

DẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
DẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH



ĐỒ ÁN TỐT NGHIỆP

Đề tài

THIẾT BỊ CHUYỂN ĐỔI NGÔN NGỮ KÝ HIỆU SANG
GIỌNG NÓI SỬ DỤNG FLEX SENSORS KẾT HỢP VỚI
HỌC MÁY

GVHD: ThS. Trần Thanh Bình
ThS. Lê Trọng Nhân
SVTH: Lê Nhật Tiên - 1912196
Nguyễn Duy Hòa - 2010276

TP. Hồ Chí Minh, tháng 12 năm 2024



Lời cam đoan

Chúng tôi xin cam đoan rằng những nội dung trình bày trong báo cáo luận văn này là công trình nghiên cứu của nhóm dưới sự hướng dẫn của thầy Trần Thanh Bình và thầy Lê Trọng Nhân. Nội dung và số liệu trong báo cáo không phải là bản sao chép từ bất kì báo cáo, tiểu luận nào có trước. Tất cả những sự giúp đỡ cho việc xây dựng bài báo cáo đều được trích dẫn và ghi nguồn đầy đủ, rõ ràng. Nếu không đúng sự thật, chúng tôi chịu mọi trách nhiệm trước các thầy, cô, và nhà trường.



Lời cảm ơn

Chúng tôi xin gửi lời cảm ơn sâu sắc nhất đến thầy Trần Thanh Bình và thầy Lê Trọng Nhân, những người đã tận tâm hướng dẫn và đồng hành cùng chúng tôi trong suốt quá trình thực hiện đề tài. Hai thầy không chỉ theo sát tiến độ mà còn nhiệt tình góp ý, chỉnh sửa những sai sót, giúp chúng tôi cải thiện và hoàn thiện công việc một cách hiệu quả hơn.

Sau một học kỳ miệt mài thực hiện, bên cạnh sự nỗ lực của từng cá nhân, sự chỉ dẫn tận tình của hai thầy là yếu tố quan trọng giúp chúng em hoàn thành đồ án đúng tiến độ và nâng cao chất lượng đề tài.

Dù đã rất cố gắng, nhưng chắc chắn bài làm vẫn không tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được sự góp ý, đóng góp quý báu từ quý thầy cô và các bạn để tiếp tục hoàn thiện hơn.

TP. Hồ Chí Minh, tháng 11 năm 2024



Tóm tắt đồ án

Đồ án tốt nghiệp Kỹ thuật Máy tính của nhóm với đề tài "Thiết bị hỗ trợ chuyển đổi ngôn ngữ ký hiệu thành giọng nói sử dụng flex sensors và học máy" được thực hiện qua hai học kỳ. Mục tiêu của đề tài là phát triển một hệ thống nhận diện cử chỉ thông qua công nghệ học máy và flex sensors, sau đó chuyển đổi cử chỉ thành giọng nói.

Cụ thể, nhóm sử dụng cảm biến uốn cong để theo dõi và đo lường sự chuyển động của các ngón tay. Dữ liệu từ cảm biến này được kết hợp với các cảm biến phụ trợ khác như cảm biến con quay gia tốc, từ đó phân tích và xây dựng một mô hình học máy dựa trên Long Short Term Memory (LSTM).

Nhóm đã tiến hành thu thập dữ liệu từ các cử chỉ thực tế và huấn luyện mô hình với tập dữ liệu này. Trong quá trình thực hiện, nhóm đã tinh chỉnh mô hình và cải tiến thiết kế hệ thống bằng các kỹ thuật nâng cao, giúp tăng độ chính xác của hệ thống trong việc nhận diện cử chỉ và chuyển đổi thành giọng nói một cách hiệu quả.



Mục lục

Lời cam đoan	I
Lời cảm ơn	II
Tóm tắt đồ án	III
Danh sách hình ảnh	VII
Danh sách bảng	IX
1 Giới thiệu đề tài	1
1.1 Giới thiệu	1
1.2 Mục tiêu	1
1.3 Phạm vi của đề tài	1
1.4 Ý nghĩa thực tiễn	1
2 Cơ sở lý thuyết	3
2.1 Giới thiệu về học máy(Machine learning)	3
2.2 Giới thiệu về Mạng nơ-ron hồi quy(RNN)	3
2.2.1 Tổng quan	3
2.2.2 Một số loại mạng nơ-ron hồi quy	4
2.2.3 Vấn đề phụ thuộc xa của RNN	4
2.3 Giới thiệu mô hình Long-short term memmory(LSTM)	6
2.3.1 Giới thiệu tổng quan	6
2.3.2 Ý tưởng đằng sau LSTM	6
2.3.3 Thứ tự các bước của LSTM	7
2.3.4 Ưu và nhược điểm của mô hình LSTM	9
2.4 Giới thiệu về Transfer Learning	10
2.4.1 Định nghĩa Transfer Learning	10
2.4.2 Ứng dụng của Transfer Learning	11
2.4.3 Lợi ích và bất lợi của Transfer Learning	12
2.5 Giới thiệu về Self-Attention	12
2.5.1 Cơ chế hoạt động của Self-Attention	12
2.5.2 Các loại Self-Attention thường gặp	13
2.5.3 Ứng dụng của Self-Attention	13
2.5.4 Lợi ích của Self-Attention	13
2.6 Generative Adversarial Networks (GANs)	13
2.6.1 Kiến trúc và hoạt động của GANs	13
2.6.2 Các biến thể của GANs	14
2.6.3 Ứng dụng của GANs	14
2.6.4 Lợi ích và hạn chế của GANs	14
2.7 Giới thiệu về thư viện Tensorflow	14
2.7.1 Nguyên lý hoạt động	15
2.7.2 Thuộc tính cơ bản trong Tensorflow	16
2.7.3 Ưu điểm của TensorFlow	18
2.8 Giới thiệu về Arduino LilyPad	19
2.9 Giới thiệu về Flex sensor	21



2.9.1	Tổng quan	21
2.9.2	Nguyên lý hoạt động	22
2.9.3	Đọc giá trị	22
2.10	Giới thiệu về cảm biến con quay gia tốc	23
2.10.1	Tổng quan	23
2.10.2	Nguyên lý hoạt động	24
2.10.2.a	Hiệu ứng điện dung;	24
2.10.2.b	Hiệu ứng áp điện (Piezoelectric):	24
2.10.3	Giới thiệu về module cảm biến ITG 3200	27
2.11	Giới thiệu về module bluetooth HC05	28
2.12	Giới thiệu về ngôn ngữ ký hiệu	29
3	Kiến trúc hệ thống	31
3.1	Module thu thập dữ liệu cử chỉ	32
3.2	Module nhận dạng cử chỉ	33
3.3	Module phát âm thanh	34
4	Hiện thực hệ thống	35
4.1	Hiện thực phần cứng	35
4.1.1	Các thành phần phần cứng chính	35
4.1.2	Sơ đồ kết nối mạch	36
4.1.3	Nhiệm vụ của phần cứng	37
4.1.4	Tóm tắt	37
4.2	Xây dựng bộ dữ liệu	39
4.2.1	Thu thập dữ liệu từ phần cứng	39
4.2.2	Xử lý và lưu trữ dữ liệu	40
4.2.3	Tóm tắt	40
4.3	Dịnh dạng dữ liệu trước khi đưa vào học máy	42
4.3.1	Đọc và tổ chức dữ liệu	42
4.3.2	Dịnh dạng dữ liệu đầu vào	42
4.3.3	Chuẩn hóa dữ liệu	42
4.3.4	Chuẩn bị dữ liệu cho mô hình LSTM	42
4.3.5	Dịnh nghĩa đầu vào và đầu ra của mô hình	43
4.3.6	Tóm tắt	43
4.4	Xây dựng mô hình học máy	44
4.4.1	Cấu trúc của mô hình	44
4.4.2	Tổng số tham số	44
4.4.3	Quy trình xây dựng mô hình	44
4.4.4	Tóm tắt mô hình	45
4.4.5	Tóm tắt	45
5	Các phương pháp cải tiến	46
5.1	Cửa sổ trượt (Sliding Window)	46
5.1.1	Khái niệm về cửa sổ trượt	46
5.1.2	Cách thức hoạt động của cửa sổ trượt	46
5.1.3	Ứng dụng cửa sổ trượt trong dự đoán	47
5.1.4	Lợi ích của cửa sổ trượt trong dự đoán	48
5.2	Khôi phục dữ liệu bị mất	48
5.2.1	Nguyên nhân gây mất dữ liệu	48



5.2.2	Phương pháp phát hiện dữ liệu bị mất	49
5.2.3	Phương pháp khôi phục dữ liệu bị mất	49
5.2.4	Tối ưu hóa giảm thiểu mất dữ liệu	49
5.3	Cải thiện chất lượng mô hình	50
5.3.1	Mô hình với Deep Convolution và Self-Attention	50
5.4	Làm giàu dữ liệu	53
5.4.1	Data Augmentation	53
5.4.2	Transfer learning	54
5.4.3	Sinh dữ liệu từ TimeGAN	54
6	Kết quả thực nghiệm	55
6.1	Mô hình ban đầu	55
6.1.1	Thiết lập thực nghiệm	55
6.1.2	Kết quả huấn luyện và kiểm tra	55
6.1.3	Kết quả dự đoán thời gian thực	56
6.1.4	Phân tích kết quả	56
6.2	Áp dụng cửa sổ trượt (Sliding Window)	57
6.2.1	Dộ chính xác dự đoán tăng theo kích thước cửa sổ	57
6.2.2	Thời gian dự đoán giảm nhẹ khi tăng kích thước cửa sổ	58
6.2.3	Kết luận	58
6.3	Khôi phục dữ liệu bị mất	58
6.3.1	Dữ liệu mất ít dòng (dưới 10 dòng)	58
6.3.2	Dữ liệu mất nhiều dòng (trên 10 dòng)	58
6.3.3	Kết luận	59
6.4	1	59
6.5	60
7	Kết luận	61
7.1	Kết quả đạt được	61
7.2	Hạn chế của hệ thống	61
7.3	Hướng cải tiến và phát triển	61
Tài liệu tham khảo		62



Danh sách hình vẽ

1	Minh họa một mạng nơ-ron hồi quy	3
2	Một số kiểu nơ-ron hồi quy phổ biến	4
3	Phụ thuộc ngắn hạn trên RNN	5
4	Phụ thuộc dài hạn trên RNN	5
5	Mô hình LSTM với 4 lớp tương tác	6
6	Điển giải các ký hiệu trong đồ thị	6
7	Đường đi của ô trạng thái (cell state) trong mạng LSTM	7
8	Một cỗng của hàm sigmoid trong LSTM	7
9	Tầng cỗng quên (forget gate layer)	8
10	Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cỗng vào và tầng ẩn hàm tanh	8
11	Ô trạng thái mới	9
12	Điều chỉnh thông tin ở đầu ra thông qua hàm tanh	9
13	Transfer learning	11
14	Lợi ích của Transfer Learning	12
15	Self Attention	12
16	Lợi ích của Transfer Learning	13
17	Thư viện Tensorflow	15
18	Nguyên lý hoặc động của Tensorflow	16
19	Hình ảnh minh họa của một tensor và so sánh với vector và ma trận.	17
20	Minh họa cho 1 graph, session, operation và variable.	18
21	Board mạch Arduino Lilypad	20
22	Flex sensor	21
23	Hình dạng uốn cong của flex sensor	22
24	Sơ đồ nối dây của flex sensor	23
25	Cấu trúc của một cảm biến gia tốc sử dụng phương pháp điện dung	24
26	Hiệu ứng Piezoelectric	25
27	Các kiểu thiết kế cơ học của cảm biến gia tốc	26
28	Cảm biến gia tốc góc ITG 3200	27
29	module bluetooth HC-05	28
30	Bảng ngôn ngữ ký hiệu	30
31	Kiến trúc tổng quan của hệ thống	31
32	Kiến trúc module thu thập dữ liệu cử chỉ	32
33	Kiến trúc model dự đoán cử chỉ	33
34	Kiến trúc module chuyển đầu ra dự đoán thành giọng nói	34
35	Sơ đồ kết nối mạch	36
36	Hình ảnh phần cứng sau khi hoàn thiện	38
37	Dạng dữ liệu được gửi	39
38	Quá trình lấy dữ liệu	41
39	Kỹ thuật Sliding Window	46
40	Cách thức hoạt động của Sliding Window	47
41	Truyền dữ liệu bằng bluetooth	48
42	Cấu trúc mô hình cũ	50
43	Mô hình mới	52
44	Kết quả thực nghiệm mô hình	56



45 Kết quả dự đoán và thời gian trung bình ứng với các giá trị sliding window . . . 57



Danh sách bảng

1	Thông số kỹ thuật của Arduino Lilypad	20
2	Định nghĩa các hành động	40
3	Bảng tóm tắt mô hình	45



1 Giới thiệu đề tài

1.1 Giới thiệu

Ngôn ngữ ký hiệu là công cụ giao tiếp quan trọng của cộng đồng người khiếm thính và câm, giúp họ truyền tải thông tin và ý tưởng trong cuộc sống hàng ngày. Tuy nhiên, một rào cản lớn là phần lớn những người không quen thuộc với ngôn ngữ ký hiệu gặp khó khăn trong việc hiểu và tương tác với nhóm đối tượng này. Điều này gây ra sự bất tiện và làm giảm khả năng hòa nhập xã hội của người khuyết tật.

Đề tài "Thiết bị chuyển đổi ngôn ngữ ký hiệu sang giọng nói sử dụng Flex Sensors kết hợp với học máy" hướng đến việc phát triển một giải pháp công nghệ nhằm thu hẹp khoảng cách này. Thiết bị được thiết kế với các flex sensors để ghi nhận chuyển động của bàn tay khi thực hiện ký hiệu. Thông qua việc kết hợp với các thuật toán học máy, hệ thống có khả năng nhận diện ngôn ngữ ký hiệu và chuyển đổi chúng thành giọng nói một cách chính xác và tự nhiên.

Ứng dụng này không chỉ giúp người khiếm thính giao tiếp dễ dàng hơn mà còn mở ra nhiều cơ hội để họ hòa nhập vào các hoạt động xã hội và nghề nghiệp. Với tiềm năng ứng dụng rộng rãi, thiết bị này hứa hẹn sẽ mang lại lợi ích thiết thực không chỉ trong lĩnh vực giao tiếp mà còn trong giáo dục, y tế và các ngành dịch vụ.

1.2 Mục tiêu

Mục tiêu của đề tài là xây dựng một hệ thống thông minh có khả năng nhận diện chính xác cử chỉ tay thông qua dữ liệu thu thập từ flex sensors và cảm biến con quay hồi chuyển, sau đó chuyển đổi các cử chỉ này thành giọng nói một cách tự nhiên và mượt mà. Hệ thống không chỉ tập trung vào tính chính xác và hiệu quả trong nhận diện mà còn hướng đến việc nâng cao khả năng ứng dụng thực tiễn, hỗ trợ giao tiếp cho cộng đồng người khiếm thính và câm, góp phần cải thiện chất lượng cuộc sống và tăng cường sự hòa nhập xã hội.

1.3 Phạm vi của đề tài

Ngôn ngữ ký hiệu bao gồm nhiều cử chỉ đa dạng và phức tạp, đòi hỏi lượng dữ liệu lớn để phân tích và xử lý một cách hiệu quả. Do đó, phạm vi nghiên cứu của đề tài sẽ được giới hạn thành hai phần chính nhằm đảm bảo tính khả thi và tiến độ thực hiện.

- Thu thập dữ liệu cử chỉ:** Tập trung vào thiết kế và hiện thực phần cứng, sử dụng các flex sensors và cảm biến con quay gia tốc để thu thập dữ liệu liên quan đến các cử chỉ tay. Phần này sẽ đảm bảo thu thập dữ liệu chính xác và đầy đủ để phục vụ quá trình xử lý.
- Phân tích, xử lý dữ liệu:** Triển khai và tối ưu hóa mô hình học máy để phân tích dữ liệu thu thập được, từ đó nhận diện và dự đoán chính xác các cử chỉ tay. Phần này là nền tảng để hệ thống chuyển đổi cử chỉ thành giọng nói.

1.4 Ý nghĩa thực tiễn

Đề tài mang lại ý nghĩa thực tiễn to lớn trong việc cải thiện khả năng giao tiếp giữa người khiếm thính hoặc câm và những người không biết ngôn ngữ ký hiệu, góp phần nâng cao chất lượng cuộc sống và khả năng hòa nhập xã hội của họ. Sản phẩm của đề tài sở hữu nhiều ưu điểm nổi bật khi áp dụng vào thực tế, cụ thể như sau:



- **Giao tiếp dễ dàng hơn:** Thiết bị cho phép người khiếm thính thực hiện các ký hiệu tay, sau đó chuyển đổi chúng thành giọng nói một cách tự động, giúp họ giao tiếp thuận tiện với mọi người mà không cần người phiên dịch.
- **Tính tiện lợi:** Nhờ thiết kế nhỏ gọn và khả năng sử dụng linh hoạt, thiết bị có thể được mang theo và sử dụng mọi lúc, mọi nơi, giúp việc giao tiếp trở nên tự nhiên hơn trong các tình huống hàng ngày.
- **Khả năng kết nối:** Thiết bị có thể tích hợp với các nền tảng công nghệ hiện đại như điện thoại di động hoặc máy tính, không chỉ hỗ trợ phát giọng nói mà còn có khả năng chuyển đổi ký hiệu thành văn bản, phục vụ cho các hình thức giao tiếp như tin nhắn hay email.
- **Tính đa dụng:** Hệ thống học máy cho phép thiết bị nhận diện và học hỏi nhiều cử chỉ từ đơn giản đến phức tạp, đáp ứng nhu cầu sử dụng của đa dạng người dùng và phù hợp với nhiều ngữ cảnh khác nhau.

Với những tính năng này, thiết bị không chỉ mở ra cơ hội giao tiếp mới cho người khiếm thính và câm, mà còn thúc đẩy sự ứng dụng công nghệ trong cuộc sống. Đề tài góp phần xây dựng một xã hội công bằng, nơi mọi người đều có cơ hội giao tiếp, phát triển và hòa nhập, đồng thời khẳng định vai trò quan trọng của công nghệ trong việc cải thiện đời sống con người.

2 Cơ sở lý thuyết

2.1 Giới thiệu về học máy(Machine learning)

Học máy (Machine Learning) là một nhánh quan trọng của trí tuệ nhân tạo (AI), tập trung vào việc nghiên cứu và phát triển các kỹ thuật cho phép hệ thống tự động học hỏi từ dữ liệu để giải quyết các vấn đề cụ thể mà không cần lập trình rõ ràng từng bước. Khái niệm học máy đã xuất hiện từ những năm 1950, khi các nhà khoa học lần đầu tiên thử nghiệm mô phỏng khả năng học tập của con người bằng máy tính. Tuy nhiên, phải đến đầu thế kỷ 21, học máy mới thực sự bùng nổ nhờ sự tiến bộ vượt bậc trong công nghệ phần cứng và khả năng tính toán.

Trong những năm gần đây, học máy đã được thúc đẩy mạnh mẽ bởi sự phát triển của các thuật toán tiên tiến, sự gia tăng của dữ liệu lớn (Big Data) và sức mạnh xử lý của các hệ thống tính toán hiện đại. Những yếu tố này đã làm cho học máy trở thành công cụ không thể thiếu trong nhiều lĩnh vực như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, y tế, tài chính, và đặc biệt là IoT (Internet of Things). Với khả năng thích ứng linh hoạt và hiệu quả, học máy không chỉ mang lại giá trị thực tiễn to lớn mà còn mở ra những cơ hội mới trong nghiên cứu và ứng dụng công nghệ.

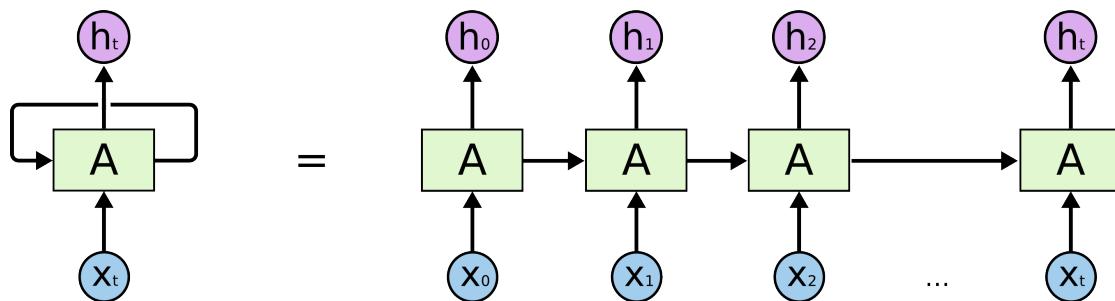
2.2 Giới thiệu về Mạng nơ-ron hồi quy(RNN)

2.2.1 Tổng quan

Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNN) là một loại mạng nơ-ron chuyên biệt trong học sâu, được thiết kế để xử lý dữ liệu tuần tự. Các chuỗi dữ liệu này thường mang tính liên kết, như các câu văn, chuỗi thời gian hay tín hiệu âm thanh. Đặc trưng của RNN là khả năng duy trì thông tin từ các bước trước đó trong chuỗi, nhờ vào các vòng hồi tiếp trong cấu trúc mạng. Điều này cho phép RNN mô hình hóa các mối quan hệ phức tạp và phụ thuộc giữa các thành phần trong dữ liệu.

RNN hoạt động bằng cách sử dụng cùng một tập tham số trên mọi bước thời gian, tái sử dụng chúng để giảm thiểu số lượng tham số cần học. Tuy nhiên, khả năng này cũng đi kèm với thách thức, khi mạng phải đối mặt với các vấn đề như khó khăn trong việc lưu trữ thông tin xa hoặc triệt tiêu/dột biến gradient trong quá trình học.

Hiện nay, mặc dù RNN vẫn được sử dụng cho nhiều bài toán liên quan đến chuỗi dữ liệu, nhưng nó đang dần được thay thế bởi các mô hình hiệu quả hơn như các mô hình dựa trên cơ chế Attention hoặc các mạng Transformer.

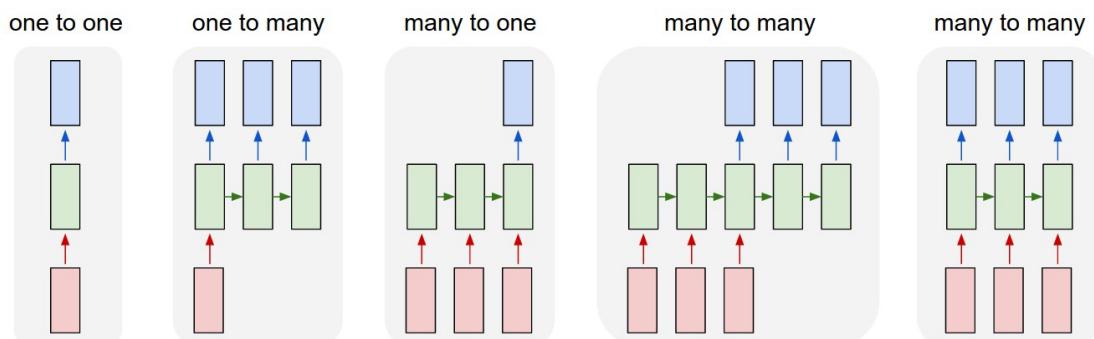


Hình 1: Minh họa một mạng nơ-ron hồi quy

2.2.2 Một số loại mạng nơ-ron hồi quy

Mạng nơ-ron hồi quy (RNN) có thể được cấu hình theo nhiều cách khác nhau tùy thuộc vào loại bài toán cần giải quyết. Dưới đây là các dạng cấu trúc RNN phổ biến:

- **Một-nhiều (One-to-Many):** Trong cấu hình này, từ một đầu vào duy nhất, mạng tạo ra một chuỗi các đầu ra. Loại này thường được ứng dụng trong các bài toán như tạo chú thích hình ảnh, nơi mạng nơ-ron tạo ra một câu hoàn chỉnh dựa trên một hình ảnh đầu vào duy nhất.
- **Nhiều-một (Many-to-One):** Đây là cấu trúc mà mạng nhận một chuỗi dữ liệu đầu vào và đưa ra một đầu ra duy nhất. Một ví dụ điển hình là phân tích cảm xúc, nơi RNN nhận một đoạn văn bản và dự đoán cảm xúc tổng quát (như tích cực, tiêu cực hoặc trung tính).
- **Nhiều-một (Many-to-Many):** Mạng này xử lý nhiều đầu vào và tạo ra nhiều đầu ra tương ứng. Đây là cấu trúc phổ biến trong các ứng dụng dịch máy, nơi mạng nhận một câu ở ngôn ngữ nguồn và chuyển đổi nó thành một câu tương ứng ở ngôn ngữ đích.

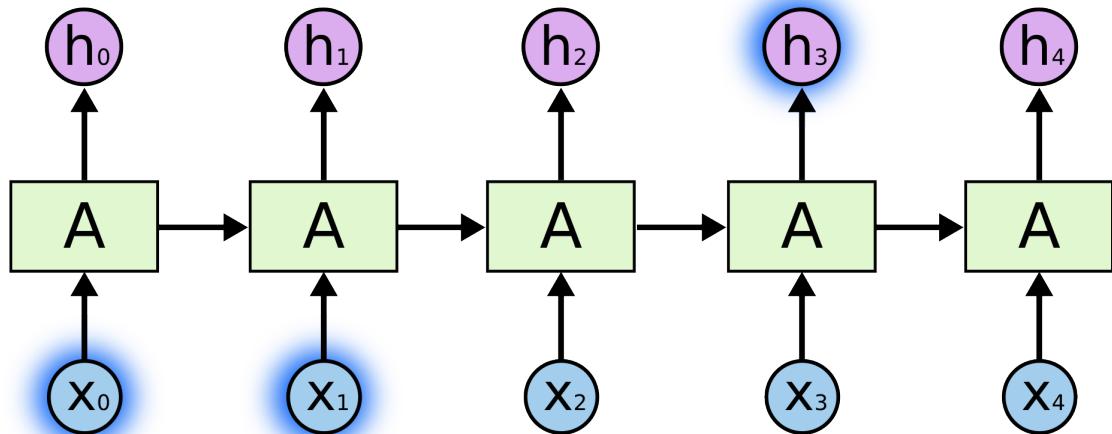


Hình 2: Một số kiểu nơ-ron hồi quy phổ biến

2.2.3 Vấn đề phụ thuộc xa của RNN

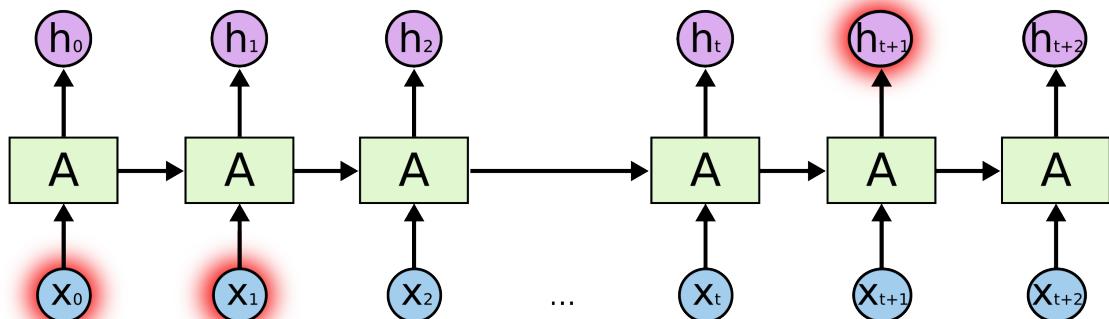
Một trong những đặc điểm nổi bật của mạng nơ-ron hồi quy (RNN) là khả năng tận dụng thông tin từ các bước trước trong chuỗi dữ liệu để dự đoán các bước tiếp theo. Điều này tương tự như cách con người sử dụng bối cảnh từ các đoạn văn trước để hiểu một đoạn văn hiện tại. Tuy nhiên, khả năng này của RNN không phải lúc nào cũng hiệu quả và còn phụ thuộc vào khoảng cách giữa thông tin cần thiết và thông tin hiện tại.

Trong các trường hợp đơn giản, khi khoảng cách giữa dữ liệu cần ghi nhớ và thông tin hiện tại là ngắn, RNN có thể hoạt động tốt. Ví dụ, khi đọc câu "Các đám mây trên bầu trời...", thông tin về cụm từ "bầu" ngay lập tức gợi ý từ "trời". Đây là một dạng phụ thuộc ngắn hạn mà RNN có thể xử lý dễ dàng.



Hình 3: Phụ thuộc ngắn hạn trên RNN

Tuy nhiên, khi cần đến các mối quan hệ phức tạp hơn với khoảng cách xa, RNN bắt đầu gặp khó khăn. Chẳng hạn, trong câu "Tôi lớn lên ở Pháp... Tôi nói tiếng Pháp rất lưu loát", thông tin gần như "Tôi nói tiếng..." không đủ để xác định đó là ngôn ngữ nào. Việc suy luận đúng cần dựa vào thông tin ở câu trước đó, "Tôi lớn lên ở Pháp". Khi khoảng cách giữa dữ liệu cần nhớ và thông tin hiện tại tăng lên, RNN thường mất khả năng ghi nhớ và học hiệu quả.



Hình 4: Phụ thuộc dài hạn trên RNN

Nguyên nhân chính là do hiện tượng triệt tiêu gradient (vanishing gradient), khiến cho các tín hiệu từ các bước trước đó bị yếu dần trong quá trình lan truyền ngược. Điều này khiến RNN không thể lưu giữ được thông tin dài hạn, gây ảnh hưởng nghiêm trọng đến hiệu quả xử lý của mạng.

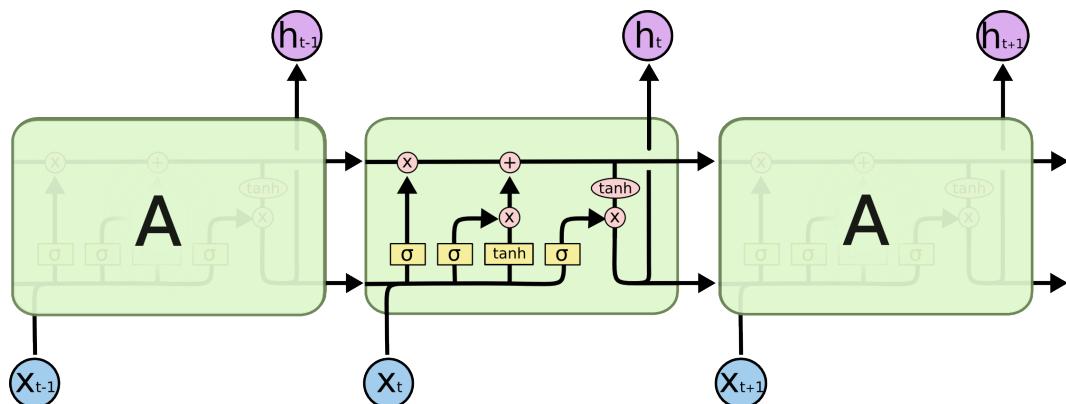
2.3 Giới thiệu mô hình Long-short term memory(LSTM)

2.3.1 Giới thiệu tổng quan

Long Short-Term Memory (LSTM) là một loại mạng nơ-ron hồi quy (RNN) đặc biệt được thiết kế để giải quyết vấn đề phụ thuộc xa trong dữ liệu tuần tự. LSTM lần đầu tiên được giới thiệu bởi Hochreiter và Schmidhuber vào năm 1997 và sau đó đã được cải tiến bởi nhiều nhà nghiên cứu khác, giúp nó trở thành một công cụ quan trọng trong lĩnh vực học sâu.

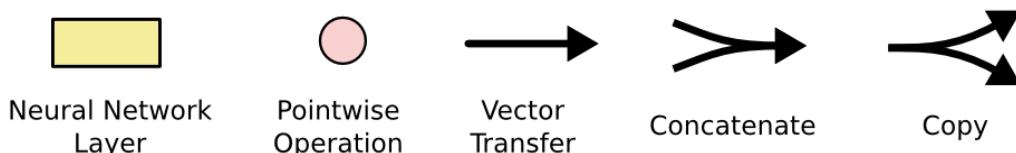
Khác với RNN truyền thống, LSTM có khả năng lưu trữ thông tin trong một khoảng thời gian dài hơn nhờ vào cấu trúc độc đáo với các cổng điều khiển thông minh. Những cổng này bao gồm cổng quên, cổng đầu vào và cổng đầu ra, cho phép mô hình lựa chọn giữ lại hay loại bỏ thông tin một cách có chọn lọc. Điều này giúp LSTM khắc phục được các hạn chế như triệt tiêu hoặc bùng nổ gradient, vốn là vấn đề phổ biến trong RNN tiêu chuẩn.

LSTM hoạt động hiệu quả trên nhiều bài toán khác nhau, từ xử lý ngôn ngữ tự nhiên, dự đoán chuỗi thời gian đến nhận dạng giọng nói và video. Chính sự linh hoạt và khả năng xử lý dữ liệu tuần tự dài hạn này đã khiến LSTM trở thành một lựa chọn phổ biến trong các ứng dụng học sâu.



Hình 5: Mô hình LSTM với 4 lớp tương tác

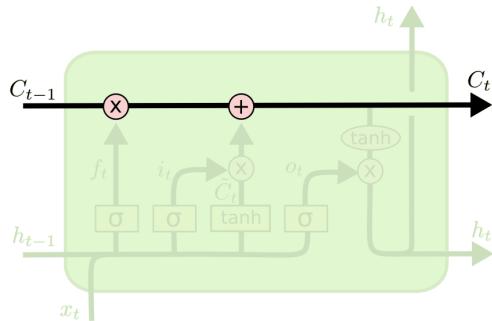
2.3.2 Ý tưởng đằng sau LSTM



Hình 6: Diễn giải các ký hiệu trong đồ thị

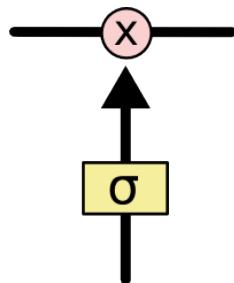
Mạng LSTM được thiết kế với mục đích giải quyết vấn đề phụ thuộc xa trong RNN bằng cách sử dụng một thành phần đặc biệt gọi là ô trạng thái (cell state). Ô trạng thái này hoạt động như một đường dẫn thẳng xuyên suốt chuỗi dữ liệu, giúp truyền tải thông tin mà không bị mất mát quá nhiều qua thời gian. Điều này giúp LSTM có thể nhớ được thông tin trong khoảng thời gian dài mà không gặp phải vấn đề triệt tiêu gradient.

- Ô trạng thái (cell state) được thể hiện qua đường chạy ngang qua đỉnh đồ thị như hình vẽ bên dưới:



Hình 7: Đường đi của ô trạng thái (cell state) trong mạng LSTM

- Ô trạng thái là một dạng băng chuyền chạy thẳng xuyên suốt toàn bộ chuỗi với chỉ một vài tương tác tuyến tính nhỏ giúp cho thông tin có thể truyền dọc theo đồ thị mạng nơ-ron ổn định.
- LSTM có khả năng xóa và thêm thông tin vào ô trạng thái và điều chỉnh các luồng thông tin này thông qua các cấu trúc gọi là cổng.
- Cổng là cơ chế đặc biệt để điều chỉnh luồng thông tin đi qua. Chúng được tổng hợp bởi một tầng ẩn của hàm activation sigmoid và với một toán tử nhân như đồ thị.



Hình 8: Một cổng của hàm sigmoid trong LSTM

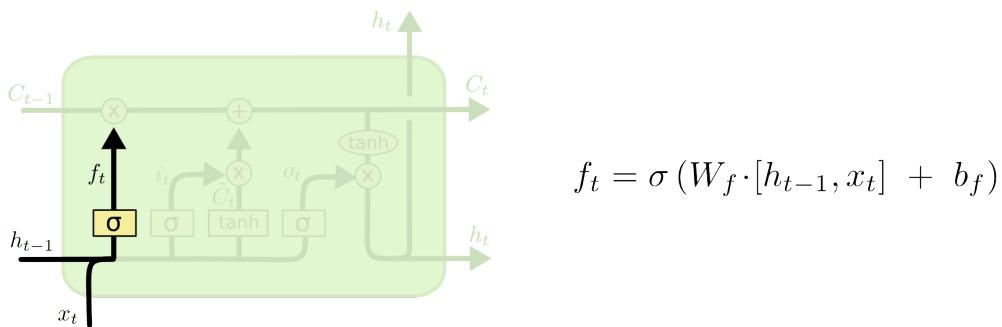
- Hàm sigmoid sẽ cho đầu ra là một giá trị xác suất nằm trong khoảng từ 0 đến 1, thể hiện rằng có bao nhiêu phần thông tin sẽ đi qua cổng. Giá trị bằng 0 ngụ ý rằng không cho phép thông tin nào đi qua, giá trị bằng 1 sẽ cho toàn bộ thông tin đi qua.
- Một mạng LSTM sẽ có 3 cổng có kiến trúc dạng này để bảo vệ và kiểm soát các ô trạng thái.

2.3.3 Thứ tự các bước của LSTM

Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Đầu tiên nó nhận đầu vào là 2 giá trị h_{t-1} và x_t và trả về một giá trị nằm

trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái C_{t-1} . Nếu giá trị bằng 1 thể hiện ‘giữ toàn bộ thông tin’ và bằng 0 thể hiện ‘bỏ qua toàn bộ chúng’.

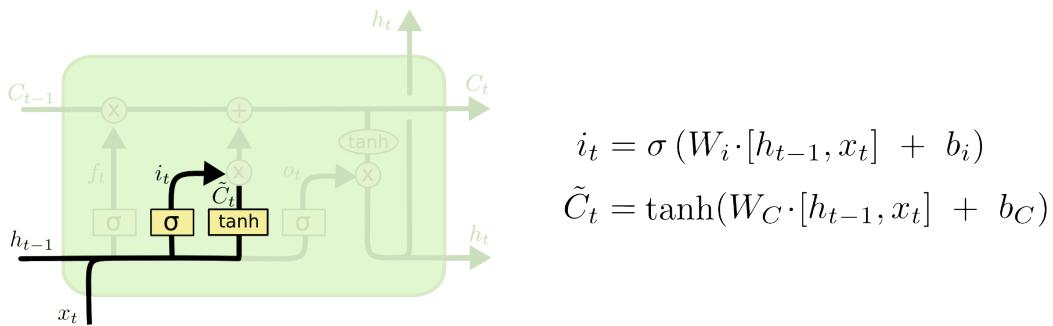
Trở lại ví dụ về ngôn ngữ, chúng ta đang cố gắng dự báo từ tiếp theo dựa trên toàn bộ những từ trước đó. Trong những bài toán như vậy, ô trạng thái có thể bao gồm loại của chủ ngữ hiện tại, để cho đại từ ở câu tiếp theo được sử dụng chính xác. Chẳng hạn như chúng ta đang mô tả về một người bạn là con trai thì các đại từ nhân xưng ở tiếp theo phải là anh, thằng, hắn thay vì cô, con ấy. Tuy nhiên chủ ngữ không phải khi nào cũng cố định. Khi chúng ta nhìn thấy một chủ ngữ mới, chúng ta muốn quên đi loại của một chủ ngữ cũ. Do đó tầng quên cho phép cập nhật thông tin mới và lưu giữ giá trị của nó khi có thay đổi theo thời gian.



Hình 9: Tầng cống quên (forget gate layer)

Bước tiếp theo chúng ta sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần. Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cống vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một véc tơ của một giá trị trạng thái mới mà có thể được thêm vào trạng thái. Tiếp theo kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái.

Trong ví dụ của mô hình ngôn ngữ, chúng ta muốn thêm loại của một chủ ngữ mới vào ô trạng thái để thay thế phần trạng thái cũ muốn quên đi.

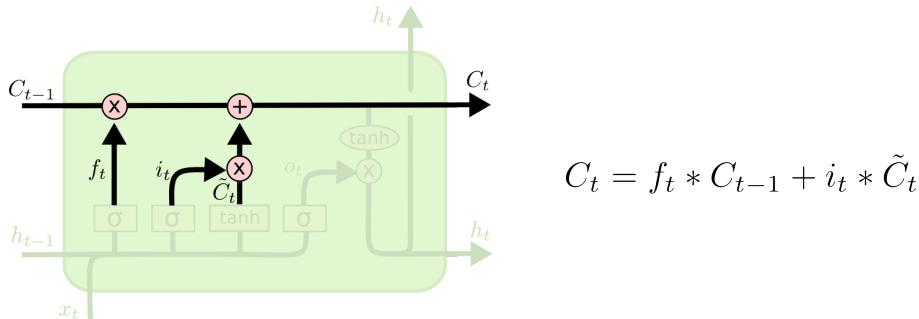


Hình 10: Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cống vào và tầng ẩn hàm tanh

Đây là thời điểm để cập nhật một ô trạng thái cũ C_{t-1} sang một trạng thái mới C_t . Những bước trước đó đã quyết định làm cái gì, và tại bước này chỉ cần thực hiện nó.

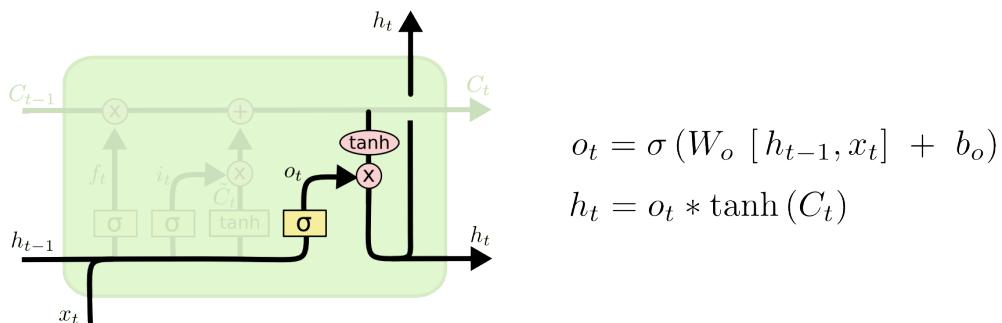
Chúng ta nhân trạng thái cũ với f_t tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử để cử \tilde{C}_{t-1} là một giá trị mới được tính toán tương ứng với bao nhiêu được

cập nhật vào mỗi giá trị trạng thái.



Hình 11: Ô trạng thái mới

Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, nhưng sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân nó với đầu ra của một cỗng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.



Hình 12: Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

2.3.4 Ưu và nhược điểm của mô hình LSTM

Ưu điểm

- LSTM được thiết kế đặc biệt để giải quyết vấn đề phụ thuộc dài hạn trong dữ liệu tuần tự. Nhờ vào ô trạng thái và cơ chế các cỗng, LSTM có khả năng lưu trữ thông tin trong thời gian dài mà không bị mất mát như các mạng RNN truyền thống.
- LSTM có thể học và theo dõi các mối quan hệ phức tạp giữa các phần tử trong chuỗi dữ liệu dài, giúp giải quyết các bài toán như dịch ngôn ngữ, nhận diện giọng nói và phân tích chuỗi thời gian.
- Các cỗng trong LSTM giúp giảm thiểu vấn đề triệt tiêu gradient, mà thường gặp phải trong các mạng RNN thông thường. Điều này cho phép mạng có thể học được thông tin từ các bước thời gian xa mà không gặp phải sự giảm sút mạnh mẽ của gradient.



- LSTM có thể được sử dụng trong nhiều bài toán khác nhau và có thể kết hợp với các mô hình học sâu phức tạp để tạo thành các mạng học sâu đa lớp, linh hoạt và hiệu quả trong việc xử lý các loại dữ liệu khác nhau.

Nhược điểm

- LSTM có cấu trúc phức tạp hơn so với các mạng nơ-ron hồi quy thông thường. Việc tính toán và tối ưu các tham số trong LSTM yêu cầu tài nguyên tính toán lớn và thời gian huấn luyện dài, đặc biệt là với các mô hình có số lớp lớn và dữ liệu phức tạp.
- Do tính toán phức tạp và yêu cầu số lượng tham số lớn, quá trình huấn luyện mô hình LSTM có thể mất nhiều thời gian, đặc biệt là khi làm việc với bộ dữ liệu lớn.
- LSTM dễ gặp phải vấn đề overfitting, đặc biệt là khi sử dụng với bộ dữ liệu huấn luyện nhỏ hoặc khi mạng quá phức tạp. Điều này có thể dẫn đến việc mô hình học quá chi tiết từ dữ liệu huấn luyện mà không có khả năng tổng quát tốt trên dữ liệu mới.
- Mặc dù LSTM giải quyết được vấn đề triệt tiêu gradient, nhưng vấn đề gradient exploding (gradient phát triển quá mức) vẫn có thể xảy ra. Điều này có thể làm mô hình trở nên không ổn định, đặc biệt trong quá trình huấn luyện với các chuỗi dữ liệu rất dài.

2.4 Giới thiệu về Transfer Learning

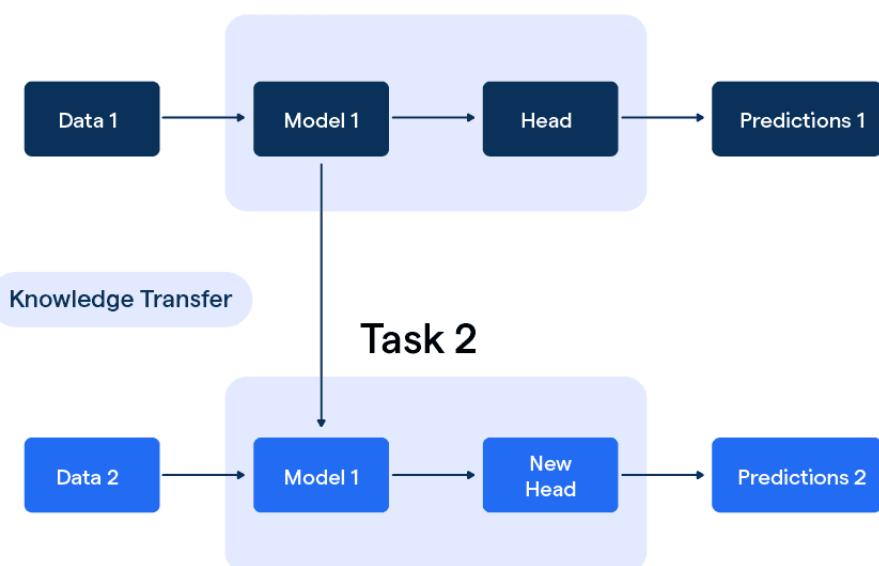
Transfer learning (học chuyển giao) là một kỹ thuật học máy trong đó một mô hình được phát triển cho một nhiệm vụ được sử dụng lại như là điểm khởi đầu cho một mô hình trên một nhiệm vụ thứ hai. Nói cách khác, kiến thức thu được trong khi giải quyết một vấn đề được áp dụng cho một vấn đề khác, nhưng có liên quan.

2.4.1 Định nghĩa Transfer Learning

Transfer learning là việc sử dụng kiến thức đã học từ một miền nguồn (source domain) để cải thiện việc học trong một miền đích (target domain) khác. Miền nguồn thường có lượng dữ liệu lớn hơn và/hoặc dễ học hơn miền đích.

Transfer Learning

Task 1



Hình 13: Transfer learning

2.4.2 Ứng dụng của Transfer Learning

Transfer learning đã được ứng dụng thành công trong nhiều lĩnh vực, bao gồm:

- **Thị giác máy tính:**

- Phân loại ảnh: Xác định các đối tượng trong ảnh (chó, mèo, xe hơi,...)
- Nhận dạng đối tượng: Xác định vị trí của các đối tượng trong ảnh.
- Phân đoạn ảnh: Phân chia ảnh thành các vùng có ý nghĩa.

- **Xử lý ngôn ngữ tự nhiên:**

- Phân loại văn bản: Phân loại các tài liệu văn bản (tin tức, đánh giá,...)
- Phân tích cảm xúc: Xác định cảm xúc được thể hiện trong văn bản.

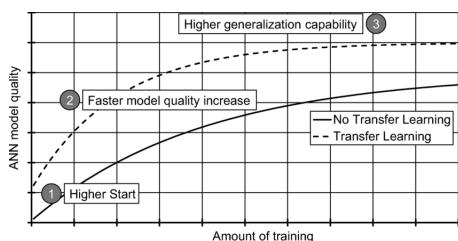
- Dịch máy: Dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác.

- **Âm thanh và lời nói:**

- Nhận dạng giọng nói: Chuyển đổi lời nói thành văn bản.
- Tổng hợp giọng nói: Tạo ra giọng nói từ văn bản.

2.4.3 Lợi ích và bất lợi của Transfer Learning

Lợi ích:



Hình 14: Lợi ích của Transfer Learning

- Cải thiện hiệu suất của mô hình, đặc biệt là khi dữ liệu huấn luyện hạn chế.
- Giảm thời gian huấn luyện và tài nguyên tính toán.
- Nâng cao khả năng khai quát hóa của mô hình.

Bất lợi:

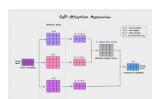
- Có thể xảy ra hiện tượng "negative transfer" nếu miền nguồn và miền đích quá khác biệt.
- Khó khăn trong việc lựa chọn mô hình tiền huấn luyện phù hợp.

2.5 Giới thiệu về Self-Attention

Self-attention là một cơ chế trong học sâu cho phép mô hình tập trung vào các phần khác nhau của dữ liệu đầu vào khi xử lý nó. Nó đã trở thành một thành phần quan trọng trong nhiều kiến trúc mạng nơ-ron hiện đại, đặc biệt là trong lĩnh vực xử lý ngôn ngữ tự nhiên.

2.5.1 Cơ chế hoạt động của Self-Attention

Self-attention hoạt động bằng cách tính toán mối quan hệ giữa các phần tử khác nhau trong một chuỗi. Mỗi phần tử được biểu diễn bằng một vector, và self-attention tính toán "attention weights" (trọng số chú ý) để xác định mức độ quan trọng của mỗi phần tử đối với các phần tử khác. Các trọng số này sau đó được sử dụng để tạo ra một biểu diễn mới cho mỗi phần tử, kết hợp thông tin từ các phần tử liên quan.



Hình 15: Self Attention



2.5.2 Các loại Self-Attention thường gặp

Có nhiều biến thể của self-attention, mỗi biến thể có những đặc điểm và ứng dụng riêng:

- * **Scaled Dot-Product Attention:** Đây là biến thể phổ biến nhất, được sử dụng trong Transformer. Nó tính toán attention weights dựa trên tích vô hướng của các vector biểu diễn.
- * **Multi-Head Attention:** Biến thể này sử dụng nhiều "head" attention song song, mỗi head tập trung vào các khía cạnh khác nhau của dữ liệu.
- * **Additive Attention:** Biến thể này sử dụng một mạng nơ-ron để tính toán attention weights.

2.5.3 Ứng dụng của Self-Attention

Self-attention đã được ứng dụng rộng rãi trong nhiều lĩnh vực:

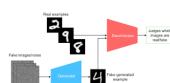
- * **Xử lý ngôn ngữ tự nhiên:** * Dịch máy * Tóm tắt văn bản * Phân tích cảm xúc * Hỏi đáp
- * **Thị giác máy tính:** * Phân loại ảnh * Nhận dạng đối tượng
- * **Các lĩnh vực khác:** * Sinh học * Âm nhạc

2.5.4 Lợi ích của Self-Attention

- * **Nắm bắt được các phụ thuộc dài hạn:** Self-attention có thể nắm bắt được mối quan hệ giữa các phần tử cách xa nhau trong chuỗi, điều mà các mô hình RNN truyền thống gặp khó khăn.
- * **Tính toán song song hiệu quả:** Self-attention có thể được tính toán song song, giúp tăng tốc độ huấn luyện và suy luận.
- * **Khả năng diễn giải tốt:** Attention weights cung cấp thông tin về các phần tử quan trọng trong chuỗi, giúp hiểu rõ hơn cách mô hình đưa ra quyết định.

2.6 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) là một lớp mô hình học sâu mạnh mẽ được sử dụng để tạo ra dữ liệu mới, giống với dữ liệu huấn luyện. GANs bao gồm hai thành phần chính: generator và discriminator, hoạt động theo một cách đối kháng - generator cố gắng sinh dữ liệu sao cho giống với dữ liệu thật nhất, discriminator cố gắng phân biệt giữa dữ liệu sinh ra và dữ liệu thật cho đến khi discriminator không thể phân biệt được nữa.



Hình 16: Lợi ích của Transfer Learning

2.6.1 Kiến trúc và hoạt động của GANs

- * **Generator:** Mạng nơ-ron này nhận vào một vector nhiễu ngẫu nhiên và cố gắng tạo ra dữ liệu giả mạo giống với dữ liệu thật.
- * **Discriminator:** Mạng nơ-ron này nhận vào dữ liệu (thật hoặc giả mạo) và cố gắng phân biệt giữa chúng.

Hai mạng này được huấn luyện đồng thời: generator cố gắng đánh lừa discriminator, trong khi discriminator cố gắng không bị đánh lừa. Quá trình huấn luyện này tạo ra một vòng lặp phản hồi, trong đó cả hai mạng đều cải thiện theo thời gian.



2.6.2 Các biến thể của GANs

Có nhiều biến thể của GANs, mỗi biến thể có những đặc điểm và ứng dụng riêng:

- * **Deep Convolutional GANs (DCGANs):** Sử dụng mạng nơ-ron tích chập để tạo ra hình ảnh chất lượng cao.
- * **Conditional GANs (cGANs):** Cho phép điều khiển quá trình tạo dữ liệu bằng cách cung cấp thông tin bổ sung (như nhãn lớp).
- * **CycleGANs:** Cho phép chuyển đổi dữ liệu từ miền này sang miền khác (ví dụ: chuyển đổi ảnh từ mùa hè sang mùa đông).
- * **Progressive Growing of GANs (PGGANs):** Tạo ra hình ảnh có độ phân giải cao bằng cách tăng dần kích thước của mạng generator và discriminator.
- * **StyleGAN:** Tạo ra hình ảnh chân thực với khả năng kiểm soát các thuộc tính tinh tế.
- * **TimeGAN:** Tạo ra dữ liệu chuỗi thời gian mẫu từ tập dữ liệu đầu vào cho sẵn

2.6.3 Ứng dụng của GANs

GANs đã được ứng dụng trong nhiều lĩnh vực:

- * **Tạo hình ảnh:** Tạo ra hình ảnh chân thực của người, vật thể, cảnh quan,...
- * **Tạo dữ liệu:** Tạo ra dữ liệu tổng hợp cho các tác vụ học máy khác.
- * **Chỉnh sửa ảnh:** Thay đổi các thuộc tính của ảnh (ví dụ: thêm nụ cười, thay đổi kiểu tóc).
- * **Siêu phân giải ảnh:** Tăng độ phân giải của ảnh.
- * **Phục hồi ảnh:** Khôi phục các phần bị thiếu hoặc bị hỏng của ảnh.

2.6.4 Lợi ích và hạn chế của GANs

Lợi ích:

- * Khả năng tạo ra dữ liệu mới, đa dạng và chân thực.
- * Ứng dụng rộng rãi trong nhiều lĩnh vực.

Hạn chế:

- * Huấn luyện GANs có thể khó khăn và không ổn định.
- * Dánh giá chất lượng của dữ liệu do GANs tạo ra có thể pha trộn.

2.7 Giới thiệu về thư viện Tensorflow

TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph - đồ thị luồng dữ liệu để dựng nên các model. Nếu bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.



TensorFlow

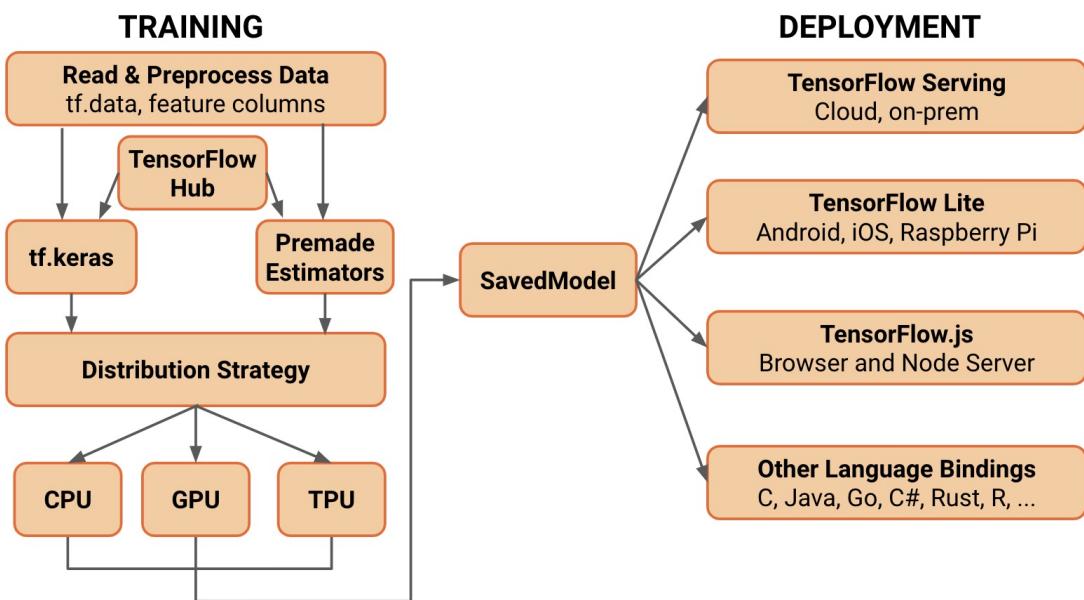
Hình 17: Thư viện Tensorflow

2.7.1 Nguyên lý hoạt động

TensorFlow là một thư viện mã nguồn mở được thiết kế để thực hiện các phép tính số phức tạp thông qua cấu trúc đồ thị (graph). Cách hoạt động của TensorFlow dựa trên việc xây dựng và xử lý các mô hình học máy thông qua đồ thị tính toán, bao gồm các bước cơ bản sau:

- **Xây dựng đồ thị tính toán:** Người dùng định nghĩa cấu trúc đồ thị, trong đó các nút (nodes) đại diện cho các phép tính toán (operations), và các cạnh (edges) biểu diễn luồng dữ liệu (dạng tensor) giữa các nút.
- **Định nghĩa hàm mất mát:** Hàm mất mát được sử dụng để đo lường sự khác biệt giữa đầu ra dự đoán của mô hình và giá trị thực tế. Đây là thước đo chính để tối ưu hóa mô hình.
- **Tối ưu hóa mô hình:** TensorFlow sử dụng các thuật toán tối ưu hóa như Gradient Descent để điều chỉnh các tham số trong mô hình, giảm giá trị của hàm mất mát.
- **Huấn luyện mô hình:** Dữ liệu huấn luyện được đưa vào mô hình, các tham số được cập nhật lặp đi lặp lại qua nhiều vòng huấn luyện để tối ưu hóa kết quả dự đoán.
- **Thực thi đồ thị qua Session:** Để thực hiện các phép tính, TensorFlow sử dụng Session, một môi trường chạy giúp thực thi đồ thị tính toán đã được xây dựng. Các session này cho phép TensorFlow phân phối các tác vụ tính toán trên nhiều thiết bị như CPU, GPU hoặc TPU.
- **Đánh giá mô hình:** Sau khi huấn luyện, mô hình được kiểm tra trên tập dữ liệu kiểm định để đánh giá hiệu suất và khả năng tổng quát.

- **Sử dụng mô hình:** Mô hình đã được huấn luyện có thể được sử dụng để dự đoán hoặc phân loại trên các dữ liệu mới, phù hợp với bài toán cụ thể.



Hình 18: Nguyên lý hoạt động của Tensorflow

2.7.2 Thuộc tính cơ bản trong Tensorflow

Về cơ bản thì các thuộc tính của TensorFlow bao gồm:

- **Tensors:**

- Tensor là cấu trúc dữ liệu trung tâm trong TensorFlow, đại diện cho các mảng đa chiều với kiểu dữ liệu đồng nhất.
- Tensor có thể mang giá trị là dữ liệu đầu vào, kết quả trung gian hoặc đầu ra của mô hình.

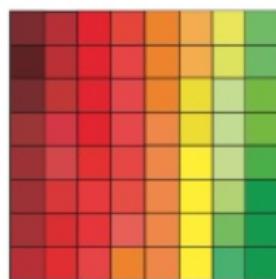
tensor = multidimensional array

vector



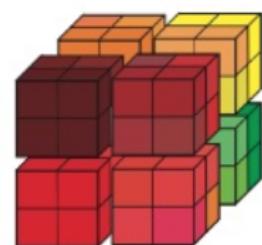
$$\mathbf{v} \in \mathbb{R}^{64}$$

matrix



$$\mathbf{X} \in \mathbb{R}^{8 \times 8}$$

tensor



$$\mathbf{X} \in \mathbb{R}^{4 \times 4 \times 4}$$

Hình 19: Hình ảnh minh họa của một tensor và so sánh với vector và ma trận.

- **Phép tính (Operations):**

- TensorFlow thực hiện các phép tính trên tensors thông qua các operations (toán tử).
- Các phép tính cơ bản bao gồm cộng, trừ, nhân, chia và nhiều phép toán học nâng cao khác.
- Các operations này được biểu diễn dưới dạng các nút trong đồ thị tính toán.

- **Biến (Variables):**

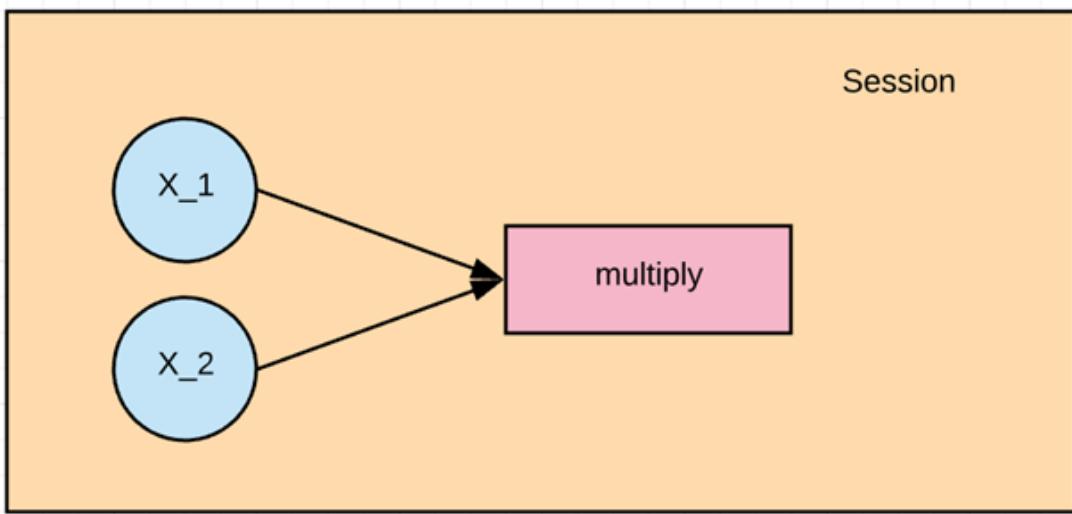
- Biến được sử dụng để lưu trữ các giá trị có thể thay đổi trong quá trình huấn luyện mô hình, chẳng hạn như các tham số của mô hình.
- Giá trị của biến có thể được khởi tạo và cập nhật trong suốt quá trình huấn luyện.

- **Đồ thị (Graphs):**

- Đồ thị là một biểu đồ tính toán biểu diễn mối quan hệ giữa các phép tính và dữ liệu.
- TensorFlow sử dụng đồ thị để tối ưu hóa việc tính toán và triển khai mô hình trên nhiều thiết bị khác nhau, bao gồm CPU, GPU và TPU.

- **Sessions:**

- Session là một phiên làm việc trong TensorFlow, nơi các phép tính trong đồ thị được thực thi.
- Session quản lý các biến, phép tính và thực hiện quá trình huấn luyện hoặc dự đoán.



Hình 20: Minh họa cho 1 graph, session, operation và variable.

- **Placeholders:**

- Placeholders là các "nơi trống" trong đồ thị, được sử dụng để nhận dữ liệu đầu vào tại thời điểm chạy.
- Điều này cho phép mô hình linh hoạt khi xử lý nhiều loại dữ liệu khác nhau trong quá trình huấn luyện hoặc kiểm tra.

Các thuộc tính trên tạo thành nền tảng mạnh mẽ, giúp TensorFlow trở thành công cụ hàng đầu trong việc xây dựng, huấn luyện và triển khai các mô hình học máy hiện đại.

2.7.3 Ưu điểm của TensorFlow

- **Tính linh hoạt cao:**

- TensorFlow hỗ trợ nhiều loại mô hình học máy, từ học có giám sát đến học không giám sát và học tăng cường.
- Người dùng có thể dễ dàng tùy chỉnh mô hình để phù hợp với các bài toán cụ thể, từ mạng nơ-ron đơn giản đến các kiến trúc phức tạp như CNN, RNN, hay Transformer.

- **Khả năng mở rộng mạnh mẽ:**

- TensorFlow được thiết kế để chạy trên nhiều nền tảng khác nhau, bao gồm máy tính cá nhân, máy chủ, GPU, TPU và thậm chí là các thiết bị di động.
- Điều này giúp người dùng triển khai mô hình trên các môi trường khác nhau mà không cần thay đổi mã nguồn.



- **Hiệu suất cao:**

- TensorFlow tận dụng sức mạnh của phần cứng, hỗ trợ tính toán song song và tối ưu hóa hiệu suất thông qua GPU và TPU.
- Công cụ này cũng cung cấp các thuật toán tối ưu hóa hiệu quả, giúp giảm thời gian huấn luyện mô hình.

- **Cộng đồng hỗ trợ lớn:**

- TensorFlow có một cộng đồng người dùng toàn cầu lớn mạnh, cung cấp nhiều tài liệu học tập, hướng dẫn và giải pháp cho các vấn đề kỹ thuật.
- Với nguồn tài liệu phong phú, người dùng dễ dàng tìm kiếm hỗ trợ từ cộng đồng hoặc các diễn đàn trực tuyến.

- **Hỗ trợ triển khai mô hình:**

- TensorFlow cho phép triển khai mô hình trên nhiều nền tảng, từ máy tính cá nhân đến các ứng dụng di động, web và đám mây.
- TensorFlow Lite giúp tối ưu hóa mô hình cho các thiết bị có hiệu năng hạn chế như điện thoại thông minh và thiết bị IoT.

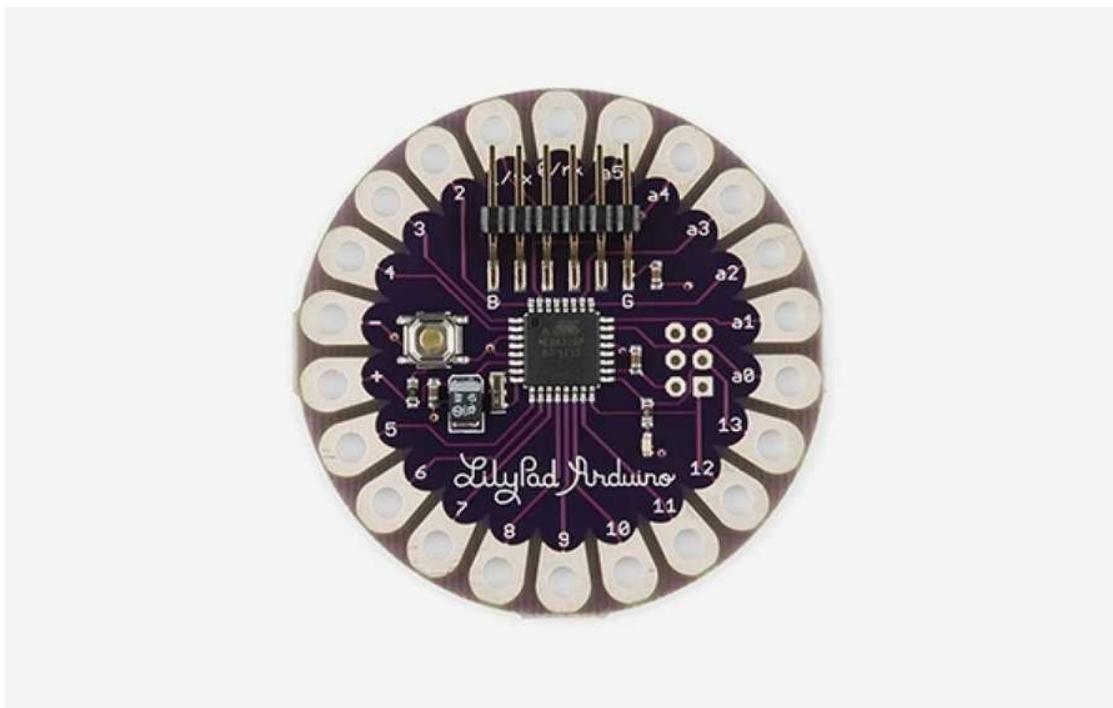
- **Thư viện phong phú:**

- TensorFlow cung cấp nhiều thư viện mở rộng như TensorFlow.js (dành cho trình duyệt web), TensorFlow Lite (dành cho thiết bị di động) và TensorFlow Extended (hỗ trợ quy trình phát triển mô hình từ đầu đến cuối).
- Người dùng có thể dễ dàng tích hợp các thư viện này vào dự án của mình.

2.8 Giới thiệu về Arduino LilyPad

Arduino LilyPad là một phiên bản đặc biệt của bo mạch Arduino, được thiết kế dành riêng cho các ứng dụng điện tử gắn vào vải (e-textiles) và thiết bị điện tử có thể đeo (wearables). Bo mạch này có cấu trúc phẳng, nhẹ và có thể được may trực tiếp vào trang phục hoặc phụ kiện bằng chỉ dệt điện.

Arduino LilyPad được phát triển bởi Leah Buechley và SparkFun Electronics, dựa trên vi điều khiển ATmega168 hoặc ATmega328V. Đây là một giải pháp lý tưởng cho các dự án điện tử sáng tạo, nơi tính thẩm mỹ và tính linh hoạt trong thiết kế được ưu tiên.



Hình 21: Board mạch Arduino Lilypad

Bảng 1: Thông số kỹ thuật của Arduino Lilypad

Vi điều khiển	ATmega168 hoặc ATmega328V
Điện áp hoạt động	2.7-5.5 V
Điện áp đầu vào	2.7-5.5 V
Số chân I/O	14
Số kênh PWM	6
Số kênh đầu vào Analog	6
Dòng điện mỗi chân	40mA
Bộ nhớ Flash	16 KB
SRAM	1 KB
EEPROM	512 bytes
Tần số xung	8 MHz

Ứng dụng tiêu biểu của Arduino Lilypad:

- **Trang phục thông minh:** Arduino LilyPad có thể được sử dụng để theo dõi thông số cơ thể, điều khiển ánh sáng hoặc tích hợp cảm biến vào trang phục.

- **Thiết bị đeo tay:** Nó có thể làm nền tảng cho các sản phẩm như găng tay thông minh, vòng tay theo dõi sức khỏe hoặc thiết bị điều khiển bằng cử chỉ.
- **Dự án nghệ thuật:** LilyPad được sử dụng rộng rãi trong các dự án nghệ thuật sáng tạo, kết hợp công nghệ với thiết kế thẩm mỹ.

Với thiết kế nhỏ gọn và khả năng tích hợp cao, Arduino LilyPad là lựa chọn lý tưởng cho các nhà sáng tạo muốn kết hợp công nghệ với thời trang và các thiết bị cá nhân hóa.

2.9 Giới thiệu về Flex sensor

2.9.1 Tổng quan

Flex Sensor (Cảm biến uốn cong) là một loại cảm biến đặc biệt hoạt động dựa trên nguyên tắc thay đổi điện trở khi bị uốn cong. Loại cảm biến này thường được sử dụng để đo lường và theo dõi mức độ uốn cong hoặc góc độ của một bề mặt hoặc cấu trúc.

Flex Sensor có thiết kế nhỏ gọn, linh hoạt và thường được sản xuất với hai kích thước phổ biến:

- 2.2 inch (khoảng 5.588 cm)
- 4.5 inch (khoảng 11.43 cm)

Nhờ khả năng thay đổi giá trị điện trở tỷ lệ thuận với mức độ uốn cong, cảm biến này trở thành một giải pháp hiệu quả cho nhiều ứng dụng, đặc biệt trong các dự án liên quan đến thiết bị đeo, robot hoặc hệ thống điều khiển bằng cử chỉ.

Cảm biến uốn cong thường được sử dụng kết hợp với các mạch điện để chuyển đổi thay đổi điện trở thành tín hiệu điện áp, từ đó dễ dàng xử lý bằng các bộ vi điều khiển hoặc các hệ thống khác.

Với độ nhạy cao và tính linh hoạt, cảm biến uốn cong là một lựa chọn lý tưởng trong các dự án cần theo dõi chuyển động hoặc đo lường độ biến dạng của bề mặt.

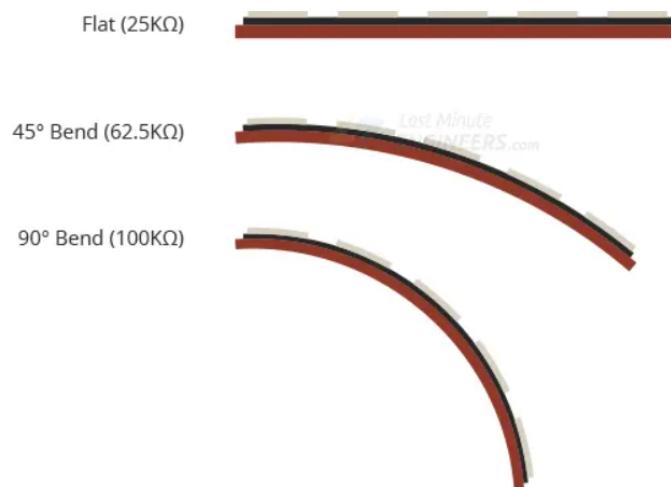


Hình 22: Flex sensor

2.9.2 Nguyên lý hoạt động

Cảm biến uốn cong hoạt động dựa trên sự thay đổi điện trở của một lớp mực dẫn đặc biệt được phủ trên bề mặt cảm biến. Khi cảm biến bị uốn cong, các đặc tính của lớp mực dẫn thay đổi, dẫn đến sự biến đổi giá trị điện trở. Cơ chế hoạt động của cảm biến bao gồm các đặc điểm chính sau:

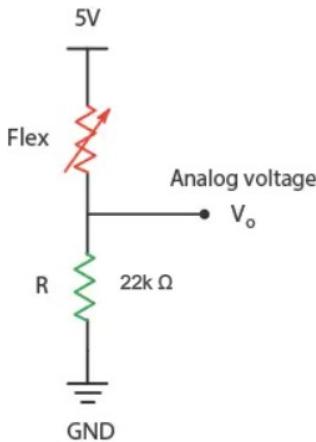
- **Trạng thái thẳng:** Khi cảm biến ở trạng thái thẳng, lớp mực dẫn không bị biến dạng, và giá trị điện trở thường dao động ở mức thấp, khoảng $25\text{k}\Omega$.
- **Trạng thái uốn cong:** Khi cảm biến bị uốn, lớp mực dẫn bị kéo căng, làm giảm diện tích tiếp xúc và tăng trở kháng của nó. Ở góc uốn cong 90° , giá trị điện trở có thể tăng lên đến $100\text{k}\Omega$ hoặc hơn, tùy thuộc vào thiết kế và loại cảm biến.
- **Trạng thái phục hồi:** Khi cảm biến được duỗi thẳng lại, lớp mực dẫn trở về trạng thái ban đầu, và giá trị điện trở cũng quay về mức trước đó.



Hình 23: Hình dạng uốn cong của flex sensor

2.9.3 Đọc giá trị

Việc đọc giá trị từ cảm biến uốn cong được thực hiện thông qua một mạch điện đơn giản, thường là mạch chia điện áp. Cảm biến uốn cong được kết hợp với một điện trở cố định để tạo ra một tín hiệu điện áp thay đổi, tương ứng với mức độ uốn cong của cảm biến. Điện áp này sau đó được đo và xử lý bởi bộ vi điều khiển hoặc mạch xử lý tín hiệu.



Hình 24: Sơ đồ nối dây của flex sensor

Lưu ý rằng điện áp đầu ra mà bạn đo là mức giảm điện áp qua trở kháng kéo xuống, không phải là giảm điện áp qua cảm biến uốn cong (Flex Sensor).

Chúng ta có thể sử dụng công thức sau để tính toán điện áp đầu ra (V_0).

$$V_0 = \frac{V_{cc} \cdot R}{R + R_{flex}}$$

Trong trường hợp này, điện áp đầu ra giảm khi bán kính uốn cong tăng lên.

2.10 Giới thiệu về cảm biến con quay gia tốc

2.10.1 Tổng quan

Cảm biến con quay gia tốc là một thiết bị được sử dụng để đo lường sự gia tốc của một vật thể theo các trục không gian. Thiết bị này hoạt động dựa trên nguyên lý của lực quán tính, thường kết hợp với công nghệ MEMS (Micro-Electro-Mechanical Systems) để đạt được độ chính xác cao trong kích thước nhỏ gọn.

Bên trong cảm biến, một khối lượng nhỏ được treo bằng các lò xo siêu nhỏ. Khi cảm biến chịu tác động của gia tốc, khối lượng này di chuyển do lực quán tính, và sự chuyển động này được chuyển đổi thành tín hiệu điện thông qua các cơ chế đo lường như:

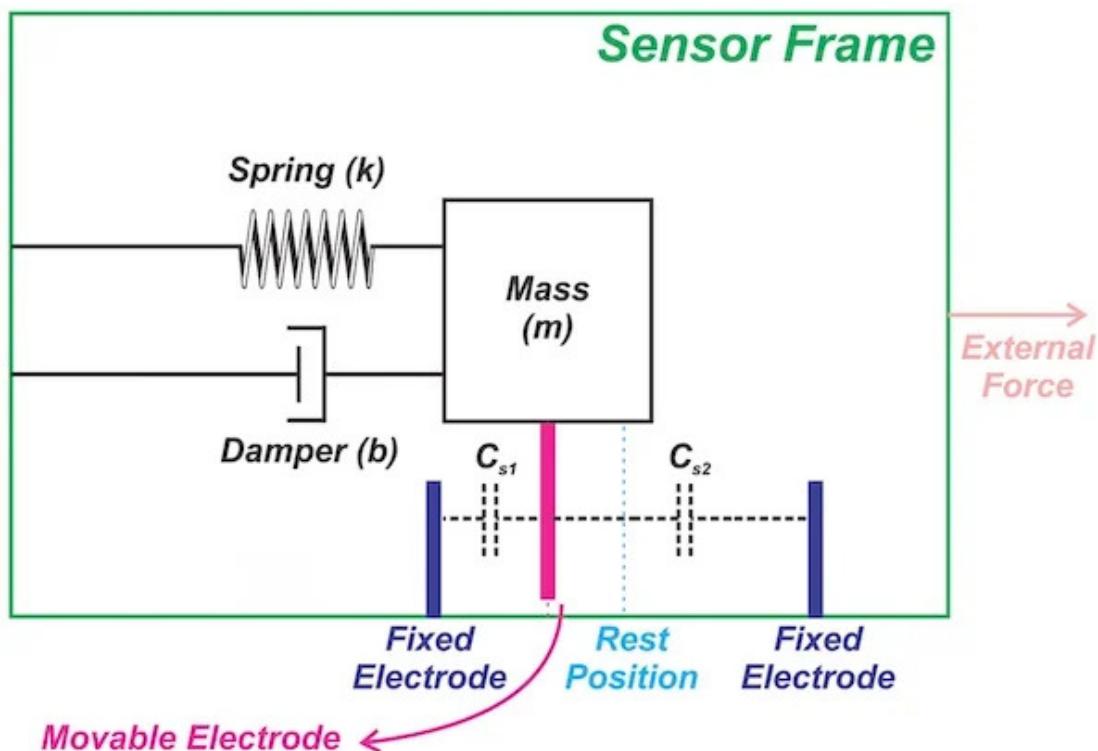
- **Hiệu ứng điện dung:** Sử dụng sự thay đổi điện dung giữa các phần tử trong cảm biến để đo mức độ di chuyển.
- **Hiệu ứng áp điện (Piezoelectric):** Sử dụng các vật liệu tạo điện áp khi chịu lực cơ học để đo sự dịch chuyển.

Cảm biến gia tốc thường được thiết kế để đo gia tốc theo ba trục: X, Y và Z. Điều này giúp xác định chính xác chuyển động, tư thế hoặc vị trí của vật thể trong không gian.

2.10.2 Nguyên lý hoạt động

2.10.2.a Hiệu ứng điện dung:

Cấu trúc lò xo - khôi nặng - giảm xóc: Cấu trúc này chuyển đổi gia tốc của khung cảm biến thành sự dịch chuyển của khôi nặng chứng minh, và phương pháp cảm biến điện dung được áp dụng để chuyển đổi sự dịch chuyển này thành tín hiệu điện tử tỷ lệ với gia tốc được áp dụng.



Hình 25: Cấu trúc của một cảm biến gia tốc sử dụng phương pháp điện dung

Phương pháp cảm biến điện dung: Có hai bản điện cực được gắn cố định vào khung cảm biến cùng với một điện cực di động được kết nối với khôi lượng tham chiếu. Thiết kế này hình thành hai tụ điện biến thiên C_1 và C_2 , như được hiển thị trong hình 29.

Khi khôi lượng tham chiếu di chuyển theo một hướng dưới ảnh hưởng của gia tốc, điện dung giữa điện cực di động và một trong hai điện cực cố định tăng lên trong khi điện dung của tụ điện kia giảm xuống. Bằng cách đo đặc sự thay đổi điện dung của hai tụ điện trên, và kết hợp với khôi lượng tham chiếu có sẵn ta có thể tính toán được gia tốc đầu vào.

2.10.2.b Hiệu ứng áp điện (Piezoelectric):

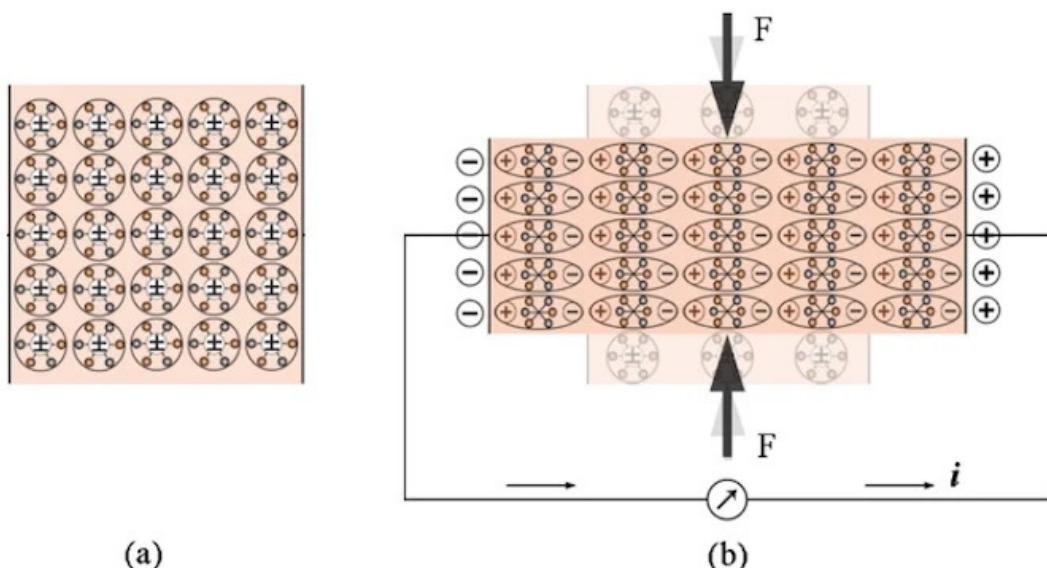
Hiệu ứng Piezoelectric là hiện tượng mà một số vật liệu nhất định có khả năng tạo ra điện tích khi chịu áp lực cơ học. Từ “Piezoelectric” bắt nguồn từ từ tiếng Hy Lạp “piezein”, có nghĩa là “ấn hoặc nén”, mô tả chính xác quá trình tạo ra điện năng thông qua áp lực.

Hiệu ứng này xảy ra ở mức độ vi mô, khi áp dụng một lực cơ học dẫn đến sự di chuyển của các điện tích dương và âm trong cấu trúc tinh thể của vật liệu. Sự dịch chuyển này tạo ra một cực điện và hình thành một điện áp đi qua vật liệu.

Hiệu ứng Piezoelectric là một quá trình có thể đảo ngược: vật liệu thể hiện hiệu ứng piezoelectric cũng sẽ bị biến dạng cơ học nếu đặt một điện áp đi qua nó.

Ví dụ, các tinh thể titanate zirconate chì sẽ tạo ra dòng điện Piezoelectric khi cấu trúc tinh của chúng bị biến dạng khoảng 0,1% so với kích thước gốc. Ngược lại, những tinh thể giống như vậy sẽ thay đổi khoảng 0,1% kích thước tinh của chúng khi áp dụng một điện trường bên ngoài.

Hiệu ứng Piezoelectric đã được khai thác trong nhiều ứng dụng hữu ích, bao gồm sản xuất và phát hiện âm thanh, in phun piezoelectric, tạo ra điện năng điện áp cao, như một bộ tạo xung trong các thiết bị điện tử, trong microbalances, để điều khiển một đầu phun siêu âm, và trong việc tập trung siêu mịn của các bộ phận quang học.

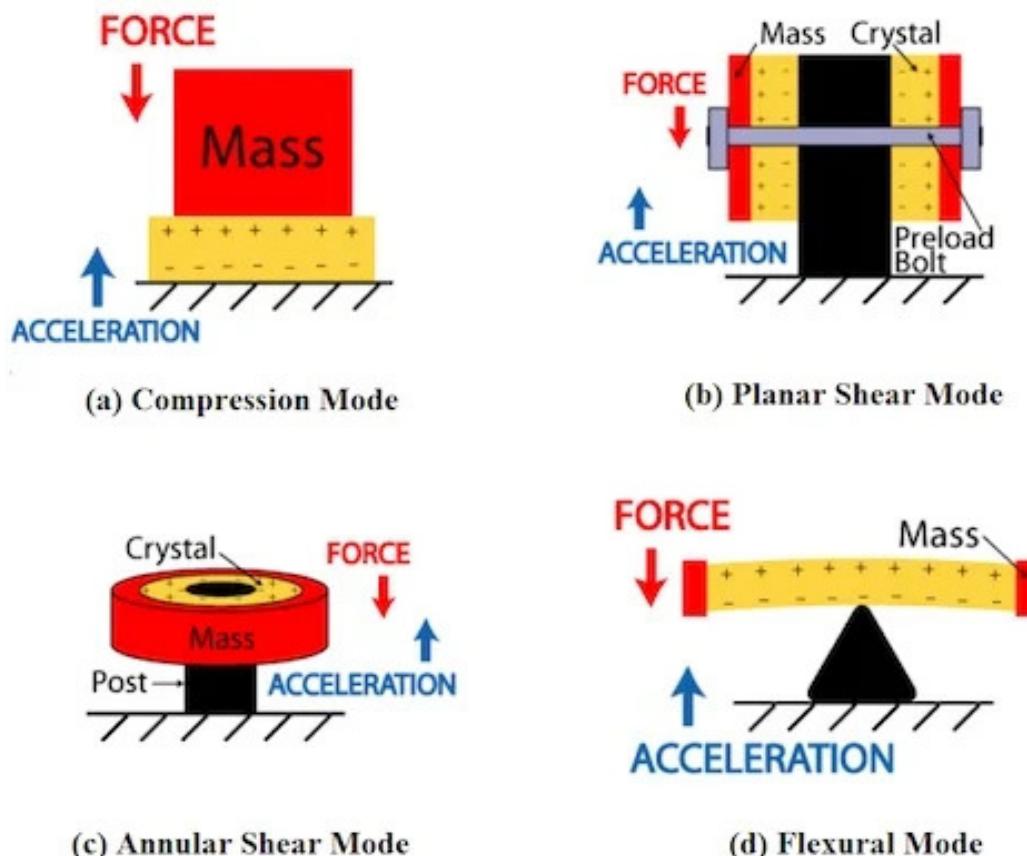


Hình 26: Hiệu ứng Piezoelectric

Cấu trúc của cảm biến tốc độ bao gồm một phần tử piezoelectric để kết nối một lượng khối lượng cố định với thân cảm biến. Khi khung cảm biến tăng tốc do lực bên ngoài, khối lượng tham chiếu có xu hướng giữ nguyên vị trí do quán tính và làm biến dạng nhẹ phần tử piezoelectric.

Nếu nối hai điện cực với nhau thông qua một sợi dây như mô tả trong 26, thì các electron tự do trong dây dẫn sẽ chảy về phía điện cực tích điện dương và tạo ra một dòng điện. Dòng điện này tích tụ các electron tự do trên điện cực dương và tạo ra một điện trường ngược hướng với trường ban đầu được tạo ra bởi hiệu ứng Piezoelectric.

Hiệu ứng này giải thích tại sao dòng điện do lực tĩnh tạo ra chỉ có thể tồn tại trong một khoảng thời gian ngắn. Dòng điện tiếp tục cho đến khi điện trường do sự tích tụ của các electron tự do triệt tiêu trường khôi hiệu ứng Piezoelectric.

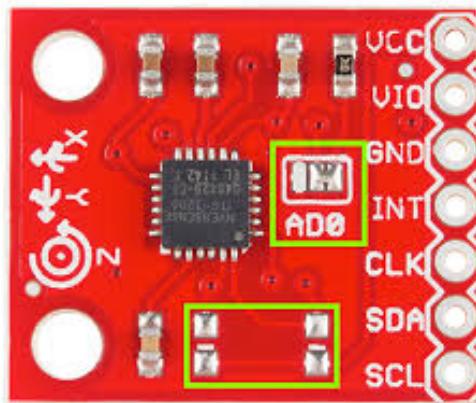


Hình 27: Các kiểu thiết kế cơ học của cảm biến gia tốc

Các kiểu thiết kế cơ học phổ biến của cảm biến gia tốc sử dụng hiệu ứng Piezoelectric:

- **Thiết kế theo chế độ nén:** Thiết kế này rất đơn giản, chỉ bao gồm một bản thiết bị có hiệu ứng Piezoelectric và một khối lượng tham chiếu. Khi xảy ra gia tốc, lực quán tính sẽ làm cho khối lượng tham chiếu ép chặt vào tấm vật liệu áp điện từ đó tạo ra dòng điện.
- **Thiết kế theo chế độ trượt:** Gồm hai loại là trượt trên một tấm phẳng hoặc thiết kế trượt trên một khối hình nhẵn. Về cơ bản thì hai kiểu thiết kế này có cùng nguyên lý hoạt động. Tinh thể được kẹp giữa trụ trung tâm và khối lượng tham chiếu bên ngoài. Khối lượng càng nhiều thì lực trượt tác dụng lên tinh thể càng lớn đối với một gia tốc nhất định. Cấu trúc này giúp gia tốc kế cứng chắc, tạo ra dải tần số cao và do tinh thể không tiếp xúc trực tiếp với cảm biến nên các hiệu ứng biến dạng và chuyển tiếp nhiệt được giảm thiểu.
- **Thiết kế theo chế độ uốn:** thiết kế này sử dụng các tấm tinh thể có hình chữ nhật hoặc hình đĩa. Sự uốn cong của tinh thể có thể xảy ra do khối lượng của chính tinh thể đối lập với gia tốc hoặc để tăng cường sự uốn cong, trọng lượng bổ sung có thể được kẹp hoặc liên kết với tinh thể. Gia tốc kế ở chế độ uốn ít cứng hơn khi so sánh với thiết kế nén hoặc trượt, cung cấp cho chúng dải tần số giới hạn. Ngoài ra, do tinh thể phải chịu mức độ căng áp lực cao nên chúng dễ bị hỏng hơn các loại khác nếu bị sốc hoặc rung quá mức.

2.10.3 Giới thiệu về module cảm biến ITG 3200



Hình 28: Cảm biến gia tốc góc ITG 3200

Module cảm biến ITG 3200 là một loại cảm biến con quay gia tốc 3 trục, sử dụng công nghệ MEMS để đo góc quay và vận tốc góc. Cảm biến này có khả năng cung cấp dữ liệu về các chuyển động xoay trên ba trục không gian X, Y và Z. ITG 3200 có độ chính xác cao và tốc độ đáp ứng nhanh, phù hợp với nhiều ứng dụng yêu cầu đo đặc chuyển động như điều khiển robot, máy bay không người lái (drone), hệ thống ổn định, và các thiết bị đo lường khác.

Các đặc điểm chính của ITG 3200:

- **Cảm biến 3 trục:** Đo lường góc quay và vận tốc góc trên ba trục không gian.
- **Độ phân giải cao:** Cảm biến có thể cung cấp các giá trị đo chính xác với độ phân giải lên đến 14-bit.
- **Tốc độ lấy mẫu:** Cảm biến hỗ trợ tốc độ lấy mẫu lên đến 8 kHz, giúp thu thập dữ liệu nhanh và chính xác.
- **Giao tiếp I2C:** ITG 3200 sử dụng giao tiếp I2C để truyền tải dữ liệu, giúp dễ dàng kết nối với các vi điều khiển và hệ thống nhúng.
- **Nguồn cấp:** Được cấp nguồn từ 2.3V đến 3.4V, giúp tiết kiệm năng lượng trong các ứng dụng di động hoặc thiết bị nhúng.
- **Bộ lọc tích hợp:** Cảm biến này có tích hợp bộ lọc kỹ thuật số để giảm nhiễu và cải thiện độ chính xác của dữ liệu.

Ứng dụng của ITG 3200:

- **Điều khiển ổn định:** ITG 3200 được sử dụng trong các hệ thống ổn định để đo chuyển động quay, giúp duy trì sự cân bằng của các thiết bị như drone và robot.
- **Thiết bị đo lường:** Sử dụng trong các thiết bị đo đặc góc quay chính xác, chẳng hạn như thiết bị đeo thông minh và các hệ thống tự động.
- **Các hệ thống điều khiển:** Được sử dụng trong các hệ thống điều khiển tự động, giúp phát hiện và xử lý chuyển động quay trong thời gian thực.

2.11 Giới thiệu về module bluetooth HC05



Hình 29: module bluetooth HC-05

Module Bluetooth HC-05 là một mô-đun giao tiếp không dây sử dụng chuẩn Bluetooth 2.0, cho phép kết nối các thiết bị điện tử với nhau qua giao thức Bluetooth. Đây là một module phổ biến và dễ sử dụng trong các dự án điện tử và IoT (Internet of Things), giúp các thiết bị như vi điều khiển, cảm biến hoặc các thiết bị di động kết nối với nhau một cách thuận tiện.

Các đặc điểm chính của HC-05:

- **Chế độ hoạt động:** HC-05 hỗ trợ hai chế độ hoạt động: Master (chủ) và Slave (phụ). Chế độ Master cho phép HC-05 kết nối với các thiết bị khác, trong khi chế độ Slave cho phép module kết nối với một thiết bị khác như điện thoại hoặc máy tính.
- **Giao tiếp Serial (UART):** HC-05 sử dụng giao tiếp UART (Universal Asynchronous Receiver/Transmitter) để trao đổi dữ liệu với các vi điều khiển và các thiết bị điện tử khác, giúp dễ dàng tích hợp vào các mạch điện tử.
- **Khoảng cách truyền tín hiệu:** HC-05 có thể truyền tín hiệu Bluetooth trong phạm vi lên tới 10 mét, tùy thuộc vào môi trường và các yếu tố xung quanh.
- **Nguồn cấp:** Module hoạt động với nguồn cung cấp từ 3.3V đến 5V, tương thích với các hệ thống sử dụng vi điều khiển phổ biến như Arduino.
- **Tốc độ truyền dữ liệu:** HC-05 có tốc độ truyền dữ liệu lên đến 3 Mbps, cho phép truyền tải thông tin nhanh chóng và hiệu quả giữa các thiết bị.

Ứng dụng của HC-05:

- **Kết nối không dây:** HC-05 có thể được sử dụng trong các dự án yêu cầu kết nối không dây giữa các thiết bị như Arduino với điện thoại thông minh, máy tính hoặc các thiết bị khác hỗ trợ Bluetooth.



- **Điều khiển từ xa:** Module này được ứng dụng trong các hệ thống điều khiển từ xa, giúp người dùng điều khiển thiết bị điện tử thông qua ứng dụng di động hoặc máy tính.
- **Hệ thống IoT:** HC-05 là một lựa chọn phổ biến trong các hệ thống IoT, nơi các thiết bị cần giao tiếp với nhau qua mạng Bluetooth để truyền dữ liệu hoặc điều khiển từ xa.

2.12 Giới thiệu về ngôn ngữ ký hiệu

Ngôn ngữ ký hiệu là phương tiện giao tiếp chính thức của cộng đồng người khiếm thính, được sử dụng để biểu đạt ý tưởng và cảm xúc thông qua các cử chỉ tay và động tác cơ thể. Mỗi ngôn ngữ ký hiệu có cấu trúc ngữ pháp và từ vựng riêng, khác biệt với ngôn ngữ nói và thường được sử dụng bởi những người không thể nghe hoặc không thể nói.

Ngôn ngữ ký hiệu không chỉ bao gồm các động tác tay mà còn sử dụng biểu cảm khuôn mặt, cử chỉ của các bộ phận cơ thể như mắt, đầu và cơ thể để truyền đạt thông điệp. Điều này giúp ngôn ngữ ký hiệu trở thành một hệ thống giao tiếp toàn diện, có thể biểu đạt đầy đủ các ý tưởng phức tạp như trong ngôn ngữ nói.

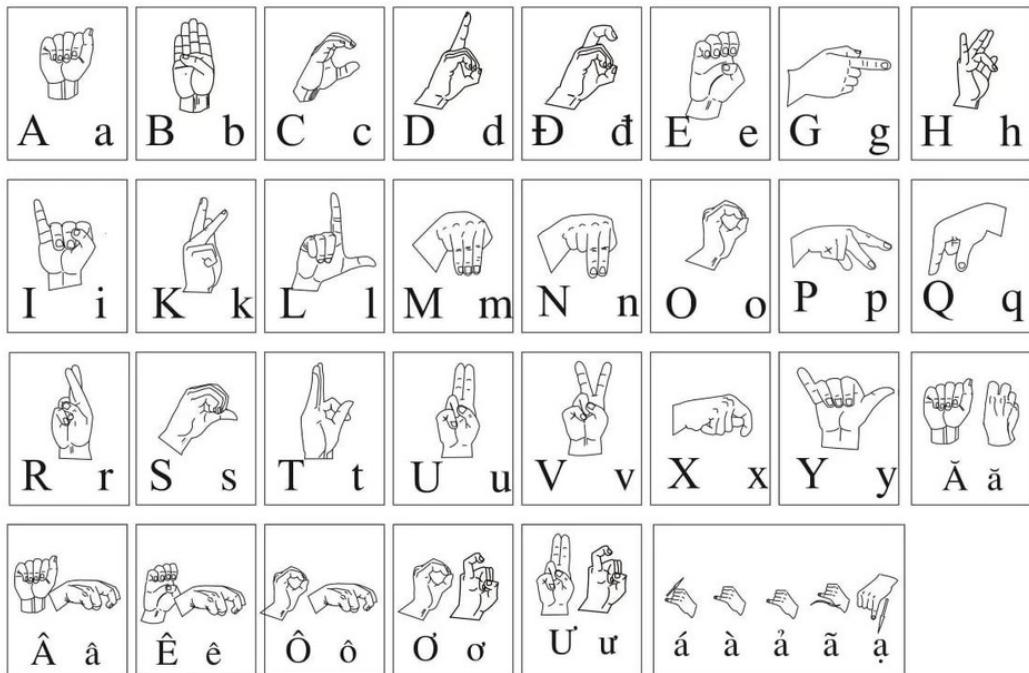
Ngôn ngữ ký hiệu ở Việt Nam đã được hình thành từ rất lâu. Nhưng do trước đây chưa có nhà khoa học nào tìm hiểu, nghiên cứu về nó nên người Việt Nam không nghĩ và đã không xem những dấu hiệu mà người điếc sử dụng là ngôn ngữ. Họ cho rằng đó chỉ là những điệu bộ khua tay của người điếc để cố gắng giao tiếp do thiếu ngôn ngữ. Mãi đến năm 1996, một tiến sĩ ngôn ngữ học người Mỹ James C. Woodward, người đã từng làm việc với William Stokoe tại trường đại học Gallaudet của Mỹ, đã sang Việt Nam thực hiện nghiên cứu về ngôn ngữ ký hiệu của cộng đồng người điếc ở Việt Nam. Theo nghiên cứu của ông, ở Việt Nam hiện có ít nhất 3 ngôn ngữ ký hiệu phổ biến (được cộng đồng người điếc sử dụng nhiều nhất). Ông đã dùng tên của những địa danh này để đặt tên cho 3 ngôn ngữ ký hiệu đó: Ngôn ngữ ký hiệu Hà Nội, ngôn ngữ ký hiệu Hải Phòng, và ngôn ngữ ký hiệu Thành phố Hồ Chí Minh.

Sau đó, đã có thêm những dự án ở Việt Nam: dự án Giáo dục hòa nhập cho trẻ điếc 1998-2001 (Viện Khoa học Giáo dục- tổ chức Pearl S. Buck, Int), dự án Giáo dục trung học và đại học cho người Điếc Việt Nam 2000 cho đến hiện tại (Sở GD-ĐT Đồng Nai và GS TS JAMES C. WOODWARD) để thực hiện việc thu thập lại những dấu hiệu của người điếc Việt Nam và tìm hiểu về ngữ pháp của ngôn ngữ này. Công việc này đã kích thích thêm nhiều nhà khoa học ở Việt Nam cũng bắt đầu tìm hiểu về ngữ pháp của ngôn ngữ ký hiệu Việt Nam.

Để một người có thể diễn đạt được ngôn ngữ ký hiệu đòi hỏi một hoặc nhiều yếu tố dưới đây:

- **Vị trí làm kí hiệu:** Vị trí làm kí hiệu là vị trí của bàn tay so với cơ thể khi làm kí hiệu. Vị trí làm kí hiệu khác nhau thể hiện những ý nghĩa khác nhau. Khoảng không gian để thể hiện kí hiệu được giới hạn từ đỉnh đầu, khoảng không gian phía trước cơ thể mở rộng đến độ rộng của hai khuỷu tay ở hai phía, lưng và hông.
- **Hình dạng bàn tay:** Hình dạng bàn tay là các hình dạng khác nhau của bàn tay khi thực hiện kí hiệu.
- **Sự chuyển động của tay:** Sự chuyển động của tay là những cử động của tay khi làm kí hiệu, bao gồm chuyển động đơn (một chuyển động trong một lần làm kí hiệu), và chuyển động kép (nhiều chuyển động trong một lần làm kí hiệu). Nhìn vào mũi tên trong hình vẽ của kí hiệu chúng ta biết được sự chuyển động của tay.
- **Chiều hướng của tay:** Chiều hướng của tay khi làm kí hiệu bao gồm chiều hướng của lòng bàn tay và chiều hướng của các ngón tay.
- **Sự diễn tả không bằng tay:** Sự diễn tả không bằng tay là những cử chỉ, điệu bộ, nét mặt, cử động của cơ thể kèm theo.

Bảng chữ cái bằng bàn tay



Số tự nhiên

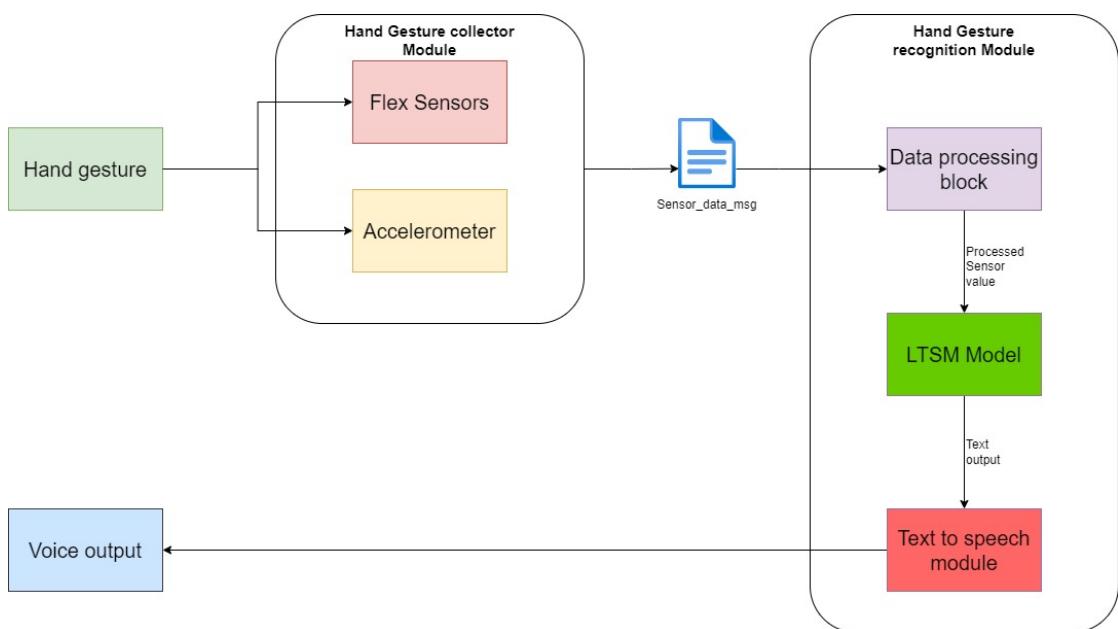


Hình 30: Bảng ngôn ngữ ký hiệu

3 Kiến trúc hệ thống

Hệ thống được thiết kế với hai module với chức năng của từng module được mô tả dưới đây:

- Module thu thập dữ liệu cử chỉ:** Đầu vào của khối này là dữ liệu cử chỉ được thu thập từ các cảm biến uốn cong và cảm biến gia tốc. Dữ liệu sau khi thu thập sẽ được đóng gói và gửi về module nhận dạng cử chỉ.
- Module nhận dạng cử chỉ** Dữ liệu được gửi về từ module thu thập sẽ được xử lý, phân loại và định hình lại dữ liệu để đưa vào mô hình LSTM. Đầu ra của mô hình LSTM sẽ là kết quả dự đoán cử chỉ dưới dạng văn bản và sẽ được chuyển đổi thành giọng nói qua khối Text-to-speech.
- Module chuyển văn bản thành giọng nói:** Khi nhận được output đầu ra, module này có chức năng chuyển đổi từ dạng one-hot-code output sang dạng văn bản, từ văn bản đó sử dụng thư viện pyttsx3 để phát ra âm thanh như mong muốn đã được cấu hình trước đó trong file config



Hình 31: Kiến trúc tổng quan của hệ thống

Mô tả luồng dữ liệu Hand Gesture (Cử chỉ tay): Cử chỉ tay là đầu vào của hệ thống. Các cảm biến uốn cong và gia tốc kê trong mô-đun thu thập cử chỉ tay sẽ ghi nhận và gửi dữ liệu này đến khối xử lý dữ liệu.

Sensor Data Message (Thông điệp dữ liệu cảm biến): Dữ liệu từ cảm biến được đóng gói thành thông điệp và truyền tới mô-đun nhận diện cử chỉ tay.

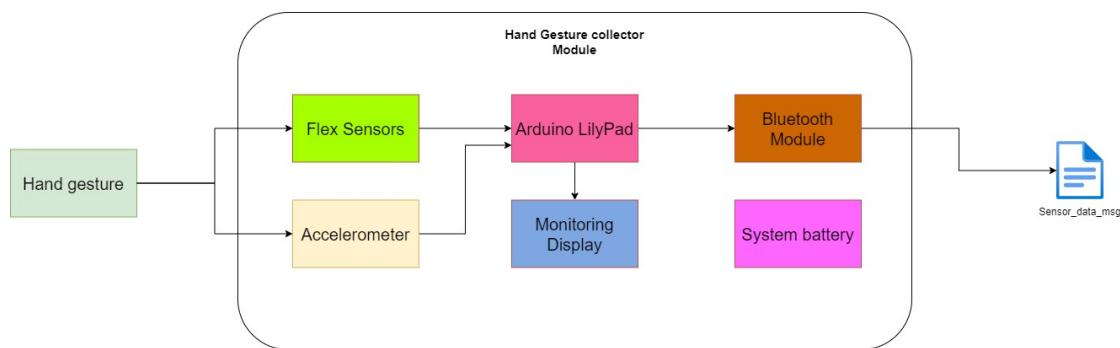
Processed Sensor Value (Giá trị cảm biến đã xử lý): Dữ liệu cảm biến được xử lý trong khối xử lý dữ liệu và các đặc trưng được trích xuất.

Text Output (Đầu ra văn bản): Mô hình LSTM dự đoán cử chỉ tay và chuyển đổi nó thành văn bản.

Voice Output (Đầu ra giọng nói): Văn bản từ mô hình LSTM được chuyển đổi thành giọng nói và phát ra ngoài thông qua mô-đun chuyển văn bản thành giọng nói.

Hệ thống này bao gồm hai mô-đun chính và một module phụ: một mô-đun thu thập cử chỉ tay, một mô-đun nhận diện cử chỉ tay và module phát âm thanh. Mô-đun thu thập cử chỉ tay sử dụng các cảm biến uốn cong và gia tốc kế để ghi nhận dữ liệu cử chỉ tay, sau đó truyền dữ liệu này đến mô-đun nhận diện cử chỉ tay. Tại đây, dữ liệu cảm biến được xử lý và đưa vào mô hình LSTM để dự đoán cử chỉ tay dưới dạng văn bản. Cuối cùng, văn bản này được chuyển đổi thành giọng nói dựa trên module phát âm thanh.

3.1 Module thu thập dữ liệu cử chỉ



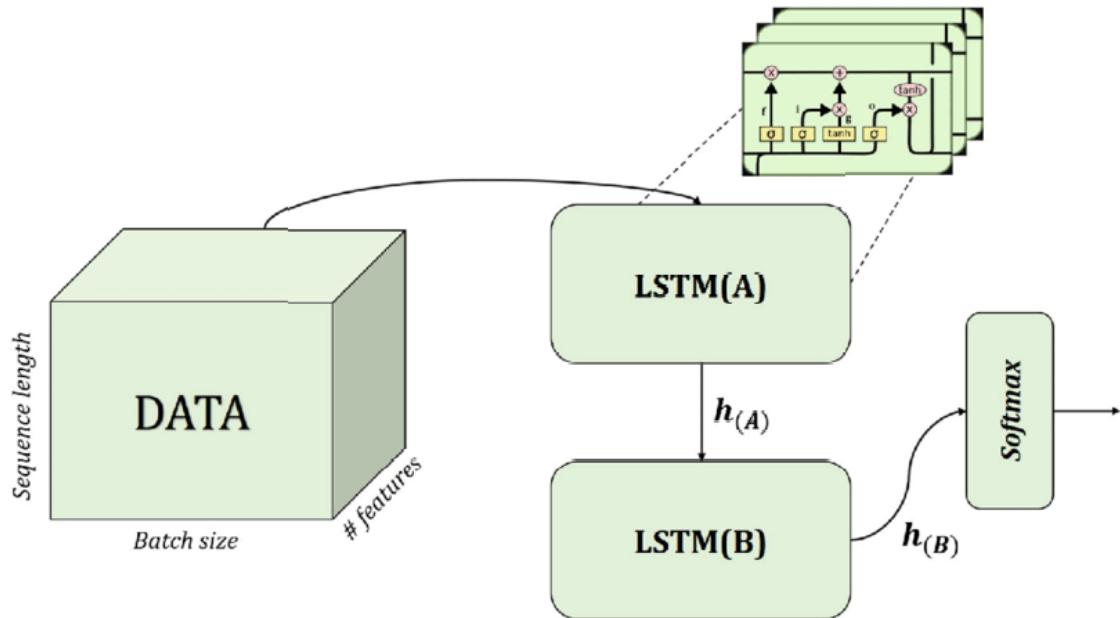
Hình 32: Kiến trúc module thu thập dữ liệu cử chỉ

Sơ đồ mô tả một hệ thống thu thập cử chỉ tay, gồm các thành phần sau:

- 1. Hand gesture (Cử chỉ tay): Đây là đầu vào của hệ thống, là các cử chỉ tay cần được nhận diện.
- 2. Flex Sensors (Cảm biến uốn cong): Nhận tín hiệu từ cử chỉ tay và chuyển tiếp đến Arduino LilyPad.
- 3. Accelerometer (Gia tốc kế): Cũng nhận tín hiệu từ cử chỉ tay và chuyển tiếp đến Arduino LilyPad.
- 4. Arduino LilyPad: Xử lý dữ liệu nhận được từ Flex Sensors và Accelerometer. Sau đó, chuyển tiếp dữ liệu này đến các thành phần khác.
- 5. Monitoring Display (Màn hình giám sát): Hiển thị dữ liệu và thông tin từ Arduino LilyPad để người dùng có thể theo dõi.
- 6. Bluetooth Module (Mô-đun Bluetooth): Truyền dữ liệu từ Arduino LilyPad tới file data
- 7. System Battery (Pin hệ thống): Cung cấp năng lượng cho toàn bộ hệ thống.

Dữ liệu được thu thập từ cử chỉ tay sẽ được xử lý và truyền đi qua mô-đun Bluetooth, sau đó lưu vào file data

3.2 Module nhận dạng cử chỉ



Hình 33: Kiến trúc model dự đoán cử chỉ

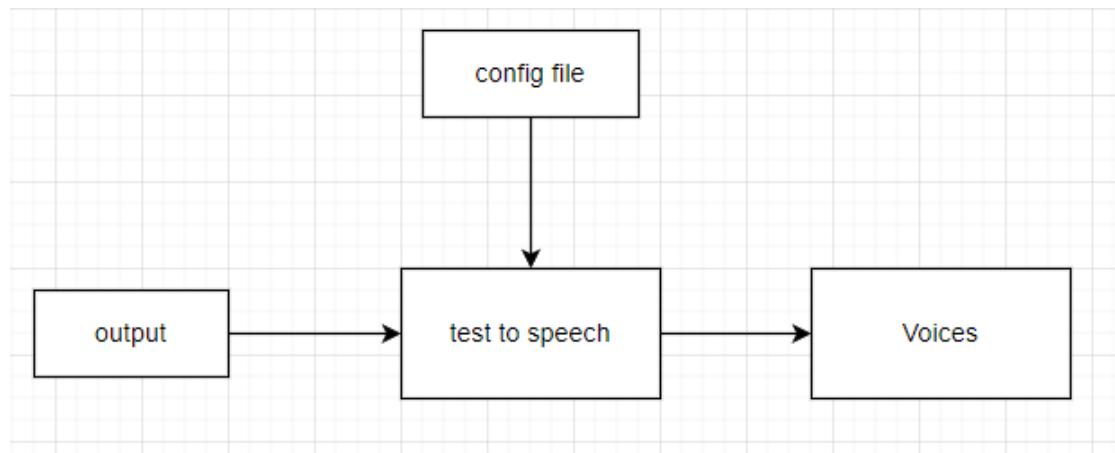
Sơ đồ mô tả một mô hình học sâu sử dụng LSTM (Long Short-Term Memory) để dự đoán xác suất của các hành động, như mua hoặc bán. Dưới đây là mô tả chi tiết của từng thành phần trong mô hình:

- 1. DATA (Dữ liệu):
 - Đầu vào của mô hình là một khối dữ liệu ba chiều với các thông số:
 - Sequence length: Độ dài chuỗi thời gian của dữ liệu.
 - Batch size: Kích thước lô, số lượng chuỗi thời gian trong một lô.
 - Features: Số lượng đặc trưng trong mỗi bước thời gian của chuỗi.
- 2. LSTM(A):
 - Dữ liệu đầu vào được đưa vào lớp LSTM đầu tiên, gọi là LSTM(A). LSTM là một loại mạng nơ-ron hồi tiếp được thiết kế để xử lý và dự đoán các chuỗi dữ liệu thời gian.
- 3. $h(A)$:
 - Đầu ra của LSTM(A) là một vector ẩn (hidden vector), ký hiệu là $h(A)$. Vector này lưu trữ thông tin trạng thái ẩn của LSTM sau khi xử lý dữ liệu đầu vào.
- 4. LSTM(B):
 - Vector ẩn $h(A)$ sau đó được đưa vào một lớp LSTM thứ hai, gọi là LSTM(B). Lớp này tiếp tục xử lý dữ liệu và tạo ra một vector ẩn mới, ký hiệu là $h(B)$.
- 5. $h(B)$:
 - Đầu ra của LSTM(B) là vector ẩn $h(B)$, lưu trữ thông tin trạng thái ẩn sau khi dữ liệu được xử lý qua hai lớp LSTM.

- 6. Softmax:
 - Vector ẩn $h(B)$ được đưa vào một lớp Softmax. Lớp này chuyển đổi vector thành các xác suất để dự đoán các hành động cụ thể.
- 7. Output:
 - Đầu ra cuối cùng của mô hình là các xác suất được tạo ra bởi lớp Softmax, ví dụ như label 0,1,2 được mã hóa dưới dạng one-hot-code

Mô hình này có thể được sử dụng cho các ứng dụng như dự đoán thị trường chứng khoán hoặc các bài toán khác liên quan đến chuỗi thời gian, nơi việc xác định các hành động dựa trên dữ liệu lịch sử là quan trọng. Sơ đồ này minh họa cách dữ liệu được xử lý qua các lớp LSTM và sau đó chuyển đổi thành các xác suất đầu ra thông qua lớp Softmax.

3.3 Module phát âm thanh



Hình 34: Kiến trúc module chuyển đầu ra dự đoán thành giọng nói

Sơ đồ mô tả module phát âm thanh sau khi dự đoán đúng một hành động sử dụng thư viện text-to-speech pyttsx3

Thư viện pyttsx3 cung cấp một giao diện đơn giản và nhất quán cho việc chuyển đổi văn bản thành giọng nói trên nhiều nền tảng. Kiến trúc của nó được thiết kế để trừu tượng hóa các chi tiết đặc thù của nền tảng, cho phép các nhà phát triển tập trung vào chức năng TTS ở mức cao. Các thành phần cốt lõi bao gồm bộ máy, trình điều khiển và cơ chế cấu hình, hoạt động cùng nhau để cung cấp tổng hợp giọng nói liền mạch và tùy chỉnh. Thiết kế của thư viện đảm bảo tính tương thích, linh hoạt và dễ sử dụng, biến nó thành công cụ mạnh mẽ để thêm khả năng TTS vào các ứng dụng Python.

Cụ thể âm thanh được phát ra khi có một output chỉ index trong file config giọng nói trước đó có sẵn, kiến trúc này sẽ lấy chính xác vị trí âm thanh giọng nói phát ra như mong muốn.



4 Hiện thực hệ thống

4.1 Hiện thực phần cứng

Hệ thống được hiện thực phần cứng thông qua sự tích hợp của nhiều thành phần điện tử, đảm bảo khả năng thu thập, xử lý tín hiệu và truyền dữ liệu một cách hiệu quả. Mục tiêu chính của phần cứng là cung cấp nền tảng ổn định, chính xác cho việc nhận diện cử chỉ ngôn ngữ ký hiệu.

4.1.1 Các thành phần phần cứng chính

- Cảm biến Flex:

- *Chức năng:* Do độ uốn cong của các ngón tay, phản ánh cử chỉ của bàn tay. Khi uốn cong, giá trị điện trở của cảm biến thay đổi, từ đó tạo ra tín hiệu điện áp tương ứng.
- *Đặc điểm:*
 - * Giá trị điện trở thay đổi từ $25\text{k}\Omega$ (khi thẳng) đến hơn $100\text{k}\Omega$ (khi uốn cong 90°).
 - * Được kết nối với vi điều khiển qua mạch chia điện áp để chuyển đổi tín hiệu thành dữ liệu số.

- Vi điều khiển Arduino LilyPad:

- *Chức năng:* Là trung tâm xử lý tín hiệu từ cảm biến Flex, cảm biến con quay hồi chuyển và điều khiển các thành phần khác.
- *Đặc điểm kỹ thuật:*
 - * Sử dụng vi điều khiển ATmega328 với 14 chân I/O số, 6 chân đầu vào analog.
 - * Kích thước nhỏ gọn, phù hợp với các thiết bị đeo.

- Mô-đun Bluetooth HC-05:

- *Chức năng:* Truyền dữ liệu không dây từ hệ thống đến các thiết bị đầu cuối như điện thoại hoặc máy tính để xử lý và hiển thị.
- *Đặc điểm:*
 - * Hỗ trợ giao tiếp trong phạm vi 10m.
 - * Giao tiếp với Arduino thông qua các chân TX và RX.

- Cảm biến con quay hồi chuyển ITG-3200:

- *Chức năng:* Do tốc độ và góc quay của bàn tay, hỗ trợ nhận diện cử chỉ một cách chính xác hơn.
- *Đặc điểm:*
 - * Giao tiếp qua giao thức I2C.
 - * Được sử dụng để bổ sung thông tin về chuyển động của bàn tay.

- Mô-đun TCA9548A:

- *Chức năng:* Là bộ chuyển mạch I2C 8 kênh, cho phép kết nối nhiều thiết bị I2C với cùng địa chỉ, tránh xung đột khi sử dụng nhiều cảm biến.
- *Đặc điểm:*

- * Hỗ trợ tối đa 8 thiết bị trên một bus I2C.
- * Hoạt động ở điện áp từ 1.8V đến 5V.

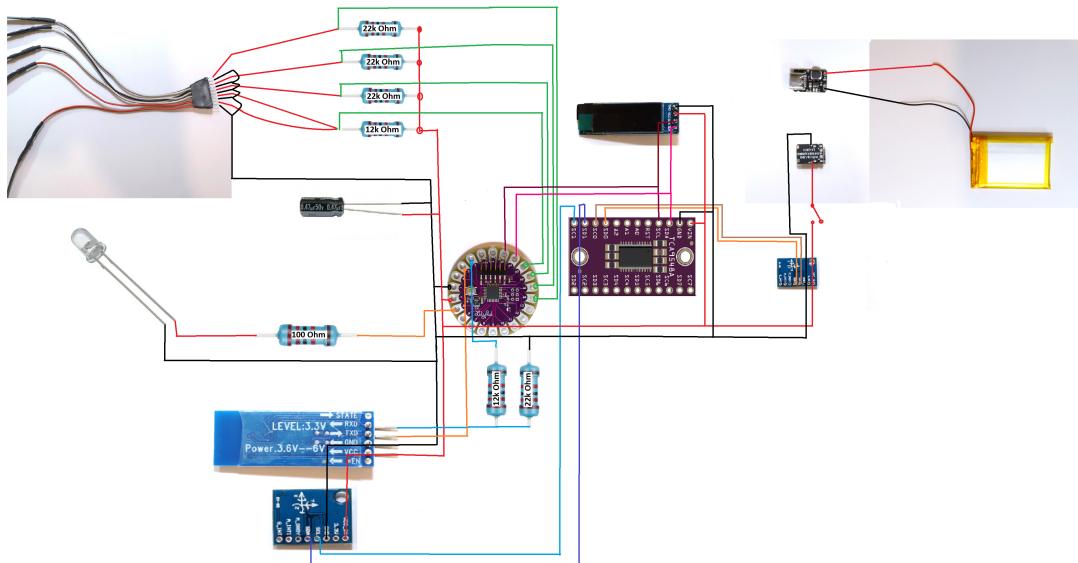
- **Nguồn pin LiPo (Lithium Polymer):**

- *Chức năng:* Cung cấp năng lượng cho toàn bộ hệ thống.
- *Dặc điểm:*
 - * Điện áp đầu ra danh định 3.7V.
 - * Nhỏ gọn, dễ dàng tích hợp vào các thiết bị di động.

- **LED và điện trở:**

- LED: Hiển thị trạng thái hoạt động của hệ thống.
- Điện trở: Được sử dụng để điều chỉnh tín hiệu từ cảm biến Flex và giới hạn dòng qua LED.

4.1.2 Sơ đồ kết nối mạch



Hình 35: Sơ đồ kết nối mạch

- **Kết nối cảm biến Flex với Arduino LilyPad:**

- Các cảm biến Flex được nối với chân analog của Arduino thông qua mạch chia điện áp (diện trở 22k Ω và 12k Ω).
- Tín hiệu điện trở từ cảm biến được chuyển thành điện áp và đưa vào Arduino để xử lý.

- **Kết nối mô-đun Bluetooth HC-05 với Arduino LilyPad:**



- Chân TX của HC-05 nối với RX của Arduino và ngược lại (RX của HC-05 nối với TX của Arduino).

- Chân VCC và GND của HC-05 được nối với nguồn từ Arduino.

- **Kết nối cảm biến ITG-3200 với TCA9548A:**

- Cảm biến ITG-3200 được nối vào các kênh riêng biệt của TCA9548A qua giao thức I2C.

- SDA và SCL của cảm biến được kết nối với các chân tương ứng trên TCA9548A.

- **Kết nối TCA9548A với Arduino LilyPad:**

- Chân SDA và SCL của TCA9548A được nối với SDA và SCL trên Arduino.

- Chân VCC và GND của TCA9548A được nối với nguồn từ Arduino.

- **Kết nối nguồn pin LiPo và LED:**

- Pin LiPo kết nối với cổng nguồn trên Arduino.

- LED được nối qua điện trở với chân I/O của Arduino để hiển thị trạng thái hoạt động.

4.1.3 Nhiệm vụ của phần cứng

- **Thu thập dữ liệu:**

- Cảm biến Flex cung cấp thông tin về độ uốn cong của ngón tay.

- Cảm biến ITG-3200 bô sung dữ liệu về chuyển động và góc quay của bàn tay.

- **Xử lý và truyền dữ liệu:**

- Arduino LilyPad xử lý dữ liệu từ các cảm biến và truyền đến các thiết bị đầu cuối qua Bluetooth HC-05.

- **Quản lý giao tiếp I2C:**

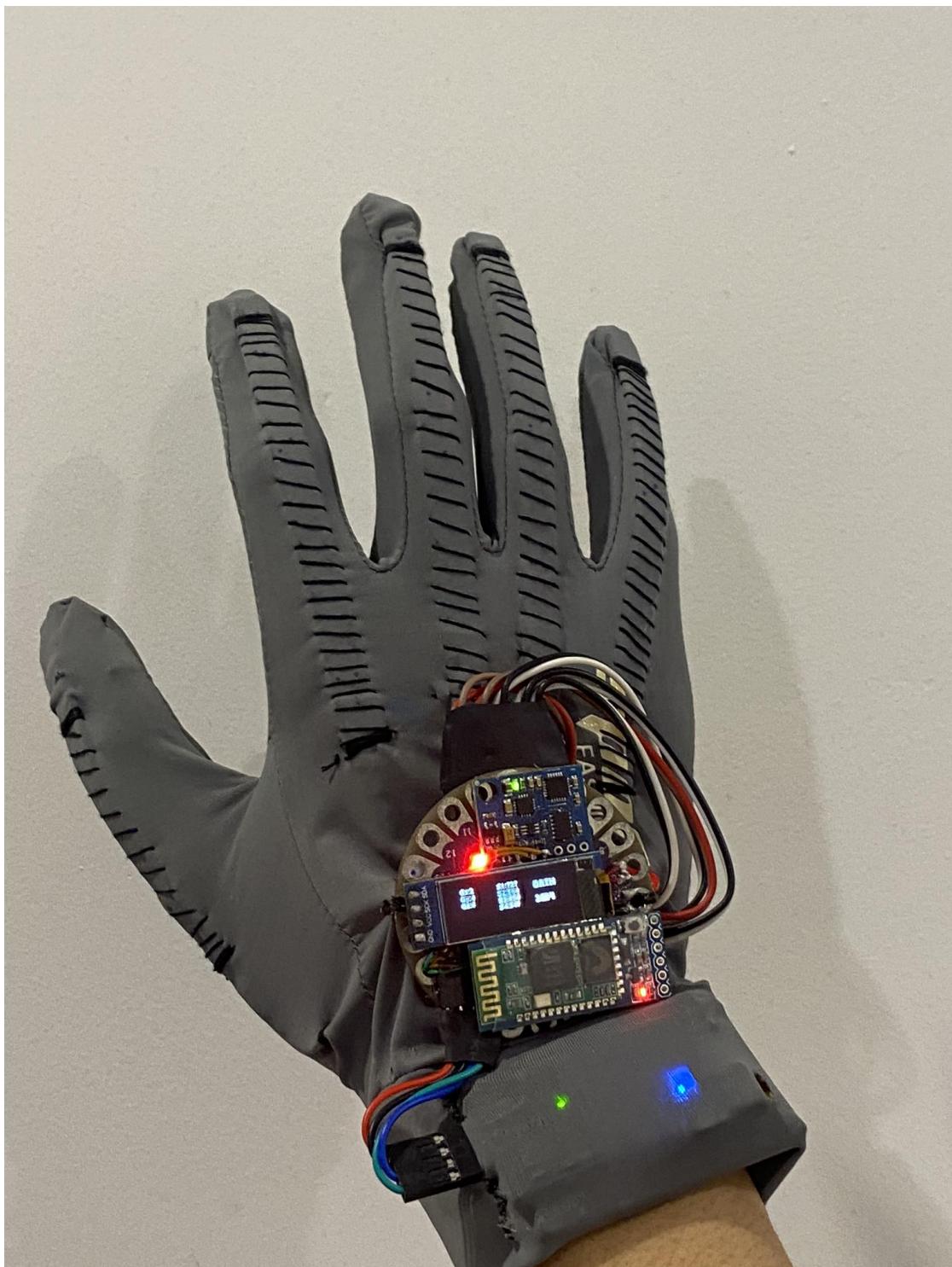
- TCA9548A giải quyết vấn đề xung đột địa chỉ I2C khi sử dụng nhiều cảm biến.

- **Hiển thị trạng thái:**

- LED hiển thị trạng thái hoạt động, màn hình OLED cung cấp thông tin trực quan.

4.1.4 Tóm tắt

Hệ thống phần cứng được thiết kế để thu thập, xử lý và truyền dữ liệu phục vụ nhận diện cử chỉ ngôn ngữ ký hiệu. Các cảm biến Flex và ITG-3200 cung cấp thông tin về độ uốn cong và chuyển động bàn tay, trong khi Arduino LilyPad đóng vai trò trung tâm xử lý và điều khiển các thành phần khác. Mô-đun TCA9548A hỗ trợ quản lý giao tiếp I2C hiệu quả, và Bluetooth HC-05 đảm bảo truyền dữ liệu không dây. Toàn bộ hệ thống được tối ưu hóa về hiệu suất, đảm bảo hoạt động chính xác và ổn định trong thực tế.



Hình 36: Hình ảnh phần cứng sau khi hoàn thiện



4.2 Xây dựng bộ dữ liệu

Phần xây dựng bộ dữ liệu tập trung vào thu thập và xử lý tín hiệu từ phần cứng nhằm tạo ra tập dữ liệu phục vụ huấn luyện và kiểm tra mô hình nhận diện cử chỉ ngôn ngữ ký hiệu.

4.2.1 Thu thập dữ liệu từ phần cứng

Dữ liệu được thu thập từ các cảm biến gia tốc và cảm biến Flex, kết nối với vi điều khiển Arduino LilyPad và truyền qua mô-đun Bluetooth HC-05 đến máy tính.

- **Cấu trúc tín hiệu đầu vào:**

- Gia tốc trên các trục X, Y, Z: Cung cấp thông tin về chuyển động của bàn tay.
- Tín hiệu analog từ cảm biến Flex 1, 2, 3, 4: Độ uốn cong của các ngón tay.
- Cách kết nối:
 - * Flex 1 và Flex 2 (ngón trỏ và ngón cái) được nối song song, sau đó nối với chân A0 của Arduino.
 - * Các Flex còn lại được nối tuần tự vào các chân analog khác (A1, A2, A3).

- **Định dạng dữ liệu truyền:**

- Dữ liệu được gửi qua Bluetooth với định dạng chuỗi:

13:38:46.492 -> 3	-1	7	711	516	663	514
13:38:46.492 -> 3	-1	7	711	516	663	514
13:38:46.492 -> 3	-1	7	711	516	663	514
13:38:46.564 -> 3	-1	7	711	516	663	514
13:38:46.564 -> 3	-1	7	711	516	663	514
13:38:46.564 -> 3	-1	7	711	516	663	514
13:38:46.564 -> 3	-1	7	711	516	663	514
13:38:46.601 -> 3	-1	7	711	516	663	514

Hình 37: Dạng dữ liệu được gửi

- Trong đó:
 - * Ba giá trị đầu là gia tốc trên các trục X, Y, Z.
 - * Bốn giá trị tiếp theo là tín hiệu analog từ các cảm biến Flex.
- Tần suất gửi dữ liệu là mỗi 10ms một lần.

- **Thu thập dữ liệu qua chương trình Python:**

- Chương trình Python giao tiếp với mô-đun Bluetooth HC-05 qua cổng COM.
- Tín hiệu nhận được từ Bluetooth được đọc, phân tách và lưu trữ dưới dạng có cấu trúc.



4.2.2 Xử lý và lưu trữ dữ liệu

- **Xử lý dữ liệu:**

- Các giá trị nhận được từ phần cứng được kiểm tra định dạng và chuyển đổi sang kiểu số nguyên.
- Mỗi hàng có 240*7 cột dữ liệu.

- **Lưu trữ dữ liệu:**

- **Dữ liệu huấn luyện và kiểm tra:**

- * Các chuỗi dữ liệu được lưu vào các tệp riêng biệt:
 - **trainx:** Dữ liệu đầu vào cho huấn luyện.
 - **testx:** Dữ liệu đầu vào cho kiểm tra.
 - * Nhãn tương ứng (**trainy, testy**) được lưu kèm theo.

- Dữ liệu được tổ chức dưới dạng từng dòng, mỗi dòng là một chuỗi giá trị đầy đủ.

- **Thông số mô hình thu thập:**

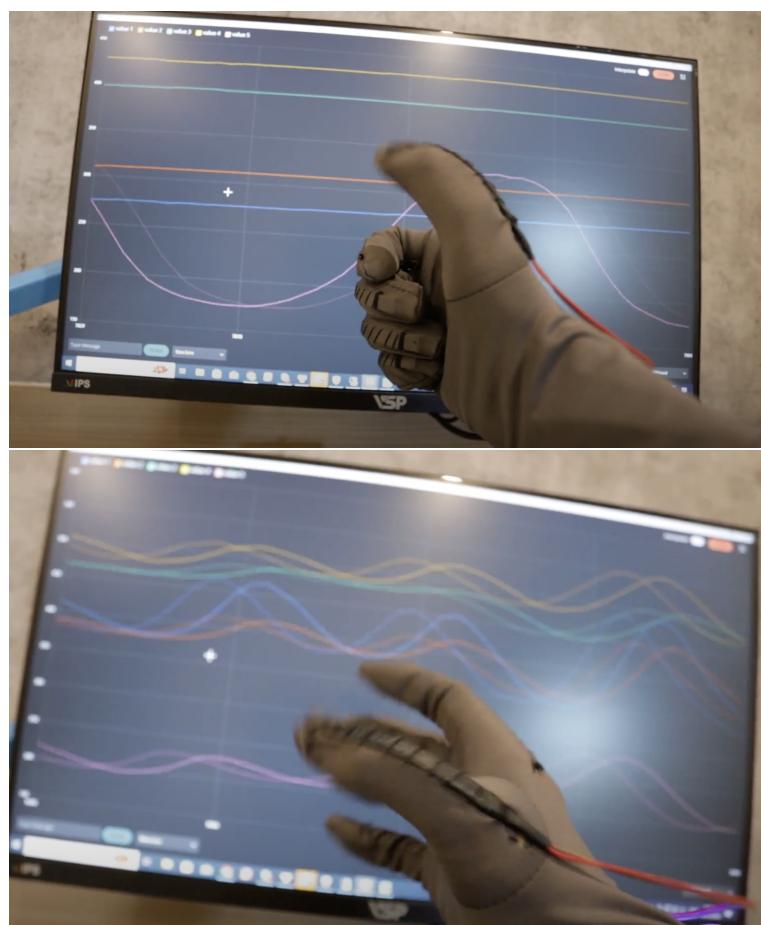
- Số mẫu huấn luyện: 200.
- Số mẫu kiểm tra: 50.
- Tổng thời gian thu thập dữ liệu được xác định bởi số mẫu và tần suất gửi tín hiệu.

4.2.3 Tóm tắt

Quy trình xây dựng bộ dữ liệu đảm bảo thu thập đầy đủ và chính xác tín hiệu từ phần cứng, với cấu trúc rõ ràng và dễ sử dụng. Bộ dữ liệu này là cơ sở quan trọng để huấn luyện mô hình học máy, hỗ trợ nhận diện chính xác các cử chỉ ngôn ngữ ký hiệu.

Bảng 2: Định nghĩa các hành động

Label 0	Xin chào
Label 1	Tôi
Label 2	Tên
Label 3	Tiến
Label 4	Tạm biệt



Hình 38: Quá trình lấy dữ liệu



4.3 Định dạng dữ liệu trước khi đưa vào học máy

Phần này tập trung vào việc chuẩn bị và chuyển đổi dữ liệu từ tệp lưu trữ thành định dạng phù hợp để sử dụng trong mô hình học máy. Quy trình được thực hiện như sau:

4.3.1 Đọc và tổ chức dữ liệu

- **Dữ liệu đầu vào:** Dữ liệu được đọc từ các tệp:
 - **trainx** và **testx**: Chứa dữ liệu đầu vào từ các cảm biến.
 - **trainy** và **testy**: Chứa nhãn tương ứng cho các mẫu dữ liệu đầu vào.
- **Quy trình xử lý:**
 - Mỗi dòng dữ liệu trong tệp đầu vào chứa 1680 giá trị, được chia thành 7 chuỗi (tương ứng với 7 giá trị cảm biến) và 240 bước thời gian.
 - Dữ liệu được tách ra và chuyển đổi thành ma trận 3 chiều với kích thước (số mẫu, 240, 7).

4.3.2 Định dạng dữ liệu đầu vào

- **Kết hợp giá trị cảm biến:**
 - Sử dụng hàm `dstack` để kết hợp dữ liệu từ 7 cảm biến thành ma trận.
 - Mỗi mẫu có định dạng:
$$(240, 7)$$
Trong đó:
 - * 240: Số bước thời gian (sequence length).
 - * 7: Số lượng đặc trưng (gia tốc các trục X, Y, Z và tín hiệu từ các cảm biến Flex).
- **Định dạng đầu ra:**
 - Chuyển nhãn (**trainy** và **testy**) sang định dạng one-hot encoding bằng hàm `to_categorical`.
 - Kích thước nhãn sau khi chuyển đổi:
$$(\text{số mẫu}, \text{số lớp})$$

4.3.3 Chuẩn hóa dữ liệu

- Sử dụng các công cụ từ thư viện `sklearn` để chuẩn hóa dữ liệu:
 - **StandardScaler**: Chuẩn hóa các giá trị về trung bình 0 và phương sai 1.
 - **MinMaxScaler**: Dưa dữ liệu về khoảng [0, 1].

4.3.4 Chuẩn bị dữ liệu cho mô hình LSTM

Dữ liệu được tổ chức thành hai tập chính:

- **Tập huấn luyện (`trainX` và `trainy`):**
 - Đầu vào: Ma trận (số mẫu huấn luyện, 240, 7).
 - Đầu ra: Ma trận (số mẫu huấn luyện, số lớp) (one-hot encoding).



- **Tập kiểm tra (testX và testy):**

- Đầu vào: Ma trận (số mẫu kiểm tra, 240, 7).
- Đầu ra: Ma trận (số mẫu kiểm tra, số lớp).

4.3.5 Định nghĩa đầu vào và đầu ra của mô hình

- **Đầu vào:**

- Chuỗi thời gian có chiều (240, 7), đại diện cho các tín hiệu liên tục từ cảm biến.

- **Đầu ra:**

- Xác suất dự đoán của từng lớp, biểu diễn bằng một vector có chiều (số lớp).

4.3.6 Tóm tắt

Quy trình định dạng dữ liệu bao gồm các bước xử lý từ tệp dữ liệu thô đến định dạng đầu vào/đầu ra phù hợp với mô hình LSTM. Điều này đảm bảo dữ liệu có cấu trúc rõ ràng và đáp ứng yêu cầu của mô hình học máy, góp phần nâng cao hiệu quả và độ chính xác trong quá trình huấn luyện.



4.4 Xây dựng mô hình học máy

Mô hình học máy được xây dựng sử dụng kiến trúc LSTM (Long Short-Term Memory) nhằm xử lý dữ liệu chuỗi thời gian từ các cảm biến và nhận diện chính xác các cử chỉ ngôn ngữ ký hiệu.

4.4.1 Cấu trúc của mô hình

Mô hình được xây dựng với các thành phần chính sau:

- **LSTM Layer:**

- Lớp LSTM đầu tiên với 128 đơn vị (units), nhận đầu vào có kích thước (240, 7) tương ứng với 240 bước thời gian và 7 đặc trưng từ cảm biến.
- Lớp này có tổng cộng **69,632 tham số** được tối ưu trong quá trình huấn luyện.

- **Dropout Layer:**

- Lớp Dropout với tỷ lệ rọi rụng là 50% giúp giảm thiểu hiện tượng overfitting trong quá trình huấn luyện.
- Lớp này không bổ sung thêm tham số nào.

- **Flatten Layer:**

- Lớp Flatten chuyển đổi đầu ra từ LSTM thành vector phẳng, kích thước (128).
- Lớp này không có tham số học.

- **Dense Layer:**

- Lớp Dense ẩn với 64 đơn vị (units) và hàm kích hoạt ReLU.
- Lớp này bổ sung **8,256 tham số**.

- **Output Dense Layer:**

- Lớp đầu ra với 5 đơn vị tương ứng với số lớp trong bài toán phân loại.
- Sử dụng hàm kích hoạt softmax để chuyển đổi đầu ra thành xác suất.
- Lớp này bổ sung **325 tham số**.

4.4.2 Tổng số tham số

- Tổng số tham số học được trong mô hình: **78,213 tham số**.
- Mô hình không chứa tham số không thể học (non-trainable).

4.4.3 Quy trình xây dựng mô hình

- **Khởi tạo mô hình:**

- Mô hình được khởi tạo sử dụng API tuần tự (Sequential) của Keras.

- **Thêm các lớp vào mô hình:**

- Lớp LSTM đầu tiên nhận đầu vào có kích thước (240, 7) và trả về đầu ra có kích thước (128).



- Thêm lớp Dropout với tỷ lệ 0.5.
- Thêm lớp Flatten để làm phẳng đầu ra của LSTM.
- Thêm lớp Dense ẩn với 64 đơn vị và hàm kích hoạt ReLU.
- Thêm lớp Dense đầu ra với 5 đơn vị và hàm kích hoạt softmax.

- **Biên dịch mô hình:**

- Sử dụng hàm mất mát `categorical_crossentropy` để tối ưu hóa bài toán phân loại.
- Trình tối ưu `adam` được lựa chọn để đảm bảo quá trình học nhanh chóng và hiệu quả.
- Đo lường hiệu suất bằng độ chính xác (`accuracy`).

- **Huấn luyện mô hình:**

- Số epoch: 300.
- Kích thước batch: 4,500 mẫu mỗi lần cập nhật trọng số.
- Bộ dữ liệu đầu vào và nhãn được đưa vào quá trình huấn luyện thông qua hàm `model.fit()`.

4.4.4 Tóm tắt mô hình

Bảng 3: Bảng tóm tắt mô hình

Layer (type)	Output Shape	Param #
LSTM	(None, 128)	69,632
Dropout	(None, 128)	0
Flatten	(None, 128)	0
Dense	(None, 64)	8,256
Dense	(None, 5)	325
Total		78,213

- Tổng số tham số: 78,213
- Kích thước bộ tham số: 305.52 KB

4.4.5 Tóm tắt

Mô hình LSTM được xây dựng và tối ưu hóa để xử lý dữ liệu chuỗi thời gian từ cảm biến. Với cấu trúc đơn giản nhưng hiệu quả, mô hình có khả năng phân loại chính xác các cử chỉ ngôn ngữ ký hiệu, đồng thời đảm bảo tính tổng quát khi áp dụng trên tập dữ liệu kiểm tra.

5 Các phương pháp cải tiến

5.1 Cửa sổ trượt (Sliding Window)

Cửa sổ trượt là một kỹ thuật quan trọng trong việc xử lý dữ liệu chuỗi thời gian, đặc biệt khi làm việc với các mô hình học máy yêu cầu dữ liệu có cấu trúc dạng chuỗi (sequence). Kỹ thuật này giúp chia nhỏ dữ liệu lớn thành các chuỗi con với chiều dài cố định, giúp mô hình dễ dàng học được các đặc trưng từ các chuỗi dữ liệu liên tục.



Hình 39: Kỹ thuật Sliding Window

5.1.1 Khái niệm về cửa sổ trượt

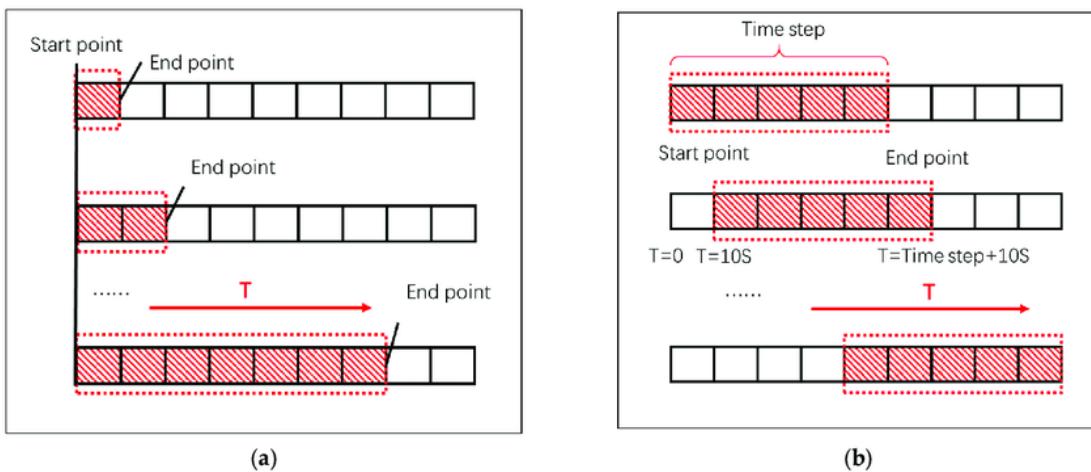
Cửa sổ trượt sử dụng một cửa sổ cố định để trượt qua dữ liệu chuỗi thời gian, tạo ra các phân đoạn nhỏ (windows) để mô hình học máy có thể xử lý. Mỗi cửa sổ trượt chứa một số mẫu nhất định, được gọi là *sequence length*. Khi cửa sổ trượt di chuyển qua dữ liệu, các mẫu trong mỗi cửa sổ được sử dụng để huấn luyện mô hình.

- **Sequence length:** Số lượng mẫu trong mỗi cửa sổ.
- **Step size:** Số mẫu mà cửa sổ trượt di chuyển qua mỗi lần.

5.1.2 Cách thức hoạt động của cửa sổ trượt

Cửa sổ trượt hoạt động theo cách chia dữ liệu thành các cửa sổ nhỏ, mỗi cửa sổ chứa một chuỗi thời gian dài một số bước thời gian xác định. Sau đó, cửa sổ này di chuyển qua các bước tiếp theo để tạo ra các mẫu huấn luyện liên tiếp.

- Dữ liệu gốc có dạng chuỗi thời gian dài (T).
- Cửa sổ trượt có kích thước cố định (ví dụ: 240 bước thời gian).
- Mỗi lần trượt, cửa sổ di chuyển qua một bước xác định (step size, ví dụ: 40 mẫu).



Hình 40: Cách thức hoạt động của Sliding Window

Ví dụ: Giả sử chúng ta có dữ liệu chuỗi thời gian dài 1000 mẫu và cửa sổ trượt có kích thước 240 mẫu. Cửa sổ trượt sẽ tạo ra các mẫu huấn luyện sau:

- Cửa sổ đầu tiên chứa mẫu 1 đến mẫu 240.
- Cửa sổ thứ hai chứa mẫu 41 đến mẫu 280 (di chuyển 40 mẫu).
- Cửa sổ thứ ba chứa mẫu 81 đến mẫu 320, và cứ thế tiếp tục.

5.1.3 Ứng dụng cửa sổ trượt trong dự đoán

Khi mô hình được huấn luyện xong, cửa sổ trượt sẽ được sử dụng để xử lý các dữ liệu đầu vào mới trong quá trình dự đoán, giúp mô hình nhận diện và dự đoán các cử chỉ ngôn ngữ ký hiệu một cách chính xác.

- Dữ liệu đầu vào cho dự đoán sẽ được chia thành các cửa sổ thời gian với kích thước cố định, giống như trong quá trình huấn luyện.
- Mỗi cửa sổ sẽ chứa một chuỗi dữ liệu mới từ các cảm biến, với chiều dài cửa sổ cố định (ví dụ: 240 bước thời gian).
- Cửa sổ trượt di chuyển qua dữ liệu mới để tạo ra các mẫu cho mô hình dự đoán.

Ví dụ: Giả sử mô hình đã được huấn luyện và bây giờ cần dự đoán cho dữ liệu mới. Nếu cửa sổ trượt có kích thước 240 và bước trượt là 40 mẫu:

- Cửa sổ đầu tiên chứa các mẫu từ 1 đến 240.
- Cửa sổ thứ hai chứa mẫu từ 41 đến 280, và cứ thế tiếp tục.
- Mỗi cửa sổ được đưa vào mô hình để dự đoán giá trị, sau đó mô hình sẽ trả về kết quả cho mỗi cửa sổ.

Dữ liệu được xử lý qua các cửa sổ trượt sẽ giúp mô hình dự đoán các cử chỉ theo từng chuỗi thời gian liên tiếp, từ đó tạo ra kết quả chính xác hơn và liên tục, đặc biệt khi dữ liệu mới được thu thập từ các cảm biến.

5.1.4 Lợi ích của cửa sổ trượt trong dự đoán

- **Dự đoán liên tục:** Cửa sổ trượt giúp mô hình duy trì khả năng dự đoán liên tục cho dữ liệu chuỗi thời gian mới, không bị gián đoạn.
- **Giữ lại thông tin trong các chuỗi thời gian dài:** Mỗi cửa sổ giúp mô hình nắm bắt thông tin từ các bước thời gian liên tiếp, từ đó cải thiện độ chính xác của dự đoán.
- **Xử lý dữ liệu theo thời gian thực:** Cửa sổ trượt cho phép mô hình xử lý dữ liệu theo dạng chuỗi thời gian liên tục, giúp mô hình thích ứng với các thay đổi trong dữ liệu theo thời gian.

5.2 Khôi phục dữ liệu bị mất

Trong quá trình truyền dữ liệu qua Bluetooth, đặc biệt với các tín hiệu liên tục như từ cảm biến, có thể xảy ra hiện tượng mất gói dữ liệu. Điều này ảnh hưởng đến chất lượng và tính liên tục của thông tin đầu vào cho mô hình học máy. Dưới đây là các phương pháp xử lý và khôi phục dữ liệu bị mất trong quá trình truyền.



Hình 41: Truyền dữ liệu bằng bluetooth

5.2.1 Nguyên nhân gây mất dữ liệu

- **Giới hạn băng thông của Bluetooth:** Khi tốc độ truyền vượt quá khả năng xử lý của kênh truyền, một số gói dữ liệu có thể bị bỏ qua.
- **Nhiều tín hiệu:** Các tín hiệu không mong muốn trong môi trường có thể gây lỗi truyền dữ liệu.
- **Khoảng cách truyền:** Khoảng cách giữa thiết bị gửi và nhận quá xa dẫn đến tín hiệu yếu, gây mất dữ liệu.



- **Quá tải bộ đệm:** Dữ liệu được gửi quá nhanh khiến bộ đệm của thiết bị nhận không xử lý kịp thời.

5.2.2 Phương pháp phát hiện dữ liệu bị mất

- **Kiểm tra tính liên tục của chuỗi dữ liệu:** Mỗi gói dữ liệu được đánh số tuần tự (*sequence number*). Khi phát hiện số thứ tự không liên tiếp, xác định đã có dữ liệu bị mất.
- **Kiểm tra tính đầy đủ của gói dữ liệu:** Dựa vào định dạng dữ liệu cố định, nếu số lượng giá trị trong gói không khớp với định dạng chuẩn (ví dụ: thiếu 7 giá trị từ cảm biến), xác định gói bị lỗi.

5.2.3 Phương pháp khôi phục dữ liệu bị mất

1. Nội suy tuyến tính (Linear Interpolation):

- Áp dụng khi mất một số ít gói dữ liệu liên tiếp.
- Dựa vào giá trị của các gói trước và sau gói bị mất để tính toán giá trị trung bình hoặc ước lượng dữ liệu:

$$x_{khôi\ phục} = x_{trước} + \frac{x_{sau} - x_{trước}}{\Delta t}$$

- Phương pháp này đảm bảo tín hiệu khôi phục mượt mà và duy trì tính liên tục.

2. Sao chép giá trị gần nhất (Forward/Backward Filling):

- Áp dụng cho các trường hợp mất dữ liệu ngắn.
- Giá trị bị mất được thay thế bằng giá trị gần nhất (trước hoặc sau gói bị mất).
- Phương pháp này đơn giản và hiệu quả khi dữ liệu thay đổi ít theo thời gian.

3. Bổ sung giá trị mặc định:

- Áp dụng khi không thể nội suy do mất nhiều gói liên tiếp hoặc không có dữ liệu tham chiếu.
- Giá trị bị mất được thay bằng giá trị mặc định (ví dụ: 0 hoặc giá trị trung bình của chuỗi).

4. Yêu cầu gửi lại dữ liệu:

- Trong trường hợp quan trọng, thiết bị nhận có thể gửi tín hiệu yêu cầu thiết bị gửi phát lại gói bị mất.
- Phương pháp này hiệu quả nhưng tăng độ trễ trong hệ thống.

5.2.4 Tối ưu hóa giảm thiểu mất dữ liệu

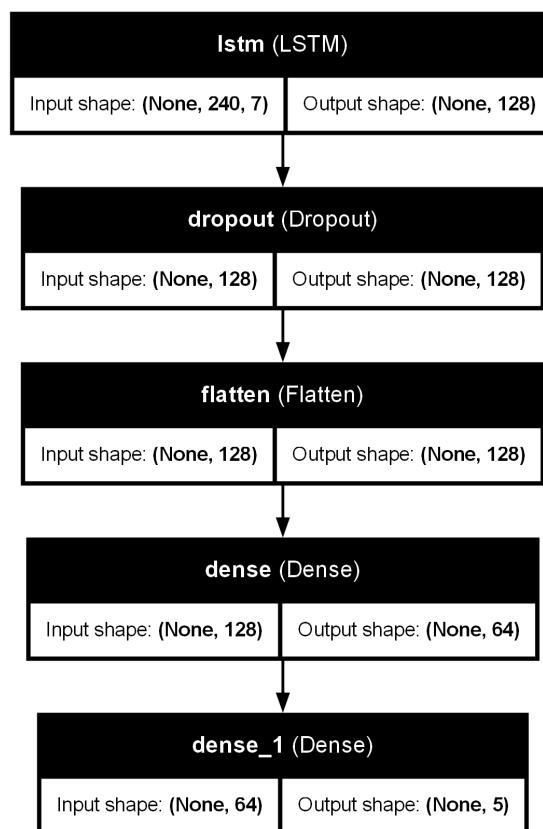
- **Điều chỉnh tốc độ truyền:** Giảm tốc độ truyền để phù hợp với khả năng xử lý của thiết bị nhận.
- **Sử dụng nén dữ liệu:** Nén tín hiệu để giảm kích thước gói, tăng tốc độ truyền.

- **Giảm nhiễu tín hiệu:** Sử dụng các kênh tần số khác hoặc cải thiện điều kiện môi trường truyền dữ liệu.
- **Tăng độ nhạy Bluetooth:** Dảm bảo khoảng cách giữa thiết bị gửi và nhận nằm trong giới hạn hỗ trợ của Bluetooth.

5.3 Cải thiện chất lượng mô hình

5.3.1 Mô hình với Deep Convolution và Self-Attention

Trong đề tài trước, mô hình học máy được sử dụng khá thô sơ và đơn giản, bao gồm các thành phần chính: một lớp *LSTM*, một lớp *Dropout*, một lớp *Flatten*, và hai lớp *Dense*.



Hình 42: Cấu trúc mô hình cũ

Kết quả đạt được từ mô hình này vẫn chưa thực sự khả quan. Để cải thiện chất lượng đầu ra, cần thực hiện hai bước quan trọng:

- **Gia tăng kích thước tập dữ liệu:** Mở rộng và đa dạng hóa dữ liệu đầu vào.
- **Nâng cấp mô hình:** Sử dụng kiến trúc học máy hiệu quả hơn để khai thác triệt để thông tin từ dữ liệu.

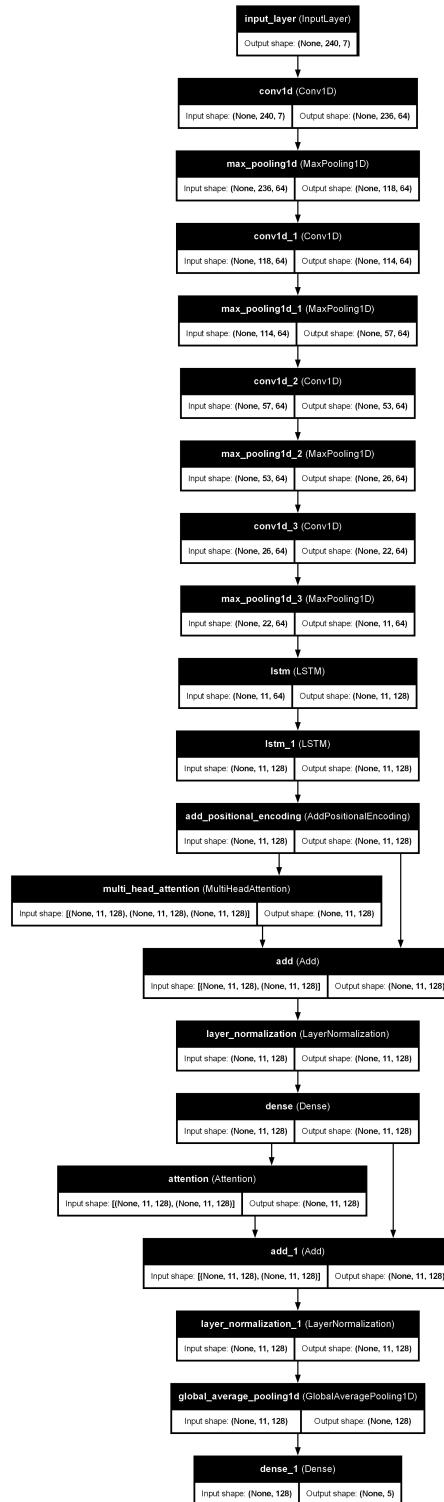


Trong đề tài này, nhóm đề xuất một mô hình cải tiến, dựa trên kiến trúc được giới thiệu trong bài báo "*Self-Attention-Based Deep Convolution LSTM Framework for Sensor-Based Badminton Activity Recognition*" ([DOI: 10.3390/s23208373](https://doi.org/10.3390/s23208373)). Mô hình này kết hợp:

1. **Mạng nơ-ron tích chập (CNN)**: Chiết xuất các đặc trưng cục bộ từ dữ liệu đầu vào.
2. **Mạng nơ-ron hồi quy (LSTM)**: Nắm bắt các phụ thuộc dài hạn trong dữ liệu.
3. **Cơ chế Self-Attention**: Tập trung vào các phần quan trọng nhất của dữ liệu.

Mô hình đề xuất bao gồm các thành phần chính sau:

- **Lớp CNN**: Chiết xuất các đặc trưng cục bộ từ dữ liệu đầu vào.
- **Lớp LSTM**: Nắm bắt và lưu giữ các thông tin phụ thuộc dài hạn trong chuỗi thời gian.
- **Cơ chế Self-Attention**: Xác định và tập trung vào các đặc trưng quan trọng nhất trong dữ liệu, cải thiện hiệu quả học hỏi của mô hình.



Hình 43: Mô hình mới



Cơ chế *Self-Attention* đóng vai trò then chốt trong việc nâng cao hiệu suất của mô hình. Nó cho phép mô hình tự động:

- Phân tích toàn bộ dữ liệu đầu vào.
- Tập trung vào những đặc trưng mang tính quyết định.

Nhờ vậy, mô hình có thể đạt được độ chính xác cao hơn trong việc dự đoán các hành động theo chuỗi thời gian.

5.4 Làm giàu dữ liệu

5.4.1 Data Augmentation

Trong học máy, *Data Augmentation* là một kỹ thuật quan trọng giúp mở rộng tập dữ liệu hiện có bằng cách tạo ra các biến thể mới của dữ liệu. Điều này không chỉ cải thiện tính đa dạng của dữ liệu mà còn giúp mô hình giảm thiểu hiện tượng quá khớp (*overfitting*) và học được nhiều đặc trưng tổng quát hơn. Dưới đây là một số kỹ thuật được áp dụng:

1. Jittering:

- **Mô tả:** Thêm nhiễu ngẫu nhiên vào dữ liệu. Jittering có thể được thực hiện bằng cách thêm một lượng nhỏ nhiễu Gaussian vào mỗi điểm dữ liệu. Điều này giúp mô hình ít bị ảnh hưởng bởi nhiễu trong dữ liệu thực tế và tăng khả năng kháng nhiễu của mô hình.
- **Lợi ích:**
 - Giúp mô hình trở nên mạnh mẽ hơn trước các biến động nhỏ trong dữ liệu thực tế.
 - Tăng khả năng tổng quát hóa bằng cách làm cho dữ liệu đầu vào đa dạng hơn.

2. Scale:

- **Mô tả:** Thay đổi tỷ lệ của dữ liệu. Scale được thực hiện bằng cách nhân mỗi điểm dữ liệu với một hệ số tỷ lệ ngẫu nhiên.
- **Lợi ích:**
 - Giúp mô hình học được các đặc trưng ở các mức độ hoặc tỉ lệ khác nhau.
 - Phù hợp với các bài toán mà dữ liệu có sự biến thiên về kích thước, chẳng hạn như tín hiệu sinh lý.

3. Magnitude Warp:

- **Mô tả:** Biến dạng biên độ của dữ liệu theo một hàm phi tuyến. Magnitude Warp có thể được thực hiện bằng cách áp dụng một hàm phi tuyến, chẳng hạn như hàm sin hoặc hàm mũ, cho biên độ của dữ liệu.
- **Lợi ích:**
 - Tạo ra các biến thể tín hiệu thực tế hơn, đặc biệt phù hợp với các bài toán liên quan đến chuỗi thời gian.
 - Làm phong phú tập dữ liệu và giúp mô hình ít phụ thuộc vào các dạng tín hiệu cụ thể.

4. Shift:



- **Mô tả:** Dịch chuyển dữ liệu theo thời gian. Shift được thực hiện bằng cách dịch chuyển toàn bộ chuỗi thời gian sang trái hoặc phải một khoảng thời gian ngẫu nhiên.
- **Lợi ích:**
 - Tăng khả năng chống lại sự thay đổi vị trí trong dữ liệu, chẳng hạn như các tín hiệu không đồng bộ.
 - Giúp mô hình học được các đặc trưng không bị phụ thuộc vào vị trí cụ thể.

Để tăng cường dữ liệu huấn luyện, nhóm đã xây dựng một script Python để áp dụng các kỹ thuật Data Augmentation đã đề cập ở trên. Script này nhận dữ liệu gốc làm đầu vào và tạo ra các phiên bản biến đổi của dữ liệu bằng cách sử dụng các kỹ thuật Jittering, Scale, Magnitude Warp và Shift.

Mỗi kỹ thuật Data Augmentation được áp dụng riêng biệt, tạo ra một phiên bản mới của tập dữ liệu với cùng kích thước với tập dữ liệu gốc. Do đó, sau khi áp dụng cả bốn kỹ thuật, nhóm thu được một tập dữ liệu mới có kích thước gấp 5 lần tập dữ liệu ban đầu (bao gồm cả dữ liệu gốc).

Việc tăng kích thước tập dữ liệu huấn luyện thông qua Data Augmentation giúp cải thiện khả năng tổng quát hóa của mô hình, giảm thiểu hiện tượng overfitting và tăng cường độ chính xác của mô hình trên dữ liệu mới.

5.4.2 Transfer learning

5.4.3 Sinh dữ liệu từ TimeGAN



6 Kết quả thực nghiệm

6.1 Mô hình ban đầu

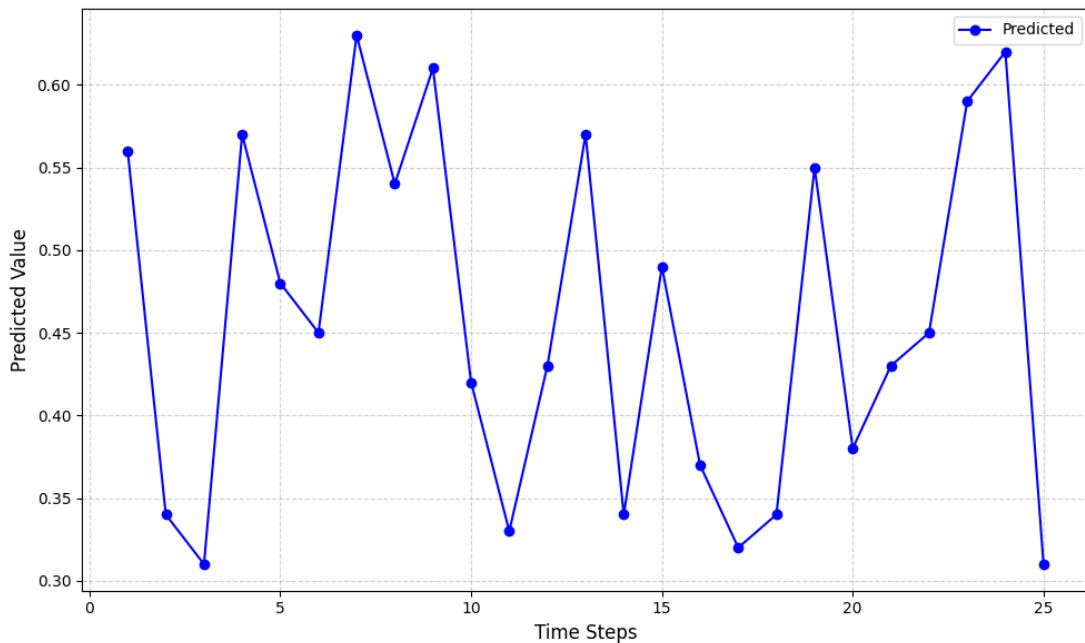
6.1.1 Thiết lập thực nghiệm

- **Dữ liệu sử dụng:** Dữ liệu được thu thập trực tiếp từ phần cứng và được tổng hợp lại đến khi đủ 1,680 dữ liệu thì đưa vào mô hình dự đoán.
- **Cấu hình mô hình:**
 - Mô hình LSTM với kiến trúc gồm một lớp LSTM (128 đơn vị), một lớp Dropout (tỷ lệ 50%), và hai lớp Dense (64 đơn vị và đầu ra 5 lớp).
 - Hàm mất mát: `categorical_crossentropy`.
 - Trình tối ưu: Adam.
 - Số epoch: 300.
 - Kích thước batch: 4,500.

6.1.2 Kết quả huấn luyện và kiểm tra

- **Độ chính xác trên tập huấn luyện:** Khoảng 70%, cho thấy mô hình đã học được một phần các đặc trưng từ dữ liệu.
- **Độ chính xác trên tập kiểm tra:** Khoảng 70%, phản ánh mô hình có khả năng tổng quát hóa ở mức trung bình.

6.1.3 Kết quả dự đoán thời gian thực



Hình 44: Kết quả thực nghiệm mô hình

- Khi thử nghiệm trên dữ liệu realtime từ các cảm biến, kết quả dự đoán của mô hình cho từng nhãn dao động trong khoảng từ 0.3 đến 0.6.
- Với ngưỡng phân loại đúng là trên 0.5, nhiều dự đoán không vượt qua ngưỡng này, dẫn đến kết quả không đáng tin cậy trong thực tế.

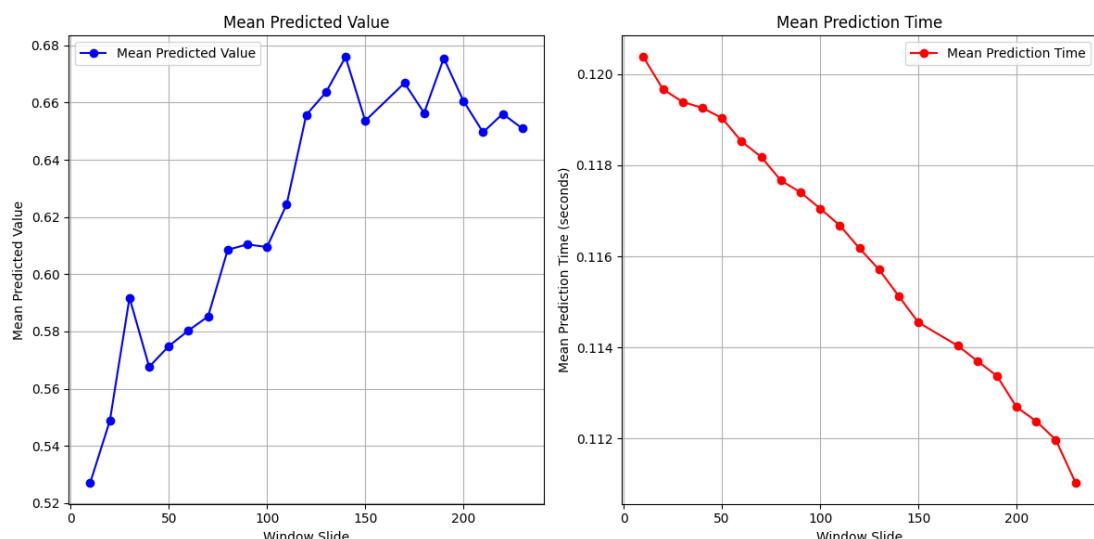
6.1.4 Phân tích kết quả

- **Mô hình hoạt động tốt hơn trên dữ liệu tĩnh:**
 - Độ chính xác trên tập kiểm tra cho thấy mô hình có khả năng học và phân loại ở mức cơ bản khi sử dụng dữ liệu tĩnh.
 - Tuy nhiên, trên dữ liệu realtime, mô hình không thể duy trì hiệu suất, đặc biệt khi các tín hiệu có nhiều hoặc thay đổi không đồng nhất.
- **Khả năng phân biệt kém:**
 - Kết quả dự đoán cho thấy nhiều nhãn có xác suất gần nhau, dao động quanh ngưỡng 0.5, thể hiện mô hình không tự tin trong việc phân loại.
 - Điều này có thể do dữ liệu huấn luyện chưa đủ phong phú hoặc chưa đại diện tốt cho các trường hợp thực tế.
- **Ảnh hưởng của dữ liệu nhiễu:**

- Trong môi trường thực tế, các yếu tố như nhiễu tín hiệu, dao động tay người dùng, hoặc sự không nhất quán trong cử chỉ có thể khiến dữ liệu đầu vào khác xa so với dữ liệu huấn luyện, làm giảm hiệu quả của mô hình.

6.2 Áp dụng cửa sổ trượt (Sliding Window)

Kỹ thuật cửa sổ trượt (sliding window) đã được tích hợp và thử nghiệm trong hệ thống nhận diện cử chỉ thời gian thực. Đây là một phương pháp quan trọng giúp cải thiện độ chính xác dự đoán bằng cách chia nhỏ dữ liệu đầu vào thành các đoạn có độ dài cố định và xử lý tuần tự từng đoạn, tạo điều kiện cho mô hình học máy phân tích sâu hơn các chuỗi dữ liệu liên tục. Trong mục này, kết quả thực nghiệm của kỹ thuật sliding window được trình bày thông qua việc so sánh độ chính xác dự đoán và thời gian xử lý khi thay đổi kích thước cửa sổ.



Hình 45: Kết quả dự đoán và thời gian trung bình ứng với các giá trị sliding window

6.2.1 Độ chính xác dự đoán tăng theo kích thước cửa sổ

Biểu đồ minh họa cho thấy rằng khi kích thước cửa sổ tăng dần từ 10 đến 230, độ chính xác dự đoán của hệ thống cũng tăng đáng kể. Cụ thể:

- Với kích thước cửa sổ nhỏ (10-50), giá trị độ chính xác trung bình dao động quanh mức 0.5. Kích thước nhỏ khiến hệ thống khó khai thác đầy đủ thông tin trong chuỗi dữ liệu, dẫn đến dự đoán kém ổn định.
- Khi kích thước cửa sổ tăng lên (từ 100 đến 230), giá trị độ chính xác cải thiện đáng kể, đạt mức từ 0.6 đến gần 0.7. Điều này cho thấy rằng việc cung cấp nhiều dữ liệu hơn trong mỗi bước xử lý giúp mô hình nắm bắt tốt hơn các đặc điểm quan trọng trong chuỗi thời gian, qua đó tăng khả năng nhận diện chính xác.



6.2.2 Thời gian dự đoán giảm nhẹ khi tăng kích thước cửa sổ

Mặc dù kích thước cửa sổ tăng, thời gian xử lý mỗi lần dự đoán giảm nhẹ từ 0.12 giây xuống 0.11 giây. Điều này có thể được giải thích bởi hai yếu tố chính:

- **Hiệu quả của mô hình học máy:** Mô hình được huấn luyện tối ưu với khả năng xử lý hiệu quả các chuỗi dữ liệu dài.
- **Độ ổn định của bộ đệm dữ liệu:** Khi kích thước cửa sổ lớn, dữ liệu trong bộ đệm ít thay đổi thường xuyên hơn, giảm chi phí xử lý từng khung dữ liệu mới.

Mặc dù sự thay đổi về thời gian là không đáng kể, nhưng nó cho thấy rằng kỹ thuật sliding window không gây ra ảnh hưởng tiêu cực đến tốc độ xử lý thời gian thực của hệ thống.

6.2.3 Kết luận

Áp dụng kỹ thuật sliding window mang lại sự cải thiện đáng kể về độ chính xác dự đoán mà không ảnh hưởng tiêu cực đến tốc độ xử lý. Sự thay đổi kích thước cửa sổ giúp mô hình học máy thích nghi tốt hơn với dữ liệu đầu vào, nâng cao hiệu quả nhận diện trong các ứng dụng thời gian thực. Phương pháp này không chỉ đáp ứng được yêu cầu thực tiễn mà còn dễ dàng tinh chỉnh để phù hợp với các hệ thống khác nhau.

6.3 Khôi phục dữ liệu bị mất

Trong quá trình thử nghiệm, dữ liệu thu thập từ phần cứng được gửi đến với tốc độ rất cao, khoảng 10ns/lần cho mỗi gói gồm 7 dữ liệu. Vì các lý do khách quan trong thực tế sẽ có hiện tượng mất dữ liệu trong khi truyền. Nhóm đã thực nghiệm nhiều lần và nhận thấy có hai trường hợp chính là mất ít dòng liên tục và nhiều dòng liên tục.

6.3.1 Dữ liệu mất ít dòng (dưới 10 dòng)

Khi dữ liệu mất ở mức độ nhỏ, chẳng hạn dưới 10 dòng (mỗi dòng chứa 7 giá trị dữ liệu), hệ thống có thể tận dụng dữ liệu gửi đến sau để bù đắp cho khoảng dữ liệu bị thiếu mà không cần áp dụng các phương pháp khôi phục dữ liệu phức tạp. Điều này mang lại hai lợi ích chính:

- **Tăng tốc độ xử lý:** Tránh việc sử dụng các thuật toán phục hồi dữ liệu tốn thời gian, đặc biệt trong các ứng dụng thời gian thực.
- **Giữ ổn định hệ thống:** Dữ liệu được bù đắp bởi các giá trị liền kề, đảm bảo chuỗi dữ liệu tiếp tục được phân tích mà không bị gián đoạn.

Phương pháp này dựa trên giả định rằng sự mất dữ liệu trong khoảng thời gian ngắn không ảnh hưởng nghiêm trọng đến việc nhận diện cử chỉ, nhờ vào tính chất liên tục của chuỗi dữ liệu từ phần cứng.

6.3.2 Dữ liệu mất nhiều dòng (trên 10 dòng)

Khi hiện tượng mất dữ liệu xảy ra trên quy mô lớn hơn, các phương pháp khôi phục dữ liệu thông thường không còn đáng tin cậy. Điều này xuất phát từ việc thiếu thông tin ngữ cảnh để tái tạo chính xác chuỗi dữ liệu bị mất. Thay vào đó, một chiến lược hiệu quả hơn được đề xuất:

- **Thiết lập cơ chế timeout:** Hệ thống sẽ theo dõi thời gian không nhận được dữ liệu từ phần cứng. Nếu vượt quá một khoảng thời gian định trước (timeout), hệ thống sẽ thực hiện việc khởi động lại quy trình nhận dữ liệu từ đầu.



- **Lợi ích của cơ chế timeout:**

- Đảm bảo rằng hệ thống không cố gắng xử lý các chuỗi dữ liệu không đầy đủ, tránh ảnh hưởng đến độ chính xác của mô hình.
- Đơn giản hóa quy trình xử lý, giảm thiểu chi phí tính toán so với các phương pháp phục hồi dữ liệu phức tạp.

6.3.3 Kết luận

Với tốc độ gửi dữ liệu cao từ phần cứng, hệ thống có thể tự điều chỉnh để đối phó với hiện tượng mất dữ liệu mà không cần áp dụng các kỹ thuật khôi phục phức tạp. Đối với dữ liệu bị mất ít dòng, dữ liệu gửi đến sau có thể bù lại được, đảm bảo tính liên tục. Ngược lại, khi mất quá nhiều dữ liệu, cơ chế timeout sẽ đảm bảo rằng hệ thống nhận dữ liệu lại từ đầu, giữ vững tính ổn định và chính xác trong các ứng dụng thời gian thực.

Chiến lược này tối ưu hóa hiệu suất của hệ thống, giảm thiểu chi phí xử lý và phù hợp với yêu cầu của các ứng dụng nhận diện cử chỉ trong thực tế.

6.4 1

Data được thu thập sau đó được phân bổ với tỉ lệ 80-20, 80% lượng data được sử dụng để train mô hình, 20% được sử dụng để kiểm tra kết quả sau khi huấn luyện

Chúng ta không thể đánh giá kỹ năng của mô hình chỉ từ một lần đánh giá. Nguyên nhân là mạng nơ-ron là ngẫu nhiên, có nghĩa là một cấu hình mô hình cụ thể khác nhau sẽ xuất hiện khi đào tạo cùng một cấu hình mô hình trên cùng một dữ liệu. Điều này là một đặc trưng của mạng, vì nó mang lại khả năng thích ứng cho mô hình, nhưng yêu cầu một quá trình đánh giá mô hình phức tạp hơn.

Chúng ta sẽ lặp lại việc đánh giá mô hình nhiều lần, sau đó tóm tắt hiệu suất của mô hình qua từng lần chạy đó. chúng ta có thể gọi hàm evaluate_model() tổng cộng 5 lần. Điều này sẽ dẫn đến một tập hợp các điểm đánh giá mô hình cần được tóm tắt.'

```
1 >#1: 90.058
2 >#2: 85.918
3 >#3: 90.974
4 >#4: 89.515
5 >#5: 90.159
6 >#6: 91.110
7 >#7: 89.718
8 >#8: 90.295
9 >#9: 89.447
10 >#10: 90.024
11
12 [90.05768578215134, 85.91788259246692, 90.97387173396675,
   89.51476077366813, 90.15948422124194, 91.10960298608755,
   89.71835765184933, 90.29521547336275, 89.44689514760775,
   90.02375296912113]
13
14 Accuracy: 89.722% (+/-1.371)
```

Cuối cùng, mẫu các điểm đánh giá được in ra, tiếp theo là giá trị trung bình và độ lệch chuẩn. Chúng ta có thể thấy rằng mô hình đã hoạt động tốt, đạt được độ chính xác phân loại khoảng



89,7% khi được huấn luyện trên dữ liệu thô, với độ lệch chuẩn là khoảng 1,3. Đây là một kết quả tốt !

6.5

Dựa vào số động tác thực hiện và số lần mô hình xác định chính xác, cho thấy mô hình chỉ đạt kết quả khoảng 70%:

Kết quả này được thực hiện như sau:

- Một động tác được duy trì trong 2.4s bằng với số frame dữ liệu được gửi đến mô hình để dự đoán
- Thực hiện động tác với label 7 và 9 vì hai động tác này có data frame khá tương đồng nhau, trong 1 phút bằng với 25 lần mô hình gửi đến model kết quả cho thấy chỉ dự đoán được 18 lần với label 7 và 17 lần với label 9

Những động tác có data frame khác biệt nhau ví dụ như label 5 và label 0 cho ra kết quả khá chính xác, lên đến hơn 90% dựa vào cách tính trên



7 Kết luận

7.1 Kết quả đạt được

Trong giai đoạn 2 này, nhóm đã thành công trong việc triển khai một sản phẩm tích hợp flexsensor và cảm biến gia tốc tốc, kết hợp với Arduino Lilypad, để thu thập dữ liệu từ chuyển động của các ngón tay. Sản phẩm này đóng vai trò quan trọng trong việc cung cấp dữ liệu đầu vào cho quá trình huấn luyện và dự đoán của mô hình LSTM.

Bên cạnh đó, nhóm đã hiện thực một mô hình LSTM có khả năng dự đoán các hành động được định nghĩa trước, thông qua quá trình huấn luyện trên tập dữ liệu chứa các mẫu dữ liệu liên quan đến các hành động đó. Điều này đặt nền tảng cho tính linh hoạt và độ chính xác của hệ thống, giúp nó có khả năng hiểu và phản ứng đáng tin cậy đối với các cử chỉ của người sử dụng.

7.2 Hạn chế của hệ thống

Một trong những thách thức lớn nhất là khả năng mô hình LSTM trực tiếp dự đoán từ dữ liệu thu thập được. Điều này xuất phát từ sự phức tạp và đa dạng của dữ liệu, đòi hỏi quá trình xử lý để tạo ra đầu ra dự đoán chính xác và hiệu quả.

Quá trình xử lý dữ liệu kéo dài thời gian giữa việc thu thập dữ liệu và quá trình dự đoán. Điều này có thể tạo ra một khoảng thời gian trễ không mong muốn, ảnh hưởng đến tính linh hoạt và phản ứng nhanh chóng được mong đợi từ hệ thống.

Hệ thống đã đạt được những sự hoạt động ổn định nhất định, tuy nhiên mô hình vẫn đưa ra các dự đoán sai ở các động tác có tính tương tự nhau. chỉ đạt được khoảng 70%

7.3 Hướng cải tiến và phát triển

Đối mặt với những thách thức và giới hạn đã được đề cập trước đó, mục tiêu quan trọng trong giai đoạn tiếp theo của chúng tôi là nâng cao mô hình LSTM để có khả năng dự đoán trực tiếp từ dữ liệu thu thập được. Điều này đòi hỏi sự tập trung vào việc tối ưu hóa và cải thiện quá trình xử lý dữ liệu, nhằm giảm thiểu độ trễ và tăng cường tính linh hoạt của hệ thống.



Tài liệu tham khảo

- [1] Bishop C. M. (1995). Neural Networks for Pattern Recognition, Nhà in Đại học Oxford. ISBN 0-19-853864-2
- [2] Microchip Technology. (truy cập lần cuối tháng 12/2023). "ATmega328P Microcontroller." Truy cập từ: [Link tham khảo](#)
- [3] Arduino Kit Vietnam. (Ngày đăng 12/06/2023). "Hướng dẫn sử dụng cảm biến uốn cong (Flex Sensor) với Arduino." Truy cập từ: [Link tham khảo](#)
- [4] Pham, D. K. (Ngày đăng 22/04/2019). "Lý thuyết về mạng LSTM." Truy cập từ: [Link tham khảo](#)
- [5] Brownlee, J. (Ngày đăng 28/08/2020). "How to Develop RNN Models for Human Activity Recognition (Time Series Classification)." Truy cập từ: [Link tham khảo](#)