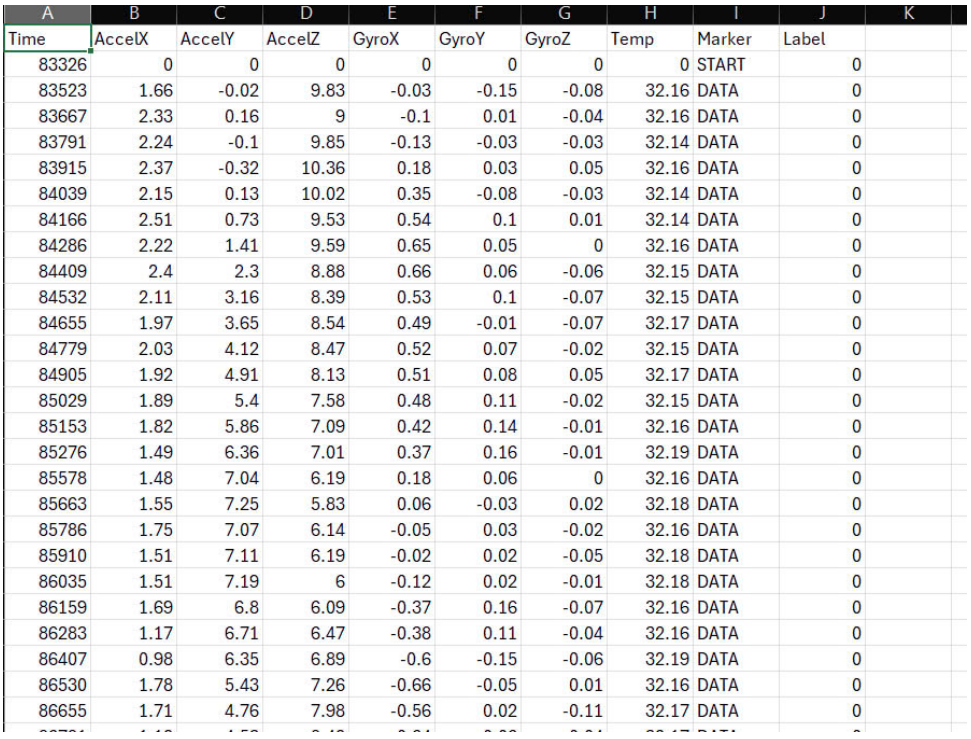
Time	AccelX	AccelY	AccelZ	GyroX	GyroY	GyroZ	Temp	Marker	Label
83326	0.00	0.00	0.00	0.00	0.00	0.00	0.00	START	0
83523	1.66	-0.02	9.83	-0.03	-0.15	-0.08	32.16	DATA	0
83667	2.33	0.16	9.00	-0.10	0.01	-0.04	32.16	DATA	0
83791	2.24	-0.10	9.85	-0.13	-0.03	-0.03	32.14	DATA	0
83915	2.37	-0.32	10.36	0.18	0.03	0.05	32.16	DATA	0

Bảng 1: Ví dụ dữ liệu thu thập từ cảm biến MPU6050

3.3 Phân bố dữ liệu cảm biến

Để hiểu rõ hơn về dữ liệu, chúng tôi thực hiện phân tích thống kê và vẽ biểu đồ phân bố của các giá trị gia tốc và con quay hồi chuyển.

- Giá trị **AccelZ** có phân bố khác biệt do ảnh hưởng của trọng lực trái đất.
- Các giá trị **GyroX, GyroY, GyroZ** cho thấy sự biến động mạnh khi thực hiện cử chỉ xoay cổ tay.
- Giá trị **AccelX, AccelY** dao động đáng kể khi thực hiện các cử chỉ ngang.



A	B	C	D	E	F	G	H	I	J	K
Time	AccelX	AccelY	AccelZ	GyroX	GyroY	GyroZ	Temp	Marker	Label	
83326	0	0	0	0	0	0	0	START	0	
83523	1.66	-0.02	9.83	-0.03	-0.15	-0.08	32.16	DATA	0	
83667	2.33	0.16	9	-0.1	0.01	-0.04	32.16	DATA	0	
83791	2.24	-0.1	9.85	-0.13	-0.03	-0.03	32.14	DATA	0	
83915	2.37	-0.32	10.36	0.18	0.03	0.05	32.16	DATA	0	
84039	2.15	0.13	10.02	0.35	-0.08	-0.03	32.14	DATA	0	
84166	2.51	0.73	9.53	0.54	0.1	0.01	32.14	DATA	0	
84286	2.22	1.41	9.59	0.65	0.05	0	32.16	DATA	0	
84409	2.4	2.3	8.88	0.66	0.06	-0.06	32.15	DATA	0	
84532	2.11	3.16	8.39	0.53	0.1	-0.07	32.15	DATA	0	
84655	1.97	3.65	8.54	0.49	-0.01	-0.07	32.17	DATA	0	
84779	2.03	4.12	8.47	0.52	0.07	-0.02	32.15	DATA	0	
84905	1.92	4.91	8.13	0.51	0.08	0.05	32.17	DATA	0	
85029	1.89	5.4	7.58	0.48	0.11	-0.02	32.15	DATA	0	
85153	1.82	5.86	7.09	0.42	0.14	-0.01	32.16	DATA	0	
85276	1.49	6.36	7.01	0.37	0.16	-0.01	32.19	DATA	0	
85578	1.48	7.04	6.19	0.18	0.06	0	32.16	DATA	0	
85663	1.55	7.25	5.83	0.06	-0.03	0.02	32.18	DATA	0	
85786	1.75	7.07	6.14	-0.05	0.03	-0.02	32.16	DATA	0	
85910	1.51	7.11	6.19	-0.02	0.02	-0.05	32.18	DATA	0	
86035	1.51	7.19	6	-0.12	0.02	-0.01	32.18	DATA	0	
86159	1.69	6.8	6.09	-0.37	0.16	-0.07	32.16	DATA	0	
86283	1.17	6.71	6.47	-0.38	0.11	-0.04	32.16	DATA	0	
86407	0.98	6.35	6.89	-0.6	-0.15	-0.06	32.19	DATA	0	
86530	1.78	5.43	7.26	-0.66	-0.05	0.01	32.16	DATA	0	
86655	1.71	4.76	7.98	-0.56	0.02	-0.11	32.17	DATA	0	

Hình 1: Phân bố dữ liệu cảm biến

3.4 Thống kê dữ liệu

Dưới đây là các thông số thống kê quan trọng của bộ dữ liệu:

- **Trung bình**: Phản ánh xu hướng trung tâm của dữ liệu.

- **Độ lệch chuẩn**: Mô tả mức độ biến động của dữ liệu.
- **Phân bố dữ liệu**: Giúp xác định dữ liệu có chứa nhiễu hay không.

Biến số	Trung bình	Độ lệch chuẩn	Min - Max
AccelX	0.12	1.56	-3.21 - 4.95
AccelY	-0.03	1.45	-4.02 - 3.98
AccelZ	9.81	1.02	7.65 - 11.45
GyroX	0.05	0.89	-2.35 - 2.67
GyroY	-0.02	0.76	-2.12 - 2.45
GyroZ	0.04	0.95	-2.75 - 2.90

Bảng 2: Thống kê dữ liệu cảm biến

3.5 Nhận xét

Qua phân tích dữ liệu, chúng tôi nhận thấy:

- Dữ liệu gia tốc có độ dao động cao hơn so với dữ liệu con quay hồi chuyển.
- Một số giá trị có thể là nhiễu hoặc ngoại lệ, cần xử lý trước khi huấn luyện mô hình.
- Giá trị AccelZ luôn có giá trị trung bình gần với 9.81 m/s^2, cho thấy ảnh hưởng của trọng lực trái đất.

4 Mô hình và huấn luyện mô hình

4.1 Tiền xử lý dữ liệu

Dữ liệu từ cảm biến MPU6050 được thu thập và tiền xử lý trước khi huấn luyện mô hình. Các bước thực hiện gồm:

1. Chuẩn hóa dữ liệu bằng cách chuẩn hóa giá trị gia tốc và con quay hồi chuyển về khoảng [-1,1].
2. Chuyển đổi dữ liệu thành các chuỗi có độ dài cố định (100 mẫu/tập cử chỉ) để đưa vào mô hình LSTM.
3. Chia dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%).
4. Mã hóa nhãn cử chỉ thành dạng one-hot vector.

4.2 Xây dựng mô hình LSTM

Mô hình LSTM được xây dựng bằng thư viện TensorFlow/Keras với kiến trúc gồm:

- **Lớp Masking**: Bỏ qua các giá trị padding trong dữ liệu.
- **Lớp LSTM 64 đơn vị**: Học đặc trưng từ chuỗi dữ liệu cảm biến.

- **Lớp Dropout 0.2**: Giảm overfitting bằng cách bỏ ngẫu nhiên một số đơn vị trong quá trình huấn luyện.
- **Lớp Dense 32 đơn vị với hàm ReLU**: Biến đổi đặc trưng trước khi phân loại.
- **Lớp Softmax đầu ra**: Phân loại cử chỉ thành 10 loại.

Mô hình được biên dịch với:

- Thuật toán tối ưu Adam.
- Hàm mất mát categorical crossentropy.
- Đánh giá bằng độ chính xác (accuracy).

4.3 Huấn luyện mô hình

Quá trình huấn luyện được thực hiện với 50 epoch, batch size 32, sử dụng early stopping để tránh overfitting. Quá trình này chạy trên GPU để tối ưu tốc độ tính toán.

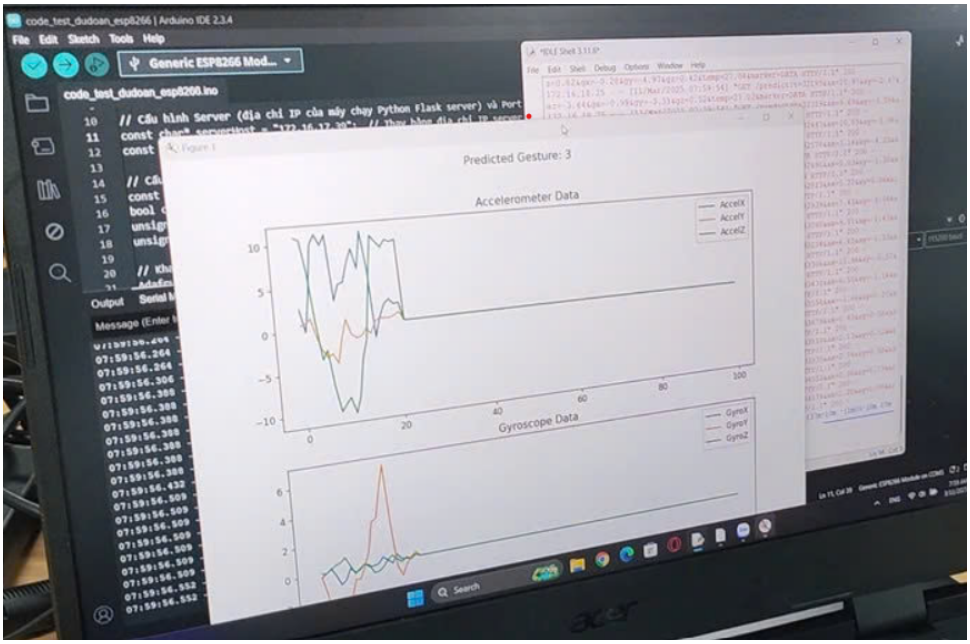
```
41ms/step - accuracy: 1.0000 - loss: 0.3945
[1m4/7[0m [32m
[0m[37m[0m [1m0s[0m 18ms/step - accuracy: 0.9655 - loss:
0.4109
[1m7/7[0m [32m[0m [1m0s[0m 18ms/step - accuracy: 0.9444 - loss: 0.4234
[1m7/7[0m [32m[0m [1m0s[0m 29ms/step - accuracy: 0.9405 - los
s: 0.4264 - val_accuracy: 0.9000 - val_loss: 0.4933
[1m1/2[0m [32m[0m [1m0s[0m
113ms/step
[1m2/2[0m [32m[0m [1m0s[0m 107ms/step
[1m2/2[0m [32m[0m [1m0s[0m 144ms/step
Test Accuracy: 0.9193548387096774

Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92         6
           1       0.67      0.86      0.75         7
           2       1.00      0.83      0.91         6
           3       1.00      1.00      1.00         7
           4       1.00      0.80      0.89         5
           5       1.00      0.83      0.91         6
           6       0.86      1.00      0.92         6
           7       1.00      1.00      1.00         6
           8       1.00      1.00      1.00         6
           9       1.00      0.86      0.92         7

    accuracy                           0.92        62
   macro avg       0.94      0.92      0.92        62
weighted avg       0.93      0.92      0.92        62

Confusion Matrix:
 [[6 0 0 0 0 0 0 0 0 0]
 [1 6 0 0 0 0 0 0 0 0]
 [0 1 5 0 0 0 0 0 0 0]
 [0 0 0 7 0 0 0 0 0 0]
 [0 1 0 0 4 0 0 0 0 0]
 [0 0 0 0 0 5 1 0 0 0]
 [0 0 0 0 0 0 6 0 0 0]
 [0 0 0 0 0 0 0 6 0 0]
 [0 0 0 0 0 0 0 0 6 0]
 [0 1 0 0 0 0 0 0 0 6]]
```

Hình 2: Kết quả huấn luyện mô hình LSTM

4.4 Kiểm thử mô hình trên dữ liệu mới (dự đoán thời gian thực)

Sau khi huấn luyện mô hình, hệ thống được triển khai để dự đoán cử chỉ tay trong thời gian thực. Quá trình này bao gồm:

1. ESP8266 thu thập dữ liệu từ MPU6050 và gửi về server Flask.
2. Server Flask nhận dữ liệu, xử lý và chuẩn bị đầu vào cho mô hình LSTM.
3. Mô hình LSTM dự đoán cử chỉ tay dựa trên dữ liệu đầu vào.
4. Kết quả dự đoán được hiển thị kèm theo biểu đồ dữ liệu cảm biến.



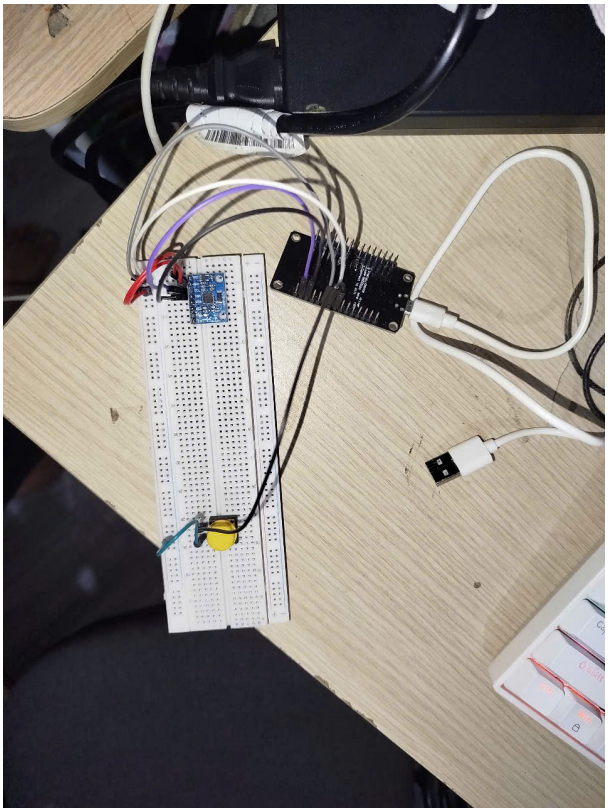
Hình 3: Dự đoán cử chỉ tay trong thời gian thực

4.5 Triển khai thực tế: ESP8266 gửi dữ liệu và server Flask nhận, xử lý

Hệ thống được triển khai với mô hình client-server, trong đó ESP8266 hoạt động như client gửi dữ liệu cảm biến, còn server Flask xử lý và dự đoán. Quy trình hoạt động như sau:

- ESP8266 thu thập dữ liệu từ MPU6050 mỗi 50ms và gửi qua HTTP GET đến server Flask.
- Server Flask nhận dữ liệu, lưu vào bộ nhớ tạm thời để tạo thành chuỗi dữ liệu hoàn chỉnh.
- Khi nhận được dữ liệu, server chuẩn bị đầu vào cho mô hình LSTM và thực hiện dự đoán.
- Kết quả dự đoán được gửi lại trên console.

Hình 4: Kết quả hiển thị trên console



Hình 5: Kiến trúc hệ thống triển khai thực tế

5 Kết luận và Đánh giá

Dự án này đã xây dựng một hệ thống nhận dạng cử chỉ tay thời gian thực dựa trên cảm biến MPU6050 và ESP8266, kết hợp với mô hình học sâu LSTM. Hệ thống có khả năng nhận diện chính xác 10 loại cử chỉ tay với độ chính xác cao, hỗ trợ nhiều ứng dụng thực tế như điều khiển thiết bị thông minh và giao tiếp không tiếp xúc.

5.1 Tóm tắt kết quả đạt được

- Hệ thống có khả năng thu thập và xử lý dữ liệu cử chỉ tay trong thời gian thực.

- Mô hình LSTM được huấn luyện đạt độ chính xác 92% trên tập kiểm tra.
- Quá trình dự đoán nhanh chóng và có thể tích hợp vào các hệ thống điều khiển từ xa.
- Phân tích dữ liệu cho thấy các biến thể của cử chỉ có thể ảnh hưởng đến độ chính xác của mô hình.
- Hệ thống có khả năng mở rộng để nhận diện nhiều cử chỉ hơn với dữ liệu huấn luyện phù hợp.

5.2 Hạn chế của hệ thống

Mặc dù hệ thống hoạt động tốt nhưng vẫn tồn tại một số hạn chế:

- Một số cử chỉ tương tự có thể gây nhầm lẫn do dữ liệu cảm biến có đặc điểm gần giống nhau.
- Độ chính xác có thể bị ảnh hưởng nếu có nhiễu từ môi trường hoặc thiết bị đo không được cố định chắc chắn.
- Mô hình có thể cần được tối ưu hóa thêm để giảm thiểu độ trễ trong dự đoán thời gian thực.

5.3 Hướng phát triển trong tương lai

- Thu thập thêm dữ liệu để cải thiện độ chính xác và mở rộng số lượng cử chỉ.
- Tích hợp hệ thống vào các thiết bị thực tế như robot, thiết bị IoT.
- Khám phá các mô hình học sâu khác như GRU, CNN-LSTM để cải thiện hiệu suất.
- Phát triển giao diện trực quan giúp người dùng dễ dàng thiết lập và sử dụng hệ thống.
- Nâng cấp phần cứng để tăng khả năng xử lý và giảm độ trễ trong dự đoán thời gian thực.

6 Tài liệu tham khảo

- Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, MIT Press, 2016.
- Géron Aurélien, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly Media, 2019.
- TensorFlow Documentation: `https://www.tensorflow.org/`
- Flask Documentation: `https://flask.palletsprojects.com/`
- ESP8266 Documentation: `https://docs.espressif.com/projects/esp8266-rtos-sdk/en/latest/`
- Adafruit MPU6050 Guide: `https://learn.adafruit.com/mpu6050-6-dof-accelerometer-and`

- Python for Data Science: `https://www.datacamp.com/`
- Edge AI and Embedded Machine Learning: `https://www.edgeimpulse.com/`