**BỘ CÔNG THƯƠNG**

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

**---------------------------------------**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**NGHIÊN CỨU HỌC SÂU HỒI QUY RNN ỨNG DỤNG NHẬN DIỆN HÀNH VI CON NGƯỜI BẰNG MEDIAPIPE POSE VÀ LSTM MODEL**

|  |  |
| --- | --- |
| **GVHD :** | **ThS. Trần Phương Nhung** |
| **Sinh viên thực hiện :** | **Mã sinh viên** |
| Bùi Anh Tuấn | 2020607726 |
| **KHOA :** | **Công nghệ Thông tin** |

**Hà Nội - 2024**

# LỜI CẢM ƠN

*Để đồ án tốt nghiệp được thực hiện thành công,em xin cảm ơn giảng viên hướng dẫn ThS. Trần Phương Nhung, Khoa Công nghệ thông tin, Trường đại học Công nghiệp Hà Nội đã chia sẻ những kiến thức quý báu, hướng dẫn tận tình và có những góp ý trong suốt quá trình thực hiện đề tài để em có thể hoàn thành đồ án một cách tốt nhất.*

*Sau khi nghiên cứu đề tài và kết thúc khóa học, chúng em đã học hỏi và tích lũy được kiến thức và kinh nghiệm từ cô đi trước để hoàn thiện và phát triển bản thân. Bên cạnh đó, đây cũng là cơ hội giúp chúng em nhận ra mình cần hoàn thiện thêm những gì để chuẩn bị cho một hành trình dài phía trước.*

*Trong quá trình hoàn thiện đề tài, em đã cố gắng hết sức để hoàn thành được mục đích đã đề ra. Chúng em rất mong nhận được sự góp ý của các cô để đề tài nghiên cứu được hoàn thiện hơn.*

*Em xin chân thành cảm ơn!*

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc168045647)

[MỤC LỤC 2](#_Toc168045648)

[DANH MỤC THUẬT NGỮ, CÁC TỪ NGỮ VIẾT TẮT VÀ KÍ HIỆU 4](#_Toc168045649)

[DANH MỤC HÌNH ẢNH 5](#_Toc168045650)

[DANH MỤC BIỂU ĐỒ 6](#_Toc168045651)

[MỞ ĐẦU 7](#_Toc168045652)

[1. Tính cấp thiết của nghiên cứu 7](#_Toc168045653)

[2. Mục đích nghiên cứu 7](#_Toc168045654)

[3. Phạm vi nghiên cứu 8](#_Toc168045655)

[4. Đối tượng nghiên cứu 8](#_Toc168045656)

[5. Bố cục đề tài 9](#_Toc168045657)

[6. Phương pháp nghiên cứu 9](#_Toc168045658)

[CHƯƠNG I. CƠ SỞ LÍ THUYẾT 10](#_Toc168045659)

[1.1. Recurrent neural network (RNN) 10](#_Toc168045660)

[1.1.1. Khái niệm 10](#_Toc168045661)

[1.1.2. Dữ liệu dạng sequence 13](#_Toc168045662)

[1.1.3. Phân loại bài toán RNN 16](#_Toc168045663)

[1.1.4. Ứng dụng bài toán RNN 17](#_Toc168045664)

[1.1.5. Mô hình bài toán RNN 17](#_Toc168045665)

[1.1.5.1 Mô hình RNN 17](#_Toc168045666)

[1.1.5.2. Loss function 19](#_Toc168045667)

[1.1.5.3. Backpropagation Through Time (BPTT) 20](#_Toc168045668)

[1.2. Tìm hiểu về Long Short Term Memory 21](#_Toc168045669)

[1.2.1 Giới thiệu chung LSTM 21](#_Toc168045670)

[1.2.2 Mô hình LSTM 24](#_Toc168045671)

[1.2.3. Cơ chế hoạt động của LSTM 26](#_Toc168045672)

[1.2.4 Ứng dụng Long Short Term Memory (LSTM) 28](#_Toc168045673)

[1.2.4.1. Ứng dụng mô hình LSTM 28](#_Toc168045674)

[1.2.4.2. Chuỗi thời gian 28](#_Toc168045675)

[1.2.4.3. Các khái niệm liên quan tới chuỗi time series 29](#_Toc168045676)

[1.3. MediaPipe 30](#_Toc168045677)

[CHƯƠNG 2. ỨNG DỤNG NHẬN DIỆN HÀNH VI 31](#_Toc168045678)

[2.1. Nhận diện hành vi 31](#_Toc168045679)

[2.2. Cơ sở phát triển một ứng dụng nhận diện 33](#_Toc168045680)

[2.3. Ứng dụng vào việc nhận diện trong thực tế 35](#_Toc168045681)

[CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM 40](#_Toc168045682)

[3.1. Tổng quan sơ đồ bài toán 40](#_Toc168045683)

[3.2. Chuẩn bị dữ liệu 40](#_Toc168045684)

[3.3. Quá trình thực hiện 41](#_Toc168045685)

[CHƯƠNG 4: KIỂM THỬ VÀ ĐÁNH GIÁ PHẦN MỀM 55](#_Toc168045686)

[Kiểm thử chức năng của ứng dụng 55](#_Toc168045687)

[KẾT LUẬN VÀ KIẾN NGHỊ 56](#_Toc168045688)

[TÀI LIỆU THAM KHẢO 57](#_Toc168045690)

# DANH MỤC THUẬT NGỮ, CÁC TỪ NGỮ VIẾT TẮT VÀ KÍ HIỆU

LSTM Model : Long short term memory

RNN : Recurrent Neural Network

Seq2seq: Sequence to Sequence Model

FPS : Frame per second

SVM : Support vector machines

HMM : Hidden Markov models

CNN: Convolutional Neural Network

ARIMA: Autoregressive Intergrated Moving Average

NLP: Neuro Linguistic Programming

# DANH MỤC HÌNH ẢNH

[Hình 1.1 Mô hình Neural Network 10](#_Toc168045003)

[Hình 1.2 RNN Computational Graph Many to many 11](#_Toc168045004)

[Hình 1.3 Video bài toán 12](#_Toc168045005)

[Hình 1.4 Seq2seq model 14](#_Toc168045006)

[Hình 1.5 Seq2seq model 14](#_Toc168045007)

[Hình 1.6 Mô hình encoder 15](#_Toc168045008)

[Hình 1.7 Mô hình decoder 15](#_Toc168045009)

[Hình 1.8 Các dạng bài toán RNN 16](#_Toc168045010)

[Hình 1.9 Ví dụ image captioning 16](#_Toc168045011)

[Hình 1.10 Mô hình RNN cho bài toán 18](#_Toc168045012)

[Hình 1.11 Mô hình RNN rút gọn 19](#_Toc168045013)

[Hình 1.12 Loss function 20](#_Toc168045014)

[Hình 1.13 Mô-đun lặp lại trong RNN tiêu chuẩn chứa một lớp duy nhất. 21](#_Toc168045015)

[Hình 1.14 Mô-đun lặp lại trong một LSTM chứa bốn lớp tương tác. 22](#_Toc168045016)

[Hình 1.15 Các kí hiệu 22](#_Toc168045017)

[Hình 1.16 Trạng thái tế bào 23](#_Toc168045018)

[Hình 1.17 Cổng (gate) 23](#_Toc168045019)

[Hình 1.18 Mô hình LSTM 24](#_Toc168045020)

[Hình 1.19 Cell state trong LSTM 26](#_Toc168045021)

[Hình 1.20 Hoạt động bên trong của một cell RNN 26](#_Toc168045022)

[Hình 1.21 Cổng LSTM có thể được coi là bộ lọc 27](#_Toc168045023)

[Hình 3.1 Sơ đồ bài toán 40](#_Toc168045024)

[Hình 3.2 Chuỗi keypoint 41](#_Toc168045025)

[Hình 3.3 Thư mục Deep-Learning 42](#_Toc168045026)

[Hình 3.4 Các thư viện cần thiết 42](#_Toc168045027)

[Hình 3.5: Label LEFT HAND MOVE 52](#_Toc168045028)

[Hình 3.6: Label HAND RIGHT MOV 52](#_Toc168045029)

[Hình 3.7: Label CLAPPING 53](#_Toc168045030)

[Hình 3.8 : Label DO NOTHING 53](#_Toc168045031)

[Hình 3.9: Câu lệnh thực thi việc điều khiển bật tắt Adudio 54](#_Toc168045032)

# DANH MỤC BIỂU ĐỒ

[Biểu đồ 3.1 Train Accuracy 43](#_heading=h.kgcv8k)

[Biểu đồ 3.2 Train loss 43](#_heading=h.34g0dwd)

[Biểu đồ 3.3 Validation Accuracy 44](#_heading=h.1jlao46)

[Biểu đồ 3.4 Validation Loss 44](#_heading=h.43ky6rz)

# MỞ ĐẦU

## Tính cấp thiết của nghiên cứu

Việc nhận diện hành vi con người là cực kỳ quan trọng trong nhiều lĩnh vực, ví dụ như y tế, an ninh, giám sát và điều khiển robot. Sử dụng phương pháp Mediapipe Pose và LSTM Model để nhận diện hành vi con người là cách tiếp cận hiệu quả và tiên tiến. Các lợi ích của việc nhận diện hành vi con người bằng Mediapipe Pose và LSTM Model bao gồm:

* **Tăng tính an toàn cho con người:** Việc nhận diện hành vi con người giúp ngăn chặn các tai nạn và sự cố bất ngờ, đặc biệt là trong các môi trường nguy hiểm như công trường xây dựng, nhà máy và tàu thủy.
* **Tăng hiệu quả trong các ứng dụng y tế:** Việc nhận diện hành vi con người có thể được sử dụng để theo dõi và chẩn đoán các bệnh nhân trong các chương trình phục hồi chức năng.
* **Tăng sự tiện lợi trong giám sát và kiểm soát:** Việc nhận diện hành vi con người có thể được sử dụng để hiểu và kiểm soát các hoạt động của con người trong các hệ thống giám sát.

Vì vậy, việc nhận diện hành vi con người bằng Mediapipe Pose và LSTM Model là một cách tiếp cận quan trọng và cấp thiết để cải thiện tính an toàn, hiệu quả và tiện ích cho các ứng dụng thực tế.

## Mục đích nghiên cứu

Đê tài nhằm nghiên cứu nhận diện hành vi con người bằng Mediapipe Pose và LSTM Model là phát triển một giải pháp phần mềm có khả năng nhận diện và phân tích hành vi con người thông qua hình ảnh hoặc video của họ. Cụ thể là tạo ra một giải pháp phần mềm nhận diện và phân tích hành vi con người để tăng tính an toàn và hiệu quả trong nhiều lĩnh vực, như giám sát an ninh, quản lý sức khỏe và điều khiển Robot. Đồng thời, giải pháp này cũng có thể đóng góp tích cực vào phát triển các ứng dụng nhận diện hành vi con người trong các lĩnh vực khác nhau. Nghiên cứu sử dụng phương pháp Mediapipe Pose và LSTM Model để đạt được mục đích nhận diện và phân tích hành vi con người vì đây là phương pháp tiên tiến, chính xác và hiệu quả nhất hiện nay trong việc phân tích hành vi con người thông qua hình ảnh hoặc video. Vậy, mục đích của nghiên cứu nhận diện hành vi con người bằng Mediapipe Pose và LSTM Model là phát triển một giải pháp phần mềm tiên tiến và chính xác cho việc nhận diện và phân tích hành vi con người, từ đó đóng góp tích cực vào sự phát triển của nhiều lĩnh vực khác nhau.

## Phạm vi nghiên cứu

Phạm vi nghiên cứu của nhận diện hành vi con người bằng Mediapipe Pose và LSTM Model sẽ tập trung vào các mục tiêu chính sau:

* Xây dựng một giải pháp phần mềm tự động nhận diện và phân tích hành vi con người từ hình ảnh hoặc video.
* Sử dụng phương pháp Mediapipe Pose để trích xuất thông tin về các bộ phận cơ thể của con người.
* Sử dụng mạng neural hồi quy LSTM để phân tích các chuỗi thời gian để nhận diện và phân tích các hành vi con người từ dữ liệu trích xuất.
* Đánh giá hiệu suất của giải pháp phần mềm trên bộ dữ liệu thử nghiệm và so sánh với một số giải pháp khác.

Trong phạm vi nghiên cứu này, sẽ xây dựng một giải pháp phần mềm sáng tạo để phát hiện và phân tích hành vi con người, từ đó có thể ứng dụng trong nhiều lĩnh vực khác nhau. Tuy nhiên, giải pháp này sẽ tập trung vào giám sát an ninh, giám sát sức khỏe, các ứng dụng điều khiển robot, và tạo điều kiện thuận lợi hơn cho việc giám sát và kiểm soát các hoạt động của con người trong các hệ thống giám sát nhà máy và công trình xây dựng.

## Đối tượng nghiên cứu

Đối tượng nghiên cứu của nhận diện hành vi con người bằng Mediapipe Pose và LSTM Model là các hoạt động của con người được thực hiện trong một số môi trường nhất định, ví dụ như trong các bài tập thể dục, đi bộ, chạy bộ, nhảy múa, giao tiếp và các hoạt động tương tác khác trong đời sống thường ngày. Nghiên cứu sẽ tập trung phát triển giải pháp phần mềm để nhận diện và phân tích các hoạt động con người thông qua hình ảnh hoặc video, và sử dụng nó cho nhiều mục đích, ví dụ như giám sát an ninh, quản lý sức khỏe, điều khiển robot hoặc giám sát và kiểm soát các hoạt động trong các hệ thống giám sát nhà máy và công trình xây dựng. Đối tượng nghiên cứu cụ thể là các bộ dữ liệu với nhiều người thực hiện các hoạt động khác nhau, và có thể đại diện cho các hoạt động trong đời sống thực tế. Đối tượng nghiên cứu cũng bao gồm các thành phần của giải pháp phần mềm, bao gồm Mediapipe Pose và mạng neural hồi quy LSTM để phân tích các chuỗi thời gian.

## Bố cục đề tài

**Chương 1:** Trình bày lý thuyết tổng quan về RNN, LSTM và Mediapipe Pose. Bên cạnh đó cũng giới thiệu sợ lược về sự ra đời, cũng như lịch sử phát triển của phương pháp.

**Chương 2.** Ứng dụng nhận diện hành vi con người bằng Mediapipe Pose và LSTM Model

**Chương 3.** Kết quả thử nghiệm

## Phương pháp nghiên cứu

***Phương pháp nghiên cứu tài liệu***

* Tra cứu tài liệu
* Tìm hiểu thống kê
* Chọn lọc

***Phương pháp thực nghiệm***

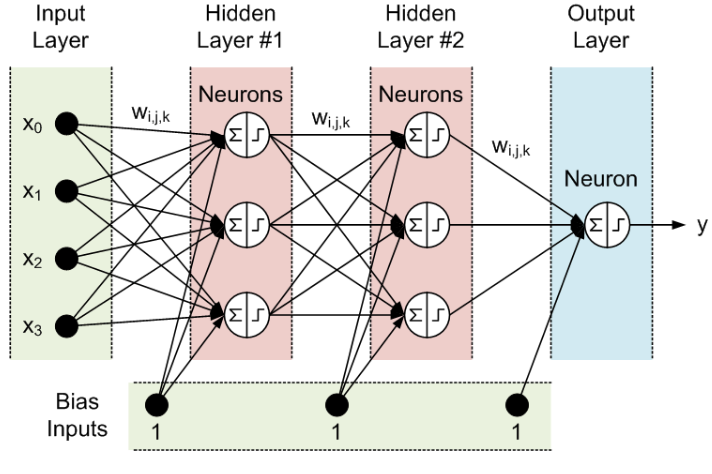
Tiến hành tìm hiểu, thực hiện chương trình để đánh giá nghiên cứu đề xuất

# CHƯƠNG I. CƠ SỞ LÍ THUYẾT

## 1.1. Recurrent neural network (RNN)

### 1.1.1. Khái niệm

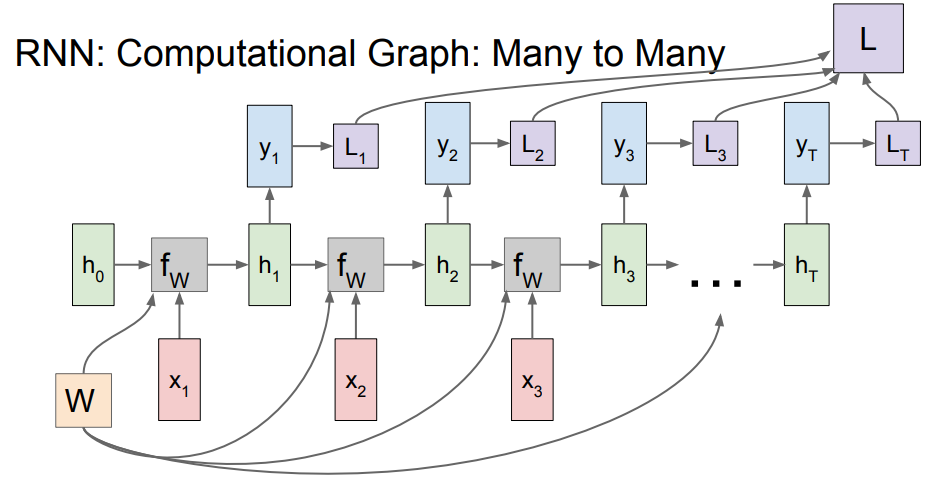
Để có thể hiểu rõ về RNN, trước tiên chúng ta cùng nhìn lại mô hình Neural Network dưới đây:



Hình 1.1 Mô hình Neural Network

Như đã biết thì Neural Network bao gồm 3 phần chính là Input layer, Hidden layer và Output layer, ta có thể thấy là đầu vào và đầu ra của mạng neuron này là độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, ... vì những dự đoán tiếp theo như từ tiếp theo phụ thuộc vào vị trí của nó trong câu và những từ đằng trước nó.

Và như vậy RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin từ từ những bước tính toán xử lý trước để dựa vào nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại. Dưới đây là mô hình mạng RNN:



Hình 1.2 RNN Computational Graph Many to many

Nếu như mạng Neural Network chỉ là input layer x đi qua hidden layer h và cho ra output layer y với full connected giữa các layer thì trong RNN, các input xt sẽ được kết hợp với hidden layer ht−1 bằng hàm fW để tính toán ra hidden layer ht hiện tại và output yt sẽ được tính ra từ ht, W là tập các trọng số và nó được ở tất cả các cụm, các L1,L2,...,Lt là các hàm mất mát sẽ được giải thích sau. Như vậy kết quả từ các quá trình tính toán trước đã được "nhớ" bằng cách kết hợp thêm ht−1 tính ra ht để tăng độ chính xác cho những dự đoán ở hiện tại. Cụ thể quá trình tính toán được viết dưới dạng toán như sau:

Ht=fW(ht−1,xt)

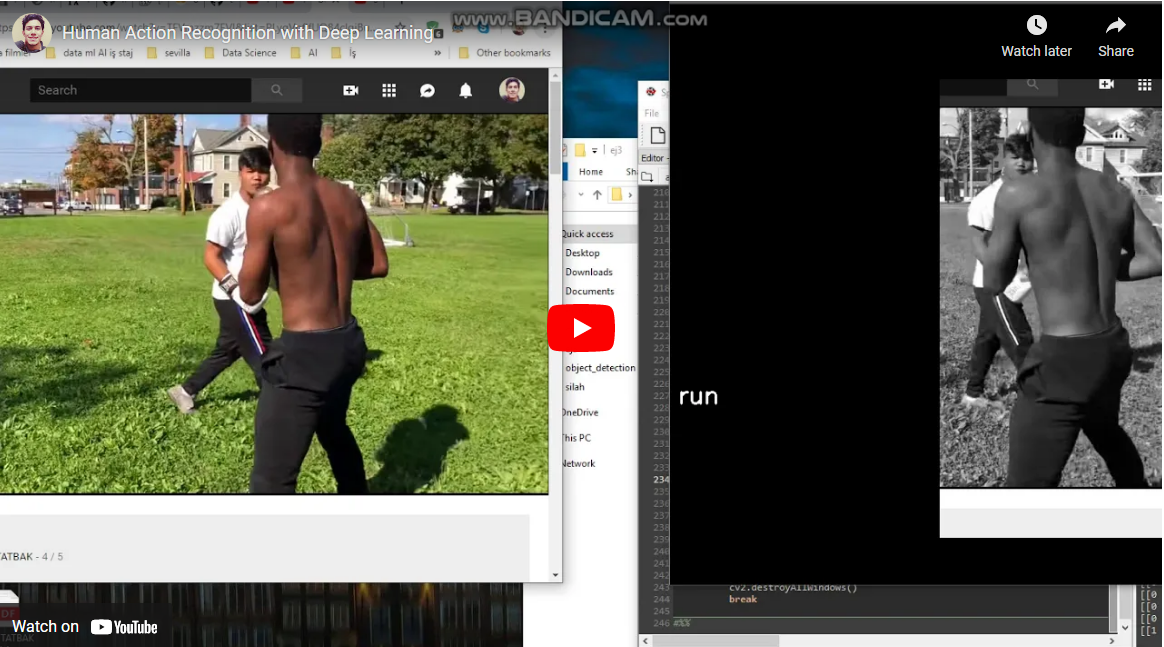
Hàm fW chúng ta sẽ xử dụng hàm tanh, công thức trên sẽ trở thành:

Ht=tanh(Whhht−1+Wxhxt)

Yt=Whyht

Đến đây có 3 thứ mới xuất hiện: Wxh,Whh,Why. Đối với mạng NN chỉ sử dụng một ma trận trọng số W duy nhất thì với RNN nó sử dụng 3 ma trận trọng số cho 2 quá trình tính toán:Whh kết hợp với "bộ nhớ trước" ht−1 và Wxh kết hợp với xt để tính ra "bộ nhớ của bước hiện tại"ht từ đó kết hợp với Why để tính ra yt.

Xét bài toán: Cần phân loại hành động của người trong video, input là video 30s, output là phân loại hành động, ví dụ: đứng, ngồi, chạy, đánh nhau, bắn súng,...



Hình 1.3 Video bài toán

*Link: https://www.youtube.com/watch?v=BmoVg3BCwF0*

Khi xử lý video ta hay gặp khái niệm FPS (frame per second) tức là bao nhiêu frame (ảnh) mỗi giây. Ví dụ 1 FPS với video 30s tức là lấy ra từ video 30 ảnh, mỗi giây một ảnh để xử lý.

Ta dùng 1 FPS cho video input ở bài toán trên, tức là lấy ra 30 ảnh từ video, ảnh 1 ở giây 1, ảnh 2 ở giây 2,... ảnh 30 ở giây 30. Bây giờ input là 30 ảnh: ảnh 1, ảnh 2,... ảnh 30 và output là phân loại hành động.

Nhận xét:

* Các ảnh có thứ tự: ảnh 1 xảy ra trước ảnh 2, ảnh 2 xảy ra trước ảnh 3,... Nếu ta đảo lộn các ảnh thì có thể thay đổi nội dung của video. Ví dụ: nội dung video là cảnh bắn nhau, thứ tự đúng là A bắn trúng người B và B chết, nếu ta đảo thứ tự ảnh thành người B chết xong A mới bắn thì rõ ràng bây giờ A không phải là kẻ giết người => nội dung video bị thay đổi.
* Ta có thể dùng CNN để phân loại 1 ảnh trong 30 ảnh trên, nhưng rõ ràng là 1 ảnh không thể mô tả được nội dung của cả video. Ví dụ: Cảnh người cướp điện thoại, nếu ta chỉ dùng 1 ảnh là người đấy cầm điện thoại lúc cướp xong thì ta không thể biết được cả hành động cướp.

=> Cần một mô hình mới có thể giải quyết được bài toán với input là sequence (chuỗi ảnh 1->30) => Như vậy RNN ra đời.

### 1.1.2. Dữ liệu dạng sequence

Dữ liệu có thứ tự như các ảnh tách từ video ở trên được gọi là sequence, time-series data.

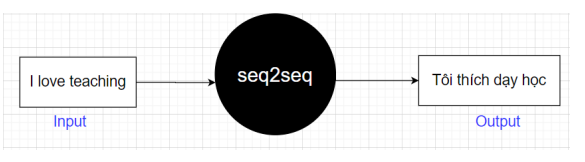
Trong bài toán dự đoán đột quỵ tim cho bệnh nhân bằng các dữ liệu tim mạch khám trước đó. Input là dữ liệu của những lần khám trước đó, ví dụ i1 là lần khám tháng 1, i2 là lần khám tháng 2, ... i8 là lần khám tháng 8. (i1, i2, ... i8) được gọi là sequence data. RNN sẽ học từ input và dự đoán xem bệnh nhân có bị đột quy tim hay không.

Ví dụ khác là trong bài toán dịch tự động với input là 1 câu, ví dụ "tôi yêu Việt Nam" thì vị trí các từ và sự xắp xếp cực kì quan trọng đến nghĩa của câu và dữ liệu input các từ [’tôi’, ’yêu’, ’việt’, ’nam’] được gọi là sequence data. Trong bài toán xử lý ngôn ngữ (NLP) thì không thể xử lý cả câu được và người ta tách ra từng từ (chữ) làm input, giống như trong video người ta tách ra các ảnh (frame) làm input.

**Mô hình Seq2seq**

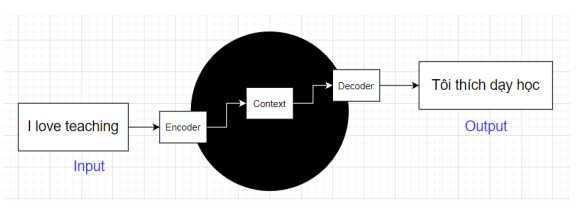
Sequence to Sequence Model (Seq2seq) là một mô hình Deep Learning với mục đích tạo ra một output sequence từ một input sequence mà độ dài của 2 sequences này có thể khác nhau. Seq2seq được giới thiệu bởi nhóm nghiên cứu của Google vào năm 2014 trong bài báo [*Sequence to Sequence with Neural Networks*](https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks). Mặc dù mục đích ban đầu của Model này là để áp dụng trong Machine Translation, tuy nhiên hiện nay Seq2seq cũng được áp dụng nhiều trong các hệ thống khác như Speech recognition, Text summarization, Image captioning,…

Input của mô hình seq2seq là một câu tiếng anh và output là câu dịch tiếng việt tương ứng, độ dài hai câu này có thể khác nhau. Ví dụ: input: I love teaching -> output: Tôi thích dạy học, input 1 câu 3 từ, output 1 câu 4 từ.



Hình 1.4 Seq2seq model

Seq2seq gồm 2 phần chính là Encoder và Decoder. Cả hai thành phần này đều được hình thành từ các mạng Neural Networks, trong đó Encoder có nhiệm vụ chuyển đổi dữ liệu đầu vào (input sequence) thành một representation với lower dimension còn Decoder có nhiệm vụ tạo ra output sequence từ representation của input sequence được tạo ra ở phần Encoder.

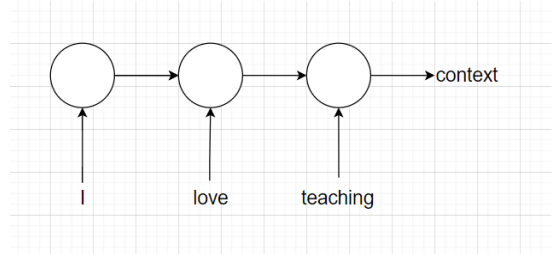


Hình 1.5 Seq2seq model

Encoder nhận input là câu tiếng anh và output ra context vector, còn decoder nhận input là context vector và output ra câu tiếng việt tương ứng. Phần encoder sử dụng mô hình RNN (nói là mô hình RNN nhưng có thể là các mô hình cải tiến như GRU, LSTM) và context vector được dùng là hidden states ở node cuối cùng. Phần decoder cũng là một mô hình RNN với s0 chính là context vector rồi dần dần sinh ra các từ ở câu dịch.

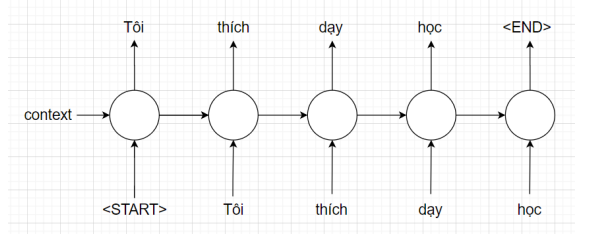
Phần decoder này giống với bài toán image captioning. Ở bài image captioning mình cũng cho ảnh qua pre-trained model để lấy được embedding vector, sau đó cho embedding vector làm s0 của mô hình RNN rồi sinh ta caption tương ứng với ảnh.

Bài trước mình đã biết model RNN chỉ nhận input dạng vector nên dữ liệu ảnh (từ) sẽ được encode về dạng vector trước khi cho vào model.



Hình 1.6 Mô hình encoder

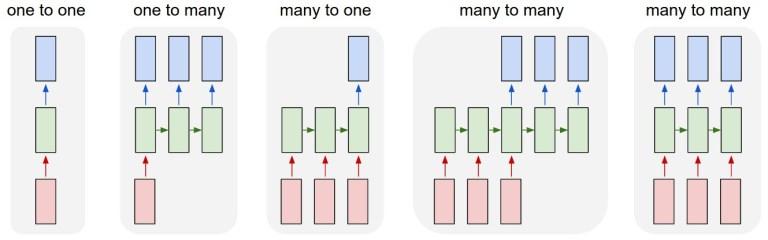
Các từ trong câu tiếng anh sẽ được embedding thành vector và cho vào mô hình RNN, hidden state ở node cuối cùng sẽ được dùng làm context vector. Về mặt lý thuyết thì context vector sẽ mang đủ thông tin của câu tiếng anh cần dịch và sẽ được làm input cho decoder.



Hình 1.7 Mô hình decoder

2 tag <START> và <END> được thêm vào câu output để chỉ từ bắt đầu và kết thúc của câu dịch. Mô hình decoder nhận input là context vector. Ở node đầu tiên context vector và tag <START> sẽ output ra chữ đầu tiên trong câu dịch, rồi tiếp tục mô hình sinh chữ tiếp theo cho đến khi gặp tag <END> hoặc đến max\_length của câu output thì dừng lại.

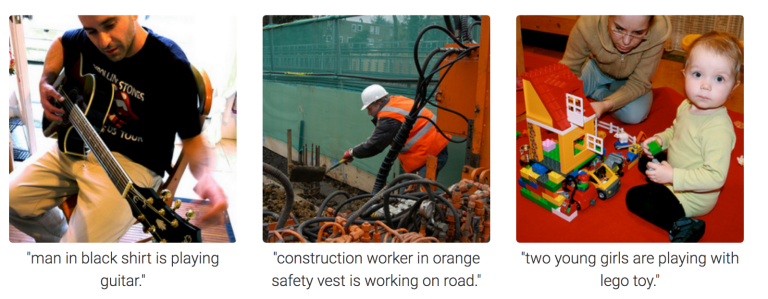
### 1.1.3. Phân loại bài toán RNN



Hình 1.8 Các dạng bài toán RNN

Các dạng bài toán RNN

* One to one: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với bài toán phân loại ảnh MNIST input là ảnh và output là ảnh đấy là số nào.
* One to many: bài toán có 1 input nhưng nhiều output, ví dụ với bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.



Hình 1.9 Ví dụ image captioning

* Many to one: bài toán có nhiều input nhưng chỉ có 1 output, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, ouptut là hành động trong video
* Many to many: bài toán có nhiều input và nhiều output, ví dụ bài toán dịch từ tiếng anh sang tiếng việt, input là 1 câu gồm nhiều chữ: "I love Vietnam" và output cũng là 1 câu gồm nhiều chữ "Tôi yêu Việt Nam". Để ý là độ dài sequence của input và output có thể khác nhau.

### 1.1.4. Ứng dụng bài toán RNN

RNN có nhiều công dụng, đặc biệt là khi dùng để dự đoán tương lai. Trong ngành tài chính, RNN có thể hữu ích trong việc dự đoán giá cổ phiếu hoặc dấu hiệu của xu hướng thị trường chứng khoán (tức là tích cực hoặc tiêu cực).

RNN rất hữu ích cho ô tô tự lái vì nó có thể tránh tai nạn ô tô bằng cách đoán trước quỹ đạo của xe.

RNN được sử dụng rộng rãi trong phân tích văn bản, speech to text: chuyển giọng nói sang text, machine translation: Bài toán dịch tự động giữa các ngôn ngữ, chú thích hình ảnh, video recognition: nhận diện hành động trong video, heart attack: dự đoán đột quỵ tim, phân tích tình cảm và dịch máy. Ví dụ, người ta có thể sử dụng (sentiment classification: phân loại số sao cho các bình luận) đánh giá phim để hiểu cảm giác mà khán giả cảm nhận sau khi xem phim. Tự động hóa nhiệm vụ này rất hữu ích khi công ty điện ảnh không có đủ thời gian để xem xét, dán nhãn, củng cố và phân tích các bài đánh giá. Máy có thể thực hiện công việc với mức độ chính xác cao hơn.

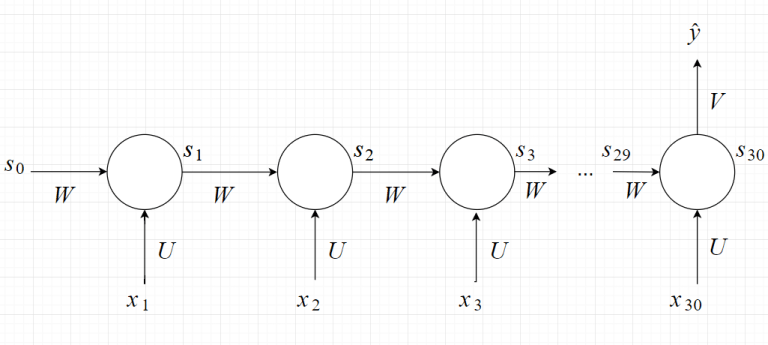
### 1.1.5. Mô hình bài toán RNN

#### 1.1.5.1 Mô hình RNN

Bài toán: Nhận diện hành động trong video 30s. Đây là dạng bài toán many to one trong RNN, tức nhiều input và 1 output.

Input ta sẽ tách video thành 30 ảnh ở mỗi giây. Các ảnh sẽ được cho qua model CNN để lấy ra các feature ([feature extraction](https://nttuan8.com/bai-9-transfer-learning-va-data-augmentation/#Feature_extractor)) thành các vector có kích thước n\*1. Vector tương ứng với ảnh ở giây thứ i là xi

Output là vector có kích thước d\*1, softmax function được sử dụng như trong [bài phân loại ảnh](https://nttuan8.com/bai-7-gioi-thieu-keras-va-bai-toan-phan-loai-anh/#Xay_dung_model).

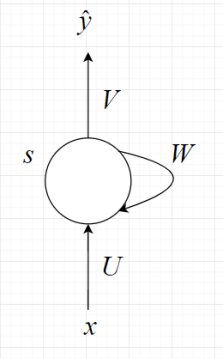


Hình 1.10 Mô hình RNN cho bài toán

Ta có:

* Mô hình có 30 input và 1 output, các input được cho vào model đúng với thứ tự ảnh trong video x1, x2, ..., *x*30​.
* Mỗi hình tròn được gọi là 1 state, state t có input là xt và st-1​(output của state trước); output là st = f(U\*xt + W\*st-1)*st*​=*f*(*U*∗*xt*​+*W*∗*st*−1​). f là activation function thường là tanh hoặc [ReLU](https://nttuan8.com/bai-10-cac-ky-thuat-co-ban-trong-deep-learning/#Mot_so_activation_thong_dung).
* Có thể thấy st mang cả thông tin từ state trước ( st-1​) và input của state hiện tại => st giống như memory nhớ các đặc điểm của các input từ x1​ đến xt
* S0​ được thêm vào chỉ cho chuẩn công thức nên thường được gán bằng 0 hoặc giá trị ngẫu nhiên. Có thể hiểu là ban đầu chưa có dữ liệu gì để học thì memory rỗng.
* Do ta chỉ có 1 output, nên sẽ được đặt ở state cuối cùng, khi đó s\_{30}*s*30​ học được thông tin từ tất cả các input y^= g(V ∗*s*30​). g là activation function, trong bài này là bài toán phân loại nên sẽ dùng softmax.

Ta thấy là ở mỗi state các hệ số W, U là giống nhau nên model có thể được viết lại thành:



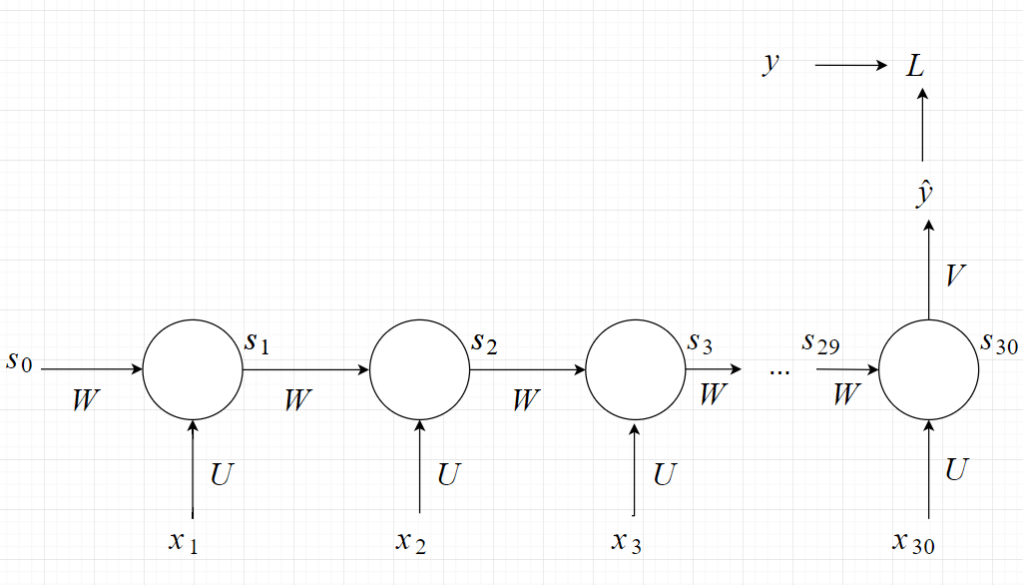
Hình 1.11 Mô hình RNN rút gọn

Tóm lại:

* xi là vector có kích thước n\*1, si là vector có kích thước m\*1, yi là vector có kích thước d\*1. U là ma trận có kích thước m\*n, W là ma trận có kích thước m\*m và V là ma trận có kích thước d\*m.
* s0 = 0, st = f(U ∗ xt +W ∗ st−1) với t >= 1
* y = g (V ∗ s30)

#### 1.1.5.2. Loss function

Loss function của cả mô hình bằng tổng loss của mỗi output, tuy nhiên ở mô hình trên chỉ có 1 output và là bài toán phân loại nên [categorical cross entropy loss](https://nttuan8.com/bai-7-gioi-thieu-keras-va-bai-toan-phan-loai-anh/#Loss_function) sẽ được sử dụng.



Hình 1.12 Loss function

#### 1.1.5.3. Backpropagation Through Time (BPTT)

Có 3 tham số ta cần phải tìm là W, U, V. Để thực hiện gradient descent, ta cần tính: , , .

Tính đạo hàm với V thì khá đơn giản: = \*

Tuy nhiên với U, W thì lại khác

= \* \*

Do s30 = f(W ∗ s29 +U ∗ x30) có s29 phụ thuộc vào W. Nên áp dụng công thức hồi cấp 3 bạn học: (f(x) ∗ g(x))’ = f’(x) ∗ g(x) + f(x) ∗ g‘(x) . Ta có:

= \* \* , trong đólà đạo hàm của s30 với W khi coi s29 là constant với W.

Tương tự trong biểu thức s29 có s28 phụ thuộc vào W, s28 có s27 phụ thuộc vào W ... nên áp dụng công thức trên và chain rule:

,

trong đó và là đạo hàm của si với W khi coi si−1 là constant với W.

Nhìn vào công thức tính đạo hàm của L với W ở trên ta có thể thấy hiện tượng vanishing gradient ở các state đầu nên ta cần mô hình tốt hơn để giảm hiện tượng vaninshing gradient => Long Short Term memory (LSTM) ra đời.

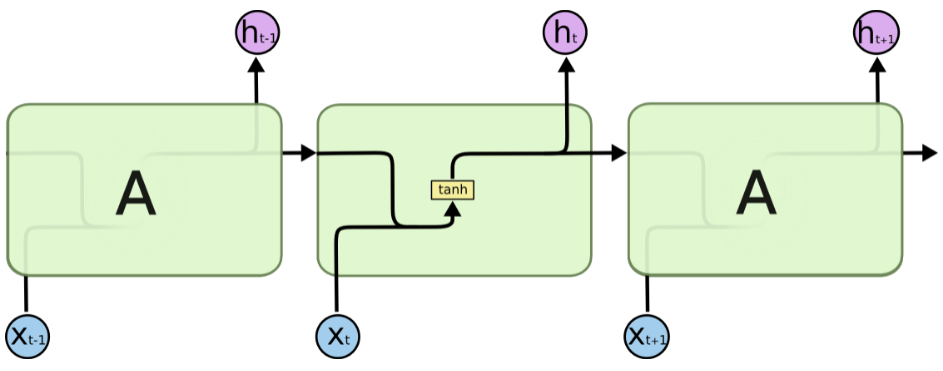
## 1.2. Tìm hiểu về Long Short Term Memory

### 1.2.1 Giới thiệu chung LSTM

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 (1997)*](http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf), 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.

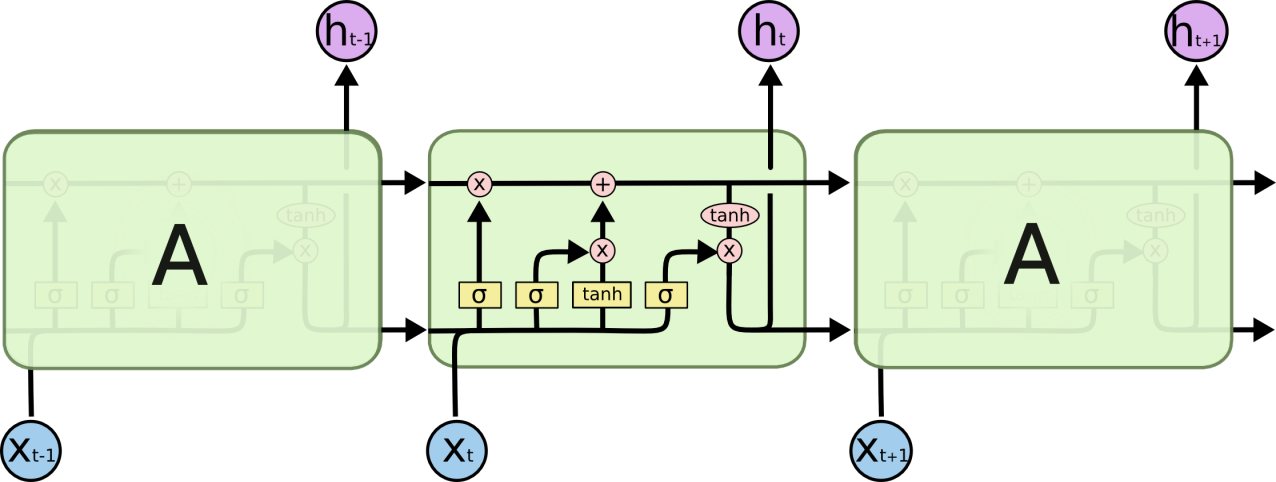
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (Long - Term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

Mọi 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-dun này có cấu trúc rất đơn giản, thường là một tầng tanh.



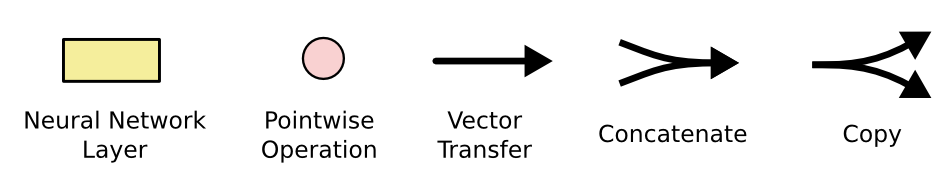
Hình 1.13 Mô-đun lặp lại trong RNN tiêu chuẩn chứa một lớp duy nhất.

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 1.14 Mô-đun lặp lại trong một LSTM chứa bốn lớp tương tác.

Giờ thì đừng hoang mang về chi tiết bên trong chúng ngay, chúng ta sẽ khám phá chúng chi tiết chúng ở bước sau. Điều bạn cần làm bây giờ là làm hãy làm quen với các kí hiệu mà ta sẽ sử dụng ở dưới đây:

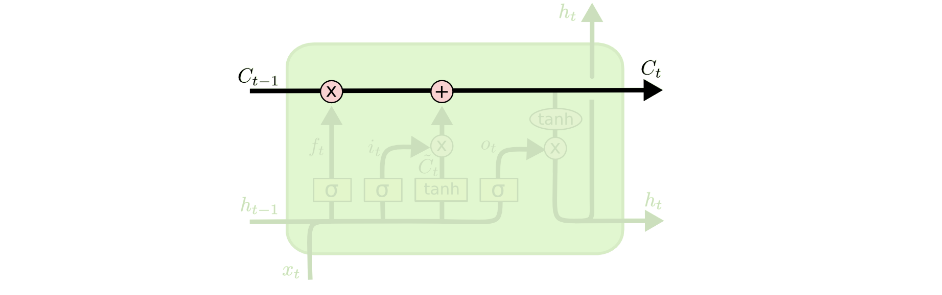


Hình 1.15 Các kí hiệu

Ở sơ đồ trên, mỗi một đường mang một véc-tơ từ đầu ra của một nút tới đầu vào của một nút khác. Các hình trong màu hồng biểu diễn các phép toán như phép cộng véc-tơ chẳng hạn, còn các ô màu vàng được sử dụng để học trong các từng mạng nơ-ron. Các đường hợp nhau kí hiệu việc kết hợp, còn các đường rẽ nhánh ám chỉ nội dung của nó được sao chép và chuyển tới các nơi khác nhau.

Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.

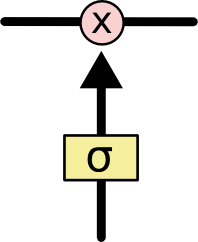
Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.



Hình 1.16 Trạng thái tế bào

LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo,chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate).

Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.



Hình 1.17 Cổng (gate)

Tầng sigmoid sẽ cho đầu ra là một số trong khoản [0, 1][0,1], mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 00 thì có nghĩa là không cho thông tin nào qua cả, còn khi là 11 thì có nghĩa là cho tất cả các thông tin đi qua nó.

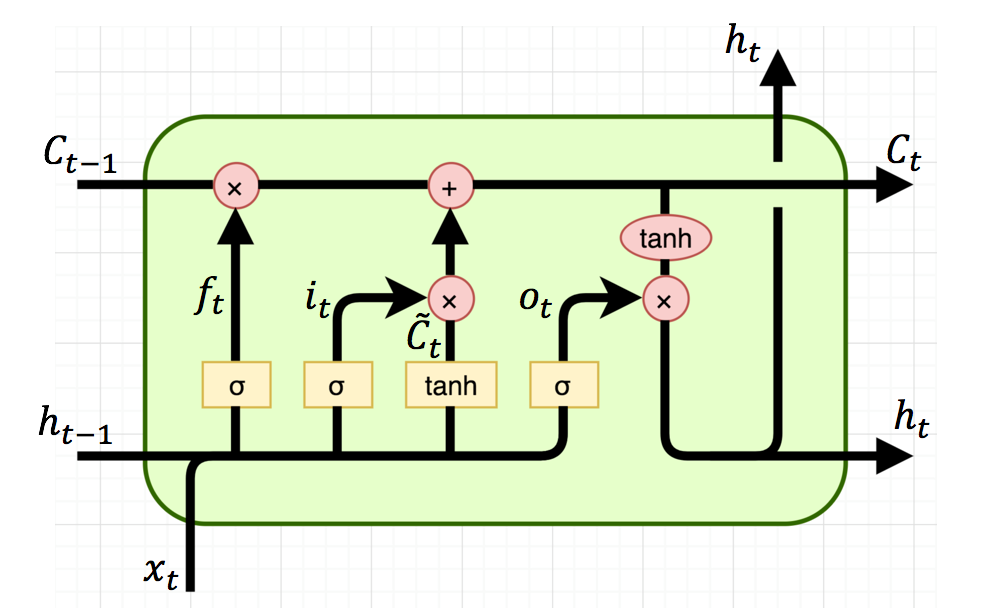
Một LSTM gồm có 3 cổng như vậy để duy trì và điều hành trạng thái của tế bào.

### 1.2.2 Mô hình LSTM

Ở state thứ t của mô hình LSTM:

* Output: *ct, ht*, ta gọi c là cell state, h là hidden state.
* Input: *ct−*1*, ht−*1*, xt*. Trong đó *xt* là input ở state thứ t của model.

*ct−*1*, ht−*1 là output của layer trước. h đóng vai trò khá giống như s ở RNN, trong khi c là điểm mới của LSTM.



Hình 1.18 Mô hình LSTM

Cách đọc biểu đồ trên:

* Bạn nhìn thấy kí hiệu *σ*, tanh ý là bước đấy dùng sigma, tanh activation function.
* Phép nhân ở đây là element-wise multiplication
* phép cộng là cộng ma trận.
* *ft,it,ot* tương ứng với forget gate, input gate và output gate.

Forget gate: *ft* = *σ*(*Uf ∗ xt* +*Wf ∗ ht−*1 +*bf*)

Forget gate quyết định thông tin nào từ long-term memory nên được lưu giữ hay loại bỏ. Điều này được thực hiện bằng cách nhân long-term memory sắp tới với một forget vectorđược tạo ra bởi đầu vào hiện tại và short-term memory sắp tới.

Input gate: *it* = *σ*(*Ui ∗ xt* +*Wi ∗ ht−*1 +*bi*)

Input gate quyết định thông tin mới nào sẽ được lưu trữ trong long-term memory. Nó chỉ hoạt động với thông tin từ đầu vào hiện tại và short-term memory từ trước đó. Vì vậy, cần phải lọc ra thông tin hữu ích và thông tin không hữu ích từ những biến đó.

Output gate: *ot* = *σ*(*Uo ∗ xt* +*Wo ∗ ht−*1 +*bo*)

Output gate sẽ lấy đầu vào hiện tại,short-term memory trước đó và long-term memory mới được tính toán để tạo ranew short-term memory/hidden state (bộ nhớ ngắn hạn mới/trạng thái ẩn) và được chuyển tới tế bào (cell) ở bước tiếp theo. Đầu ra hiện tại cũng có thể được rút ra từ trạng thái ẩn này.

**Nhận xét**: 0 *< ft,it,ot <* 1; *bf ,bi,bo* là các hệ số bias; hệ số W, U giống như trong bài RNN.

*c*˜*t* = tanh(*Uc ∗ xt* +*Wc ∗ ht−*1 +*bc*), bước này giống hệt như tính *st* trong RNN.

*ct* = *ft ∗ ct−*1 + *it ∗ c*˜*t*, forget gatequyết định xem cần lấy bao nhiêu từ cell state trước và inputgate sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước.

*ht* = *ot ∗tanh*(*ct*), output gatequyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state. Ngoài ra *ht* cũng được dùng để tính ra output *yt* cho state t.

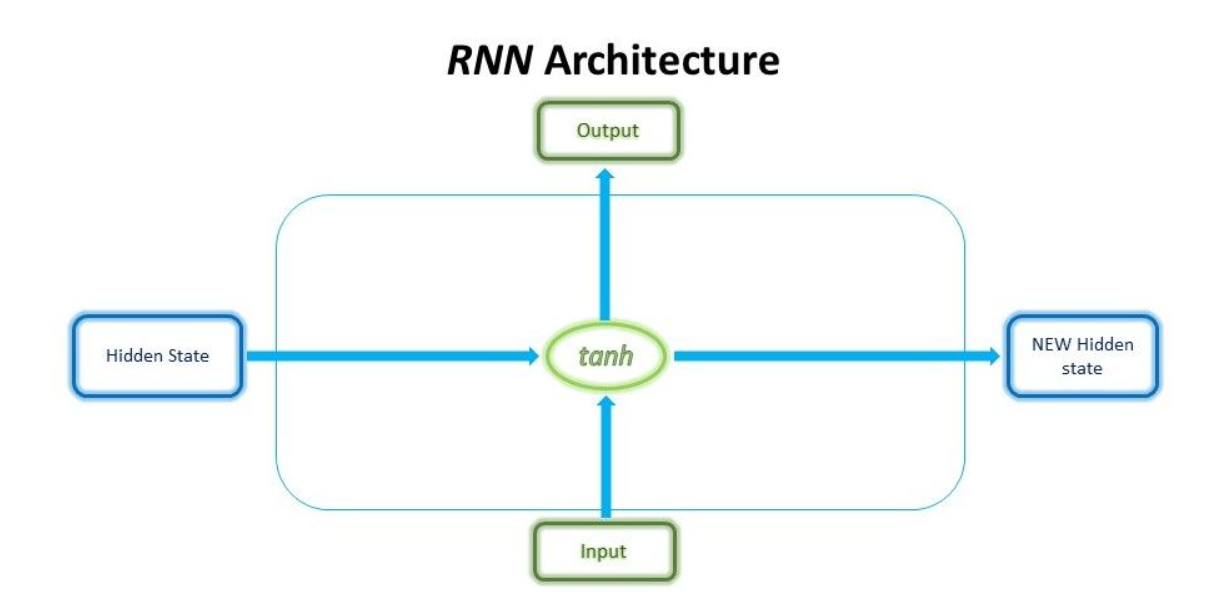
**Nhận xét:** *ht, c*˜*t* khá giống với RNN, nên model có short term memory. Trong khi đó *ct* giống như một băng chuyền ở trên mô hình RNN vậy, thông tin nào cần quan trọng và dùng ở sau sẽ được gửi vào và dùng khi cần => có thể mang thông tin từ đi xa=> long term memory. Do đó mô hình LSTM có cả short term memory và long term memory.



Hình 1.19 Cell state trong LSTM

### 1.2.3. Cơ chế hoạt động của LSTM

Mấu chốt của LSTM nằm ở cơ chế gating với mỗi tế bào LSTM. Tuy nhiên, trong tế bào RNN thông thường, đầu vào (input) tại bước t và trạng thái ẩn phía trước (hidden state) t-1 được tính toán qua “tanh” (như trong hình) để ra được trạng thái ẩn mới và đầu ra (a new hidden state and output).

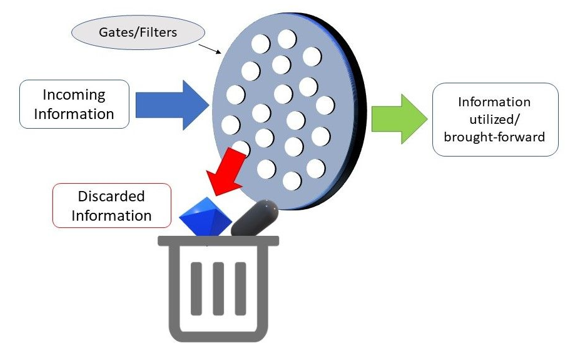
[](https://gpuhub.net/long-short-term-memory-tu-zero-toi-hero-voi-pytorch-p1/)

Hình 1.20 Hoạt động bên trong của một cell RNN

Trong khi đó LSTM có cấu trúc phức tạp hơn một chút. Về cơ bản mô hình của LSTM không khác mô hình truyền thống của RNN, nhưng chúng sử dụng hàm tính toán khác ở các trạng thái ẩn. Bộ nhớ của LSTM được gọi là tế bào (Cell) và bạn có thể tưởng tượng rằng chúng là các hộp đen nhận đầu vào là trạng thái phía trước h(t−1) và đầu vào hiện tại x(t). Bên trong hộp đen này sẽ tự quyết định cái gì cần phải nhớ và cái gì sẽ xoá đi. Sau đó, chúng sẽ kết hợp với trạng thái phía trước, nhớ hiện tại và đầu vào hiện tại (the short-term memory from the previous cell (similar to hidden states in RNNs), lastly the long-term memory and the current input data). Vì vậy mà ta ta có thể truy xuất được quan hệ của các từ phụ thuộc xa nhau rất hiệu quả.

Bộ nhớ ngắn hạn thường được gọi là trạng thái ẩn và bộ nhớ dài hạn thường được gọi là trạng thái tế bào.

Những cổng (gates) này có thể được coi giống như máy lọc nước. Vai trò của các cổng này được cho là loại bỏ có chọn lọc bất kỳ thông tin không liên quan nào, tương tự như cách máy lọc nước loại bỏ tạp chất. Đồng thời, chỉ có nước và các chất dinh dưỡng có lợi có thể đi qua các máy lọc này, giống như cách các cổng chỉ giữ thông tin hữu ích. Tất nhiên, các cổng này cần được đào tạo để lọc chính xác những gì hữu ích và những gì không.

[](https://gpuhub.net/long-short-term-memory-tu-zero-toi-hero-voi-pytorch-p1/)

Hình 1.21 Cổng LSTM có thể được coi là bộ lọc

Các cổng này được gọi là Cổng đầu vào (Input gate), Cổng quên (Forget gate) và Cổng đầu ra (Output gate). Có nhiều biến thể cho tên của các cổng này; tuy nhiên, tính toán và hoạt động của các cổng này hầu hết giống nhau.

### 1.2.4 Ứng dụng Long Short Term Memory (LSTM)

#### 1.2.4.1. Ứng dụng mô hình LSTM

* [Quản lý robot](https://vi.wikipedia.org/w/index.php?title=Qu%E1%BA%A3n_l%C3%BD_robot&action=edit&redlink=1)
* [Dự đoán chuỗi thời gian](https://vi.wikipedia.org/w/index.php?title=D%E1%BB%B1_%C4%91o%C3%A1n_chu%E1%BB%97i_th%E1%BB%9Di_gian&action=edit&redlink=1)
* [Nhận dạng tiếng nói](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_ti%E1%BA%BFng_n%C3%B3i)
* Học giai điệu (nhịp điệu)
* Sáng tác nhạc
* Học ngữ pháp
* [Nhận dạng chữ viết tay](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_ch%E1%BB%AF_vi%E1%BA%BFt_tay)
* Nhận dạng hành động của con người
* [Ngôn ngữ ký hiệu](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_k%C3%BD_hi%E1%BB%87u)
* Phát hiện tương đồng protein
* Dự đoán vị trí dưới tế bào của protein
* Phát hiện chuỗi thời gian bất thường
* Một số tác vụ dự đoán trong lĩnh vực quản lý quy trình kinh doanh
* Dự đoán trong các lộ trình chăm sóc y tế
* [Phân tích ngữ nghĩa](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A2n_t%C3%ADch_ng%E1%BB%AF_ngh%C4%A9a&action=edit&redlink=1)
* [Đồng phân đoạn đối tượng](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%93ng_ph%C3%A2n_%C4%91o%E1%BA%A1n_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng&action=edit&redlink=1)
* Quản lý hành khách sân bay
* [Dự báo giao thông vận tải](https://vi.wikipedia.org/w/index.php?title=D%E1%BB%B1_b%C3%A1o_giao_th%C3%B4ng_v%E1%BA%ADn_t%E1%BA%A3i&action=edit&redlink=1) ngắn hạn
* Thiết kế thuốc

#### 1.2.4.2. Chuỗi thời gian

Chuỗi thời gian là một chuỗi các phép đo được thực hiện theo thời gian, thường thu được ở các khoảng cách đều nhau, có thể là hàng ngày, hàng tháng, hàng quý hoặc hàng năm. Phân tích chuỗi thời gian bao gồm các phương pháp để phân tích dữ liệu chuỗi thời gian để trích xuất số liệu thống kê có ý nghĩa và các đặc điểm khác của dữ liệu. Dự báo chuỗi thời gian là việc sử dụng một mô hình để dự đoán các giá trị trong tương lai dựa trên các giá trị được quan sát trước đó.

Nói cách khác, chuỗi thời gian là một chuỗi các điểm dữ liệu được ghi lại tại các thời điểm cụ thể.

Một số ví dụ về chuỗi thời gian có thể là:

* Nhiệt độ không khí hàng ngày hoặc lượng mưa hàng tháng ở Bangalore, Ấn Độ
* Lưu lượng dòng chảy hàng năm của sông Ganga tại Patna
* Dữ liệu dân số Ấn Độ hàng năm
* Giá cổ phiếu đóng cửa hàng ngày
* Lãi suất hàng tuần

#### 1.2.4.3. Các khái niệm liên quan tới chuỗi time series

**Stationary**: Tính dừng.

Một chuỗi thời gian có tính dừng khi các giá trị mean, variance, autocorrelation không thay đổi theo thời gian. Với hầu hết các phương pháp thống kê dự báo, ta đều phải đảm bảo tính dừng của chuỗi dữ liệu vì thế việc kiểm tra tính dừng là rất quan trọng.

**Lag**:

Lag-1 của chuỗi thời gian có thể thu được bằng cách dịch chuyển thời gian về quá khứ 1 đơn vị. Tương tự Lag-n thu được bằng cách dịch chuyển n đơn vị thời gian về quá khứ. Ví dụ: Khi kí hiệu chuỗi thời gian là X(t) thì ta có lag-n của chuỗi thời gian

**Specification**:

Là bước kiểm tra mối quan hệ tuyến tính hay phi tuyến của các biến phụ thuộc bằng cách sử dụng các models như ARIMA, ARCH, GARCH, VAR, Cointegration, etc,…

**Differencing**:

Là phương pháp để biến đổi một chuỗi time series thành chuỗi dừng (stationary), để loại bỏ xu hướng (trend), hay sự tự tương quan (auto-correlation). Ví dụ:Differencing bậc 1 với khoảng interval 1 của chuỗi thời gian X(t) được tính bằng công thức:

D\_1 = X(t) – X(t-1)

## 1.3. MediaPipe

**Giới thiệu về MediaPipe:**

Về tổng quan, MediaPipe là tập hợp của một loạt các giải pháp Machine Learning đa nền tảng, có thể can thiệp được và cực kỳ lightweight. Một số ưu điểm có thể kể tới của giải pháp này bao gồm:

Cung cấp một giải pháp inference nhanh chóng: Google khẳng định rằng bộ công cụ này có thể chạy ổn định trên hầu hết các cấu hình phần cứng thông dụng.

Dễ dàng cài đặt và triển khai: Việc cài đặt cực kỳ dễ dàng và tiện lợi, có thể triển khai trên nhiều nền tảng khác nhau như Mobile (Android/iOS), Desktop/Cloud, Web và IoT devices.

Mã nguồn mở và miễn phí: Toàn bộ source code được công khai trên MediaPipe, người dùng hoàn toàn có thể sử dụng và tùy chỉnh trực tiếp để phù hợp với bài toán của mình.

# CHƯƠNG 2. ỨNG DỤNG NHẬN DIỆN HÀNH VI

## 2.1. Nhận diện hành vi

Nhận diện hành vi (Behavioral recognition) trong AI là quá trình nhận biết và dự đoán hành vi của con người hoặc động vật thông qua các dữ liệu thu thập được từ các cảm biến như camera, microphone, cảm biến chuyển động, vv. Hành vi được xác định bằng cách phân tích các dữ liệu này và áp dụng các thuật toán và mô hình học máy để dự đoán hành động tiếp theo của người hoặc động vật.

Cơ sở lý thuyết của nhận diện hành vi bao gồm các khái niệm trong lĩnh vực nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và các kỹ thuật học máy. Các phương pháp và thuật toán như deep learning, neural networks, decision trees, support vector machines (SVM), hidden Markov models (HMM) và probabilistic graphical models (PGM) được áp dụng trong việc xử lý dữ liệu và dự đoán hành vi.

Nhận diện hành vi là một chủ đề quan trọng trong lĩnh vực trí tuệ nhân tạo và ứng dụng rộng rãi trong nhiều lĩnh vực như an ninh, y tế, giải trí, tài chính.

Sau đây là một số khía cạnh khác liên quan đến nhận diện hành vi mà nhận được nhiều sự quan tâm:

* **Phương pháp và kỹ thuật:** Các phương pháp và kỹ thuật nhận diện hành vi khác nhau bao gồm việc sử dụng các mô hình học máy như máy vector hỗ trợ, mạng nơ-ron, cây quyết định và thuật toán rừng ngẫu nhiên. Các phương pháp này có thể được sử dụng để phân tích dữ liệu từ các cảm biến khác nhau như camera, micro, cảm biến chuyển động, vv.
* **Ứng dụng:** Nhận diện hành vi có nhiều ứng dụng khác nhau như giám sát an ninh, chẩn đoán y tế, phân tích thị trường, vv. Ví dụ, trong lĩnh vực y tế, việc theo dõi các hành vi của bệnh nhân có thể giúp đánh giá tình trạng sức khỏe của họ, phát hiện các bệnh lý sớm và cải thiện chất lượng chăm sóc sức khỏe.
* **Thách thức:** Tuy nhiên, việc sử dụng các công nghệ nhận diện hành vi cũng đặt ra nhiều thách thức. Một trong những thách thức lớn nhất là bảo vệ quyền riêng tư của cá nhân và đảm bảo tính minh bạch trong việc thu thập và sử dụng dữ liệu. Ngoài ra, việc đảm bảo tính chính xác và độ tin cậy của các thuật toán và mô hình cũng là một vấn đề quan trọng.
* **Công cụ và thư viện:** Hiện nay có rất nhiều công cụ và thư viện được phát triển để hỗ trợ việc phân tích và nhận diện hành vi, bao gồm OpenCV, TensorFlow, PyTorch, và thư viện scikit-learn trong Python. Việc sử dụng các công cụ này có thể giúp đơn giản hóa quá trình phát triển và triển khai các hệ thống nhận diện hành vi.

Các kỹ thuật phân tích hành vi: Có nhiều kỹ thuật phân tích hành vi được sử dụng trong nhận diện hành vi, bao gồm nhận diện hình ảnh và video, phân tích giọng nói và phân tích dữ liệu vận động (kinetic data analysis). Điều này cho phép hệ thống đánh giá các hành vi của con người dựa trên nhiều mặt khác nhau, từ hình ảnh đến dữ liệu vận động và giọng nói.

Ứng dụng của nhận diện hành vi trong thực tế: Nhận diện hành vi có nhiều ứng dụng trong thực tế, bao gồm giám sát giao thông, giám sát hoạt động của con người trong các nhà máy và dây chuyền sản xuất, phát hiện các hành vi bất thường trong các hệ thống an ninh và giám sát hành vi của khách hàng trong các cửa hàng.

Những thách thức của nhận diện hành vi: Tuy nhiên, việc nhận diện hành vi còn đối mặt với nhiều thách thức, bao gồm độ chính xác của hệ thống, độ chính xác của dữ liệu đầu vào, các vấn đề về quyền riêng tư và đạo đức, vv.

Các công nghệ và công cụ hỗ trợ: Các công nghệ và công cụ hỗ trợ nhận diện hành vi ngày càng phát triển, từ các hệ thống nhận diện hình ảnh và video, các công nghệ nhận diện giọng nói và các hệ thống phân tích dữ liệu vận động. Nhiều công ty cũng đã phát triển các công cụ phần mềm để hỗ trợ việc nhận diện hành vi, cho phép các nhà nghiên cứu và các chuyên gia phát triển các hệ thống phân tích và nhận diện hành vi một cách dễ dàng và nhanh chóng.

Tiềm năng của nhận diện hành vi trong tương lai: Với sự phát triển của trí tuệ nhân tạo và các công nghệ liên quan, việc nhận diện hành vi sẽ có tiềm năng lớn để phát triển trong tương lai, đặc biệt là trong các lĩnh vực như y tế, an ninh và quản lý sản xuất.

## 2.2. Cơ sở phát triển một ứng dụng nhận diện

Để phát triển và thực hiện một ứng dụng nhận diện hành vi, có một số bước cơ bản cần thực hiện:

**Thu thập dữ liệu:** Để phát triển một hệ thống nhận diện hành vi, cần có một lượng lớn dữ liệu để huấn luyện mô hình. Việc thu thập dữ liệu cần được thực hiện theo một phương pháp hợp lý và đảm bảo tính đại diện.

**Xử lý dữ liệu:** Sau khi thu thập dữ liệu, cần phải xử lý và chuẩn hóa các dữ liệu này để đảm bảo tính nhất quán và độ chính xác của dữ liệu.

**Chọn phương pháp và mô hình:** Sau khi có dữ liệu đã được xử lý, cần chọn phương pháp và mô hình phù hợp để phân tích và nhận diện hành vi. Có nhiều phương pháp và mô hình khác nhau có thể được sử dụng, tùy thuộc vào mục đích sử dụng và đặc điểm của dữ liệu.

**Huấn luyện mô hình:** Sau khi chọn phương pháp và mô hình, cần huấn luyện mô hình trên dữ liệu đã chuẩn hóa. Quá trình huấn luyện sẽ giúp tối ưu hóa các tham số của mô hình và cải thiện độ chính xác của nó.

**Đánh giá và kiểm tra mô hình:** Sau khi huấn luyện xong, cần đánh giá và kiểm tra độ chính xác của mô hình trên các tập dữ liệu độc lập. Nếu mô hình không đạt được độ chính xác cần thiết, cần tiến hành điều chỉnh mô hình hoặc sử dụng các phương pháp cải tiến để cải thiện hiệu suất của nó.

**Triển khai và sử dụng:** Sau khi đạt được độ chính xác đủ tốt, mô hình có thể được triển khai và sử dụng trong ứng dụng thực tế. Việc triển khai và sử dụng cần được thực hiện cẩn thận và đảm bảo tính đúng đắn, an toàn và bảo mật cho người dùng.

Trong dự án nghiên cứu khoa học kĩ thuật, nhóm nghiên cứu chọn mô hình phát triển ứng dụng phát triển nhận diện hành vi dựa trên mô hình kết hợp Mediapipe Pose và LSTM Model. Mô hình Mediapipe Pose và LSTM Model được mô tả như sau:

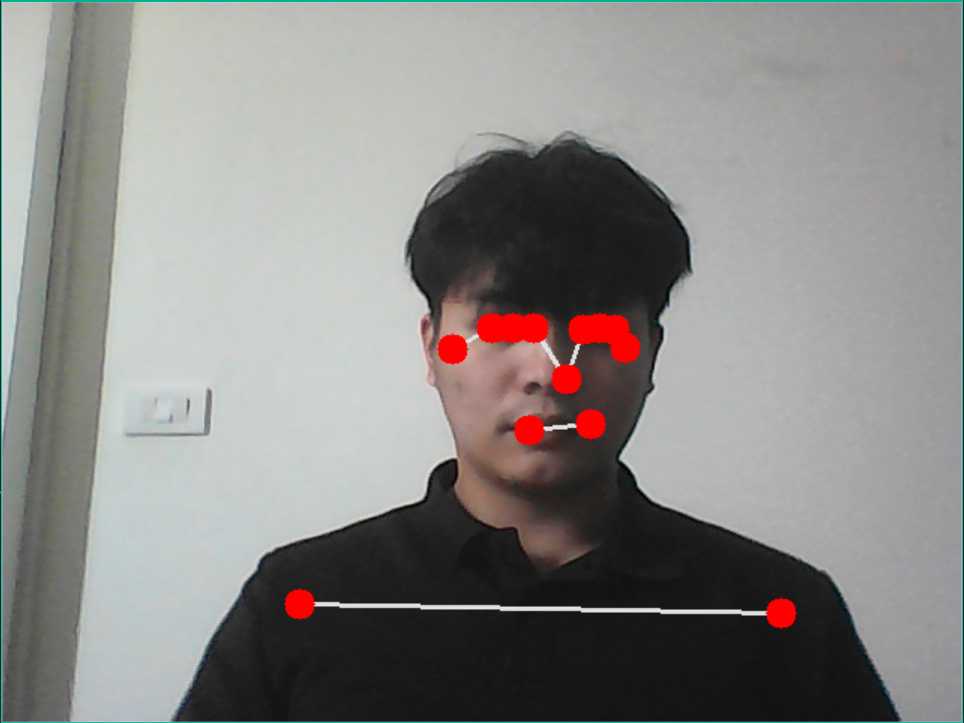
Mediapipe Pose là một thư viện mã nguồn mở do Google phát triển, được sử dụng để nhận diện và theo dõi vị trí và hướng của các khớp trong cơ thể con người thông qua việc phân tích dữ liệu từ camera hoặc video. Thư viện này sử dụng mạng nơ-ron tích chập (CNN) để xác định vị trí và hướng của các khớp trong cơ thể. Nó cung cấp một API dễ sử dụng cho các nhà phát triển để tích hợp việc phát hiện và theo dõi vị trí và hướng của các khớp trong cơ thể vào các ứng dụng của họ.

LSTM (Long Short-Term Memory) là một loại mạng nơ-ron thường được sử dụng trong việc mô hình hóa chuỗi dữ liệu, đặc biệt là trong các ứng dụng xử lý ngôn ngữ tự nhiên và âm nhạc. Một trong những đặc điểm nổi bật của LSTM là khả năng lưu trữ thông tin trong thời gian dài và tự học được mối quan hệ phức tạp giữa các đặc trưng của dữ liệu đầu vào. LSTM thường được sử dụng để giải quyết các vấn đề liên quan đến dự đoán chuỗi thời gian, ví dụ như dự đoán giá cổ phiếu hoặc dự báo thời tiết.

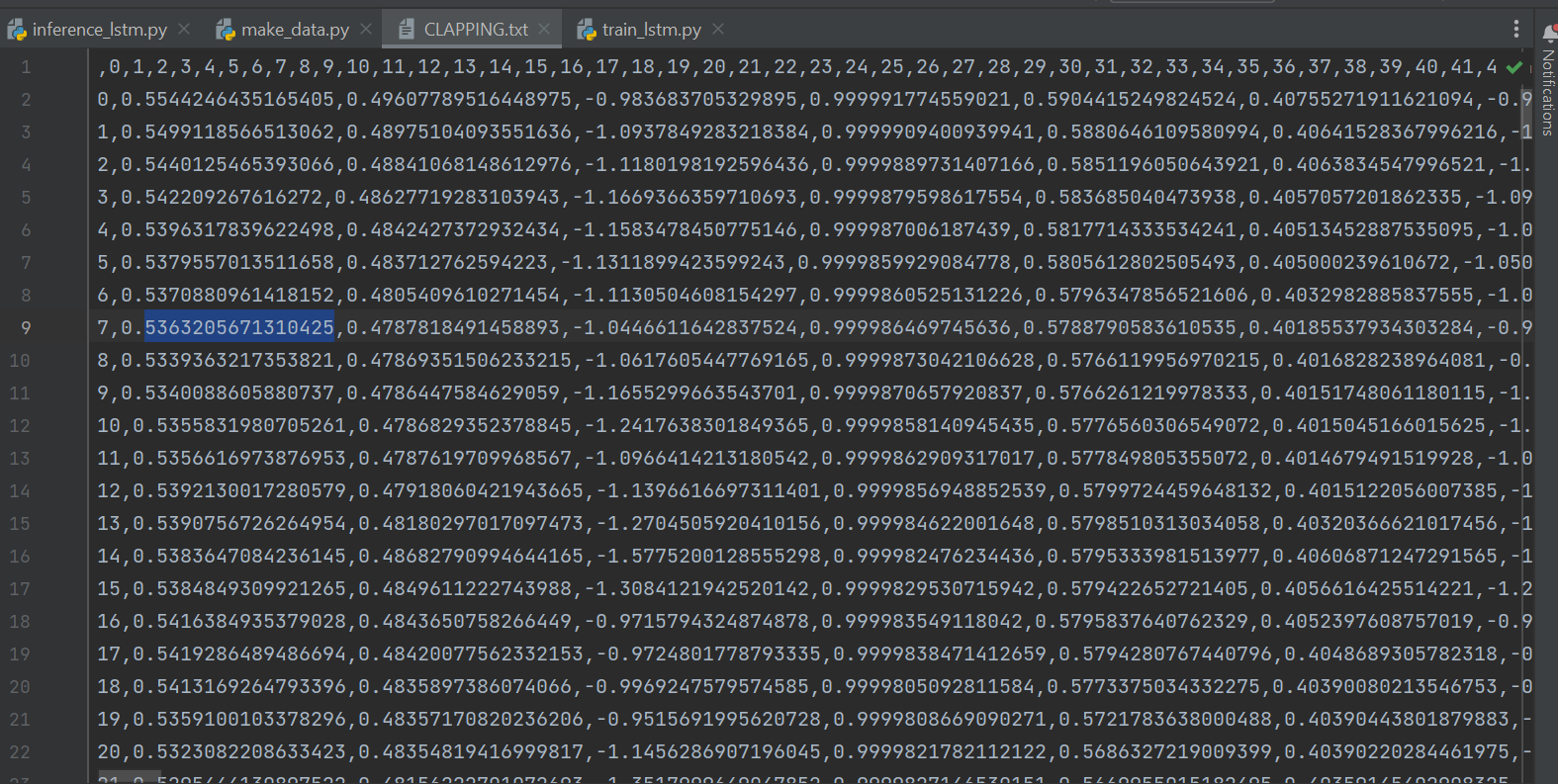
Kết hợp Mediapipe Pose và LSTM Model có thể được sử dụng để phân tích và dự đoán hành vi của con người trong thời gian thực. Việc sử dụng Mediapipe Pose để theo dõi vị trí và hướng của các khớp trong cơ thể, kết hợp với LSTM để phân tích chuỗi thời gian của dữ liệu đầu vào, có thể giúp dự đoán và nhận diện hành vi của con người, ví dụ như nhận diện các động tác của người dùng hoặc theo dõi các bệnh nhân trong lĩnh vực y tế,….

## 2.3. Ứng dụng vào việc nhận diện trong thực tế

**Thu thập dữ liệu:** Thực tế dữ liệu đầu vào được thu thập từ camera, các video là các Frame nối liên tiếp với nhau. Dự vào các đặc tính trên khuôn mặt và các bộ phận cơ thể để tạo các điểm liên kết.

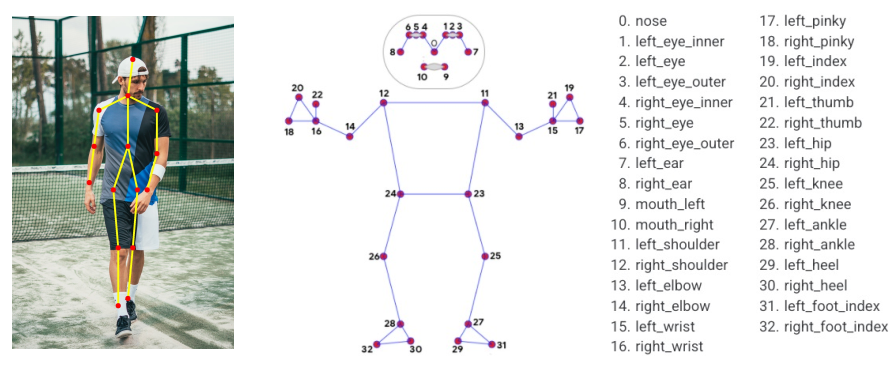


**Xử lý dữ liệu:** Phụ thuộc vào sự thay đổi của các điểm liên kết qua các Frame liên tiếp để tạo ra được file chứa tọa độ thay đổi của các điểm liên kết, từ đó gắn nhãn cho các hành động được tạo data.

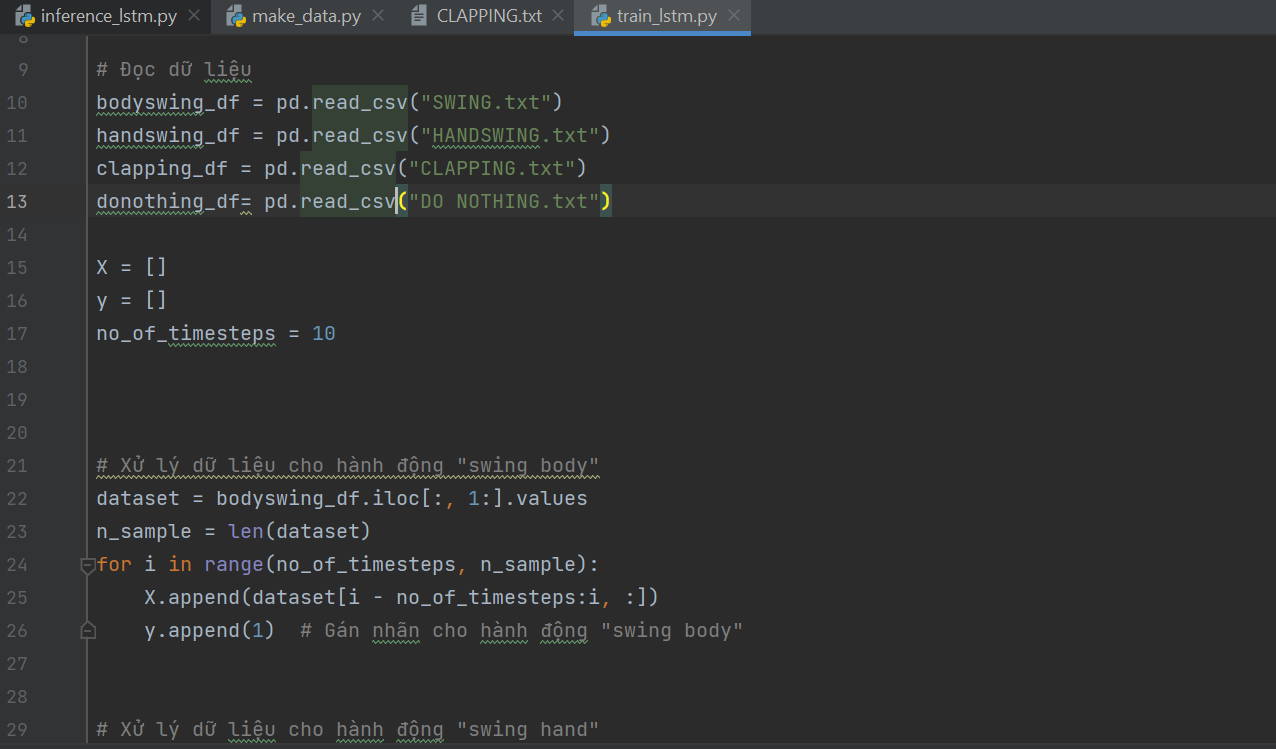


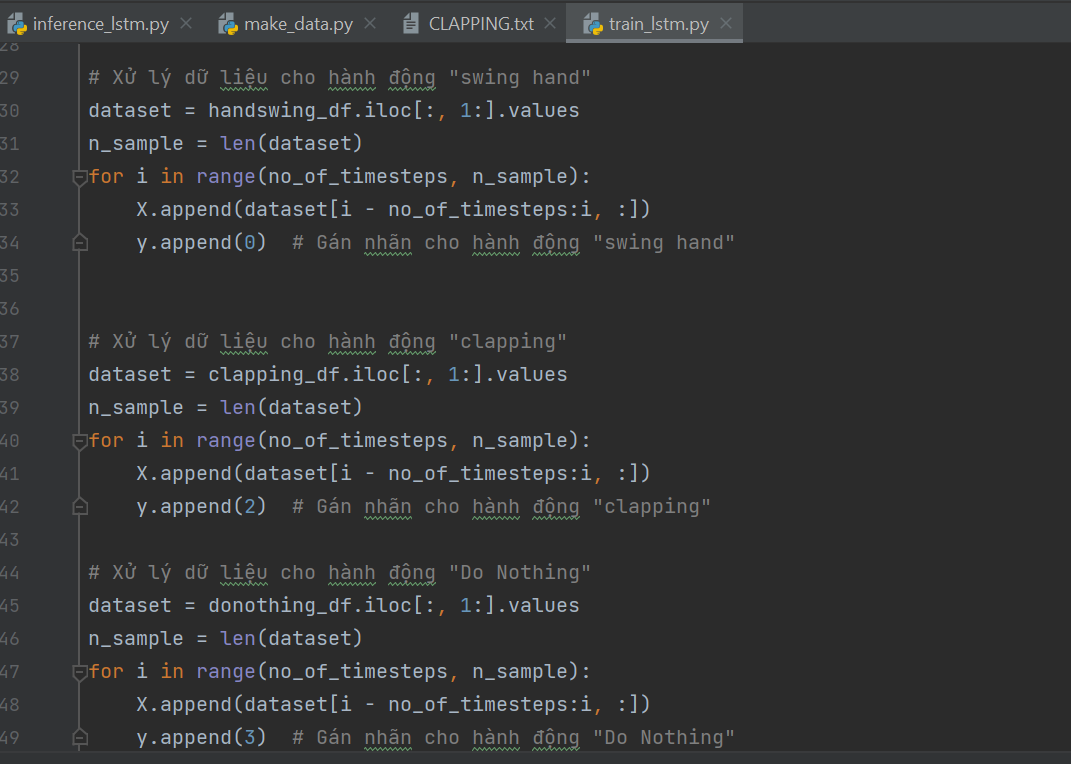
**Chọn phương pháp và mô hình:** Mô hình sử dụng là mô hình LSTM. Một mô hình phân tích dữ liệu dựa sự thay đổi của các Frame

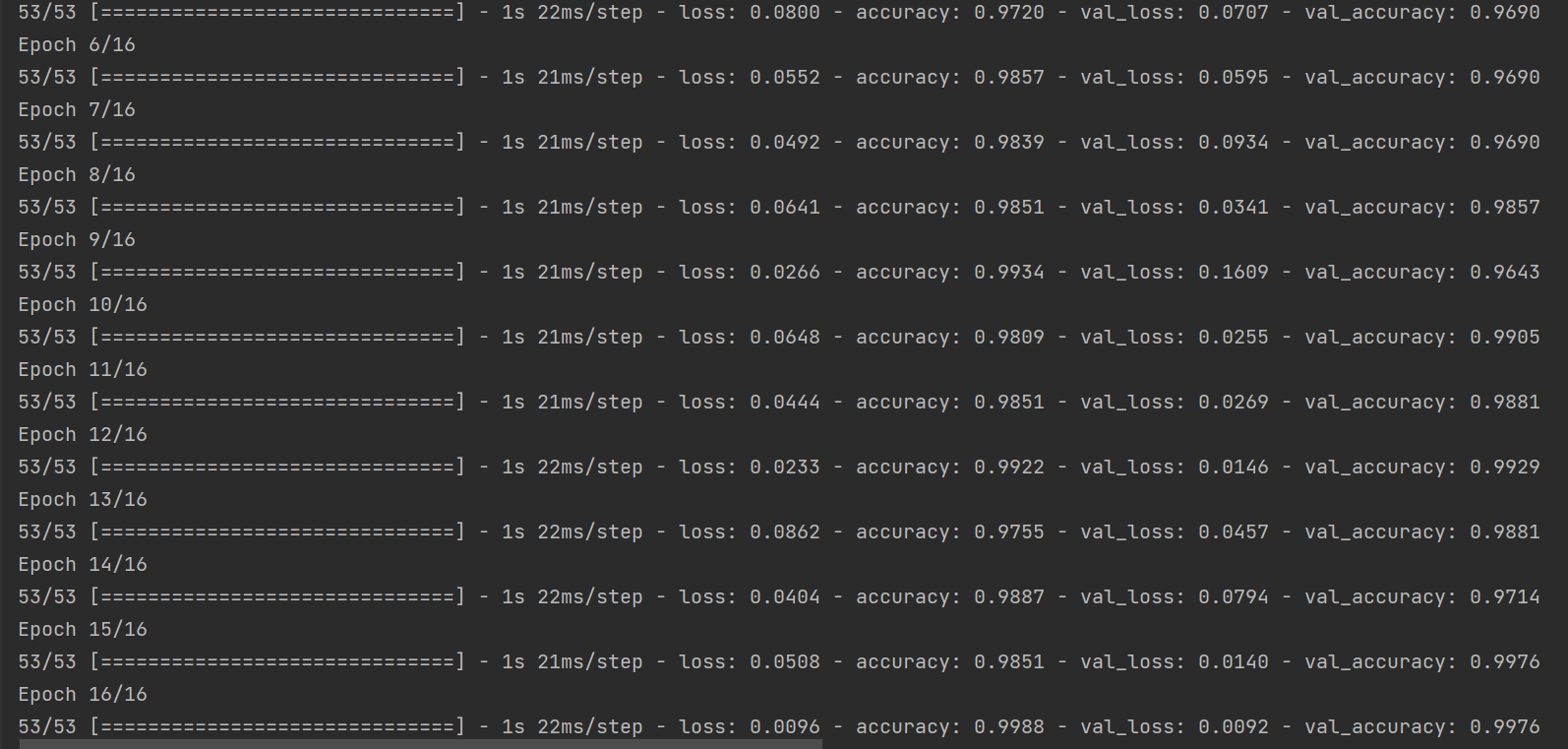




**Huấn luyện mô hình:** Việc huấn luyện mô hình bằng cách tạo các bộ dataset được đọc từ các file lưu dữ liệu về việc thay đổi tọa độ của các điểm liên kết qua các Frame. Khi được train các tọa độ sẽ được lưu vào bộ dataset để những lần lấy dữ liệu thực tế để so sánh với các sự thay đổi của các tọa độ đó. Từ đó lấy ra các nhãn gần đúng với các hành động thực tế



** Đánh giá và kiểm tra mô hình:** Sau khi huấn luyện xong, cần đánh giá và kiểm tra độ chính xác của mô hình trên các tập dữ liệu độc lập. Mô hình đã đạt được độ chính xác cao và giá trị hàm mất mát không quá ngưỡng cho phép.



**Triển khai và sử dụng:**  Tại giao diện interface, hình ảnh thực thu được từ camera, các hình ảnh sẽ được tách ra thành các frame, các frame này cũng sẽ được nối các điểm liên kết, dựa vào sự thay đổi của các điểm liên kết này sẽ được so sánh với các bộ dataset được train trước đó. Từ việc so sánh này sẽ trả về các giá trị label tương ứng với các hành động.

Trong dự án nghiên cứu khoa học kĩ thuật, nhóm nghiên cứu chọn mô hình phát triển ứng dụng phát triển nhận diện hành vi dựa trên mô hình kết hợp Mediapipe Pose và LSTM Model. Mô hình Mediapipe Pose và LSTM Model được mô tả như sau:

Mediapipe Pose là một thư viện mã nguồn mở do Google phát triển, được sử dụng để nhận diện và theo dõi vị trí và hướng của các khớp trong cơ thể con người thông qua việc phân tích dữ liệu từ camera hoặc video. Thư viện này sử dụng mạng nơ-ron tích chập (CNN) để xác định vị trí và hướng của các khớp trong cơ thể. Nó cung cấp một API dễ sử dụng cho các nhà phát triển để tích hợp việc phát hiện và theo dõi vị trí và hướng của các khớp trong cơ thể vào các ứng dụng của họ.

LSTM (Long Short-Term Memory) là một loại mạng nơ-ron thường được sử dụng trong việc mô hình hóa chuỗi dữ liệu, đặc biệt là trong các ứng dụng xử lý ngôn ngữ tự nhiên và âm nhạc. Một trong những đặc điểm nổi bật của LSTM là khả năng lưu trữ thông tin trong thời gian dài và tự học được mối quan hệ phức tạp giữa các đặc trưng của dữ liệu đầu vào.

Kết hợp Mediapipe Pose và LSTM Model có thể được sử dụng để phân tích và dự đoán hành vi của con người trong thời gian thực. Việc sử dụng Mediapipe Pose để theo dõi vị trí và hướng của các khớp trong cơ thể, kết hợp với LSTM để phân tích chuỗi thời gian của dữ liệu đầu vào, có thể giúp dự đoán và nhận diện hành vi của con người, ví dụ như nhận diện các động tác của người dùng hoặc theo dõi các bệnh nhân trong lĩnh vực y tế,….

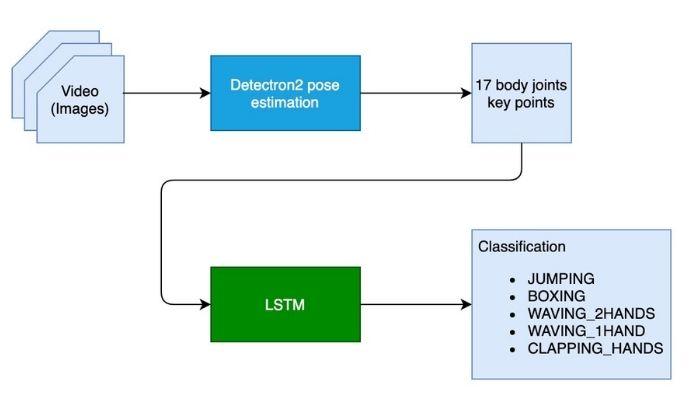
# CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM

## 3.1. Tổng quan sơ đồ bài toán

Để phân loại một hành động, trước tiên chúng ta cần xác định vị trí các bộ phận cơ thể khác nhau trong mọi khung hình, sau đó phân tích chuyển động của các bộ phận đó theo thời gian.

Bước đầu tiên đạt được bằng cách sử dụng Detectron2, nó xuất ra tư thế của cơ thể sau khi quan sát một khung hình trong video.

Bước thứ hai là phân tích chuyển động của cơ thể theo thời gian và đưa ra dự đoán được thực hiện bằng mạng LSTM. Đầu vào là các Keypoints từ một chuỗi khung được, đầu ra là loại hành đồng được dự đoán.



Hình 3.1 Sơ đồ bài toán

## 3.2. Chuẩn bị dữ liệu

Đối với ứng dụng này, chúng ta chỉ cần huấn luyện mạng LSTM để phân loại các hành động, còn phần Pose Estimation thì chúng ta sẽ sử dụng pre-trained có sẵn cung cấp bởi Detectron2.

Bộ dữ liệu được dùng để huấn luyện mạng LSTM được tạo thành bằng cách sử dụng OpenPose trên các video của tập dữ liệu Berkeley Multimodal Human Action Database (MHAD). Sử dụng cách thức tương tự, chúng ta cũng có thể tạo ra bộ dữ liệu của riêng mình.

Training Data bao gồm các chuỗi 17 Keypoints kết hợp với một nhãn tương ứng. Mỗi Keypoint là một cặp tọa độ (x,y).

A picture containing text, footwear, person, clothing

Description automatically generated

Hình 3.2 Chuỗi keypoint

Mỗi lần phân loại, chúng ta sẽ sử dụng 32 frames liên tiếp nhau. Như vậy thì kích thước dữ liệu của một Input Data sẽ là 32x34:

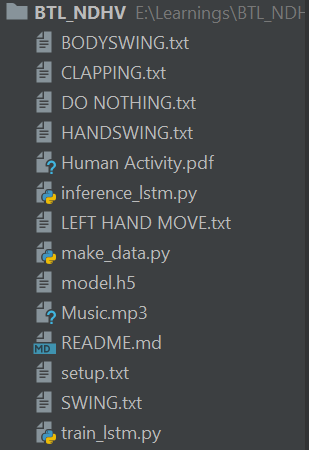
A picture containing text, font, white, typography

Description automatically generated

## 3.3. Quá trình thực hiện

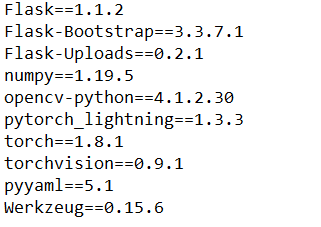
Sau khi khởi tạo file thực hiện tuần tự các bước sau:

**Bước 1: Tạo thư mục lưu trữ**



Hình 3.3 Thư mục Deep-Learning

**Bước 2: Cài đặt các thư viện cần thiết**

****

Hình 3.4 Các thư viện cần thiết

**Bước 3: Khởi tạo Human Pose Estimation model**

cfg = get\_cfg()

cfg.merge\_from\_file(model\_zoo.get\_config\_file("COCO-Keypoints/keypoint\_rcnn\_R\_50\_FPN\_3x.yaml"))

cfg.MODEL.ROI\_HEADS.SCORE\_THRESH\_TEST = 0.5

cfg.MODEL.WEIGHTS=model\_zoo.get\_checkpoint\_url("COCO-Keypoints/keypoint\_rcnn\_R\_50\_FPN\_3x.yaml")

pose\_detector = DefaultPredictor(cfg)

**Bước 4: Định nghĩa LSTM model**

#lstm classifier definition

class ActionClassificationLSTM(pl.LightningModule):

def \_\_init\_\_(self, input\_features, hidden\_dim, learning\_rate=0.001):

super().\_\_init\_\_()

self.save\_hyperparameters()

self.lstm = nn.LSTM(input\_features, hidden\_dim, num\_layers=2, batch\_first=True)

self.linear = nn.Linear(hidden\_dim, TOT\_ACTION\_CLASSES)

def forward(self, x):

lstm\_out, (ht, ct) = self.lstm(x)

return self.linear(ht[-1])

def training\_step(self, batch, batch\_idx):

x, y = batch

y = torch.squeeze(y)

y = y.long()

y\_pred = self(x)

loss = F.cross\_entropy(y\_pred, y)

prob = F.softmax(y\_pred, dim=1)

pred = prob.data.max(dim=1)[1]

acc = torchmetrics.functional.accuracy(pred, y)

dic = {

'batch\_train\_loss': loss,

'batch\_train\_acc': acc

}

self.log('batch\_train\_loss', loss, prog\_bar=True)

self.log('batch\_train\_acc', acc, prog\_bar=True)

return {'loss': loss, 'result': dic}

def training\_epoch\_end(self, training\_step\_outputs):

avg\_train\_loss = torch.tensor([x['result']['batch\_train\_loss'] for x in training\_step\_outputs]).mean()

avg\_train\_acc = torch.tensor([x['result']['batch\_train\_acc'] for x in training\_step\_outputs]).mean()

self.log('train\_loss', avg\_train\_loss, prog\_bar=True)

self.log('train\_acc', avg\_train\_acc, prog\_bar=True)

def validation\_step(self, batch, batch\_idx):

x, y = batch

y = torch.squeeze(y)

y = y.long()

y\_pred = self(x)

loss = F.cross\_entropy(y\_pred, y)

prob = F.softmax(y\_pred, dim=1)

pred = prob.data.max(dim=1)[1]

acc = torchmetrics.functional.accuracy(pred, y)

dic = {

'batch\_val\_loss': loss,

'batch\_val\_acc': acc

}

self.log('batch\_val\_loss', loss, prog\_bar=True)

self.log('batch\_val\_acc', acc, prog\_bar=True)

#return dict

return dic

def validation\_epoch\_end(self, validation\_step\_outputs):

avg\_val\_loss = torch.tensor([x['batch\_val\_loss']

for x in validation\_step\_outputs]).mean()

avg\_val\_acc = torch.tensor([x['batch\_val\_acc']

for x in validation\_step\_outputs]).mean()

self.log('val\_loss', avg\_val\_loss, prog\_bar=True)

self.log('val\_acc', avg\_val\_acc, prog\_bar=True)

def configure\_optimizers(self):

optimizer = optim.Adam(self.parameters(), lr=self.hparams.learning\_rate)

scheduler = optim.lr\_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=10, min\_lr=1e-15, verbose=True)

**Bước 5: Huấn luyện mô hình**

import numpy as np  
import pandas as pd  
  
from keras.layers import LSTM, Dense, Dropout  
from keras.models import Sequential  
  
from sklearn.model\_selection import train\_test\_split  
  
# Đọc dữ liệu  
bodyswing\_df = pd.read\_csv("SWING.txt")  
handswing\_df = pd.read\_csv("HANDSWING.txt")  
clapping\_df = pd.read\_csv("CLAPPING.txt")  
donothing\_df= pd.read\_csv("DO NOTHING.txt")  
lefthand\_df = pd.read\_csv("LEFT HAND MOVE.txt")  
X = []  
y = []  
no\_of\_timesteps = 10  
  
  
  
# Xử lý dữ liệu cho hành động "swing body"  
dataset = bodyswing\_df.iloc[:, 1:].values  
n\_sample = len(dataset)  
for i in range(no\_of\_timesteps, n\_sample):  
 X.append(dataset[i - no\_of\_timesteps:i, :])  
 y.append(1) # Gán nhãn cho hành động "swing body"  
  
  
# Xử lý dữ liệu cho hành động "swing hand"  
dataset = handswing\_df.iloc[:, 1:].values

n\_sample = len(dataset)

for i in range(no\_of\_timesteps, n\_sample):

X.append(dataset[i - no\_of\_timesteps:i, :])  
 y.append(0) # Gán nhãn cho hành động "swing hand"  
  
  
# Xử lý dữ liệu cho hành động "clapping"  
dataset = clapping\_df.iloc[:, 1:].values  
n\_sample = len(dataset)  
for i in range(no\_of\_timesteps, n\_sample):  
 X.append(dataset[i - no\_of\_timesteps:i, :])  
 y.append(2) # Gán nhãn cho hành động "clapping"  
  
# Xử lý dữ liệu cho hành động "Do Nothing"  
dataset = donothing\_df.iloc[:, 1:].values  
n\_sample = len(dataset)  
for i in range(no\_of\_timesteps, n\_sample):  
 X.append(dataset[i - no\_of\_timesteps:i, :])  
 y.append(4) # Gán nhãn cho hành động "Do Nothing"  
  
# Xử lý dữ liệu cho hành động "left hand move"  
dataset = lefthand\_df.iloc[:, 1:].values  
n\_sample = len(dataset)  
for i in range(no\_of\_timesteps, n\_sample):

X.append(dataset[i - no\_of\_timesteps:i, :])  
 y.append(3) # Gán nhãn cho hành động "left hand move"  
  
  
X, y = np.array(X), np.array(y)  
print(X.shape, y.shape)  
  
# Chia dữ liệu thành tập huấn luyện và tập kiểm tra  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
  
# Xây dựng mô hình  
model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X.shape[1], X.shape[2])))

model.add(Dropout(0.2))

model.add(LSTM(units=50, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=50, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=50))

model.add(Dropout(0.2))  
model.add(Dense(units=5, activation="softmax")) # Số lượng hành động là 3: swing hand, swing body, clapping

model.compile(optimizer="adam", metrics=['accuracy'],

loss="sparse\_categorical\_crossentropy")  
  
# Huấn luyện mô hình

model.fit(X\_train, y\_train, epochs=16

, batch\_size=32, validation\_data=(X\_test, y\_test))  
# Lưu mô hình  
model.save("model.h5")

**Kết quả huấn luyện model:**

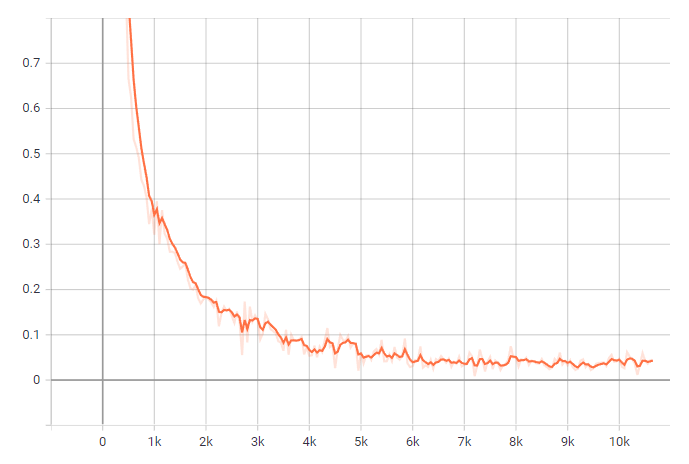
Train Accuracy:

A picture containing plot, line, text, diagram

Description automatically generated

*Biểu đồ 3.1 Train Accuracy*

Train Loss:



*Biểu đồ 3.2 Train loss*

Validation Accuracy:

A picture containing plot, line, diagram, text

Description automatically generated

*Biểu đồ 3.3 Validation Accuracy*

Validation Loss:

A picture containing line, plot, diagram, parallel

Description automatically generated

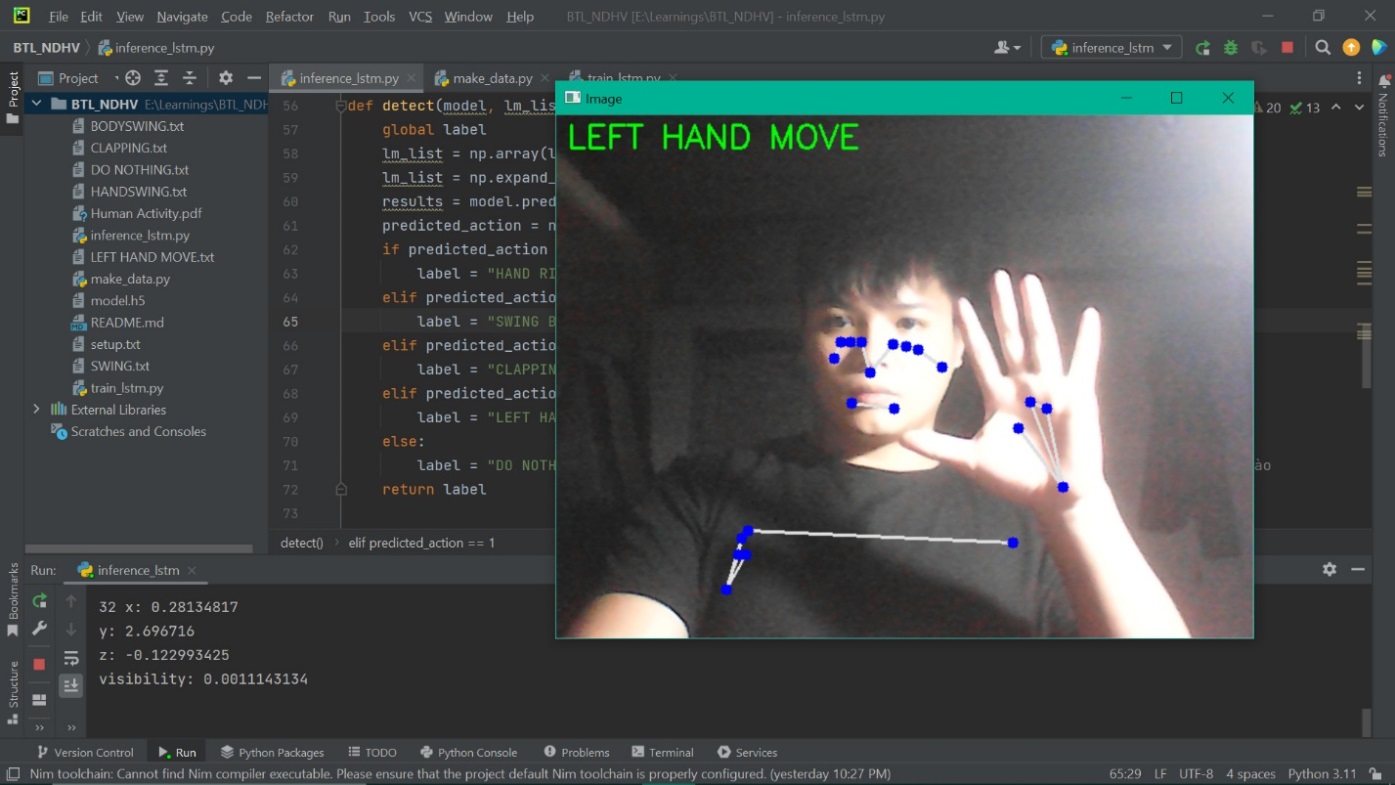
*Biểu đồ 3.4 Validation Loss*

**Bước 6: Thực hiên Inference**

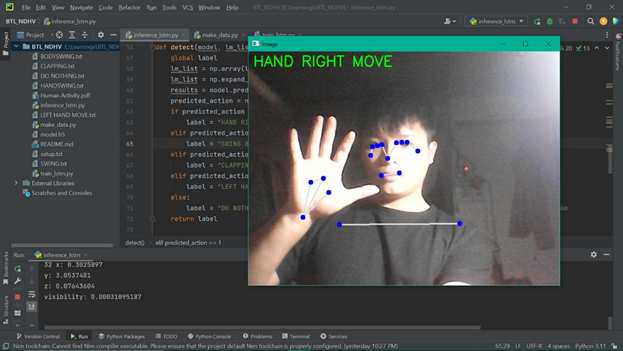
import cv2  
import mediapipe as mp  
import numpy as np  
import tensorflow as tf  
import pygame  
  
label = "Warmup...."  
n\_time\_steps = 10  
lm\_list = []  
  
mpPose = mp.solutions.pose  
pose = mpPose.Pose()  
mpDraw = mp.solutions.drawing\_utils  
  
model = tf.keras.models.load\_model("model.h5")  
  
cap = cv2.VideoCapture(0)  
  
# Khởi tạo Pygame để điều khiển nhạc  
pygame.mixer.init()  
music\_file = 'music.mp3' # Đường dẫn tới file nhạc của bạn trong cùng thư mục  
pygame.mixer.music.load(music\_file)  
  
def make\_landmark\_timestep(results):  
 c\_lm = []  
 for id, lm in enumerate(results.pose\_landmarks.landmark):  
 c\_lm.append(lm.x)  
 c\_lm.append(lm.y)  
 c\_lm.append(lm.z)  
 c\_lm.append(lm.visibility)  
 return c\_lm  
  
def draw\_landmark\_on\_image(mpDraw, results, img):  
 mpDraw.draw\_landmarks(img, results.pose\_landmarks, mpPose.POSE\_CONNECTIONS)  
 for id, lm in enumerate(results.pose\_landmarks.landmark):  
 h, w, c = img.shape  
 cx, cy = int(lm.x \* w), int(lm.y \* h)  
 cv2.circle(img, (cx, cy), 5, (255, 0, 0), cv2.FILLED)  
 return img  
  
def draw\_class\_on\_image(label, img):  
 font = cv2.FONT\_HERSHEY\_SIMPLEX  
 bottomLeftCornerOfText = (10, 30)  
 fontScale = 1  
 fontColor = (0, 255, 0)  
 thickness = 2  
 lineType = 2  
 cv2.putText(img, label,  
 bottomLeftCornerOfText,  
 font,  
 fontScale,  
 fontColor,  
 thickness,  
 lineType)  
 return img  
  
def detect(model, lm\_list):  
 global label  
 lm\_list = np.array(lm\_list)  
 lm\_list = np.expand\_dims(lm\_list, axis=0)  
 results = model.predict(lm\_list)  
 predicted\_action = np.argmax(results)  
  
 if predicted\_action == 0:  
 label = "HAND RIGHT MOVE"  
 if pygame.mixer.music.get\_busy():  
 pygame.mixer.music.pause()  
 elif predicted\_action == 1:  
 label = "DO NOTHING"  
 elif predicted\_action == 2:  
 label = "CLAPPING"  
 elif predicted\_action == 3:  
 label = "LEFT HAND MOVE"  
 if not pygame.mixer.music.get\_busy():  
 pygame.mixer.music.play()  
 else:  
 label = "DO NOTHING"  
 return label  
  
i = 0  
warmup\_frames = 60  
  
while True:  
 success, img = cap.read()  
 imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
 results = pose.process(imgRGB)  
 i += 1  
 if i > warmup\_frames:  
 print("Start detect....")  
  
 if results.pose\_landmarks:  
 c\_lm = make\_landmark\_timestep(results)  
  
 lm\_list.append(c\_lm)  
 if len(lm\_list) == n\_time\_steps:  
 detect(model, lm\_list)  
 lm\_list = []  
  
 img = draw\_landmark\_on\_image(mpDraw, results, img)  
  
 img = draw\_class\_on\_image(label, img)  
 cv2.imshow("Image", img)  
 if cv2.waitKey(1) == ord('q'):  
 break  
  
cap.release()  
cv2.destroyAllWindows()  
pygame.mixer.quit()

**Kết quả thực hiện :**

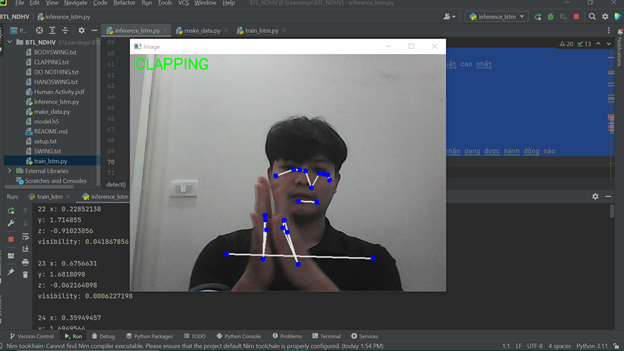
Khi phần mềm chương trình hoạt động, hinh ảnh sẽ được thu lại từ camera, và dựa vào các dữ liệu được train trong model, ứng vỡi mỗi hành động thì sẽ hiển thị ra các label được gắn với các hành động đó.



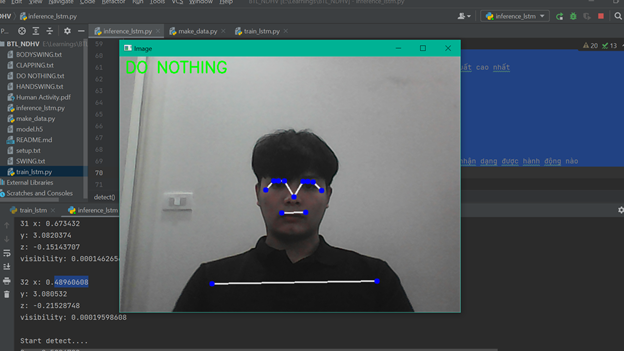
Hình 3.5: Label LEFT HAND MOVE



Hình 3.6: Label HAND RIGHT MOV

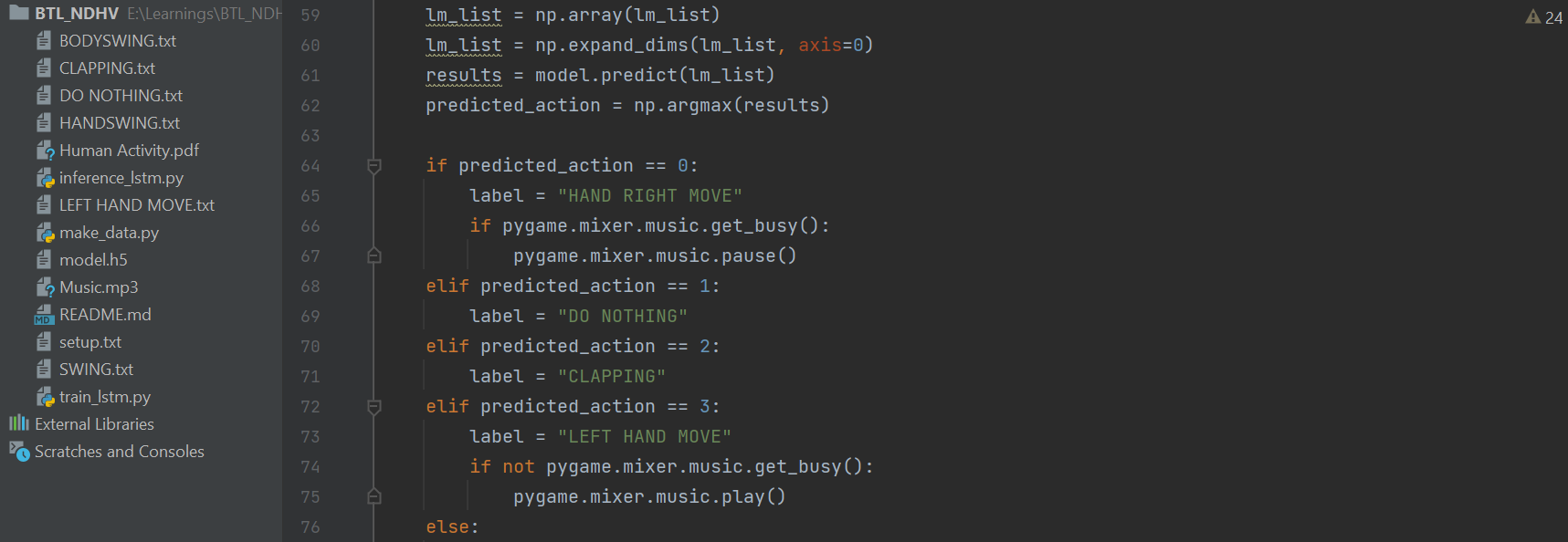


Hình 3.7: Label CLAPPING



Hình 3.8 : Label DO NOTHING

Ngoài việc có thể nhận diện được các hành vi, và dán nhãn cho chúng, phần mềm còn có một ứng dụng về việc bật tắt audio, cụ thể là ở đây là một file nhạc. Nếu hành động phát hiện ra hành động và được gắn nhãn là LEFT HAND MOVE, thì file music.mp3 sẽ được phát, còn ngược lại với trường hợp hành động với nhãn HAND RIGHT MOVE thì file mp3 sẽ được tắt đi.



Hình 3.9: Câu lệnh thực thi việc điều khiển bật tắt Adudio

# CHƯƠNG 4: KIỂM THỬ VÀ ĐÁNH GIÁ PHẦN MỀM

## Kiểm thử chức năng của ứng dụng

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Các chức năng | Bộ dữ liệu | Kết quả mong đợi |
| 1 | Kiểm thử về chức năng vẽ điểm | Click vào giao diện PyCharm file make\_data.py chọn compile | Hiện thị ra các tọa độ x , y z, visibility của các điểm nối trên Video =>Thành công |
| 2 | Kiểm thử về chức năng thu thập dữ liệu | Click vào compile trong make\_data.py khi đủ frames hành động thì ấn vào nút “q” để kết thúc tạo data | Tạo ra các bộ data được gắn label từ trước dưới file csv => Thành công |
| 3 | Kiểm thử chức năng train dữ liệu | Click vào compile trong giao diện file train.py. Các file csv được tạo từ make\_data sẽ được đọc và tạo ra các model | Hiển thị model.h5 phục vụ cho viện nhận diện trong inference => Thành công |
| 4 | Kiểm thử chức năng nhận diện | Click vài compile trong giao diện inference.py . | Hiện các label hành động, hành vi => Thành công |

# KẾT LUẬN VÀ KIẾN NGHỊ

## 1.Kết quả đạt được

Sau quá trình nghiên cứu và tìm hiểu, nhóm có cái nhìn tổng quan về RNN, kiến thức mạng bộ nhớ ngắn hạn (LSTM) - một mô hình kỹ thuậ hiệu quả rong bài toán xử lý chuỗi và hiện đang được các nhà nghiên cứu sử dụng rất nhiều. Với những cải tiến so với RNN thuần, LSTM đã và đang được sử dụng phổ biến. Trên thực tế, cách cài đặt LSTM cũng rất đa dạng và linh hoạt theo bài toán, tuy nhiên vẫn dựa trên LSTM chuẩn như trên.

Đồng thời, nhóm cũng đã rút ra những bài học kinh nghiệm sâu sắc về kỹ năng tìm kiếm, tổng hợp thông tin, kỹ năng quản lý thời gian, quản lý đội nhóm, kỹ năng trình bày, thuyết trình nội dung để chuẩn bị tốt hơn cho những phần nghiên cứu và tìm hiểu trong tương lai sau này.

**2.Hạn chế**

Trong quá trình nghiên cứu, do kinh nghiệm còn có phần thiếu sót. Nhóm tác sẽ cố gắng tìm và nghiên cứu những kiến thức để có thể tiếp cận mô hình một cách toàn diện và hoàn thiện hơn.

**3. Kiến nghị và đề xuất**

Để được đề tài lí thuyết được sâu rộng, nên được áp dụng và phát triển trong các lĩnh vực về y khoa giúp phát hiện ra những biểu hiện của những căn bệnh từ sớm và có pháp đồ điều trị kịp thời bên cạnh đó cũng có thể áp dụng trong các lĩnh vực khác như nghiên cứu tâm lí học, lĩnh vực an ninh, phòng chống tối phạm,..

# TÀI LIỆU THAM KHẢO

[1] Giáo trình nhập môn CSDL - TS. Phạm Văn Hà- Khoa CNTT - Trường ĐH Công nghiệp Hà Nội NXB KHKT - 2021

[2] [1409.1556v6] Very Deep Convolutional Networks for Large-Scale Image Recognition (arxiv.org)

[3] Học sâu – Bách khoa toàn thư Việt Nam: https://bktt.vn/Học\_sâu

[5] Hiện tượng vanishing Gradient. The Vanishing Gradient Problem. The Problem, Its Causes, Its… | by Chi-Feng Wang | Towards Data Science

[6] Bài 11: Object detection với Faster R-CNN | Deep Learning cơ bản (nttuan8.com) [7] Deep face recognition with Keras, Dlib and OpenCV - Martin Krasser's Blog (krasserm.github.io)

[8] ipazc/mtcnn: MTCNN face detection implementation for TensorFlow, as a PIP package. (github.com)

[9] Face detection with OpenCV and deep learning - PyImageSearch

[10] [1805.07836] Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels (arxiv.org)

[11] Deep Residual Learning for Image Recognition 1512.03385.pdf (arxiv.org)