

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 em 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 em 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 cảm biến uốn cong (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 cảm biến uốn cong, 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	VI
Danh sách bảng	VII
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
2.2.3 Một số loại nơ-ron hồi quy	3
2.2.3.1 Vấn đề phụ thuộc xa của RNN	4
2.2.4 Giới thiệu mô hình Long-short term memmory(LSTM)	5
2.2.4.1 Giới thiệu tổng quan	5
2.2.4.2 Ý tưởng đằng sau LSTM	6
2.2.4.3 Thứ tự các bước của LSTM	7
2.2.4.4 Ưu và nhược điểm của mô hình LSTM	9
2.2.5 Giới thiệu về Transfer Learning	10
2.2.5.1 Định nghĩa Transfer Learning	10
2.2.5.2 Ứng dụng của Transfer Learning	10
2.2.5.3 Lợi ích và bất lợi của Transfer Learning	11
2.2.6 Giới thiệu về Self-Attention	12
2.2.6.1 Cơ chế hoạt động của Self-Attention	12
2.2.6.2 Các loại Self-Attention thường gặp	12
2.2.6.3 Ứng dụng của Self-Attention	12
2.2.6.4 Lợi ích của Self-Attention	13
2.2.7 Nội suy tuyến tính (Linear Interpolation)	13
2.2.7.1 Lợi ích của nội suy tuyến tính	14
2.2.7.2 Hạn chế của nội suy tuyến tính	14
2.2.8 Generative Adversarial Networks (GANs)	14
2.2.8.1 Kiến trúc và hoạt động của GANs	14
2.2.8.2 Các biến thể của GANs	15
2.2.8.3 Ứng dụng của GANs	15
2.2.8.4 Lợi ích và hạn chế của GANs	15
2.2.9 Giới thiệu về thư viện Tensorflow	16
2.2.9.1 Nguyên lý hoạt động	16
2.2.9.2 Thuộc tính cơ bản trong Tensorflow	17



2.9.3	Ưu điểm của TensorFlow	19
2.10	Giới thiệu về Arduino LilyPad	19
2.11	Giới thiệu về cảm biến uốn cong(Flex sensor)	21
2.11.1	Tổng quan	21
2.11.2	Nguyên lý hoạt động	21
2.11.3	Đọc giá trị	22
2.12	Giới thiệu về cảm biến con quay gia tốc	23
2.12.1	Nguyên lý hoạt động	23
2.12.1.a	Sử dụng hiệu ứng Piezoelectric	23
2.12.1.b	Sử dụng phương pháp điện dung	26
2.12.2	Giới thiệu về module cảm biến ITG 3200	27
2.13	Giới thiệu về module bluetooth HC05	28
2.14	Giới thiệu về ngôn ngữ ký hiệu	29
3	Kiến trúc hệ thống	33
3.1	Module thu thập dữ liệu cử chỉ	34
3.2	Module nhận dạng cử chỉ	35
3.3	Module phát âm thanh	36
4	Hiện thực hệ thống	37
4.1	Hiện thực phần cứng	37
4.2	Xây dựng bộ dữ liệu	40
4.2.1	Dữ liệu được thu thập trên arduino	40
4.2.2	Dữ liệu nhận được từ gateway	40
4.3	Dịnh dạng data trước khi đưa vào học máy	45
4.4	Xây dựng mô hình	46
5	Các phương pháp cải tiến	47
5.1	Sliding Window	47
5.2	Khắc phục vấn đề mất dữ liệu	47
5.3	Cải thiện chất lượng mô hình	47
5.3.1	Mô hình với Deep Convolution và Self-Attention	47
5.4	Làm giàu dữ liệu	50
5.4.1	Data Augmentation	50
5.4.2	Transfer learning	51
5.4.3	Sinh dữ liệu từ TimeGAN	52
6	Kết quả thực nghiệm	53
6.1	Mô hình ban đầu	53
6.2	Sau khi cải tiến	53
6.3	53
7	Kết luận	55
7.1	Kết quả đạt được	55
7.2	Hạn chế của hệ thống	55
7.3	Hướng cải tiến và phát triển	55
Tài liệu tham khảo		56



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	10
14	Lợi ích của Transfer Learning	11
15	Self Attention	12
16	Nội suy tuyến tính	13
17	Generative Adversarial Networks (GANs)	14
18	Thư viện Tensorflow	16
19	Nguyên lý hoặc động của Tensorflow	17
20	Hình ảnh minh họa của một tensor và so sánh với vector và ma trận.	18
21	Minh họa cho 1 graph, session, operation và variable.	19
22	Board mạch Arduino Lilypad	20
23	Flex sensor	21
24	Hình dạng uốn cong của flex sensor	22
25	Sơ đồ nối dây của flex sensor	22
26	Hiệu ứng Piezoelectric	24
27	Các kiểu thiết kế cơ học của cảm biến gia tốc	25
28	Cấu trúc của một cảm biến gia tốc sử dụng phương pháp điện dung	26
29	Cảm biến gia tốc góc ITG 3200	27
30	module bluetooth HC-05	28
31	Bảng chữ số của ngôn ngữ ký hiệu	31
32	Bảng chữ cái của ngôn ngữ ký hiệu	32
33	Kiến trúc tổng quan của hệ thống	33
34	Kiến trúc module thu thập dữ liệu cử chỉ	34
35	Kiến trúc model dự đoán cử chỉ	35
36	Kiến trúc module chuyển đầu ra dự đoán thành giọng nói	36
37	Sơ đồ kết nối mạch	37
38	Hình ảnh sau khi hoàn thiện	39
39	Dạng dữ liệu được gửi	40
40	kiểm tra các cổng com đang mở	41
41	Quá trình lấy dữ liệu	44
42	Cấu trúc mô hình cũ	47
43	Mô hình mới	49
44	Dữ liệu sau khi chỉnh sửa	52



Danh sách bảng

1	Thông số kỹ thuật của Arduino Lilypad	20
2	File readme mô tả hành động được thu nhập và gán nhãn	42
3	File config là các đoạn text được chuyển sang giọng nói	42



1 Giới thiệu đề tài

1.1 Giới thiệu

Giao tiếp là một phần không thể thiếu trong cuộc sống của con người. Giao tiếp giúp con người có thể dễ dàng trao đổi thông tin, bày tỏ cảm xúc, từ đó gây dựng nên kết nối xã hội giữa người với người. Tuy nhiên, đối với những người có khiếm khuyết như câm hoặc điếc thì việc giao tiếp là một trở ngại cực kì lớn đối với họ. Theo một thống kê của Tổ chức Y tế thế giới(WHO) vào năm 2023, trên thế giới hiện nay có khoảng 70 triệu người khiếm thính, phần lớn trong số đó cũng ghi nhận tình trạng mất khả năng nói và khoảng hơn 430 triệu người có vấn đề về thính giác.

Phương tiện giao tiếp với những người có khiếm khuyết về thính giác cũng như giọng nói được sử dụng chủ yếu hiện nay là ngôn ngữ ký hiệu hay còn gọi là thủ ngữ. Dù vậy, tỉ lệ người bình thường có thể sử dụng thành thạo hiện nay là rất nhỏ. Điều đó đặt ra rất nhiều thách thức trong việc giao tiếp giữa người bình thường và người gặp khiếm khuyết. Chính vì thế, nhóm chọn đề tài "Thiết bị hỗ trợ chuyển đổi ngôn ngữ ký hiệu thành giọng nói sử dụng flex sensors và học máy" nhằm mục đích thiết kế và hiện thực một hệ thống có thể chuyển ngôn ngữ ký hiệu thành giọng nói để hỗ trợ trong việc giao tiếp của người gặp khiếm khuyết.

1.2 Mục tiêu

Mục tiêu của đề tài là hiện thực một hệ thống có thể nhận diện được cử chỉ từ dữ liệu thu thập bằng cảm biến uốn cong và cảm biến con quay hồi chuyển và chuyển đổi cử chỉ đó thành giọng nói.

1.3 Phạm vi của đề tài

Ngôn ngữ ký hiệu là một ngôn ngữ với nhiều cử chỉ đa dạng và phức tạp, yêu cầu một lượng dữ liệu rất lớn để có thể phân tích và xử lý. Cho nên nhóm sẽ giới hạn phạm vi của đề tài thành hai phần chính để đảm bảo tiến độ cũng như tính khả thi của đề tài. Hai phần trọng tâm của đề tài bao gồm:

- **Thu thập dữ liệu cử chỉ:** thiết kế, hiện thực phần cứng để thu thập dữ liệu cử chỉ bằng các cảm biến uốn cong và cảm biến con quay gia tốc.
- **Phân tích, xử lý dữ liệu:** xây dựng mô hình học máy để đưa ra kết quả dự đoán cử chỉ từ những dữ liệu đã thu thập được.

1.4 Ý nghĩa thực tiễn

Với đề tài này, nhóm hy vọng có thể có ý nghĩa thực tiễn rất lớn trong việc cải thiện khả năng giao tiếp giữa người khiếm thính hoặc câm và người không biết ngôn ngữ ký hiệu, từ đó giúp cải thiện đời sống của họ. Sản phẩm của đề tài có một số ưu điểm nổi bật khi áp dụng vào thực tế như sau

- **Giao tiếp dễ dàng hơn:** Thiết bị này giúp người khiếm thính có thể giao tiếp với những người không biết ngôn ngữ ký hiệu một cách dễ dàng hơn. Họ chỉ cần thực hiện các ký hiệu, và thiết bị sẽ chuyển đổi chúng thành giọng nói.
- **Tính tiện lợi:** Thiết bị có thể được sử dụng mọi lúc mọi nơi, giúp người khiếm thính có thể giao tiếp một cách tự nhiên và thoải mái hơn trong cuộc sống hàng ngày



- **Khả năng kết nối:** Thiết bị có thể kết nối với các thiết bị khác như điện thoại di động hoặc máy tính, giúp việc chuyển đổi và phát giọng nói trở nên dễ dàng và thuận tiện hơn. Không những thế còn có thể chuyển đổi thành văn bản để sử dụng trong đối thoại bằng tin nhắn hoặc email.
- **Tính đa dụng:** Với việc sử dụng học máy, thiết bị có thể học và nhận biết một loạt các ký hiệu từ đơn giản đến phức tạp, giúp nó có thể phục vụ một lượng lớn người dùng và áp dụng cho nhiều ngữ cảnh khác nhau.

Như vậy, đòn tài này không chỉ mở ra cơ hội mới cho người khiếm thính hoặc câm trong việc giao tiếp mà còn đẩy mạnh tiến trình ứng dụng công nghệ vào cuộc sống, nhằm tạo ra một xã hội công bằng và tiến bộ hơn.

2 Cơ sở lý thuyết

2.1 Giới thiệu về học máy(Machine learning)

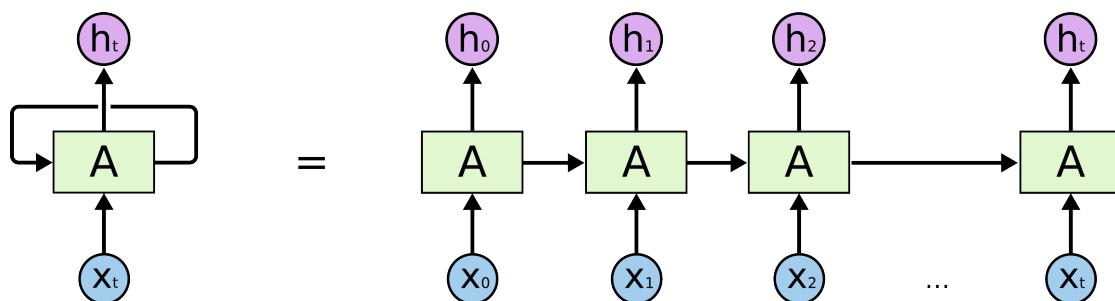
Học máy (machine learning) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kỹ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Lịch sử học máy bắt đầu từ những năm 1950 với các nỗ lực đầu tiên để mô phỏng khả năng học của con người thông qua máy tính. Tuy nhiên, cho đến đầu thập kỷ 21, học máy mới trở nên phổ biến và mạnh mẽ nhờ vào sự tiến bộ của phần cứng và khả năng tính toán.

Trong thập kỷ gần đây, sự phát triển của các thuật toán học máy, kết hợp với dữ liệu lớn và sức mạnh tính toán ngày càng tăng, đã đưa học máy vào tầm tay của nhiều người và ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm nhận dạng hình ảnh, ngôn ngữ tự nhiên và IoT (Internet of Things).

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

2.2.1 Tổng quan

Mạng nơ-ron hồi quy (RNN) là một mô hình học sâu được đào tạo để xử lý và chuyển đổi đầu vào dữ liệu tuần tự thành đầu ra dữ liệu tuần tự cụ thể. Dữ liệu tuần tự là dữ liệu, chẳng hạn như từ, câu hoặc dữ liệu chuỗi thời gian, trong đó các thành phần tuần tự tương quan với nhau dựa trên ngữ nghĩa phức tạp và quy tắc cú pháp. RNN là một hệ thống phần mềm gồm nhiều thành phần được kết nối với nhau theo cách con người thực hiện chuyển đổi dữ liệu tuần tự, chẳng hạn như dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác. Phần lớn RNN đang được thay thế bằng trí tuệ nhân tạo (AI) dựa trên công cụ biến đổi và các mô hình ngôn ngữ lớn (LLM), hiệu quả hơn nhiều trong việc xử lý dữ liệu tuần tự.



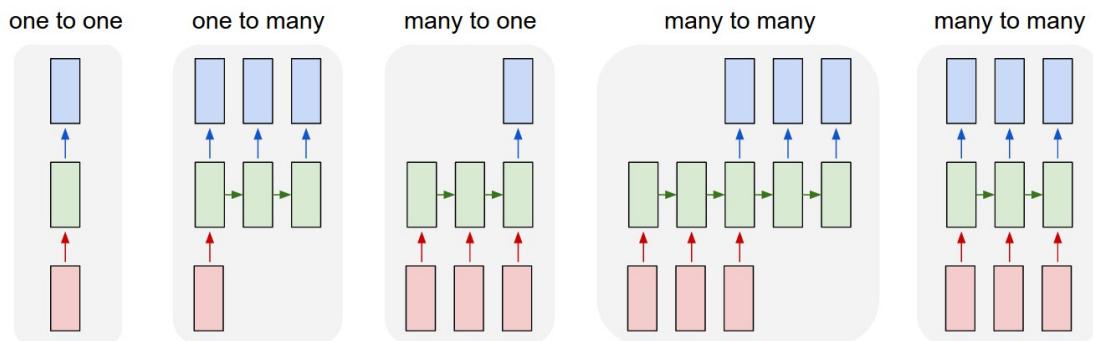
Hình 1: Minh họa một mạng nơ-ron hồi quy

2.3 Một số loại nơ-ron hồi quy

RNN thường có đặc trưng là kiến trúc môt-môt: một chuỗi đầu vào được liên kết với một đầu ra. Tuy nhiên, bạn có thể điều chỉnh linh hoạt thành các cấu hình khác nhau cho các mục đích cụ thể. Sau đây là một số loại RNN phổ biến.

- **Một-nhiều:** Loại RNN này dẫn một đầu vào đến một số đầu ra. Loại này tạo điều kiện cho các ứng dụng ngôn ngữ như chú thích hình ảnh bằng cách tạo một câu từ một từ khóa duy nhất.

- **Nhiều-nhiều:** Mô hình sử dụng nhiều đầu vào để dự đoán nhiều đầu ra. Ví dụ: bạn có thể tạo một công cụ dịch ngôn ngữ bằng RNN, với khả năng phân tích câu và cấu trúc chính xác các từ trong một ngôn ngữ khác.
- **Nhiều-một:** Một số đầu vào được ánh xạ đến một đầu ra. Loại này rất hữu ích trong các ứng dụng như phân tích cảm xúc, trong đó mô hình dự đoán cảm xúc của khách hàng như tích cực, tiêu cực và trung lập từ lời chứng thực đầu vào.

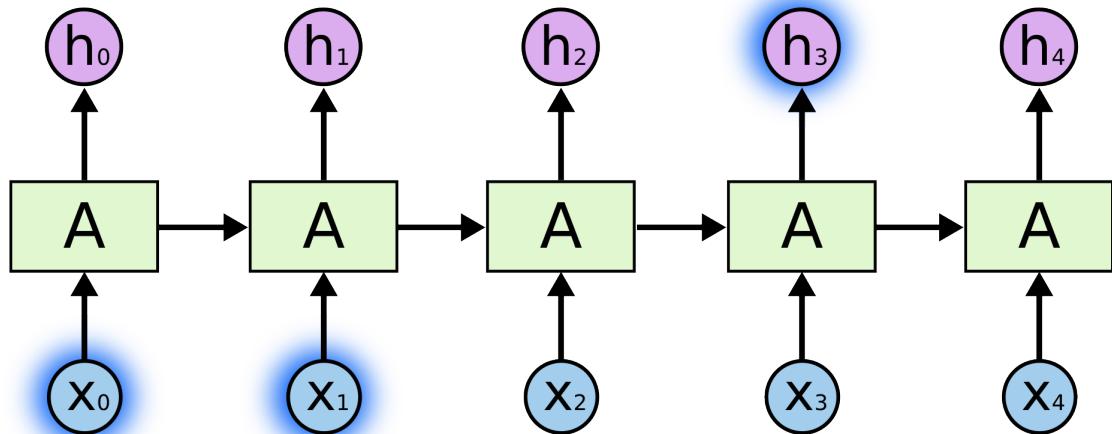


Hình 2: Một số kiểu nơ-ron hồi quy phổ biến

2.3.1 Vấn đề phụ thuộc xa của RNN

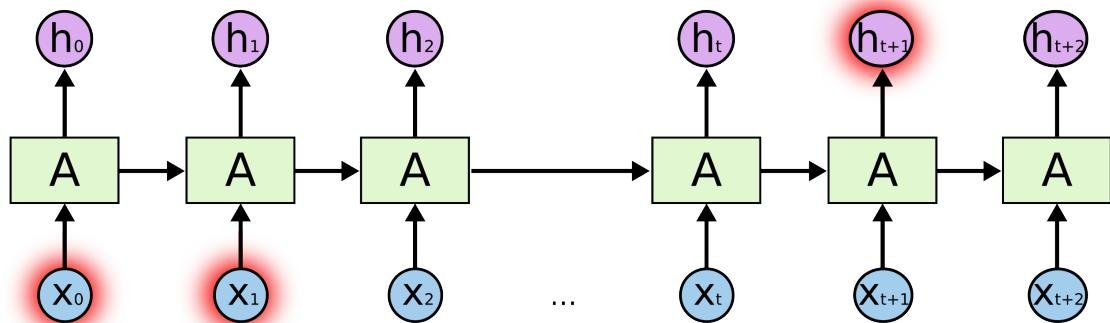
Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như ta sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời. Nếu mà RNN có thể làm được việc đó thì chúng sẽ cực kì hữu dụng, tuy nhiên liệu chúng có thể làm được không? Câu trả lời là còn tùy.

Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Ví dụ, ta có câu: “các đám mây trên bầu trời” thì ta chỉ cần đọc tới “các đám mây trên bầu” là đủ biết được chữ tiếp theo là “trời” rồi. Trong tình huống này, khoảng cách tối thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được.



Hình 3: Phụ thuộc ngắn hạn trên RNN

Nhưng trong nhiều tình huống ta buộc phải sử dụng nhiều ngữ cảnh hơn để suy luận. Ví dụ, dự đoán chữ cuối cùng trong đoạn: “I grew up in France... I speak fluent French.”. Rõ ràng là các thông tin gần (“I speak fluent”) chỉ có phép ta biết được đằng sau nó sẽ là tên của một ngôn ngữ nào đó, còn không thể nào biết được đó là tiếng gì. Muốn biết là tiếng gì, thì ta cần phải có thêm ngữ cảnh “I grew up in France” nữa mới có thể suy luận được. Rõ ràng là khoảng cách thông tin lúc này có thể đã khá xa rồi.



Hình 4: Phụ thuộc dài hạn trên RNN

Thật không may là với khoảng cách càng lớn dần thì RNN bắt đầu không thể nhớ và học được nữa.

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

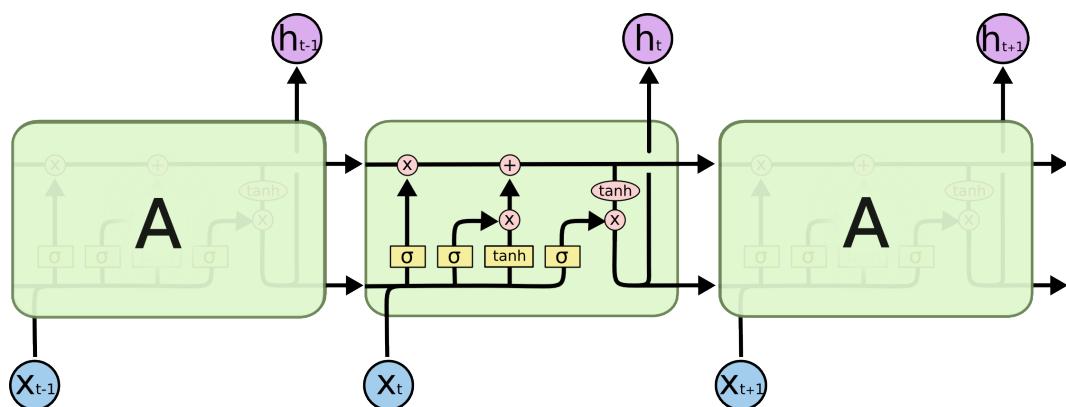
2.4.1 Giới thiệu tổng quan

Mạng bộ nhớ dài - ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới

thiệu bởi Hochreiter & Schmidhuber vào năm 1997, và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

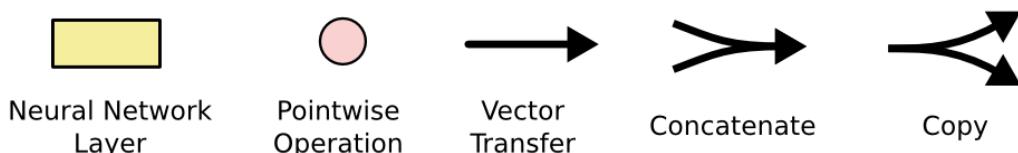
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

Mỗi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Với mạng RNN chuẩn, các mô-đun này có cấu trúc rất đơn giản, thường là một tầng tanh. LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.



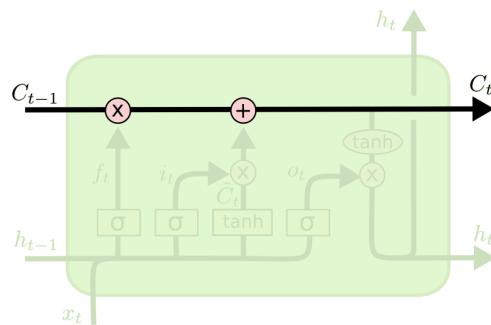
Hình 5: Mô hình LSTM với 4 lớp tương tác

2.4.2 Ý tưởng đằng sau LSTM



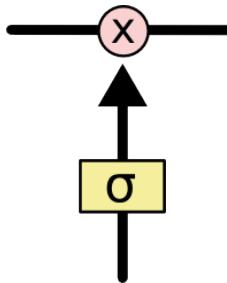
Hình 6: Diễn giải các kí hiệu trong đồ thị

- Ý tưởng chính của LSTM là thành phần 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.



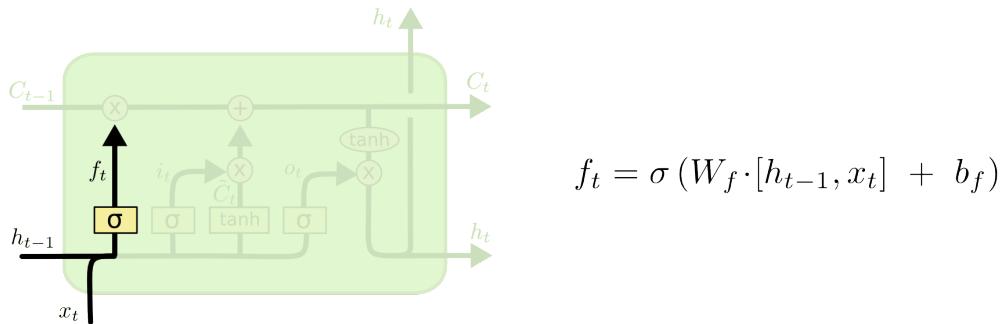
Hình 8: Một cổng của hàm sigmoid trong LSTM

- Hàm sigmoid sẽ cho đầu ra là một giá trị xác xuất 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.

2.4.3 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’.

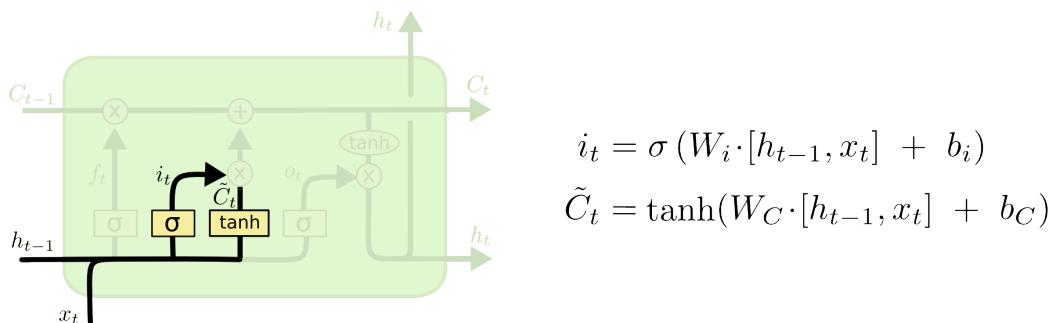
Trở lại ví dụ về ngôn ngữ, chúng ta đang cố gắng dự báo từ tiếp theo dựa trên toàn bộ những từ trước đó. Trong những bài toán như vậy, ô trạng thái có thể bao gồm loại của chủ ngữ hiện tại, để cho đại từ ở câu tiếp theo được sử dụng chính xác. Chẳng hạn như chúng ta đang mô tả về một người bạn là con 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 cổng 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 cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một vec 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.

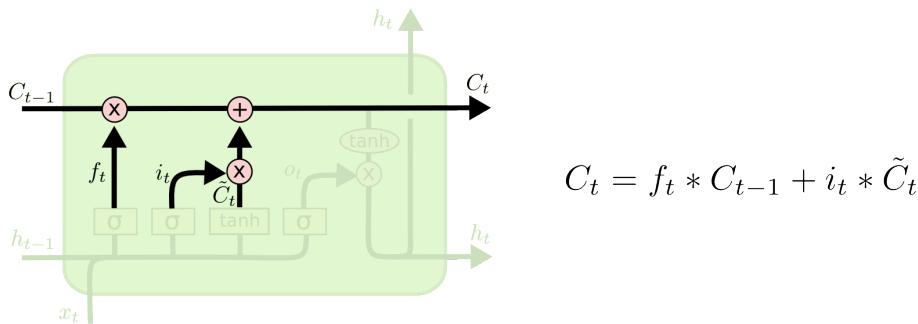
Trong ví dụ của mô hình ngôn ngữ, chúng ta muốn thêm loại của một chủ ngữ mới vào ô trạng thái để thay thế phần trạng thái cũ muốn quên đi.



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

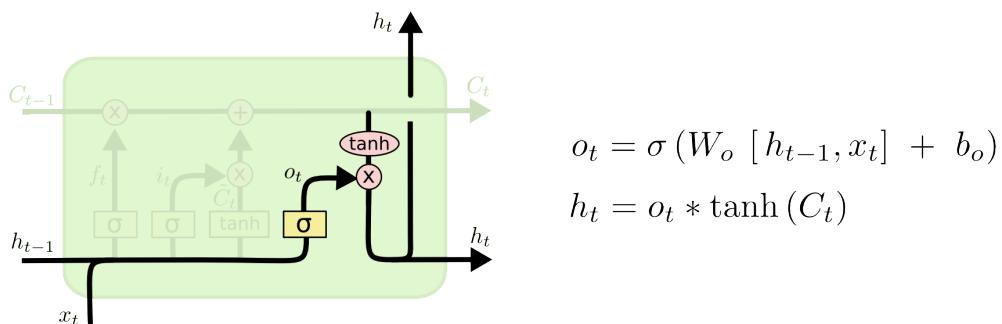
Dây là thời điểm để cập nhật một ô trạng thái cũ C_{t-1} sang một trạng thái mới C_t . Những bước trước đó đã quyết định làm cái gì, và tại bước này chỉ cần thực hiện nó.

Chúng ta nhân trạng thái cũ với f_t tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử đề 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

2.4.4 Ưu và nhược điểm của mô hình LSTM

- Ưu điểm

- LSTM được thiết kế để xử lý dữ liệu chuỗi và giữ thông tin từ quá khứ, giúp nó phù hợp cho nhiều loại dự đoán chuỗi như ngôn ngữ tự nhiên, dự đoán chuỗi thời gian, v.v.
- LSTM có khả năng học và giữ lại thông tin lâu dài, giúp chúng phát hiện và theo dõi các mối quan hệ phức tạp trong dữ liệu dài hạn.
- LSTM sử dụng cỗng quên (forget gate) để kiểm soát việc giữ và quên thông tin, giúp giảm hiện tượng triệt tiêu gradient, vấn đề thường gặp trong các mô hình RNN.
- Có thể kết hợp nhiều lớp LSTM để tạo thành các mô hình học sâu phức tạp hơn, linh hoạt trong việc xử lý nhiều loại dữ liệu và nhiều mức độ phức tạp.

- Nhược điểm

- Mô hình LSTM có khả năng tính toán khá phức tạp, đặc biệt khi được sử dụng trong các mô hình lớn với nhiều tham số.

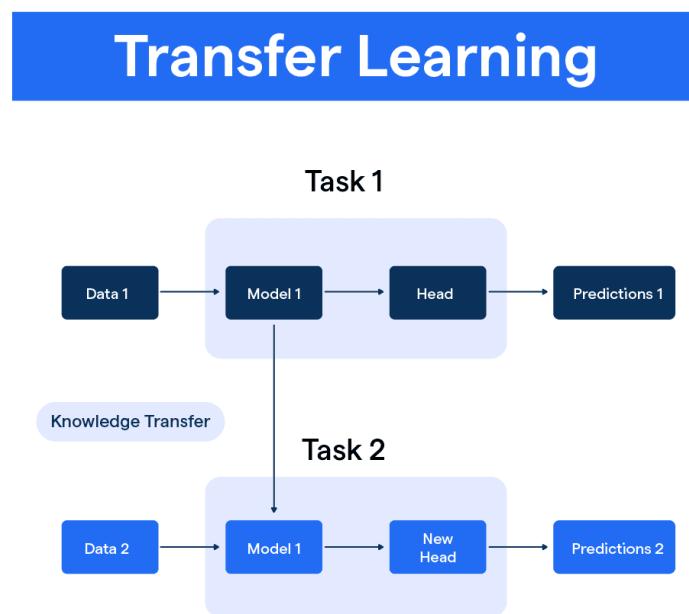
- Do tính toán phức tạp, việc huấn luyện mô hình LSTM có thể mất nhiều thời gian, đặc biệt là khi làm việc với dữ liệu lớn.
- Mô hình LSTM có nguy cơ cao về overfitting, đặc biệt khi kích thước dữ liệu huấn luyện ít và mô hình quá phức tạp.
- Mặc dù LSTM giảm vấn đề triệt tiêu gradient, nhưng không giải quyết hoàn toàn vấn đề gradient exploding.

2.5 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.

2.5.1 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.



Hình 13: Transfer learning

2.5.2 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:

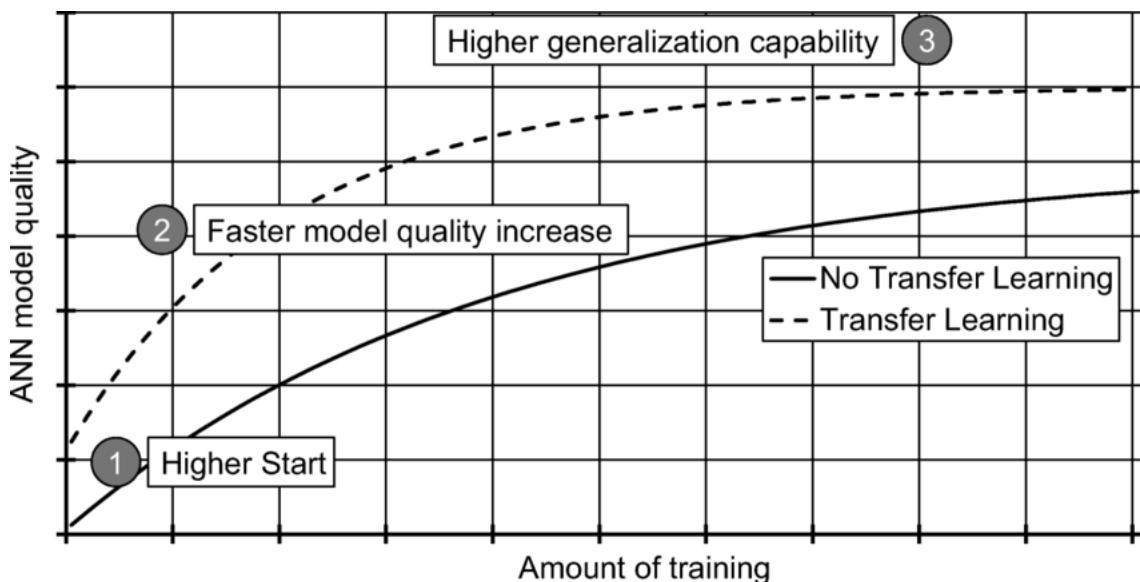
- Phân loại văn bản: Phân loại các tài liệu văn bản (tin tức, đánh giá,...)
- Phân tích cảm xúc: Xác định cảm xúc được thể hiện trong văn bản.
- Dịch máy: Dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác.

- 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.

2.5.3 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 khái quát hóa của mô hình.

Bất lợi:

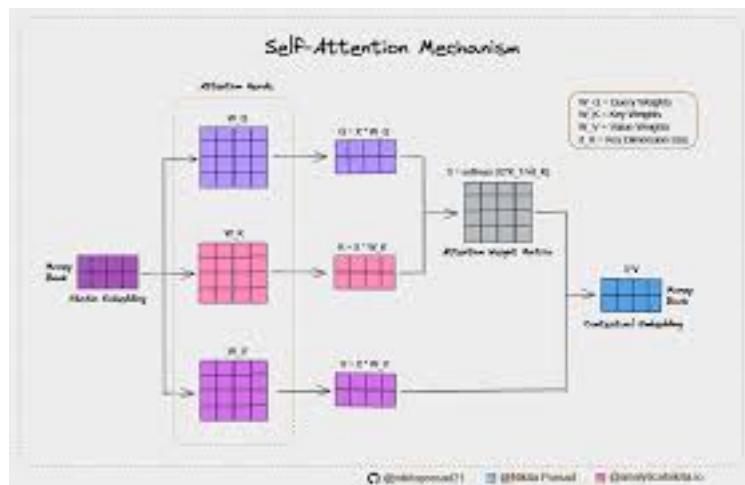
- Có thể xảy ra hiện tượng "negative transfer" nếu miền nguồn và miền đích quá khác biệt.
- Khó khăn trong việc lựa chọn mô hình tiền huấn luyện phù hợp.

2.6 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.

2.6.1 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

2.6.2 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.

2.6.3 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

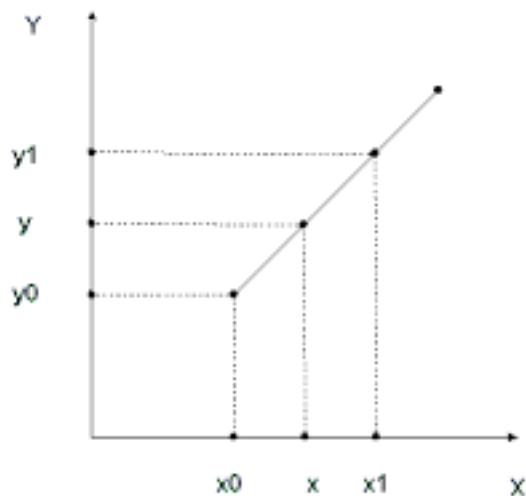
- **Các lĩnh vực khác:**

- Sinh học
- Âm nhạc

2.6.4 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.

2.7 Nội suy tuyến tính (Linear Interpolation)



Hình 16: Nội suy tuyến tính

Nội suy tuyến tính là một phương pháp ước tính giá trị của một hàm số tại một điểm chưa biết, dựa trên giá trị của hàm số tại hai điểm đã biết.

Giả sử ta có hai điểm dữ liệu (x_1, y_1) và (x_2, y_2) , và ta muốn ước tính giá trị của hàm số $y = f(x)$ tại điểm x nằm giữa x_1 và x_2 . Nội suy tuyến tính giả định rằng hàm số $f(x)$ là một đường thẳng giữa hai điểm đã biết.

Công thức nội suy tuyến tính được cho bởi:

$$y = y_1 + \frac{(x-x_1)}{(x_2-x_1)}(y_2-y_1)$$

Trong lĩnh vực học máy, nội suy tuyến tính thường được sử dụng để tạo ra các điểm dữ liệu mới từ tập dữ liệu hiện có. Ví dụ, trong bài toán xử lý ảnh, nội suy tuyến tính có thể được sử dụng để phóng to hoặc thu nhỏ ảnh.

2.7.1 Lợi ích của nội suy tuyến tính

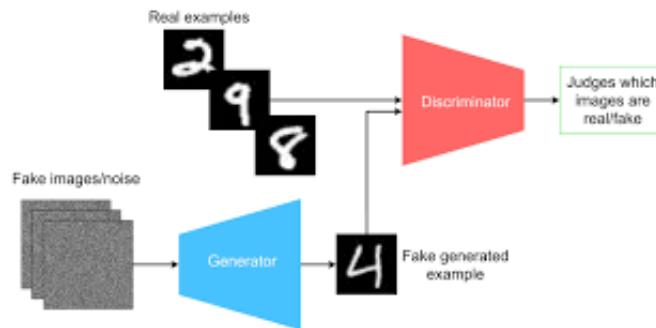
- Đơn giản:** Nội suy tuyến tính là một phương pháp đơn giản và dễ hiểu để ước lượng giá trị của một hàm số tại một điểm chưa biết.

2.7.2 Hạn chế của nội suy tuyến tính

- Giả định tuyến tính:** Nội suy tuyến tính giả định rằng hàm số là một đường thẳng giữa hai điểm đã biết, điều này có thể không phản ánh đúng mối quan hệ giữa các điểm dữ liệu.

2.8 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 17: Generative Adversarial Networks (GANs)

2.8.1 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.

2.8.2 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

2.8.3 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.

2.8.4 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 phức tạp.

2.9 Giới thiệu về thư viện Tensorflow

TensorFlow là một thư viện mã nguồn mở được phát triển bởi Google, được sử dụng để xây dựng các mô hình học máy và mạng nơ-ron. TensorFlow sẽ giúp giải quyết các bài toán nhanh chóng và đơn giản hơn thông qua việc tạo các mô hình tính toán trong Machine Learning trên máy tính.



TensorFlow

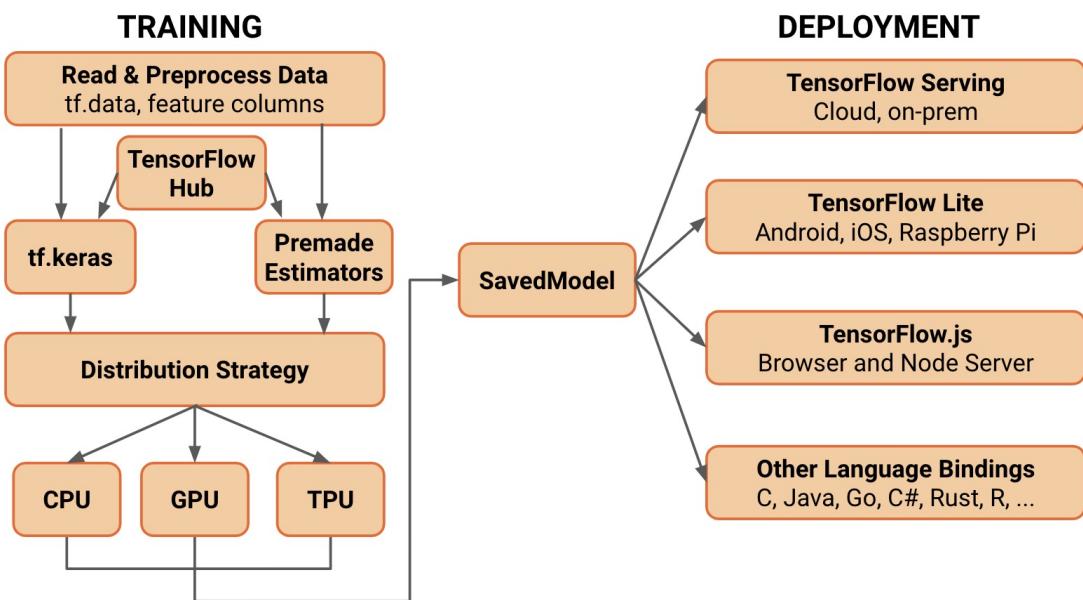
Hình 18: Thư viện Tensorflow

2.9.1 Nguyên lý hoạt động

Tensorflow là một thư viện tính toán số và biểu diễn dữ liệu bằng cấu trúc đồ thị (graph) để tạo và huấn luyện các mô hình học máy. Các đồ thị này bao gồm các nút (nodes) và các cạnh (edges) được sử dụng để biểu diễn các phép tính và dữ liệu tương ứng trong mô hình. Một cách tổng quát, quá trình huấn luyện mô hình học máy trong TensorFlow trải qua các bước sau:

- **Xây dựng đồ thị tính toán:** Người dùng xác định cấu trúc đồ thị tính toán bằng cách khai báo các biến (variables) và các phép tính (operations) trong mô hình.
- **Định nghĩa hàm mất mát:** Hàm mất mát (loss function) được định nghĩa để đo lường sự khác biệt giữa đầu ra dự đoán của mô hình và giá trị thực tế.
- **Tối ưu hóa mô hình:** Quá trình tối ưu hóa được sử dụng để tìm ra các giá trị tham số tối ưu nhằm giảm thiểu hàm mất mát.
- **Huấn luyện mô hình:** Dữ liệu huấn luyện được đưa vào mô hình để huấn luyện và cập nhật các giá trị tham số.

- **Đánh giá mô hình:** Dữ liệu kiểm tra được sử dụng để đánh giá hiệu suất của mô hình.
- **Sử dụng mô hình:** Mô hình đã huấn luyện được sử dụng để dự đoán và phân loại các dữ liệu mới.



Hình 19: Nguyên lý hoạt động của Tensorflow

2.9.2 Thuộc tính cơ bản trong Tensorflow

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

- **Tensors:** Là đối tượng chính của TensorFlow, đại diện cho dữ liệu và kết quả tính toán. Tensors là một mảng đa chiều có các phần tử có cùng kiểu dữ liệu.

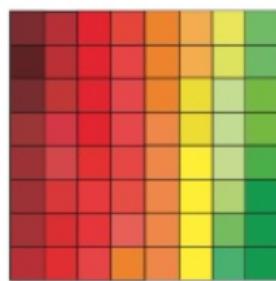
tensor = multidimensional array

vector



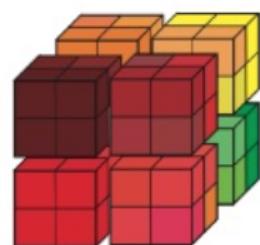
$$\mathbf{v} \in \mathbb{R}^{64}$$

matrix



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

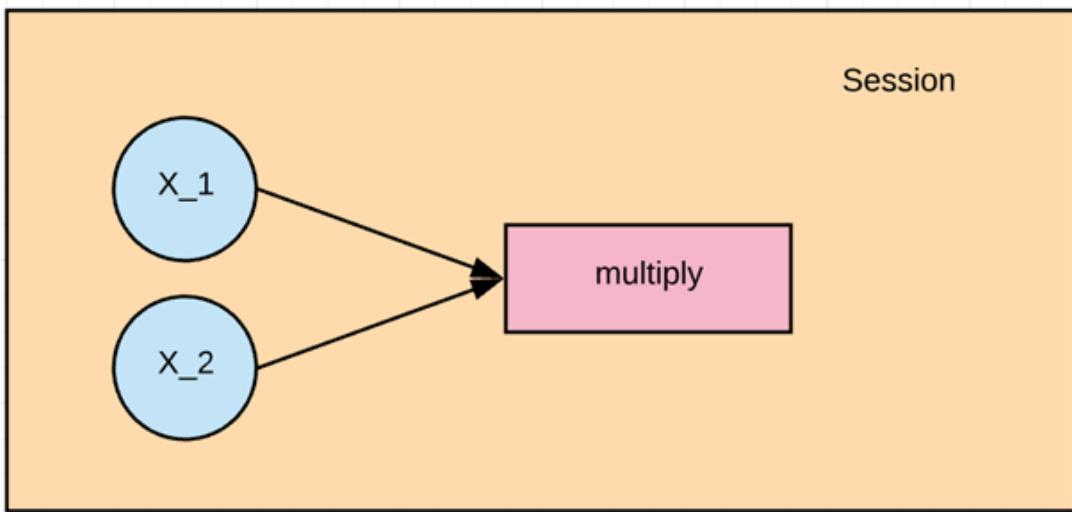
tensor



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

Hình 20: Hình ảnh minh họa của một tensor và so sánh với vector và ma trận.

- **Phép tính toán(Operations):** Là các hoạt động được thực hiện trên các tensors. Một số hoạt động cơ bản của TensorFlow bao gồm phép cộng, trừ, nhân và chia.
- **Biến(Variables):** Là các đối tượng được sử dụng để lưu trữ trạng thái thay đổi được trong quá trình huấn luyện mô hình. Variables có thể được khởi tạo với giá trị cố định hoặc giá trị ngẫu nhiên và có thể được cập nhật trong quá trình huấn luyện.
- **Đồ thị(Graphs):** Là các biểu đồ đại diện cho các phép tính và các kết nối giữa chúng. Graphs được sử dụng để mô tả cấu trúc của mô hình học máy và tạo ra các tính toán hiệu quả trên nhiều thiết bị tính toán.
- **Sessions:** Là một phiên làm việc TensorFlow, chứa tất cả các biến và phép tính cần thiết để thực hiện tính toán. Sessions được sử dụng để thực thi các tính toán trong TensorFlow.



Hình 21: Minh họa cho 1 graph, session, operation và variable.

- **Placeholders:** Là các đối tượng được sử dụng để đại diện cho các tensors được cung cấp vào mô hình trong quá trình huấn luyện hoặc kiểm tra. Placeholders được sử dụng để đảm bảo rằng dữ liệu đầu vào có thể được cung cấp cho mô hình.

Các thuộc tính này là những thành phần cơ bản của TensorFlow và cung cấp các công cụ cần thiết để xây dựng và huấn luyện các mô hình học máy hiệu quả.

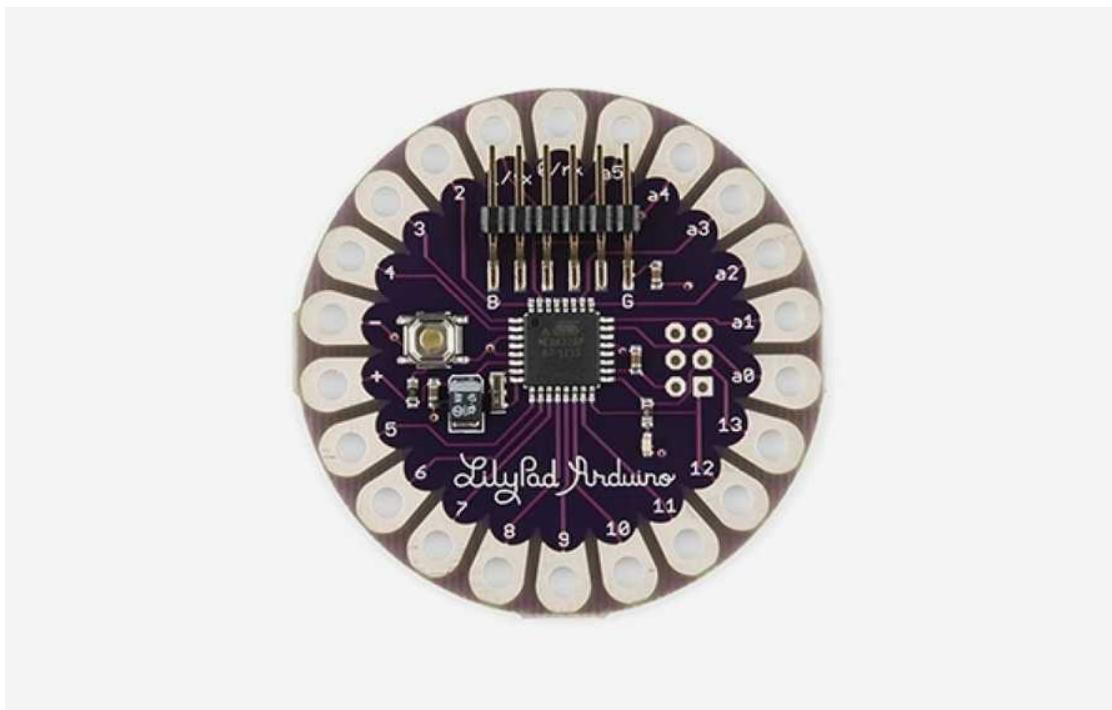
2.9.3 Ưu điểm của TensorFlow

TensorFlow cung cấp các công cụ hỗ trợ cho việc xây dựng, huấn luyện và triển khai các mô hình học máy một cách linh hoạt và hiệu quả trên nhiều nền tảng khác nhau. TensorFlow sở hữu nhiều ưu điểm vượt trội như sau:

- Tính linh hoạt: TensorFlow không chỉ hỗ trợ một loạt các thuật toán học máy và học sâu, mà còn cho phép người dùng tạo ra các mô hình học máy tùy chỉnh của riêng họ.
- Khả năng mở rộng: TensorFlow có thể chạy trên nhiều nền tảng khác nhau, từ máy tính cá nhân đến các máy chủ lớn, và thậm chí là các thiết bị di động.
- Hiệu suất cao: TensorFlow tận dụng tối đa sức mạnh của phần cứng bằng cách sử dụng đa luồng và tính toán song song.
- Hỗ trợ cộng đồng: Với một cộng đồng người dùng lớn và tích cực, TensorFlow cung cấp nhiều tài liệu học tập và hỗ trợ từ cộng đồng.

2.10 Giới thiệu về Arduino LilyPad

LilyPad Arduino là một dạng bo mạch Arduino được thiết kế đặc biệt cho các thiết bị điện tử mặc được và e-textiles. Mạch có thể được may vào vải và gắn với các nguồn điện, cảm biến và actuator bằng chỉ dẫn điện.



Hình 22: Board mạch Arduino Lilypad

Arduino Lilypad được thiết kế và phát triển bởi Leah Buechley và SparkFun Electronics. Board mạch được thiết kế dựa trên chip điều khiển ATmega168V hoặc ATmega328V. Dưới đây là thông số kỹ thuật của Arduino LilyPad:

Bảng 1: Thông số kỹ thuật của Arduino Lilypad

Vị trí 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

Một số ứng dụng của Arduino Lilypad:

- **Quần áo thông minh:** Lilypad có thể được may vào trang phục kết hợp với các cảm biến khác tạo ra một bộ trang phục thông minh với các chức năng theo dõi thông số cơ thể.
- **Thiết bị đeo tay thông minh:** Có thể dùng Lilypad như như một đồng hồ đeo tay thông minh, găng tay thông minh để theo dõi cử chỉ bàn tay, hoặc cử chỉ của cơ thể.

2.11 Giới thiệu về cảm biến uốn cong(Flex sensor)

2.11.1 Tổng quan

Cảm biến uốn cong (flex sensor) thực chất là một biến trở có khả năng thay đổi điện trở khi được uốn cong. Vì điện trở biến thiên tỷ lệ thuận trực tiếp với độ uốn cong, nên nó thường được gọi là Potentiometer Uốn Cong.

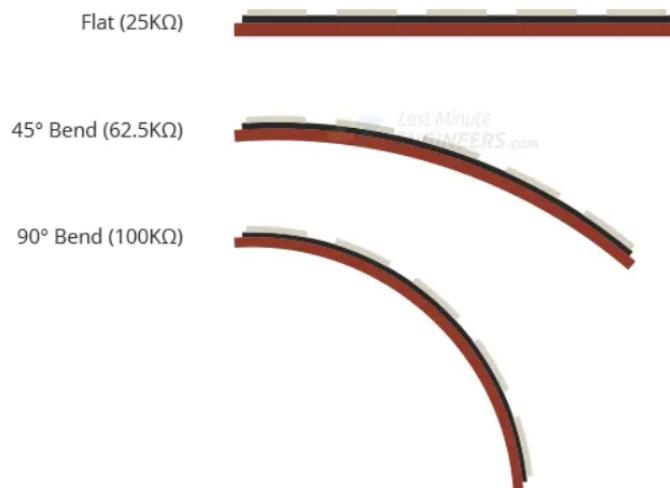
Cảm biến uốn cong thường có hai kích thước phổ biến: 2.2 inch (5.588cm) và 4.5 inch (11.43cm).



Hình 23: Flex sensor

2.11.2 Nguyên lý hoạt động

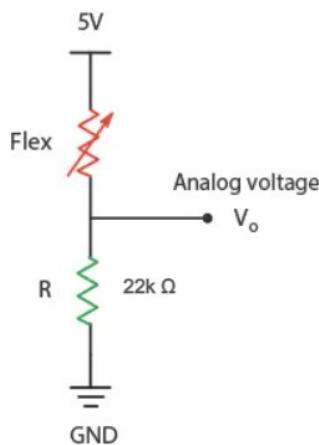
- Mực dẫn trên cảm biến chính là một loại trở kháng.
 - Khi cảm biến thẳng, trở kháng này khoảng 25k.
 - Khi cảm biến được uốn cong, lớp mực dẫn bị căng, dẫn đến giảm diện tích tiếp diện (hãy tưởng tượng như việc căng một sợi dây cao su) và tăng trở kháng. Ở góc uốn cong 90°, trở kháng này khoảng 100K.
 - Khi cảm biến được thẳng lại, trở kháng trở về giá trị ban đầu.
- Bằng cách đo trở kháng, ta có thể xác định được mức độ uốn cong của cảm biến.



Hình 24: Hình dạng uốn cong của flex sensor

2.11.3 Đọc giá trị

Cách đơn giản nhất để đọc cảm biến uốn cong (Flex Sensor) là kết hợp với một trở kháng tĩnh để tạo thành một bộ chia điện áp, từ đó tạo ra một điện áp biến đổi có thể đọc được bởi bộ chuyển đổi tương tự sang số của vi điều khiển.



Hình 25: 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.

2.12 Giới thiệu về cảm biến con quay gia tốc

Cảm biến gia tốc hoạt động dựa trên nguyên lý của lực quán tính, thường thông qua cấu trúc vi cơ điện tử (MEMS). Bên trong cảm biến, có một khối lượng nhỏ gọi là vi khối lượng, được gắn vào một khung bằng các lò xo siêu nhỏ. Khi cảm biến trải qua gia tốc, vi khối lượng này sẽ dịch chuyển do lực quán tính. Sự dịch chuyển này được chuyển đổi thành tín hiệu điện bằng cách sử dụng các phương pháp như cảm ứng điện dung, hiệu ứng piezoelectric hoặc cầu Wheatstone.

Nhờ vào cấu trúc và nguyên lý hoạt động tinh vi, cảm biến gia tốc có thể đo lường chính xác gia tốc theo nhiều trục, thường là ba trục (x, y, z), giúp cung cấp thông tin chi tiết về chuyển động và vị trí của vật thể trong không gian.

2.12.1 Nguyên lý hoạt động

2.12.1.a Sử dụng hiệu ứng Piezoelectric

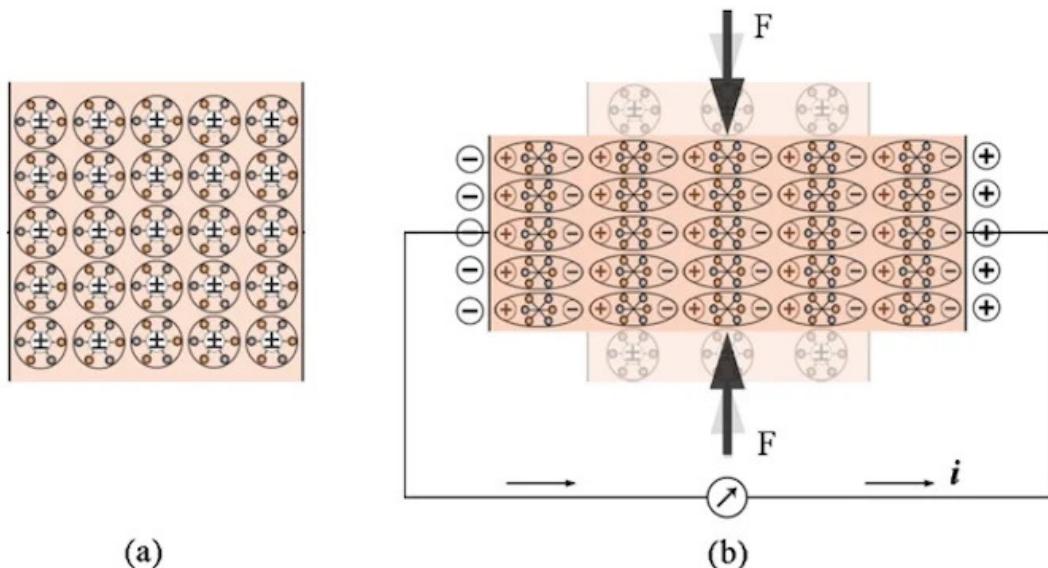
Hiệu ứng Piezoelectric là hiện tượng mà một số vật liệu nhất định có khả năng tạo ra điện tích khi chịu áp lực cơ học. Từ “Piezoelectric” bắt nguồn từ từ tiếng Hy Lạp “piezein”, có nghĩa là “ấn hoặc nén”, mô tả chính xác quá trình tạo ra điện năng thông qua áp lực.

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 chỉ 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 nhau 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.

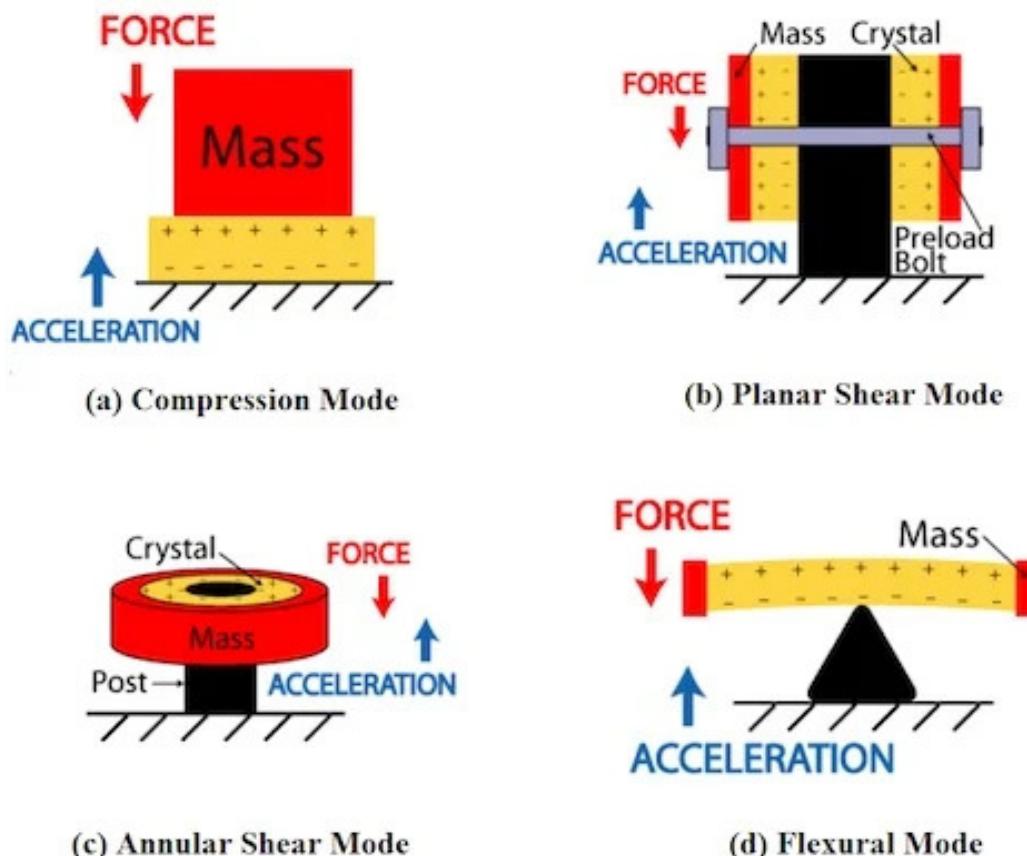


Hình 26: Hiệu ứng Piezoelectric

Cấu trúc của cảm biến gia tốc 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.



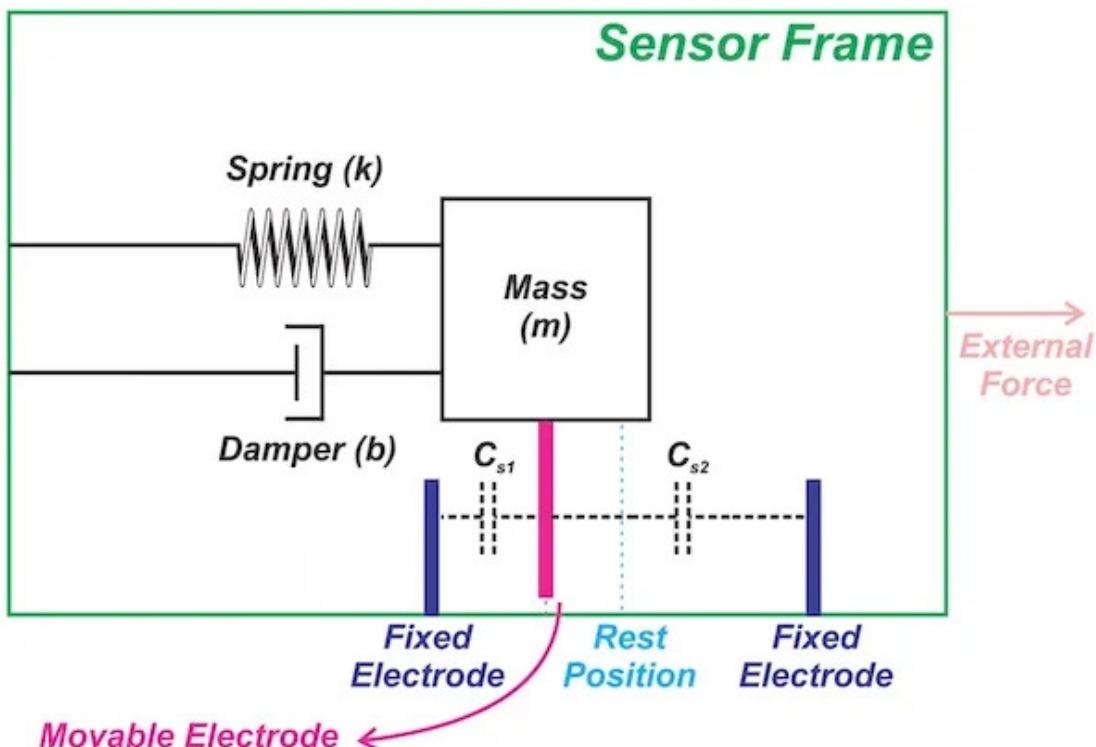
Hình 27: Các kiểu thiết kế cơ học của cảm biến gia tốc

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.

2.12.1.b Sử dụng phương pháp điện dung

Cấu trúc lò xo-khối nặng-giảm xóc: Cấu trúc này chuyển đổi tốc độ của khung cảm biến thành sự dịch chuyển của khối 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 tốc độ được áp dụng.

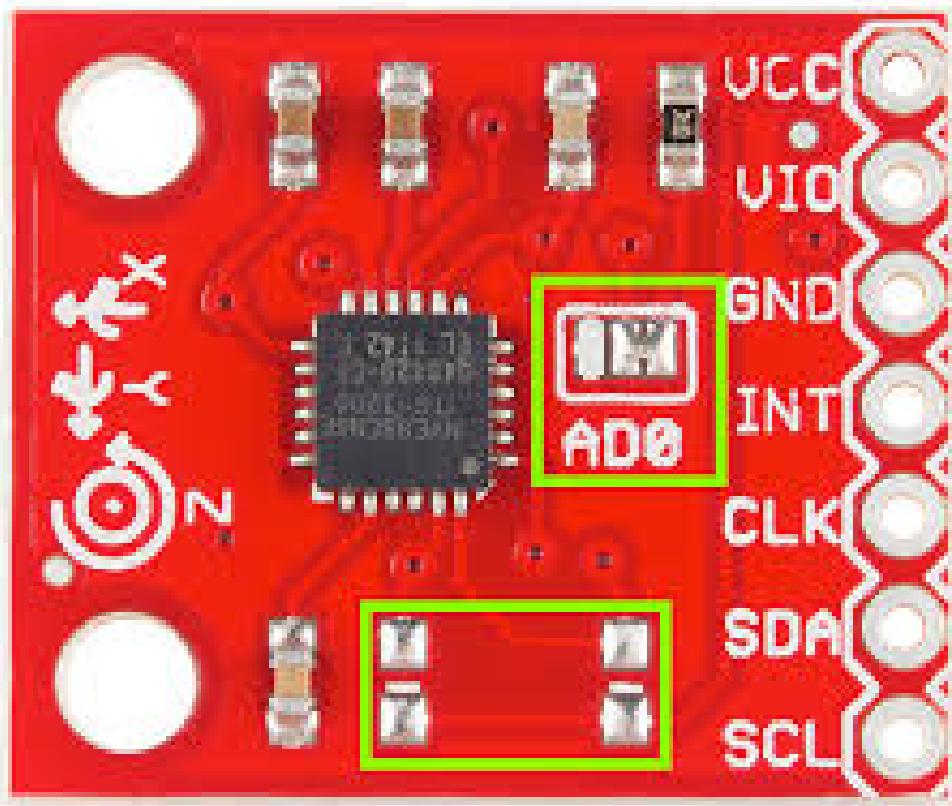


Hình 28: Cấu trúc của một cảm biến tốc sử dụng phương pháp điện dung

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 khối lượng tham chiếu. Thiết kế này hình thành hai tụ điện biến thiên C_1 và C_2 , như được hiển thị trong hình 30.

Khi khối lượng tham chiếu di chuyển theo một hướng dưới ảnh hưởng của tốc độ, 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 khối lượng tham chiếu có sẵn ta có thể tính toán được tốc độ đầu vào.

2.12.2 Giới thiệu về module cảm biến ITG 3200



Hình 29: Cảm biến gia tốc góc ITG 3200

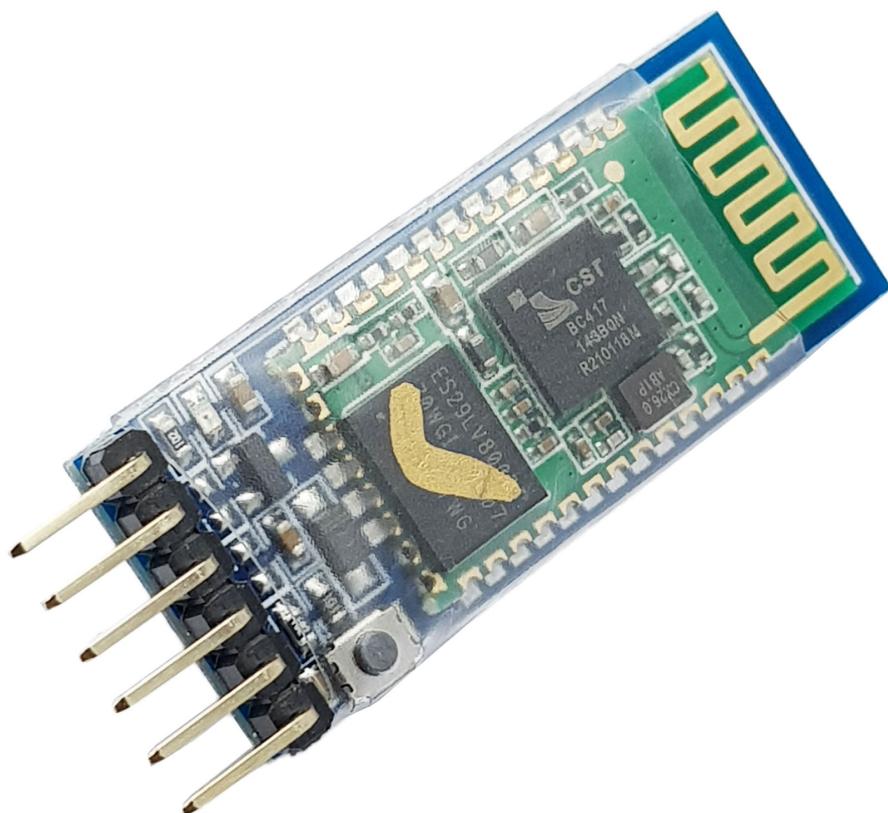
Cảm biến gia tốc góc ITG-3200 là một loại cảm biến gia tốc góc (gyroscope) được sản xuất bởi InvenSense. Đây là một cảm biến quan trọng được sử dụng rộng rãi trong các ứng dụng đo lường và điều khiển chuyển động. Dưới đây là một số đặc điểm chính và thông tin cơ bản về ITG-3200:

Đặc điểm chính

- **Trục đo:** ITG-3200 là cảm biến 3 trục, có khả năng đo gia tốc góc trên ba trục X, Y và Z.
- **Dải đo (Range):** Cảm biến này có thể đo gia tốc góc lên đến ± 2000 độ/giây, phù hợp cho nhiều ứng dụng khác nhau từ điều khiển bay của máy bay không người lái đến các thiết bị chơi game.
- **Tốc độ lấy mẫu (Sampling Rate):** Cảm biến hỗ trợ tốc độ lấy mẫu lên đến 8 kHz, giúp thu thập dữ liệu nhanh và chính xác.
- **Giao tiếp:** ITG-3200 sử dụng giao tiếp I2C (Inter-Integrated Circuit), giúp dễ dàng kết nối với các vi điều khiển và các hệ thống nhúng khác.
- **Nguồn điện:** Cảm biến hoạt động với điện áp từ 2.1V đến 3.6V, và thường được cấp điện áp 3.3V.

- **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.

2.13 Giới thiệu về module bluetooth HC05



Hình 30: module bluetooth HC-05

HC-05 là thiết bị Bluetooth tốt nhất sử dụng giao thức truyền thông UART. HC-05 Bluetooth có nhiều tính năng khác biệt so với tất cả các thiết bị Bluetooth khác vì có nhiều chân và chức năng.

Module thường sử dụng giao tiếp nối tiếp UART với các chân TX và RX ở tốc độ baud 9600. Có giao tiếp truyền dữ liệu hai chiều và có thể hoạt động như một slave và master.



Module Bluetooth chỉ cung cấp khả năng giao tiếp trong khoảng cách ngắn do có giới hạn, hầu hết do đảm bảo tốc độ và tính bảo mật của nó.

Mô tả sơ đồ chân

- **Chân VCC** Giống như mọi thiết bị khác, HC05 Modules cũng phụ thuộc vào nguồn điện để hoạt động và chân VCC cấp nguồn điện từ bên ngoài.
- **Chân GND** Chân nối đất module.
- **Chân TX** Chân truyền dữ liệu giao thức UART
- **Chân RX** Chân nhận dữ liệu trong giao tiếp UART.
- **Chân State** Báo trạng thái kết nối của Bluetooth.
- **Chân Enable/key C** là chân thay đổi chế độ giữa chế độ dữ liệu và chế độ dòng lệnh bằng cách cấp tín hiệu bên ngoài. Cấp logic cao sẽ chuyển sang chế độ dòng lệnh và trạng thái logic thấp sẽ chuyển sang chế độ dữ liệu. Chế độ thiết bị mặc định là chế độ dữ liệu.
- **Nút button** Các chế độ dữ liệu và lệnh có thể thay đổi thông qua một nút nhấn có trên module.
- **Đèn LED:** Đèn LED hiển thị trạng thái của Module HC-05.

Đặc tính Bluetooth HC-05

- Module Bluetooth HC-05 cung cấp hai giao tiếp trong khoảng cách ngắn hơn với tốc độ nhanh.
- Có chân enable cho phép chuyển đổi giữa chế độ dòng lệnh và dữ liệu. Module có giao thức UART dễ dàng giao tiếp với bất kỳ bộ vi điều khiển hoặc hệ thống nào.
- Phạm vi giao tiếp lên đến 8 - 10 mét nhưng sẽ giảm xuống khi có vật cản.
- Thiết bị sử dụng nguồn điện 5V.
- Module có thể làm Master hoặc Slave.
- Hỗ trợ tốc độ truyền:
 - 9600
 - 19200
 - 38400
 - 57600
 - 115200
 - 230400
 - 460800

2.14 Giới thiệu về ngôn ngữ ký hiệu

Ngôn ngữ ký hiệu, còn được gọi là ngôn ngữ dấu hiệu hoặc thủ ngữ, là ngôn ngữ sử dụng các biểu hiện của bàn tay thay cho âm thanh của tiếng nói. Ngôn ngữ ký hiệu được người khiếm thính tạo ra nhằm giúp họ có thể giao tiếp với nhau trong cộng đồng của mình và tiếp thu tri thức của xã hội.

Ở Việt Nam, ngôn ngữ ký hiệu đã được hình thành từ rất lâu. Tuy nhiên, 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. Mãi đến năm 1996, một tiến sĩ ngôn ngữ học người Mỹ là 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 Nam². 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.

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.

			
0 không	1 một	2 hai	3 ba
			
4 bốn	5 năm	6 sáu	7 bảy
			
8 tám	9 chín	10 mười	11 mười một
			
12 mười hai	23 hai mươi ba	33 ba mươi ba	40 bốn mươi
			
80 tám mươi	90 chín mươi	100 một trăm	1.000 một nghìn
			
5.000 năm nghìn	10.000 mười nghìn	1.000.000 một triệu	1.000.000.000 một tỷ

Hình 31: Bảng chữ số của ngôn ngữ kí hiệu

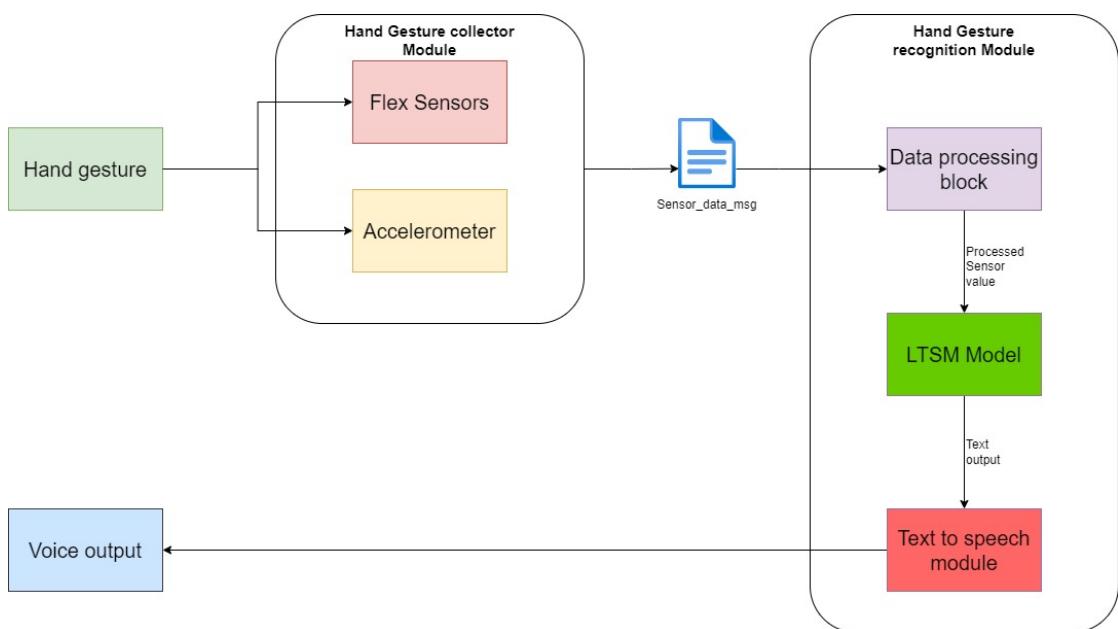


Hình 32: Bảng chữ cái của ngôn ngữ ký hiệu

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 33: 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.

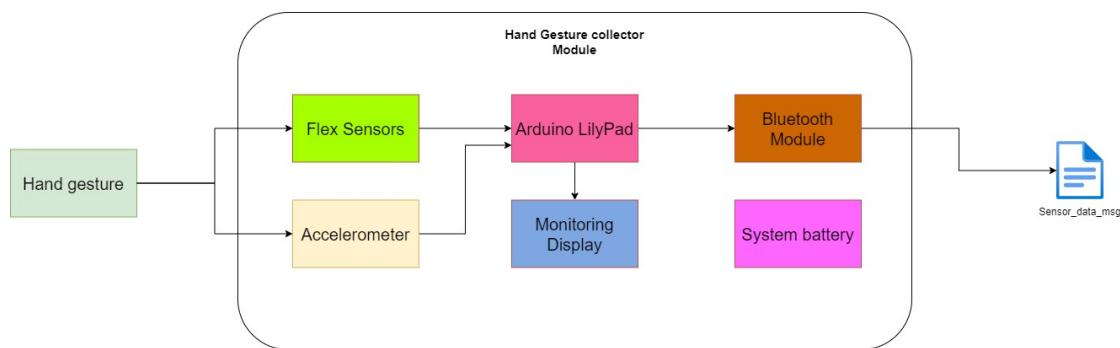
Processed Sensor Value (Giá trị cảm biến đã xử lý): Dữ liệu cảm biến được xử lý trong khối xử lý dữ liệu và các đặc trưng được trích xuất.

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



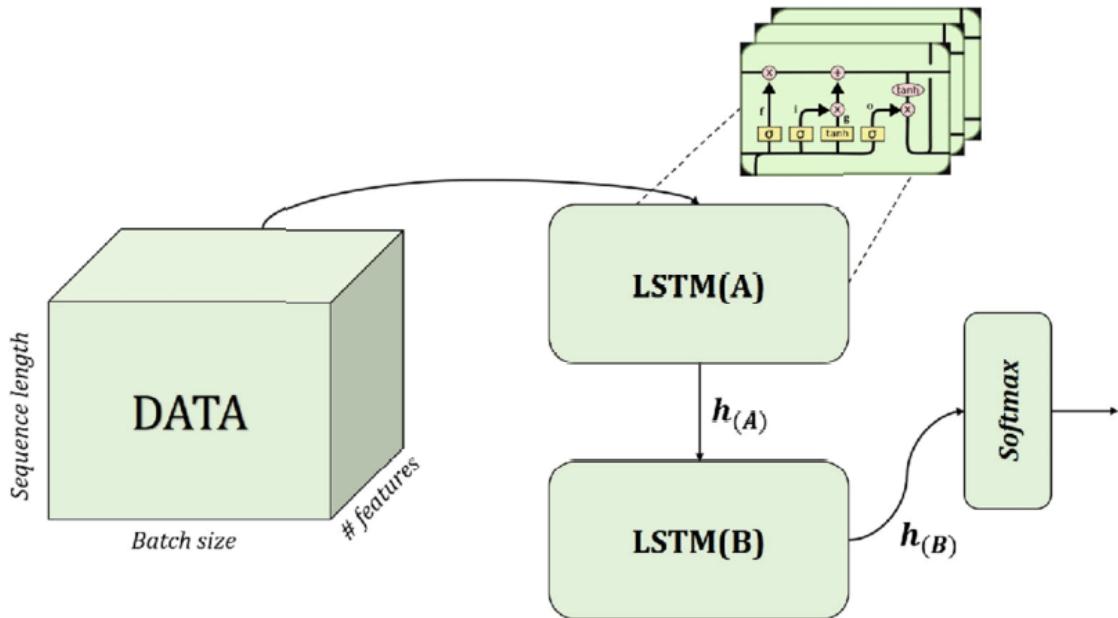
Hình 34: Kiến trúc module thu thập dữ liệu cử chỉ

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 35: Kiến trúc model dự đoán cử chỉ

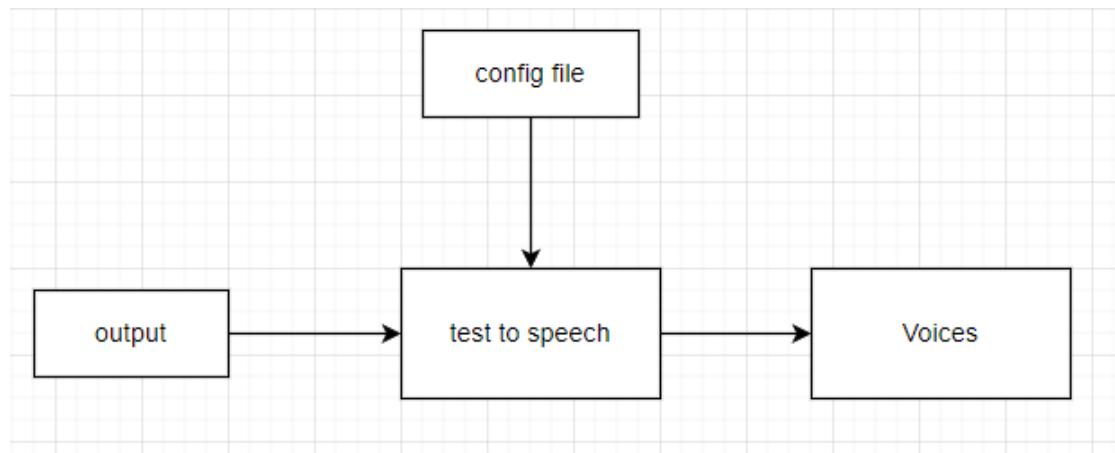
Sơ đồ mô tả một mô hình học sâu sử dụng LSTM (Long Short-Term Memory) để dự đoán xác suất của các hành động, như mua hoặc bán. Dưới đây là mô tả chi tiết của từng thành phần trong mô hình:

- 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 36: Kiến trúc module chuyển đầu ra dự đoán thành giọng nói

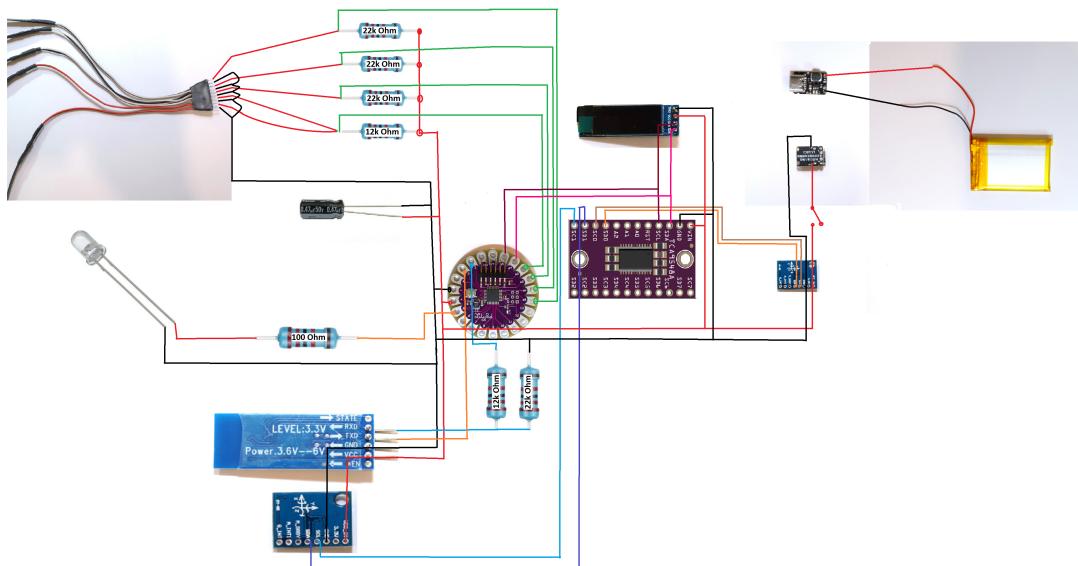
Sơ đồ mô tả module phát âm thanh sau khi dự đoán đúng một hành động sử dụng thư viện text-to-speech pyttsx3

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



Hình 37: Sơ đồ kết nối mạch

Sơ đồ này mô tả kết nối của một mạch điện tử bao gồm nhiều thành phần như cảm biến, Arduino, màn hình hiển thị, mô-đun Bluetooth, và nguồn pin. Dưới đây là mô tả chi tiết các thành phần và kết nối của chúng:

- Cảm biến Flex:
 - Cảm biến này có nhiều dây kết nối với các điện trở (22k Ohm và 12k Ohm).
 - Các dây này sau đó được kết nối với vi điều khiển (Arduino LilyPad).
- LED và Điện trở (100 Ohm):
 - LED được kết nối với một điện trở 100 Ohm và sau đó nối với vi điều khiển.
- Arduino LilyPad:
 - Dây là vi điều khiển trung tâm trong mạch, chịu trách nhiệm thu thập và xử lý dữ liệu từ các cảm biến.
 - Nó được kết nối với các cảm biến Flex, màn hình hiển thị, mô-đun Bluetooth và các điện trở khác.
- Màn hình hiển thị:
 - Màn hình hiển thị nhỏ (có thể là OLED) được kết nối với Arduino để hiển thị dữ liệu hoặc thông tin cần thiết.
- Mô-đun Bluetooth:
 - Mô-đun Bluetooth được kết nối để truyền dữ liệu không dây. Nó được cấp nguồn từ Arduino và được kết nối với các chân TX, RX của vi điều khiển.

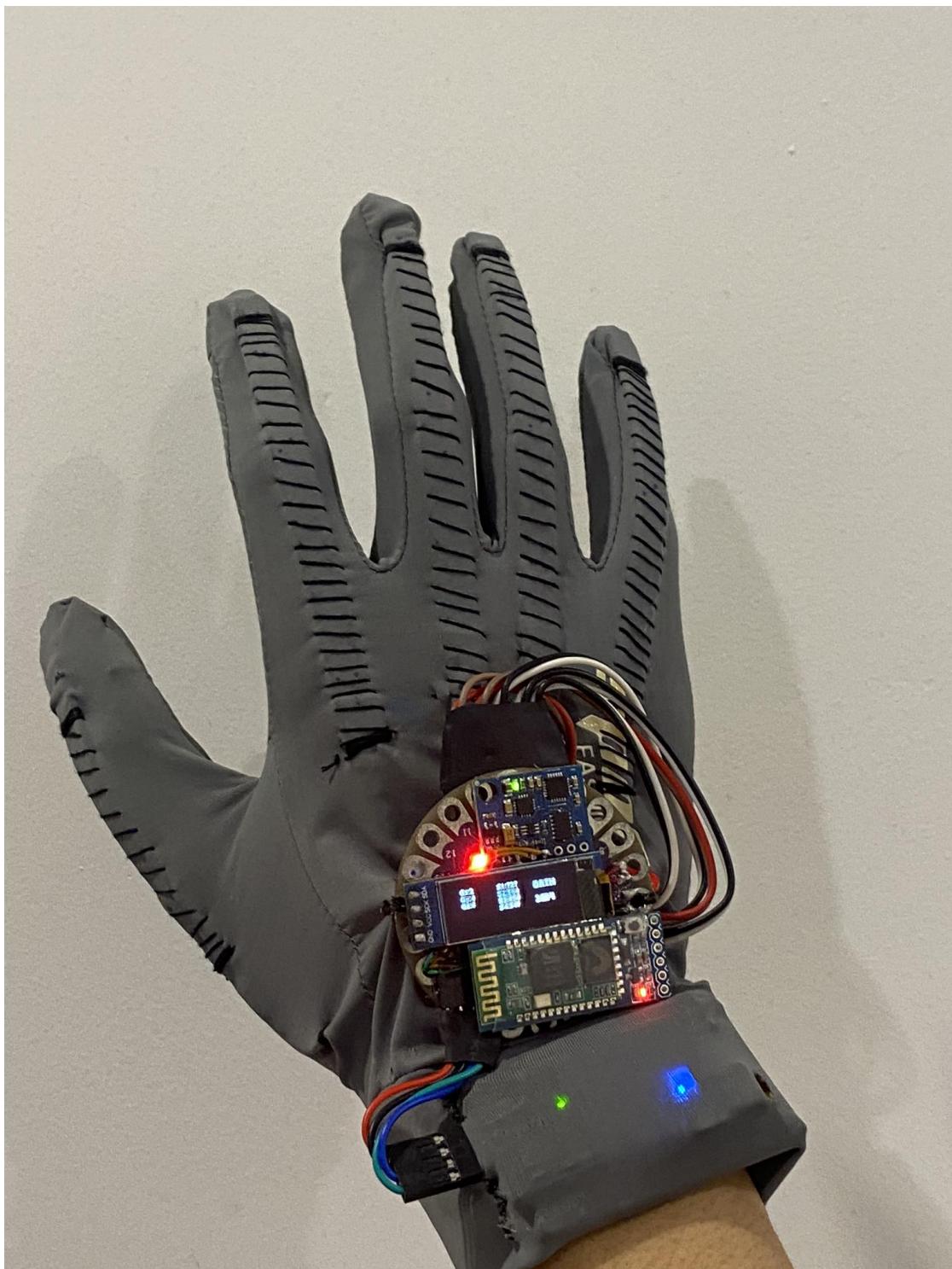


- Nguồn pin:
 - Một pin Lithium-Polymer (LiPo) được sử dụng để cung cấp năng lượng cho toàn bộ mạch.
- Các điện trở khác:
 - Có các điện trở khác trong mạch để điều chỉnh dòng điện và điện áp đi qua các thành phần.
- Mạch sạc và quản lý pin:
 - Một mạch quản lý pin (TP4056) được kết nối với pin để đảm bảo việc sạc an toàn và hiệu quả.

Dưới đây là mô tả kết nối chi tiết:

- Các dây từ cảm biến Flex được kết nối với các chân trên Arduino LilyPad thông qua các điện trở 22k Ohm và 12k Ohm. - LED được kết nối với một điện trở 100 Ohm trước khi nối với Arduino LilyPad. - Màn hình hiển thị được kết nối với các chân của Arduino LilyPad để hiển thị dữ liệu. - Mô-đun Bluetooth được kết nối với các chân TX, RX, VCC, và GND của Arduino LilyPad. - Pin LiPo cung cấp năng lượng qua mạch sạc TP4056, từ đó kết nối với Arduino và các thành phần khác để cấp nguồn.

Sơ đồ này minh họa rõ ràng cách các thành phần điện tử được kết nối với nhau để tạo thành một hệ thống hoàn chỉnh.



Hình 38: Hình ảnh sau khi hoàn thiện



4.2 Xây dựng bộ dữ liệu

Dữ liệu được thu thập từ module thu thập dữ liệu cảm biến sau đó được lưu vào file text

4.2.1 Dữ liệu được thu thập trên arduino

Dữ liệu được gửi từ Arduino có 7 thành phần chính:

```
1 gyro.readGyro(&ix,&iy,&iz);
2 ix = filter_gX.updateEstimate(ix);
3 iy = filter_gY.updateEstimate(iy);
4 iz = filter_gZ.updateEstimate(iz);
5     readFlexSensor(flexSensor);
6 flexSensor[0] = filter_flex0.updateEstimate(flexSensor[0]);
7 flexSensor[1] = filter_flex1.updateEstimate(flexSensor[1]);
8 flexSensor[2] = filter_flex2.updateEstimate(flexSensor[2]);
9 flexSensor[3] = filter_flex3.updateEstimate(flexSensor[3]);
```

Các dữ liệu được gửi lần lượt là giá tốc trục x, giá tốc trục y, giá tốc trục z, và các tín hiệu analog 1,2,3,4. với flex 1,2 (ngón trỏ và ngón cái) được nối song song với nhau sau đó nối tiếp đến A0, và các ngón còn lại được nối tuần tự

Đầu ra của tín hiệu được gửi qua bluetooth có dạng như sau:

```
1 "%lld\t%lld\t%lld\t%d\t%d\t%d\t%d\n"
```

dữ liệu được gửi với mỗi 10ms một lần

```
1 13:38:46.492 -> 3      -1      7      711      516      663      514
2 13:38:46.492 -> 3      -1      7      711      516      663      514
3 13:38:46.492 -> 3      -1      7      711      516      663      514
4 13:38:46.564 -> 3      -1      7      711      516      663      514
5 13:38:46.564 -> 3      -1      7      711      516      663      514
6 13:38:46.564 -> 3      -1      7      711      516      663      514
7 13:38:46.564 -> 3      -1      7      711      516      663      514
8 13:38:46.601 -> 3      -1      7      711      516      663      514
```

Hình 39: Dạng dữ liệu được gửi

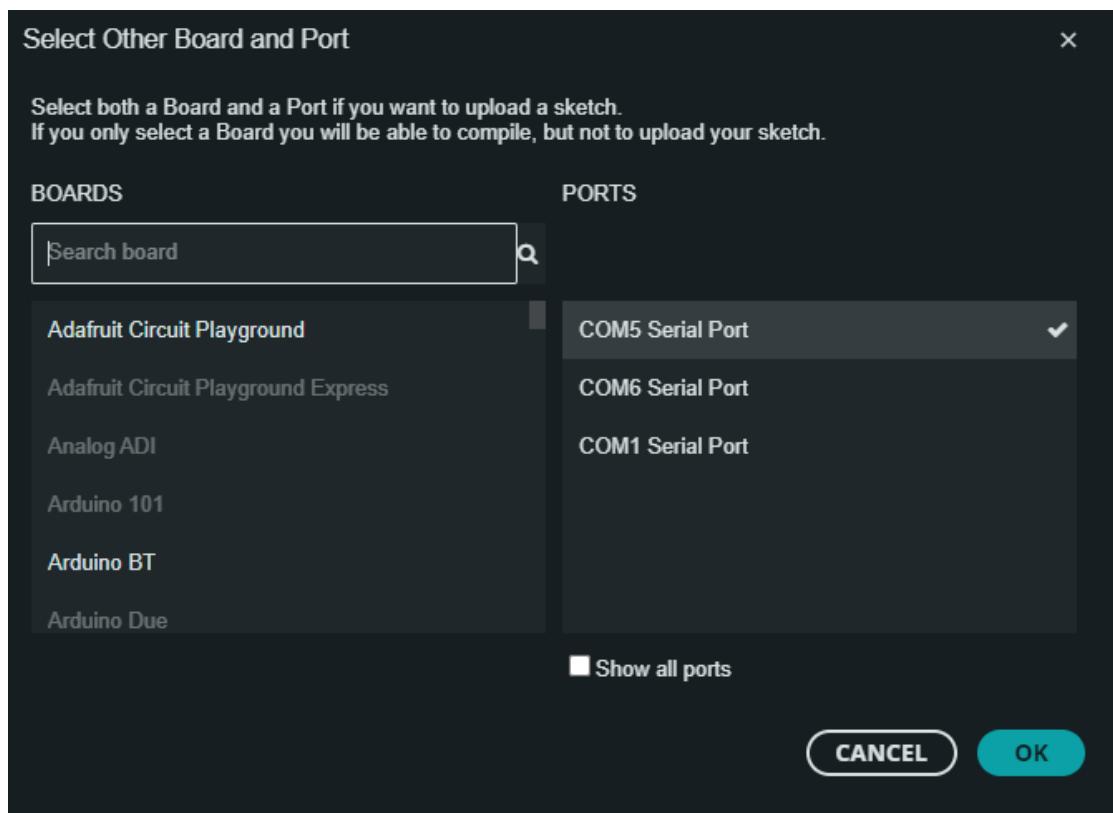
4.2.2 Dữ liệu nhận được từ gateway

```
1 SERIAL_PORT = 'COM5'
2 SERIAL_RATE = 38400
3 label = 14
4 DATA_FILE_NAME = './datatrain1/trainx_'+str(label)+'.txt' #
    Specify the output file name
5 LABEL_FILE_NAME = './datatrain1/trainy_'+str(label)+'.txt' #
    Specify the output file name
6 DATA_TEST_FILE_NAME = './datatrain1/testx_'+str(label)+'.txt' #
    Specify the output file name
7 LABEL_TEST_FILE_NAME = './datatrain1/testy_'+str(label)+'.txt' #
    Specify the output file name
```



```
8 test_counter = 300
9 queue_size = 240
10 windowSlide = 60
11 training_times = 200
12 testing_times = 40
```

- **SERIAL_PORT** là cổng kết nối với thiết bị Sau khi kết nối với bluetooth với arduino, kiểm tra cổng COM đã kết nối bằng cách mở phần mềm arduino



Hình 40: kiểm tra các cổng com đang mở

Ví dụ trong bài luận văn này là cổng 5

- **SERIAL_RATE** ở đây là baudrate khi khởi tạo ở arduino, ở đây module bluetooth HC-05 sử dụng 38400 để đảm bảo thời gian trễ thấp nhất khi một frame dữ liệu được gửi qua
- **label** là nhãn của dữ liệu khi được train, nhãn này được định nghĩa ở file config và readme như sau



Bảng 2: File readme mô tả hành động được thu nhập và gắn nhãn

label 0	hành động nắm tay
label 1	hành động tạm biệt (hành động chậm)
label 2	hành động tạm biệt (hành động chậm)
label 3	chỉ vào bản thân (ngón cái đưa lên)
label 4	không biết (bàn tay xòe thẳng và xoay quanh trục ngón giữa)
label 5	trạng thái nghỉ
label 6	chỉ ra ngoài (chỉ một ngón trỏ đưa lên)
label 7	hành động đi bộ (ngón giữa và ngón trỏ)
label 8	ngón trỏ chỉ xuống đất và xoay vòng
label 9	đùa giỡn (ngón trỏ và ngón giữa co giãn cùng một lúc)
label 10	chỉ tại đây (ngón cái và ngón giữa chỉ xuống, ngón cái đưa lên)
label 11	hành động xin chào (hành động nhanh)
label 12	giới thiệu tên
label 13	thể hiện trạng thái vui vẻ (bàn tay xòe thẳng và xoay vòng quanh cổ tay)
label 14	hành động gấp nhau (ngón trỏ chỉ lên, các ngón còn lại co)

Bảng 3: File config là các đoạn text được chuyển sang giọng nói

Giữ	0
Tạm Biệt	1
Tốt lắm	2
Tôi	3
Không biết	4
-	5
Bạn	6
Đi bộ	7
Xung quanh	8
Dùa thôi	9
Tại đây	10

Continued on next page

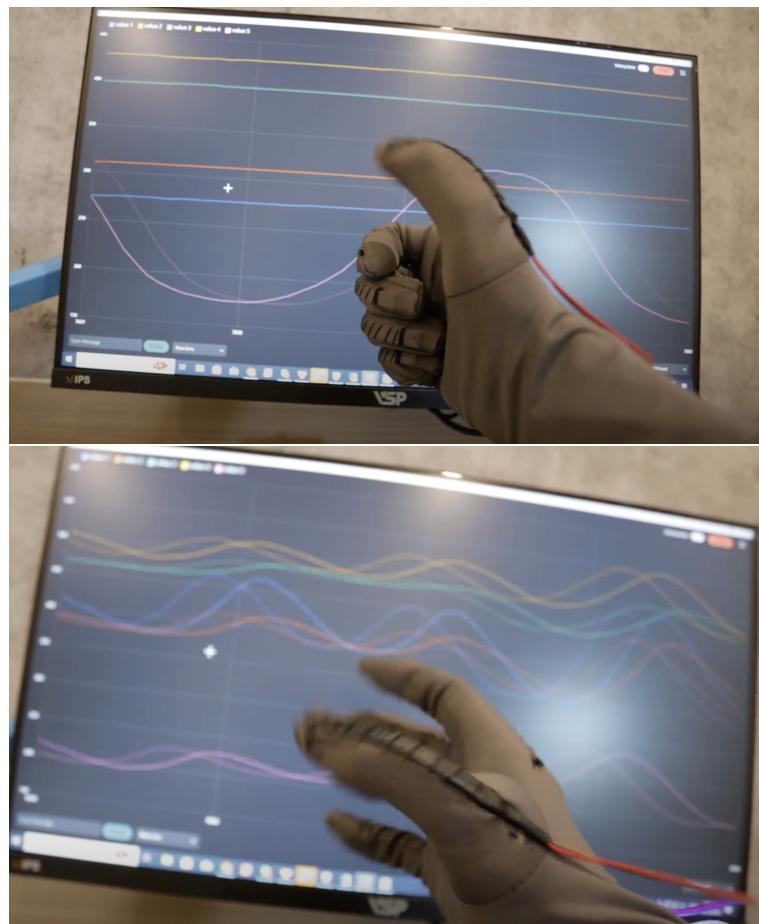


Bảng 3: File config là các đoạn text được chuyển sang giọng nói (Continued)

Xin chào	11
Tên là Chánh (điều chỉnh để phát ra tên như mong muốn)	12
Rất vui	13
Gặp	14

- **DATA_FILE_NAME:** nơi lưu trainx
- **LABEL_FILE_NAME:** nơi lưu trainy
- **DATA_TEST_FILE_NAME:** nơi lưu testx
- **LABEL_TEST_FILE_NAME:** nơi lưu testy
- **test_counter:** số lần test dữ liệu, được định tính trước khi bắt đầu thu thập
- **queue_size:** một frame dữ liệu để gắn nhãn cho frame đó
- **windowSlide:** cửa sổ trượt, một lượng dữ liệu một để thêm vào một frame, trùng lặp với frame trước đó được tính bằng $queue_size - windowSlide$
- **training_times:** số frame dữ liệu được ghi vào file train
- **testing_times:** số frame dữ liệu được ghi vào file test

Tiến hành thu thập dữ liệu với các động tác tay được định nghĩa như ở file readme



Hình 41: Quá trình lấy dữ liệu

Các file data được lưu có kích thước cho một hành động như sau

- Một hàng có 240*7 cột dữ liệu
- Có tổng cộng 200 hàng dữ liệu cho file train và 20 cho trainy



4.3 Định dạng data trước khi đưa vào học máy

```
1 line = next(testx).strip().split()
2     if len(line) >= 1680:
3         for i in range(240):
4             x1.append(int(line[i]))
5             x2.append(int(line[i + 240]))
6             x3.append(int(line[i + 480]))
7             x4.append(int(line[i + 720]))
8             x5.append(int(line[i + 960]))
9             x6.append(int(line[i + 1200]))
10            x7.append(int(line[i + 1440]))
11    else:
12        print("Error test data.")
13    x1 = array(x1)
14    x2 = array(x2)
15    x3 = array(x3)
16    x4 = array(x4)
17    x5 = array(x5)
18    x6 = array(x6)
19    x7 = array(x7)
20    line_dataset = dstack([x1, x2, x3, x4, x5, x6, x7])
21
22    line_dataset = line_dataset.reshape(1, 240, 7)
    testx_data.append(line_dataset)
```

Có hai loại tín hiệu chính trong dữ liệu gốc: gia tốc và flexsensor. Gia tốc có 3 trục dữ liệu, flexsensor có 4 trục dữ liệu. Mỗi loạt dữ liệu được ghi với thời gian là 2.4 giây tương đương với 240 bước thời gian. Các cửa sổ dữ liệu này tương ứng với các cửa sổ đặc trưng đã được tạo ra trước đó. Điều này có nghĩa là một hàng dữ liệu có $(240 * 7)$, tức là 1680 phần tử.

Dữ liệu sau khi được tải từ file text, sau đó được phân chia thành từng frame dữ liệu: Mỗi frame dữ liệu có các tính chất như sau:

- Mỗi cột dữ liệu được ghi 240 lần
- Có tổng cộng 7 cột dữ liệu tương ứng với 7 feature của một frame

Có thể thấy hàm reshape sẽ đảm nhận việc đó



4.4 Xây dựng mô hình

Sau khi train thành công dựa trên bộ dữ liệu, các tham số của mô hình được lưu vào file h5. file này được sử dụng để dự đoán các hành động

5 Các phương pháp cải tiến

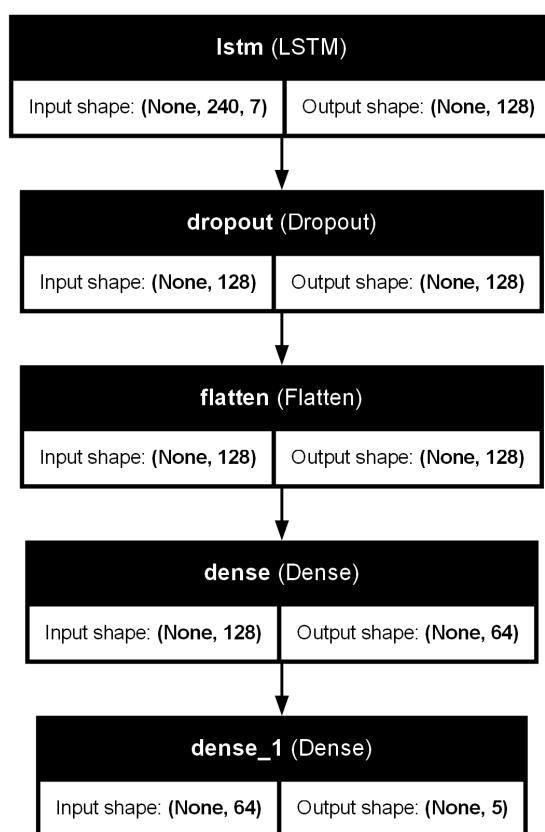
5.1 Sliding Window

5.2 Khắc phục vấn đề mất dữ liệu

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.

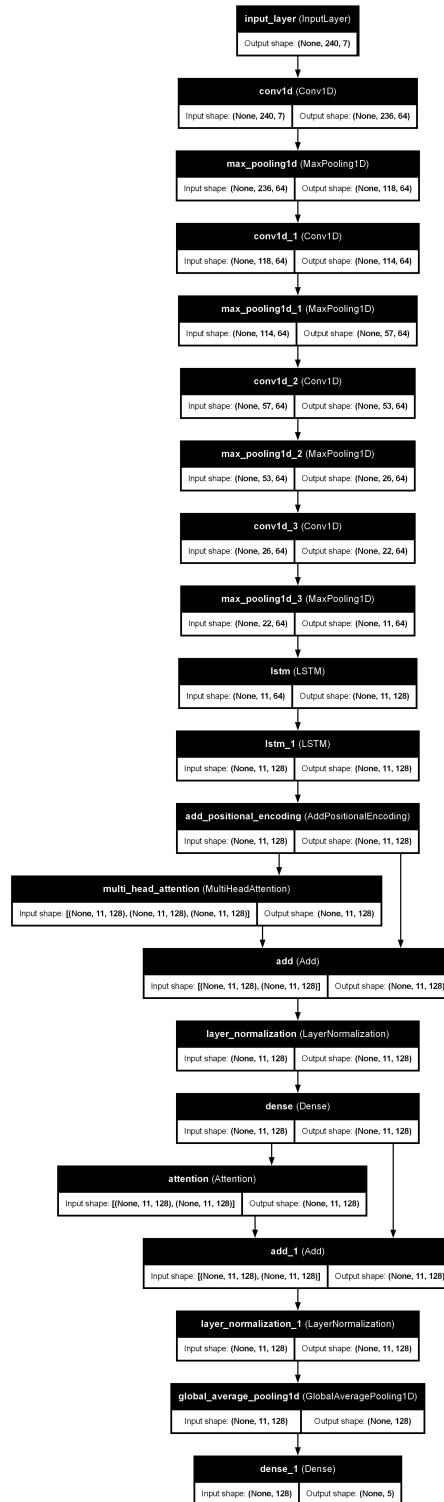


Trong đề tài này, nhóm đề xuất một mô hình cải tiến, dựa trên kiến trúc được giới thiệu trong bài báo "*Self-Attention-Based Deep Convolution LSTM Framework for Sensor-Based Badminton Activity Recognition*" ([DOI: 10.3390/s23208373](https://doi.org/10.3390/s23208373)). Mô hình này kết hợp:

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

Một cách cải thiện hiệu suất của mô hình là sử dụng *Transfer learning*. Transfer learning là một kỹ thuật học máy mà mô hình được huấn luyện trước trên một tập dữ liệu lớn và sau đó được sử dụng để giải quyết một bài toán mới. Nhóm sử dụng một tập dữ liệu có tên là "DU-ASL-DATA-GLOVE-DB" được công bố trên trang web [Figshare](#). Tập dữ liệu này chứa dữ liệu về ngôn ngữ ký hiệu Mỹ (ASL) được thu thập từ cảm biến găng tay thông minh.

- **Tổng quan:** Đây là một tập dữ liệu được thu thập bởi 1 nhóm sinh viên chuyên ngành kỹ thuật sinh học tại Ấn Độ, họ thu thập dữ liệu từ 25 người khác nhau (19 nam và 6 nữ). Mỗi người thực hiện 40 cử chỉ khác nhau (26 cử chỉ trong bảng chữ cái và 14 từ). Dữ liệu được thu thập từ cảm biến với tần số lấy mẫu 100Hz và mỗi cử chỉ có thời lượng 1.5 giây và được thực hiện 10 lần cho mỗi đối tượng.
- **Thông tin phần cứng:** Dữ liệu được thu thập từ 5 cảm biến flex 2.2 inch SEN-10264 và 1 cảm biến IMU MPU6050.
- **Tổ chức dữ liệu:** Dữ liệu được lưu trong 25 thư mục được đánh số từ 001 đến 025, mỗi thư mục chứa dữ liệu của một người. Mỗi thư mục chứa 40 file csv tương ứng với 40 cử chỉ khác nhau. Mỗi file csv chứa dữ liệu của 10 lần thực hiện cử chỉ.
- **Tiền xử lý:** Dữ liệu cần được xử lý để có thể phù hợp với mô hình của nhóm. Mô hình học máy nhóm sử dụng yêu cầu dữ liệu đầu vào phải có timestep là 250 và số chiều là 7. Do đó nhóm cần thực hiện 2 bước tiền xử lý chính:
 - **Thay đổi tập dữ liệu** Vì tín hiệu ở cột thứ 4 được tính bằng tổng tín hiệu của cảm biến từ ngón áp út và ngón út, nhóm cần phải tính lại giá trị của cột này bằng cách tính tổng điện trở của f4 và f5 trong dataset. Nhóm cũng chỉ sử dụng giá trị từ flex sensor và cảm biến góc quay vì thiết bị không có cảm biến gia tốc.

		θ	1	2	3	4	5	6
0	20		58	72	77	-0.068702	-0.015267	0.015267
1	20		58	70.7531	76.3766	-0.0734613	-0.015267	0.0200263
2	20		58	70.7406	75.7531	-0.076336	-0.0171515	0.022901
3	20		58	72.6109	75.1297	-0.076336	-0.0219108	0.022901
4	19.5063		58	73	76.9749	-0.076336	-0.022901	0.022901

Hình 44: Dữ liệu sau khi chỉnh sửa

- **Upscaling dữ liệu:** Dữ liệu được thu thập với tần số lấy mẫu 100Hz với thời gian thực hiện cùi chỉ là 1.5 giây. Điều đó có nghĩa là dữ liệu thu được có số lượng timestep là 150. Để phù hợp với mô hình nhóm sử dụng, nhóm cần phải upscaling dữ liệu lên 250 timestep. Kỹ thuật upscaling được sử dụng là nội suy tuyến tính.
- **Huấn luyện mô hình:** Mô hình được sử dụng là mô hình cải tiến mà nhóm đã giới thiệu ở trên. Mô hình được huấn luyện trên tập dữ liệu ASL. Weight của mô hình được lưu lại và sử dụng cho việc huấn luyện mô hình trên tập dữ liệu của nhóm.

5.4.3 Sinh dữ liệu từ TimeGAN



6 Kết quả thực nghiệm

6.1 Mô hình ban đầu

6.2 Sau khi cải tiến

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 !

6.3

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](#)