BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN NGÀNH
NGÀNH KHOA HỌC MÁY TÍNH

Đề tài

XÂY DỰNG GAME KẾT HỢP ĐIỀU KHIỂN BẰNG CỬ CHỈ

Sinh viên thực hiện : Lương Đức Huy
Mã số : B2007184
Khóa: 46

Cần Thơ, 05/2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN NGÀNH

NGÀNH KHOA HỌC MÁY TÍNH

Đề tài

XÂY DỰNG GAME KẾT HỢP ĐIỀU KHIỂN BẰNG CỬ CHỈ

Giảng viên hướng dẫn:

PSG. TS. Phạm Nguyên Khang

Sinh viên thực hiện:

Lương Đức Huy

Mã số: B2007184

Khóa: 46

Cần Thơ, 05/2024

NHẬN XÉT CỦA GIẢNG VIÊN

Cần Thơ, ngày tháng 05 năm 2024
(GVHD ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để có thể hoàn thành được bài niên luận này, em xin được bày tỏ lòng biết ơn chân thành và sâu sắc đến Thầy Phạm Nguyên Khang – người đã trực tiếp tận tình hướng dẫn, giúp đỡ em. Trong suốt quá trình thực hiện niên luận, nhờ những sự chỉ bảo và hướng dẫn quý giá đó mà bài niên luận này được hoàn thành một cách tốt nhất.

Em cũng xin gửi lời cảm ơn chân thành đến các Thầy Cô là Giảng viên Đại học Cần Thơ, đặc biệt là các Thầy Cô ở trường Công nghệ thông tin & Truyền thông, những người đã truyền đạt những kiến thức quý báu trong thời gian qua.

Tuy có nhiều cố gắng trong quá trình thực hiện niên luận, nhưng không thể tránh khỏi những sai sót. Em rất mong nhận được sự đóng góp ý kiến quý báu của quý Thầy Cô và các bạn để bài niên luận hoàn thiện hơn.

Cần Thơ, ngày tháng 05 năm 2024

Người viết

Lương Đức Huy

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

DANH MỤC CÁC TỪ VIẾT TẮT

STT	Từ viết tắt	Diễn giải
1	LSTM	Long Short-Term Memory
2	OpenCV	Open Computer Vision Library
3	RNN	Recurrent Neural Network
4	UDP	User Datagram Protocol

ABSTRACT

The goal of this thesis is to create a video game that the user can manipulate with body language. To do this, a Long Short-Term Memory (LSTM) model is used to identify and comprehend actions. human motions in the moment. The project aims to develop a new kind of interactive gaming where users may use their bodies to naturally and fluidly participate with the game.

The method of creating a video game that integrates the LSTM model and gesture control is explained in the essay. Players' gestures are gathered, the data is preprocessed to match the LSTM model, the LSTM model is trained to identify various gesture behaviors, and the model is integrated into the game. Control game components by playing.

The thesis summarizes the findings and assesses the LSTM model's effectiveness in game control and gesture detection in the experimental section. It also addresses the difficulties and restrictions associated with this methodology, as well as prospective avenues for future research to enhance game functionality and performance.

The thesis concludes by outlining some potential uses and applications for gesture control technology and LSTM modeling in the video gaming business as well as other industries including sports and medical. education as well as economics.

The thesis concludes by discussing the value of utilizing deep learning models like LSTM with gesture control technologies in the development of novel, interactive gaming experiences. It also creates a plethora of options for further study and advancement in this area.

TÓM TẮT

Bài luận văn này tập trung vào việc phát triển một trò chơi điện tử mà người chơi có thể điều khiển bằng cử chỉ của cơ thể, và sử dụng một mô hình Long Short-Term Memory (LSTM) để nhận dạng và hiểu các hành vi cử chỉ của con người trong thời gian thực. Mục tiêu của nghiên cứu là tạo ra một trải nghiệm chơi game mới mẻ và tương tác, trong đó người chơi có thể tương tác với trò chơi một cách tự nhiên và linh hoạt bằng cách sử dụng cử chỉ của cơ thể.

Bài luận mô tả quy trình xây dựng trò chơi điện tử kết hợp điều khiển bằng cử chỉ và mô hình LSTM. Điều này bao gồm việc thu thập dữ liệu cử chỉ từ người chơi, tiền xử lý dữ liệu để phù hợp với mô hình LSTM, huấn luyện mô hình LSTM để nhận dạng các hành vi cử chỉ khác nhau và tích hợp mô hình vào trò chơi để điều khiển các yếu tố trong trò chơi.

Trong phần thực nghiệm, bài luận văn trình bày các kết quả và đánh giá hiệu suất của mô hình LSTM trong việc nhận dạng cử chỉ và điều khiển trò chơi. Nó cũng thảo luận về những thách thức và hạn chế của phương pháp này, cùng với các hướng nghiên cứu tiềm năng để cải thiện hiệu suất và tính khả dụng của trò chơi.

Cuối cùng, bài luận văn đề xuất một số ứng dụng và tiềm năng phát triển của công nghệ điều khiển bằng cử chỉ và mô hình LSTM trong ngành công nghiệp trò chơi điện tử, cũng như trong các lĩnh vực khác như thể thao, y tế, và giáo dục.

Kết luận, bài luận văn đề cập đến sự quan trọng của việc sử dụng công nghệ điều khiển bằng cử chỉ và mô hình học sâu như LSTM trong việc tạo ra trải nghiệm chơi game mới mẻ và tương tác. Nó cũng mở ra nhiều cơ hội cho nghiên cứu và phát triển trong tương lai trong lĩnh vực này.

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Trong thời đại của công nghệ thông tin ngày nay, việc tương tác với máy tính và thiết bị điện tử thông qua cử chỉ, hành vi của con người đang trở nên phổ biến và hấp dẫn hơn bao giờ hết. Công nghệ này không chỉ mở ra những trải nghiệm mới mẻ trong lĩnh vực giải trí như trò chơi điện tử, mà còn có tiềm năng ứng dụng rộng rãi trong các lĩnh vực như y tế, giáo dục và công nghiệp.

Trong lĩnh vực trò chơi điện tử, việc sử dụng cử chỉ của người chơi để điều khiển trò chơi không chỉ tạo ra trải nghiệm tương tác sâu sắc hơn, mà còn giúp giải quyết vấn đề về sự chậm trễ và khó khăn khi sử dụng các thiết bị đầu vào truyền thống như bàn phím hoặc chuột.

Tuy nhiên, mặc dù có nhiều nỗ lực nghiên cứu và phát triển trong lĩnh vực này, việc nhận dạng và phản ứng với cử chỉ của con người vẫn đang gặp phải nhiều thách thức. Cần phải xây dựng các hệ thống nhận dạng hành vi của con người có khả năng hoạt động trong thời gian thực và đáng tin cậy, để tạo ra trải nghiệm chơi game mượt mà và tự nhiên.

Trong ngữ cảnh này, việc áp dụng mô hình LSTM (Long Short-Term Memory) trong việc nhận dạng cử chỉ của người chơi trở thành một phương pháp tiềm năng. LSTM là một loại mạng nơ-ron hồi tiếp (RNN) có khả năng xử lý dữ liệu chuỗi và nhớ thông tin trong thời gian dài, phù hợp để xử lý dữ liệu từ cử chỉ liên tục của người chơi trong trò chơi điện tử.

Do đó, bài luận đặt ra mục tiêu xây dựng một mô hình LSTM để nhận dạng hành vi của con người trong thời gian thực, từ đó điều khiển trò chơi điện tử thông qua cử chỉ. Mục tiêu này không chỉ là một bước tiến quan trọng trong việc tạo ra trò chơi điện tử tương tác và thú vị hơn, mà còn mở ra cánh cửa cho ứng dụng rộng rãi của công nghệ điều khiển bằng cử chỉ trong các lĩnh vực khác nhau của cuộc sống.

2. Mục tiêu đề tài

Xây dựng một mô hình LSTM để nhận dạng cử chỉ của con người trong thời gian thực và tích hợp vào trò chơi điện tử để có thể điều khiển trò chơi bằng cử chỉ của con người, từ đó tạo ra trải nghiệm tương tác mới mẻ và đáng chú ý hơn cho người chơi.

3. Đối tượng và phạm vi nghiên cứu

Đối tượng của nghiên cứu này là các hệ thống và công nghệ liên quan đến xây dựng trò chơi điện tử kết hợp với việc điều khiển bằng cử chỉ, đặc biệt là việc sử dụng

mô hình LSTM để nhận dạng hành vi của con người. Phạm vi nghiên cứu tập trung vào các khía cạnh sau:

- **Xây dựng mô hình LSTM**: Tập trung vào việc phát triển và cải tiến mô hình LSTM để nhận dạng hành vi của con người trong thời gian thực, dựa trên dữ liệu cử chỉ được thu thập từ camera.
- **Nhận dạng và phân loại cử chỉ**: Nghiên cứu các phương pháp và thuật toán để nhận dạng và phân loại các cử chỉ của người chơi, bao gồm cử chỉ tay, cử chỉ vận động và các cử chỉ phức tạp khác.
- **Tích hợp vào trò chơi điện tử**: Áp dụng mô hình LSTM vào trò chơi điện tử, thiết lập giao diện để người chơi có thể tương tác và điều khiển trò chơi bằng cử chỉ.
- **Đánh giá hiệu suất**: Đánh giá hiệu suất của hệ thống nhận dạng cử chỉ trong việc điều khiển trò chơi.

Với phạm vi nghiên cứu này, mục tiêu là phát triển một hệ thống điều khiển bằng cử chỉ đáng tin cậy và hiệu quả, tạo ra trải nghiệm chơi game tương tác và hấp dẫn cho người chơi.

4. Phương pháp nghiên cứu

Trong nội dung bài luận này có 2 nội dung chính, đi kèm theo phương pháp nghiên cứu như sau:

1. **Game**: Nghiên cứu và xây dựng một game sử dụng Unity (trong phạm vi bài luận này sẽ lấy một mã nguồn mẫu game miễn phí trên trang chủ Unity và điều chỉnh lại để phù hợp với việc điều khiển bằng cử chỉ so với tài nguyên có sẵn trên laptop cá nhân)
2. **Mô hình nhận dạng hành vi**: Nghiên cứu các mô hình để lấy khung xương của con người => Tạo và thu thập dữ liệu video => Từ dữ liệu video và mô hình lấy khung xương tiến hành xử lý và thu thập dữ liệu dưới dạng csv => Xây dựng và huấn luyện mô hình nhận dạng hành vi => Kiểm thử và cải tiến mô hình => Tích hợp mô hình xây dựng được vào trò chơi => Kiểm thử việc điều khiển trò chơi bằng cử chỉ và cải tiến.

5. Nội dung nghiên cứu:

- Nghiên cứu cách lập trình game với Unity.
- Nghiên cứu các mô hình trích xuất khung xương.
- Nghiên cứu cách xây dựng và huấn luyện một mô hình LSTM.
- Nghiên cứu cách giao tiếp giữa mô hình với trò chơi thông qua tính năng UDP trong Unity.

6. Bố cục luận văn

Phần giới thiệu

Giới thiệu tổng quát về đề tài.

Phần nội dung

Chương 1: Giới thiệu tổng quan.

Chương 2: Xây dựng game.

Chương 3 : Chuẩn bị dữ liệu.

Chương 4: Xây dựng và huấn luyện mô hình.

Chương 5: Tích hợp điều khiển vào trò chơi.

Phần kết luận

Trình bày kết quả đạt được và hướng phát triển hệ thống.

PHẦN NỘI DUNG
CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN

1. Mô tả chi tiết bài toán

Đối với đề tài "Xây dựng game kết hợp điều khiển bằng cử chỉ", bài toán cốt lõi là nhận dạng được cử chỉ của con người và truyền tín hiệu điều khiển đến trò chơi. Mục tiêu của bài toán làm sao nhận dạng được cử chỉ của con người và truyền tín hiệu để điều khiển trò chơi một cách hiệu quả nhất.

2. Hướng giải quyết vấn đề

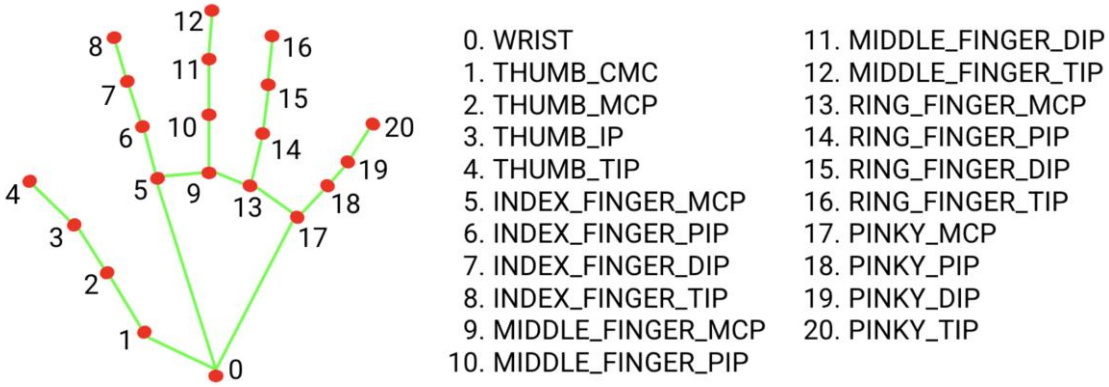
Để giải quyết vấn đề, việc xây dựng một mô hình học sâu để nhận dạng cử chỉ là một giải pháp tối ưu. Việc lựa chọn dạng mô hình học sâu cũng ảnh hưởng đến kết quả cuối cùng. Dữ liệu đầu vào cho việc nhận dạng cử chỉ cũng bao gồm nhiều loại: cảm biến, video, webcam,… Đa số đều là dạng dữ liệu chuỗi thời gian, lựa chọn mô hình LSTM để nhận dạng cử chỉ sẽ hiệu quả hơn so với các mô hình học sâu khác như CNN, RNN. Trong phạm vi bài luận này, dữ liệu đầu vào bắt nguồn từ webcam, vì vậy, vấn đề tiếp theo cần giải quyết là trích xuất đặc trưng dữ liệu từ webcam, một trong các dạng đặc trưng phù hợp với dữ liệu này và đề tài này là các điểm mốc quan trọng trên cơ thể của con người. Hiện tại có rất nhiều giải pháp cho việc phát hiện các điểm mốc quan trọng trên cơ thể như: OpenPose, MoveNet, MediaPipe,… trong phạm vi bài luận, giải pháp MediaPipe của Google được sử dụng vì giải pháp này có hiệu suất tốt hơn các giải pháp khác, ngoài ra MediaPipe cũng được Google tối ưu rất tốt cho CPU và cũng được sử dụng rộng rãi ở thời điểm hiện tại cho các vấn đề liên quan đến thị giác máy tính, máy học, học sâu,…

3. Tổng quan lý thuyết

3.1. Hand landmarks detection [1]

MediaPipe cung cấp một số giải pháp để phát hiện các điểm mốc (landmarks) như: Face landmark detection dùng để phát hiện các điểm mốc trên khuôn mặt, Hand landmarks detection phát hiện các điểm mốc trên bàn tay, Pose landmark detection phát hiện các điểm mốc trên cơ thể người,… Trong phạm vi bài luận này, để dễ dàng cho việc điều khiển trò chơi bằng cử chỉ, sử dụng tay và giải pháp Hand landmarks detection được lựa chọn.

Hand landmarks detection cho phép phát hiện các điểm mốc chính của bàn tay. Đầu vào của mô hình có thể là dữ liệu tĩnh (hình ảnh) hoặc dữ liệu liên tục (video, xử lý từng khung hình trong video), kết quả đầu ra trả về tọa độ 21 điểm chính của bàn tay, mỗi tọa độ là một điểm trong không gian Oxyz đã được chuẩn hóa. 21 điểm tọa độ mô hình có thể phát hiện được thể hiện như **Hình 1**.



Hình 1: 21 điểm mốc chính được phát hiện bởi Hand landmarks detection [1]

Hand landmarks detection có thể phát hiện cả 2 bàn tay (tay trái/phải) cùng lúc, giải pháp này cũng cung cấp cho các môi trường phát triển bao gồm Android, Python và Web.

3.2. Long short-term memory (LSTM)

Trước khi đi vào LSTM, cần đi qua các vấn đề liên quan đến hạn chế của mạng truyền thẳng và Recurrent Neural Network (RNN) để đưa ra nhận định về việc lựa chọn LSTM cho đề tài.

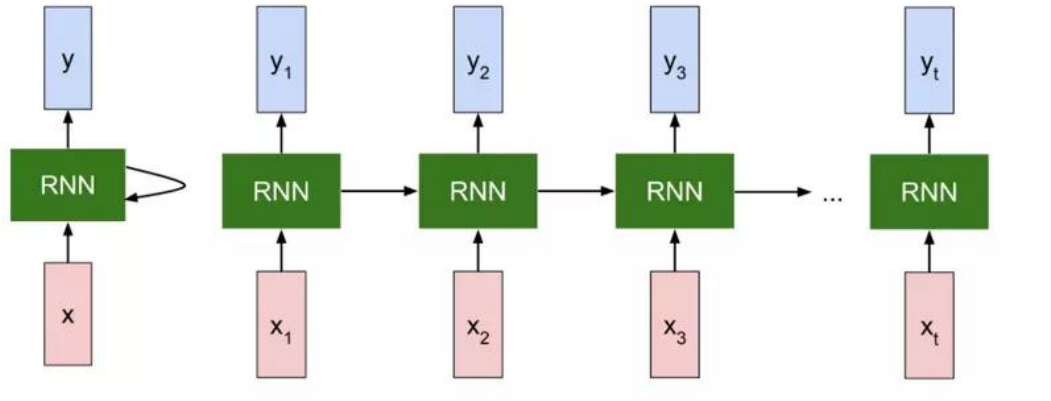
3.2.1. Hạn chế của mạng truyền thẳng

Đối với mạng neural thông thường, tất cả dữ liệu được đưa vào cùng một lúc. Nếu dữ liệu mang ý nghĩa trình tự, mạng truyền thẳng sẽ gặp phải vấn đề nếu trình tự dữ liệu bị thay đổi, kết quả bị ảnh hưởng theo.

Vấn đề này dễ dàng nhìn thấy qua các dữ liệu dạng văn bản. Ví dụ, với hai câu "Bạn làm bài tập chưa" và "Bạn chưa làm bài tập". Nếu tách thành từng từ, chúng ta được bộ từ vựng bao gồm: Bạn, làm, bài, tập, chưa. Thực hiện one hot encoding và cho vào một mạng truyền thẳng, sẽ không có sự phân biệt nào giữa hai câu trên. Việc dữ liệu bị đảo thứ tự sẽ làm sai lệch, ảnh hưởng kết quả dự đoán là một vấn đề hạn chế trong mạng truyền thẳng. Giải pháp để giải quyết vấn đề này là chúng ta cần một mạng có thể xử lý tuần tự, và đó cũng là lý do RNN ra đời.

3.2.2. Recurrent Neural Network (RNN) [2]

Để dễ hình dung, cách thức hoạt động của một mô hình RNN được thể hiện như **Hình 2**.



Hình 2: Mô hình RNN [2]

Trong một mạng RNN sẽ bao gồm nhiều block RNN, cách thức mạng hoạt động là mỗi block RNN sẽ lấy thông tin từ các block trước kết hợp input hiện tại để làm đầu vào.

Mỗi x đại diện cho dữ liệu vào lần lượt được chia theo timestep, ví dụ với timestep là 5, lấy 5 từ liên tiếp trong một câu để đưa vào mạng, x_t đại diện cho timestep thứ t và y_t là output của một step.

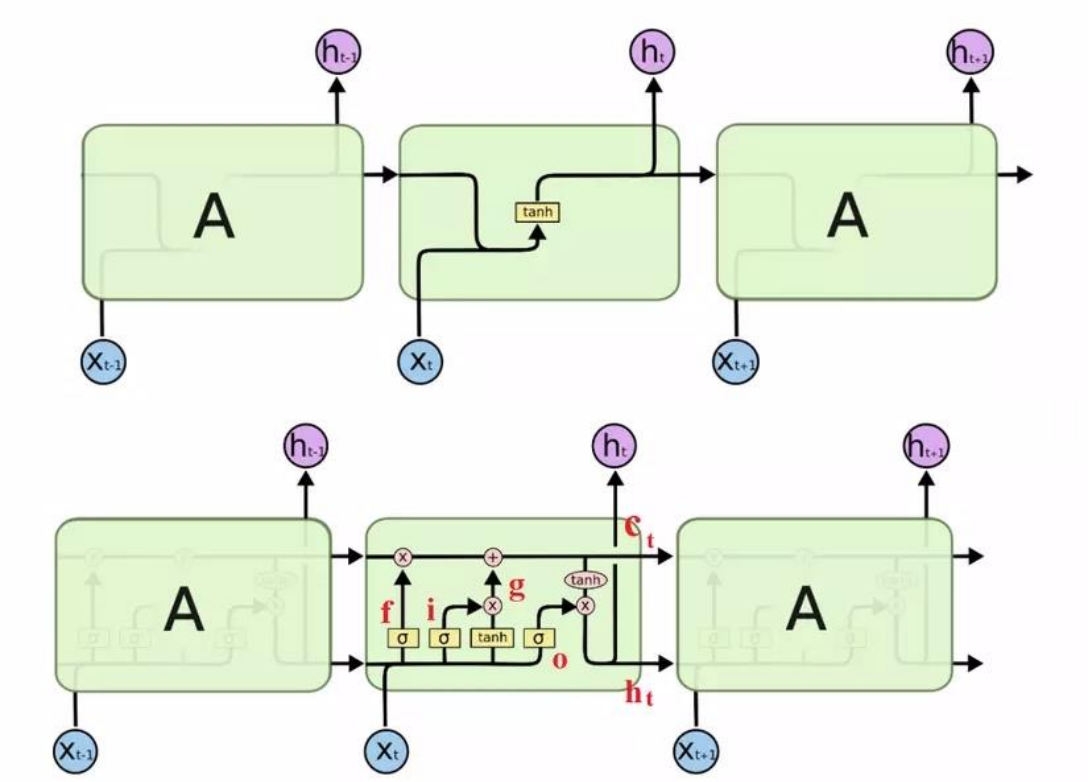
Mỗi một block RNN (khối xanh lá) là một **hidden state** h_t (một số tài liệu ký hiệu s_t). h_t là sự tổng hợp thông tin của hidden state trước (h_{t-1}) với input tại thời điểm timestep t (tức x_t).

Tại từng timestep, một block RNN có 2 output h_t và y_t. Trong đó, h_t tổng hợp thông tin các state trước để tiếp tục truyền trong mạng, còn y_t là output của từng timestep một.

Để có thể hoạt động theo cách thức này, RNN sử dụng thuật toán lan truyền ngược theo thời gian (Backpropagation Through Time - BTT), lan truyền ngược tại mỗi thời điểm t. Tuy nhiên, việc áp dụng lan truyền ngược theo thời gian dẫn tới 2 vấn đề lớn: vanishing gradient (đạo hàm bị triệt tiêu) hoặc exploding gradient (bùng nổ đạo hàm). Từ đây dẫn tới biến thể LSTM dùng để chống lại các vấn đề RNN gặp phải.

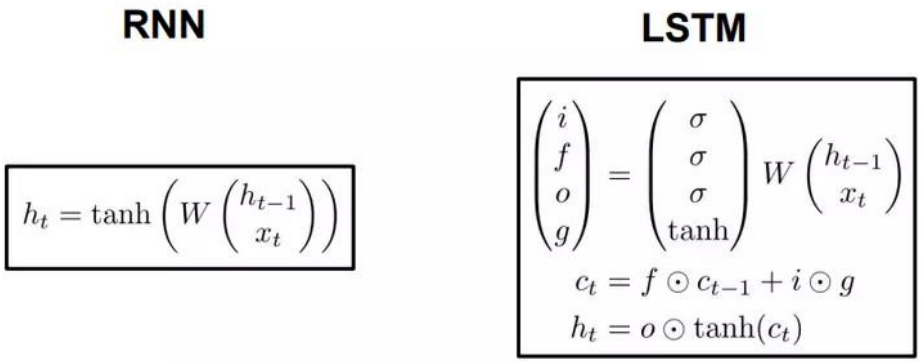
3.2.3. Long short-term memory (LSTM)

Ý tưởng của LSTM cũng tương tự như RNN, tuy nhiên việc tính toán trong mỗi hidden state của LSTM khác hơn so với RNN, trong LSTM mỗi hidden state thêm một số phép tính toán. Sự khác nhau giữa 2 mô hình RNN và LSTM được thể hiện như **Hình 3**.



Hình 3: So sánh kiến trúc mô hình RNN (trên) và LSTM (dưới) [2]

Việc tính toán trong hidden state của RNN và LSTM được thể hiện như **Hình 4**.



Hình 4: Tính toán trong mỗi hidden state giữa RNN và LSTM [2]

Đối mới mô hình LSTM, có thêm các i, f, o, g có công thức gần giống nhau, chỉ khác ma trận tham số và ma trận này sẽ quyết định chức năng cho từng cổng.

- Input gate i – cổng vào: cổng này sẽ quyết định bao nhiêu lượng thông tin đầu vào sẽ ảnh hưởng đến state mới.
- Forget gate f – cổng quên: cổng này quyết định bỏ đi bao nhiêu lượng thông tin đến từ trạng thái trước đó.

- Output gate o – cổng ra: điều chỉnh lượng thông tin có thể ra ngoài y_t và lượng thông tin truyền tới trạng thái tiếp theo
- g là một trạng thái ẩn được tính dựa vào x_t và h_{t-1} dùng để cập nhật trạng thái mới.
- c_t là bộ nhớ trong của mỗi hidden state, nó là sự tổng hợp của bộ nhớ trước c_{t-1} đã được lọc thông qua f, cộng với g được lọc bởi cổng i. c_t sẽ mang thông tin nào quan trọng truyền đi xa hơn và sẽ được dùng khi cần, đây được gọi là **long term memory**. Sau khi có c_t, đưa qua cổng ra để lọc thông tin một lần nữa, thu được trạng thái mới h_t.

Chính nhờ cơ chế các cổng về việc điều chỉnh lượng thông tin, LSTM đã phần nào giải quyết các vấn đề về vanishing gradient so với RNN.

3.3. Unity

Unity là một nền tảng phát triển trò chơi và ứng dụng đa nền tảng mạnh mẽ, được phát triển bởi Unity Technologies. Nó được sử dụng rộng rãi trong ngành công nghiệp trò chơi, giáo dục, giải trí tương tác, kiến trúc 3D, và nhiều lĩnh vực khác. Dưới đây là một tổng quan về lý thuyết cơ bản và các khái niệm chính của Unity:

1. Game Objects và Components:
   - Game Objects: Đại diện cho các thực thể trong trò chơi, như nhân vật, vật phẩm, môi trường,…
   - Components: Là các phần cụ thể của Game Objects, như hình dạng, vị trí, vật lý, hành vi,…
2. Scene:
   - Scene: Là một không gian môi trường chứa các Game Objects và Components, mô tả một cảnh hoặc màn chơi cụ thể.
3. Scripting:
   - Scripting: Sử dụng ngôn ngữ lập trình như C# hoặc JavaScript để tạo ra hành vi và tương tác của trò chơi.
   - MonoBehaviour: Là lớp cơ sở cho các script trong Unity, cung cấp các phương thức như Start(), Update(), FixedUpdate(),…
4. Physics:
   - Physics Engine: Unity tích hợp một bộ xử lý vật lý cho việc xử lý va chạm, vật lý và động lực trong trò chơi.
   - Rigidbody: Component để xử lý vật lý cho Game Objects, như làm cho chúng rơi tự do hoặc phản ứng với lực.
5. Rendering:

- Rendering Engine: Unity sử dụng một hệ thống rendering mạnh mẽ để tạo ra đồ họa 2D và 3D.
- Shaders: Định nghĩa cách mà Game Objects được hiển thị và tương tác với ánh sáng và màu sắc.

6. Audio:
   - Audio Engine: Hỗ trợ âm thanh và âm nhạc trong trò chơi thông qua các Component như AudioSource và AudioListener.
7. Cross-platform Development:
   - Cross-platform: Unity cho phép phát triển một lần và triển khai trên nhiều nền tảng như iOS, Android, PC, Console, Web,…
8. Asset Pipeline:
   - Asset Pipeline: Quản lý và quản lý tài nguyên như mô hình 3D, texture, âm thanh,…
   - Asset Store: Cung cấp một thư viện lớn các tài nguyên và công cụ từ cộng đồng và các nhà phát triển chuyên nghiệp.
9. Networking:
   - Networking: Hỗ trợ kết nối mạng và đa người chơi thông qua các API như Unity Networking hoặc các giải pháp bên thứ ba.
10. AI:
    - Artificial Intelligence: Unity cung cấp các công cụ và thư viện cho phát triển trí tuệ nhân tạo để tạo ra hành vi của nhân vật và hệ thống thông minh.

Unity cung cấp một môi trường linh hoạt và mạnh mẽ cho việc phát triển trò chơi và ứng dụng đa nền tảng, từ các dự án nhỏ đến các dự án lớn và phức tạp. Điều này giúp cho nhà phát triển tập trung vào việc sáng tạo nội dung và trải nghiệm người dùng mà không cần lo lắng về các vấn đề kỹ thuật cơ bản.

Ngoài ra Unity còn cung cấp một nền tảng học tập trực tuyến Unity Learn, được thiết kế để cung cấp tài nguyên học tập và đào tạo chất lượng cao về việc sử dụng Unity để phát triển trò chơi và ứng dụng đa nền tảng. Trên Unity Learn, chúng ta có thể truy cập các khóa học, hướng dẫn, dự án thực hành, và tài liệu hữu ích, từ cơ bản đến nâng cao, giúp bạn tiếp cận kiến thức và kỹ năng cần thiết để trở thành một nhà phát triển trò chơi thành công trên nền tảng Unity.

Unity còn cung cấp một tính năng UDP (User Datagram Protocol), một tính năng cho phép truyền tin mạng không đồng bộ, giúp tối ưu hiệu suất và phản ứng trong trò chơi đa người chơi. Đặc biệt thích hợp cho các ứng dụng yêu cầu phản ứng nhanh và hiệu suất cao, UDP trong Unity dễ dàng tích hợp và hỗ trợ đa nền tảng, là thứ giúp truyền tín hiệu điều khiển trong phạm vi đề tài này.

CHƯƠNG 2: XÂY DỰNG TRÒ CHƠI

Để thuận tiện cho việc đánh giá độ hiệu quả của mô hình LSTM, thể loại game được trong phạm vi bài luận này là Endless runner [3]. Khái quát, endless runner hay infinite runner là dạng game mà nhân vật do người chơi hóa thân sẽ chạy trong khoảng thời gian và không gian vô hạn, trong quá trình chạy người chơi phải điều khiển nhân vật để tránh các chướng ngại vật và ăn các vật phẩm bổ trợ.

Vì cốt lõi của bài luận này là xây dựng được một mô hình máy học có khả năng nhận dạng được cử chỉ/hành vi con người nên sẽ không đi quá sâu vào việc xây dựng game từ những bước đầu, mà sử dụng một mẫu game mã nguồn mở có sẵn và tùy chỉnh lại để phù hợp cho việc điều khiển bằng cử chỉ.

Mẫu game được sử dụng là Endless Runner – Sample Game [4], đây là một mẫu game mã nguồn mở thuộc thể loại endless runner do Unity cung cấp được công bố trên Unity Asset Store. Thông tin về mẫu game được thể hiện thông qua **Bảng 1**.

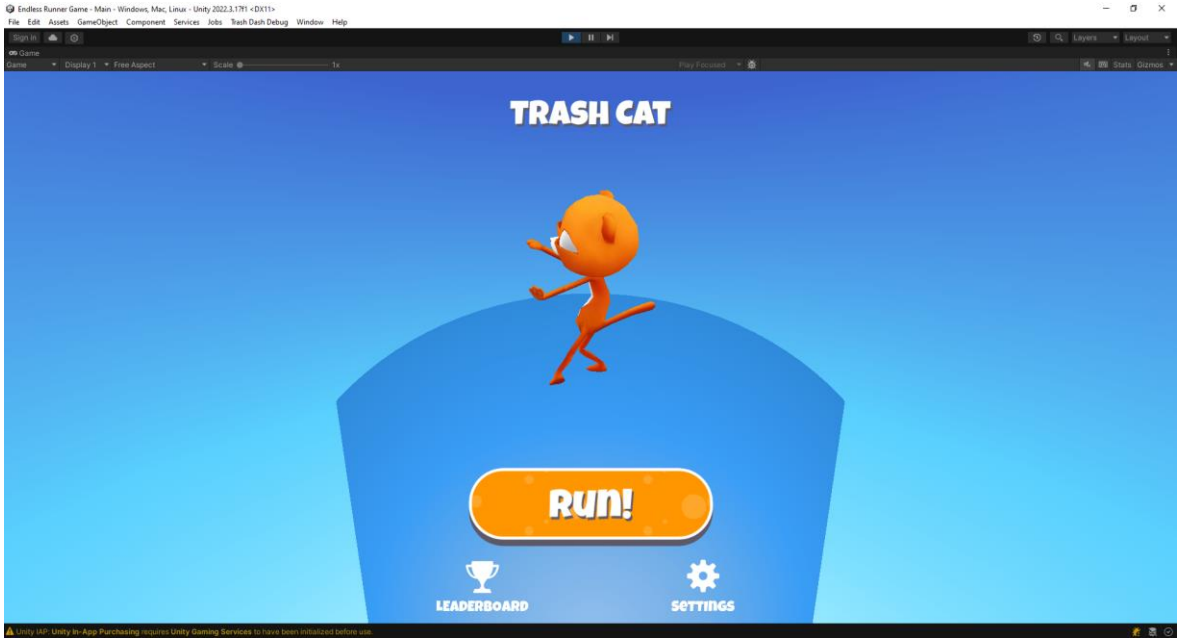
Bảng 1: Thông tin về Endless Runner - Sample Game

Tên	Endless Runner – Sample Game
Thể loại	Endless runner
Nội dung	Hóa thân vào nhân vật một chú mèo, chú mèo sẽ chạy trên một con đường dài vô hạn, trên được sẽ được bố trí các chướng ngại vật và các vật phẩm, nhiệm vụ của người chơi là điều khiển chú mèo tránh các chướng ngại vật và ăn những vật phẩm
Cách chơi	Dùng các phím mũi tên lên, xuống, trái, phải để điều khiển

Để phù hợp và tăng độ hiệu quả cho việc điều khiển game bằng cử chỉ, game được điều chỉnh một số thứ như số lượng chướng ngại vật, số lượng mạng trong một màn chơi, các thông số về chiều cao, độ xa khi nhân vật nhảy, trượt,…

Ngoài ra, để chuẩn bị cho việc nhận tín hiệu điều khiển, tạo thêm một script cho game để thiết lập việc nhận tín hiệu bằng giao thức UDP thông qua địa chỉ IP 127.0.0.1 và port 5052, sử dụng namespace **System.Net.Sockets**.

Game sau khi được điều chỉnh và hoàn thiện sẽ có giao diện được thể hiện thông qua **Hình 5**, **Hình 6**.



Hình 5: Giao diện game ở màn hình chính

Hình 6: Giao diện game lúc đang chơi

CHƯƠNG 3: CHUẨN BỊ DỮ LIỆU

Quá trình chuẩn bị dữ liệu là một phần quan trọng trong việc xây dựng và huấn luyện mô hình máy học. Trong phạm vi đề tài này, để tiện lợi cho việc quy định cử chỉ và gán nhãn dữ liệu, dữ liệu sẽ được thu thập bằng cách tự quay video.

1. Thu thập dữ liệu

Trước khi thu thập dữ liệu cần quy định những cử chỉ/hành động dùng để điều khiển game. Việc điều khiển game sẽ dựa vào cử chỉ tay, quy định các cử chỉ được trình bày như Bảng 2.

Bảng 2: Quy định cử chỉ điều khiển game

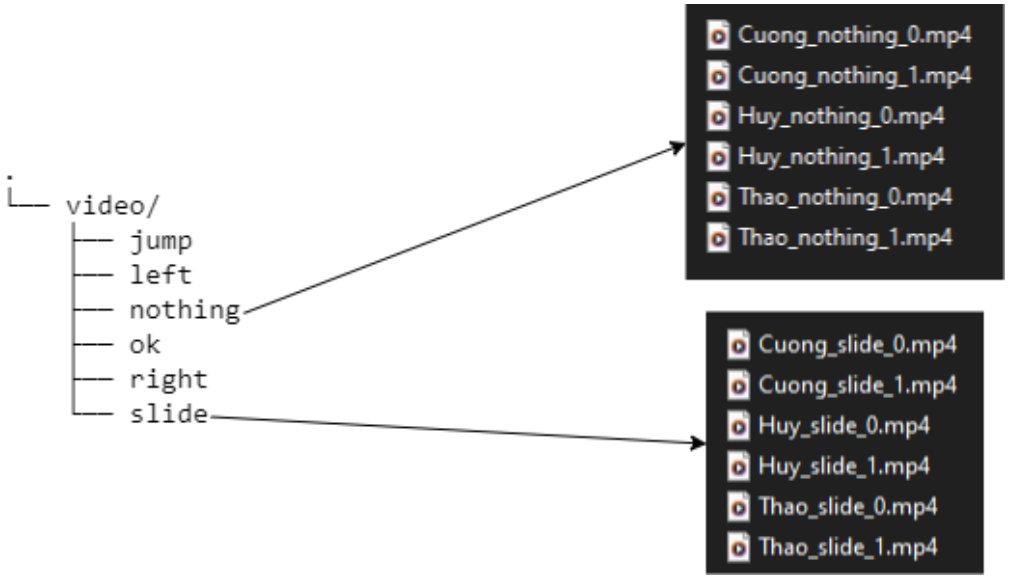
Cử chỉ/hành động	Phím tương đương	Nhân vật
Không làm gì	Không có phím tương đương	Không tác động đến nhân vật
Vẫy tay lên trên	Mũi tên đi lên	Nhảy lên cao
Vẫy tay xuống dưới	Mũi tên đi xuống	Trượt trên đường
Vẫy tay qua trái	Mũi tên qua trái	Di chuyển qua trái
Vẫy tay qua phải	Mũi tên qua phải	Di chuyển qua phải
Đưa tay ký hiệu OK	Nhấn chuột trái	Dùng cho việc xác nhận vào trận khi đang ở màn hình chính của game

Sau khi quy định các cử chỉ/hành động, tiến hành thu thập dữ liệu bằng cách quay video từ 2-3 người và cách quay như sau:

- Mỗi người sẽ quay 12 video.
- Mỗi video quay việc thực hiện 1 cử chỉ/hành động, người quay sẽ liên tục lặp đi lặp lại cử chỉ/hành động đó. Mỗi hành động sẽ quay với từng tay (trái/phải), tức 12 video sẽ bao gồm 6 hành động theo **Bảng 2**, trong đó mỗi hành động bao gồm 2 video nhưng thực hiện khác tay.
- Mỗi video có độ dài khoảng 30-45s (với thiết lập 30fps).
- Khi quay lựa chọn góc có vùng ánh sáng tốt, rõ ràng, thuận lợi cho việc phát hiện các điểm mốc khi đi vào quá trình tiền xử lý dữ liệu.
- Khi việc quay video hoàn thành, thực hiện việc tổng hợp thành một dataset dạng video. bằng cách phân chia các video có cùng kiểu cử chỉ/hành động vào cùng một thư mục được đặt tên theo tên của hành động. Quy định tên hành động như sau: Không làm gì = nothing, vẫy tay lên

trên = jump, vẫy tay xuống dưới = slide, vẫy tay qua trái = left, vẫy tay qua phải = right, đưa tay ký hiệu OK = ok.

Dữ liệu sau khi phân chia có cấu trúc như **Hình 7**.



Hình 7: Cấu trúc dataset dạng video

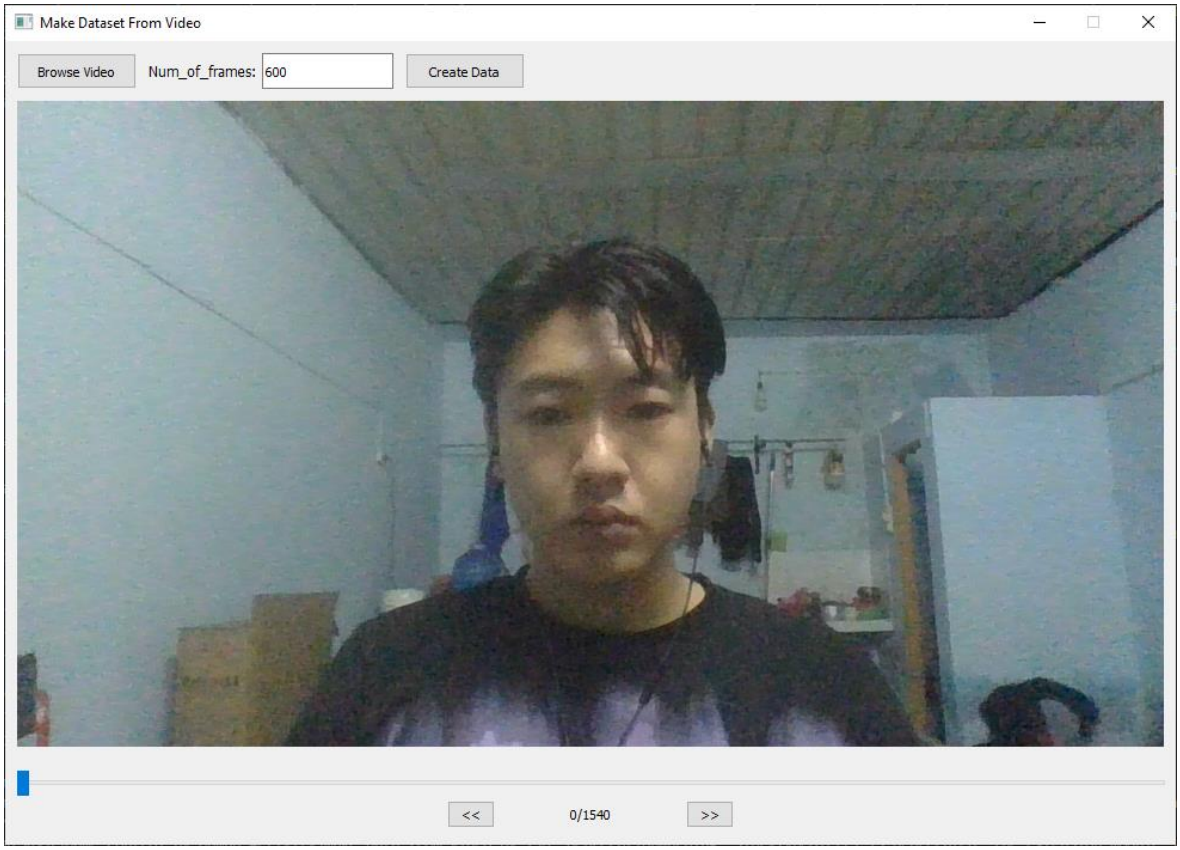
2. Tiền xử lý dữ liệu

Đối với các mô hình máy học, dữ liệu được đưa vào phải là dữ liệu ở dạng số. Vì vậy, với tập dữ liệu gốc là video không thể trực tiếp đưa vào mô hình, cần phải xử lý dữ liệu một cách phù hợp để có thể đưa vào mô hình.

Với mô hình LSTM, là mô hình dùng cho dữ liệu dạng chuỗi thời gian. Vậy với dữ liệu dạng video, dữ liệu sẽ được đưa vào mô hình dưới dạng từng frame liên tục trong một video (số lượng frame tương ứng với số lượng timestep của mô hình). Tức video sẽ được phân tích thành từng hình ảnh liên tục nhau (mỗi ảnh tương ứng với một frame).

Để mô hình có thể học và dự đoán được cử chỉ/hành vi của con người, đối với mỗi ảnh/frame cần trích xuất đặc trưng phù hợp để mô hình hiệu quả, đặc trưng này sẽ quy về dạng số để có thể đưa vào mô hình. Trong phạm vi bài luận này, việc điều khiển game phụ thuộc vào cử chỉ của bàn tay, nên đặc trưng phù hợp nhất sẽ là các điểm mốc trên bàn tay.

Để việc xử lý dữ liệu trở nên hiệu quả, thực hiện việc xây dựng một công cụ có giao diện, công cụ được xây dựng bằng ngôn ngữ Python và sử dụng thư viện PyQt5 để tạo giao diện. Công cụ cho phép chọn video từ thư mục và hiển thị video lên giao diện, công cụ sử dụng Hand landmarks detection [1] để phát hiện các điểm mốc trên bàn tay. Giao diện công cụ được thể hiện như **Hình 8**.

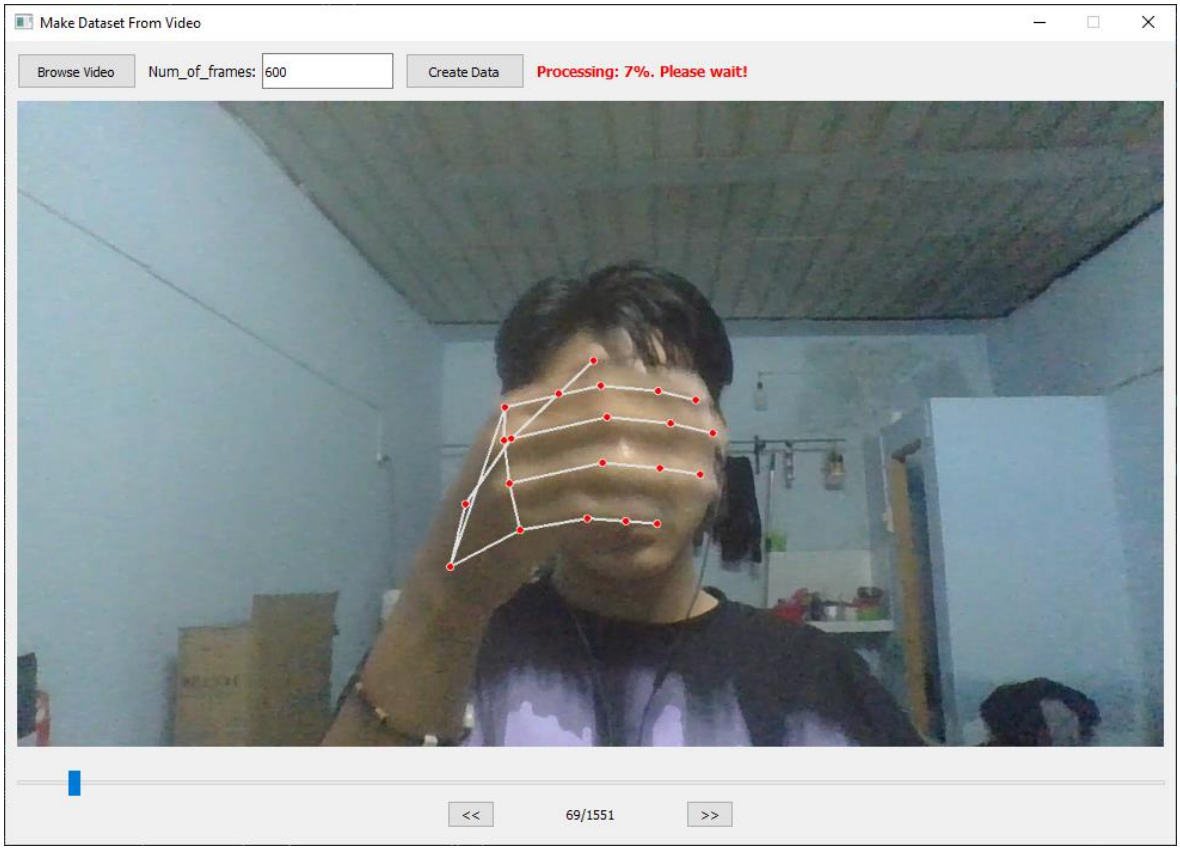


Hình 8: Công cụ tạo dữ liệu từ video

Công cụ gồm các thành phần:

- Nút Browse Video: chọn video từ thư mục và hiển thị lên giao diện
- Nút Create Data: Tạo dữ liệu từ video
- Ô input Num_of_frames: Nhập số lượng frame sẽ được lấy để tạo dữ liệu
- Thanh slider: dùng để kéo đến frame cần đánh dấu điểm bắt đầu tạo dữ liệu

Cách công cụ hoạt động được thể hiện như **Hình 9**. Sau khi chọn video và hiển thị lên giao diện, kéo thanh slider đến vị trí frame bắt đầu cử chỉ/hành động, nhập số lượng frame sẽ để tạo dữ liệu. Nhấn nút Create Data, công cụ sẽ duyệt qua từng frame bắt đầu từ vị trí frame đã đánh dấu, duyện đến khi đủ số lượng frame đã nhập trong ô Num_of_frames. Với từng frame được duyệt qua, công cụ sẽ chuyển về dạng màu RGB và đưa vào mô hình Hand landmarks detection [1], mô hình trả về tọa độ 21 điểm mốc trên bàn tay, mỗi điểm bao gồm 3 giá trị x, y, z là tọa độ đã được chuẩn hóa so với kích thước của frame đầu vào. Các dữ liệu được lưu trữ để tạo dữ liệu.



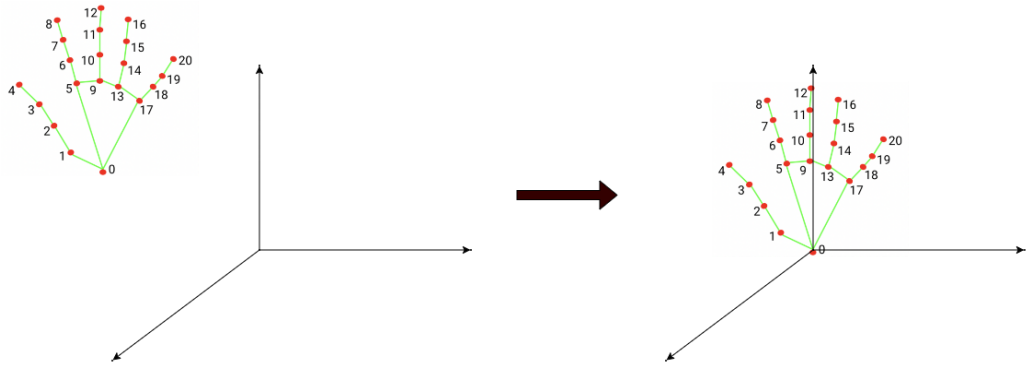
Hình 9: Cách công cụ hoạt động

Tuy nhiên, nếu trực tiếp lưu trữ các điểm tọa độ này để làm dữ liệu sẽ xảy ra các vấn đề sai lệch cho mô hình. Nguyên nhân chính gây ra vấn đề này bao gồm 2 nguyên nhân sau:

- **Tọa độ đã được chuẩn hóa so với kích thước frame**: Các giải pháp phát hiện các điểm mốc do MediaPipe cung cấp, cũng cung cấp kèm các chức năng vẽ khung xương cơ thể dựa trên các điểm mốc, vì vậy tọa độ các điểm mốc được phát hiện Mediapipe sẽ chuẩn hóa so với kích thước frame để có thể vẽ khung xương chính xác khi hiển thị frame lên màn hình. Nếu giữ nguyên tọa độ này để làm dữ liệu cho mô hình sẽ gây ra hiện tượng sai lệch nhãn dự đoán nếu vị trí tay ở frame đầu vào không giống với vị trí của dữ liệu dùng để huấn luyện. Và vấn đề quan tâm ở đây là tư thế của bàn tay tạo ra cử chỉ chứ không phải vị trí của bàn tay trên frame.
- **Tỉ lệ cơ thể giữa mỗi người khác nhau**: Mỗi người có một tỉ lệ cơ thể khác nhau, có người bàn tay to, có người bàn tay nhỏ, khoảng cách giữa các điểm mốc trên bàn tay giữa mỗi người cũng sẽ khác nhau. Điều này cũng sẽ ảnh hưởng tới việc dự đoán nhãn trong thực tế cho nhiều người.

Vì vậy, trước khi lưu tọa độ các điểm mốc lại thành dữ liệu cần xử lý qua 2 bước sau:

1. Chuẩn hóa tọa độ các điểm về trục tọa độ Oxyz, trong đó tọa độ của điểm thứ 0 sẽ được chuẩn hóa về gốc tọa độ Oxyz, tức giá trị x, y, z của điểm thứ 0 là (0,0,0). Các điểm còn lại sẽ bằng tọa độ chính nó trừ đi tọa độ ban đầu của điểm thứ 0 (cần lưu trữ giá trị x, y, z của điểm thứ 0 trước khi chuẩn hóa về gốc tọa độ). Việc chuẩn hóa như này mô hình sẽ không bị ảnh hưởng bởi tọa độ đã bị chuẩn hóa bởi Mediapipe. Việc chuẩn hóa được thể hiện như **Hình 10**.



Hình 10: Chuẩn hóa tọa độ các điểm mốc về gốc tọa độ

2. Chuẩn hóa khoảng cách giữa các điểm tọa độ về một tỉ lệ cố định, thực hiện theo các bước sau:
   a. Tính tọa độ điểm trung tâm giữa 21 điểm mốc:

$$center = \left(\frac{1}{21}\sum_{i=0}^{20} x^i, \frac{1}{21}\sum_{i=0}^{20} y^i, \frac{1}{21}\sum_{i=0}^{20} y^i\right)$$

   b. Đối với mỗi điểm trong 21 điểm mốc, tiến hành tính khoảng cách so với điểm trung tâm sử dụng công thức Euclid, sau khi có khoảng cách tính hệ số tỉ lệ chuẩn hóa:

$$distance^i = \sqrt{(x^i - center_x)^2 + (y^i - center_y)^2 + (z^i - center_x)^2}$$

$$scale\_factor^i = 1.0/distance^i$$

   c. Cuối cùng, chuẩn hóa tỉ lệ các điểm trong 21 điểm mốc bằng cách nhân tọa độ điểm với hệ số tỉ lệ:

$$point^i = (x^i * scale\_factor^i, y^i * scale\_factor^i, z^i * scale\_factor^i$$

Công cụ sẽ duyệt từ frame được đánh dấu với số lượng frame bằng số được nhập trong ô Num_of_frames. Đối với mỗi frame duyệt qua sẽ trích xuất 21 điểm mốc trên bàn tay và đưa qua 2 bước chuẩn hóa vừa nêu trên và lưu trữ lại thành mỗi danh sách, sau khi duyệt xong ghi danh sách thành một file csv để tạo thành tập dữ liệu mới. Dữ liệu mới sẽ có dạng như **Hình 11**.

Hình 11: Dữ liệu sau khi xử lý

Dữ liệu dưới dạng file csv, trong đó có 63 cột là tọa độ x, y, z của 21 điểm (21*3), mỗi hàng tương ứng với dữ liệu đã trích xuất từ một frame trong quá trình xử lý. Số lượng hàng bằng với số được nhập ở ô Num_of_frames.

Với mỗi video từ dữ liệu gốc cho ra một file csv, tổng hợp và phân chia các file csv vào các folder để được tập dữ liệu sau xử lý. Tập dữ liệu mới có cấu trúc như **Hình 12**.

Hình 12: Cấu trúc tập dữ liệu sau khi xử lý

CHƯƠNG 4: XÂY DỰNG VÀ HUẤN LUYỆN MÔ HÌNH

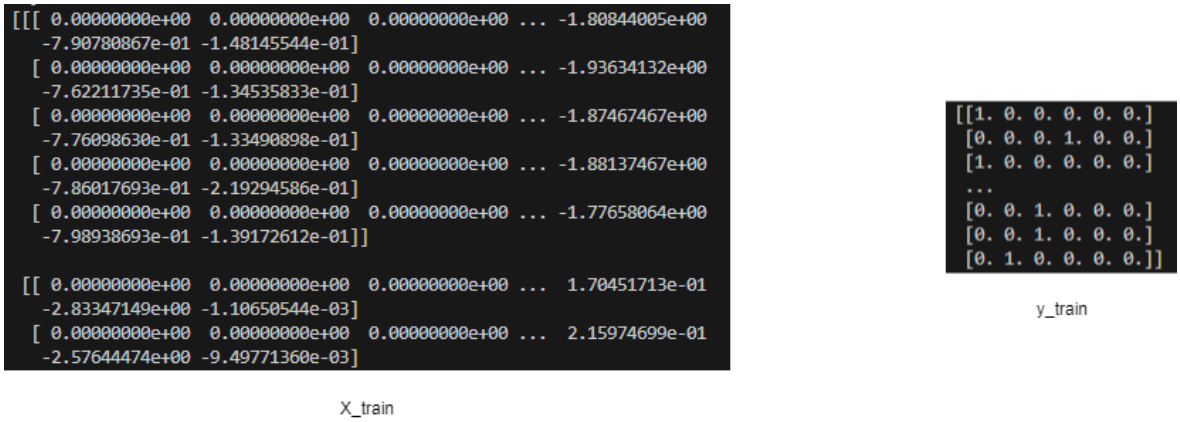
1. Đọc dữ liệu

Tập dữ liệu sau khi tiền xử lý là các file dưới dạng csv, trong đó mỗi hàng là tọa độ của 21 điểm mốc trên bàn tay. Vì vậy, trước khi xây dựng mô hình LSTM, cần đọc tập dữ liệu sao cho phù hợp với kiểu đầu vào của một mô hình LSTM là dữ liệu chuỗi thời gian.

Tiến hành đọc dữ liệu theo các bước sau:

- Chuyển hóa các nhãn về dạng số, ví dụ với jump chuyển thành 0, left thành 1, nothing thành 2,…
- Tạo 2 danh sách để lưu thuộc tính và nhãn, lần lượt tạm gọi là X, y. Với từng nhãn, đọc tất cả các file dữ liệu của nhãn đó. Với mỗi file được đọc vào, duyệt lần lượt qua từng dòng dữ liệu trong file. Mỗi lần duyệt sẽ lấy n dòng dữ liệu tiếp theo tính từ dòng hiện tại để đưa vào X (n là số lượng timesteps), mỗi lần thêm n dữ liệu vào X song song thêm 1 số để gán nhãn cho n dữ liệu được đưa vào đó vào y, ví dụ nếu đang duyệt jump thì thêm 0 vào y.
- Sau khi đã có X và y, tiến hành chia tập dữ liệu làm 2 phần huấn luyện và kiểm tra. Trong đó 80% của dữ liệu làm tập huấn luyện và 20% còn lại làm tập kiểm thử. Cuối cùng, vì tập dữ liệu có nhiều hơn 2 nhãn nên cần chuyển đổi mảng các nhãn thành dạng one-hot encoding.

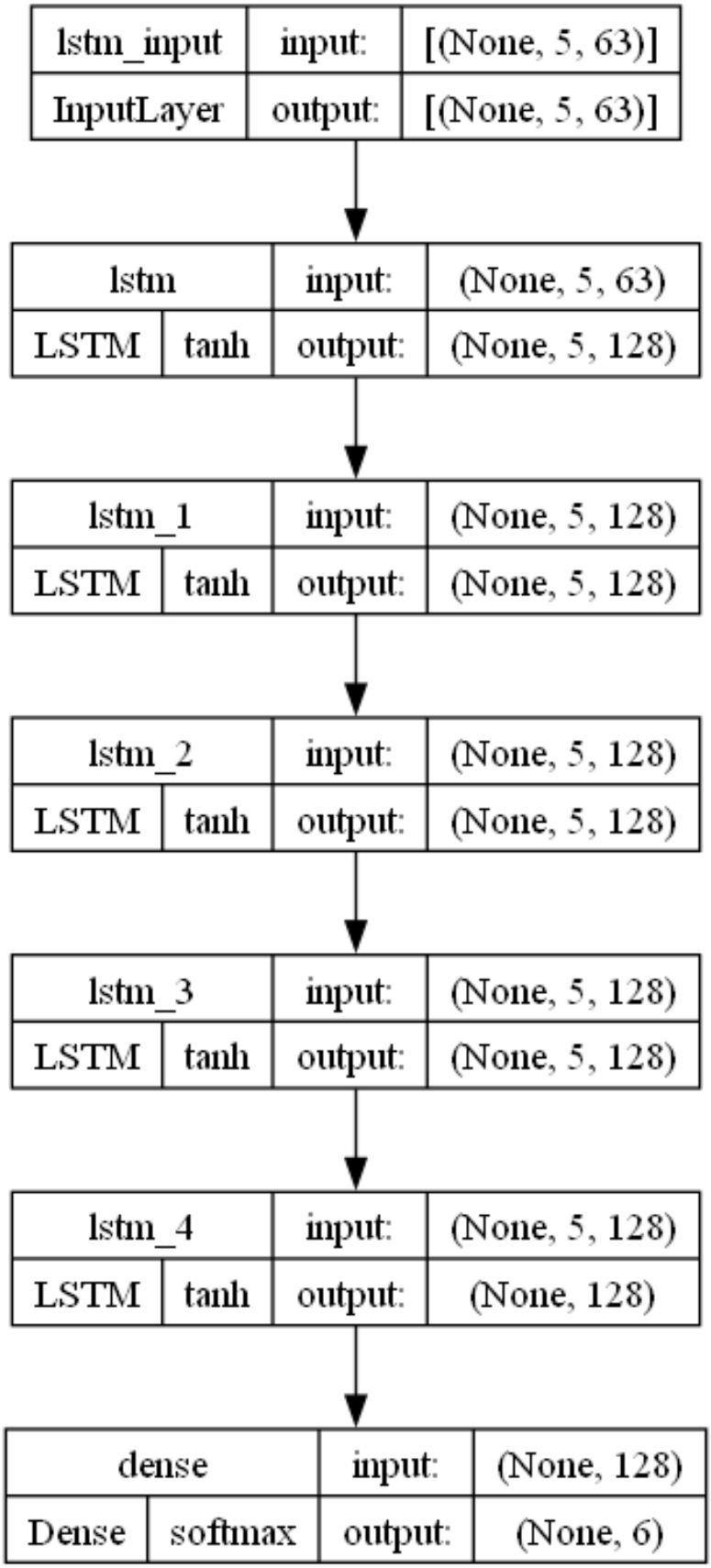
Dữ liệu sau khi đọc vào được thể hiện như **Hình 13**.



Hình 13: Dữ liệu được đọc vào trước khi đưa vào huấn luyện mô hình

2. Xây dựng mô hình

Kiến trúc mô hình LSTM nhận dạng cử chỉ/hành vi con người được xây dựng như **Hình 14**.



Hình 14: Kiến trúc mô hình LSTM nhận dạng cử chỉ/hành vi con người

Mô tả kiến trúc mô hình trên như sau:

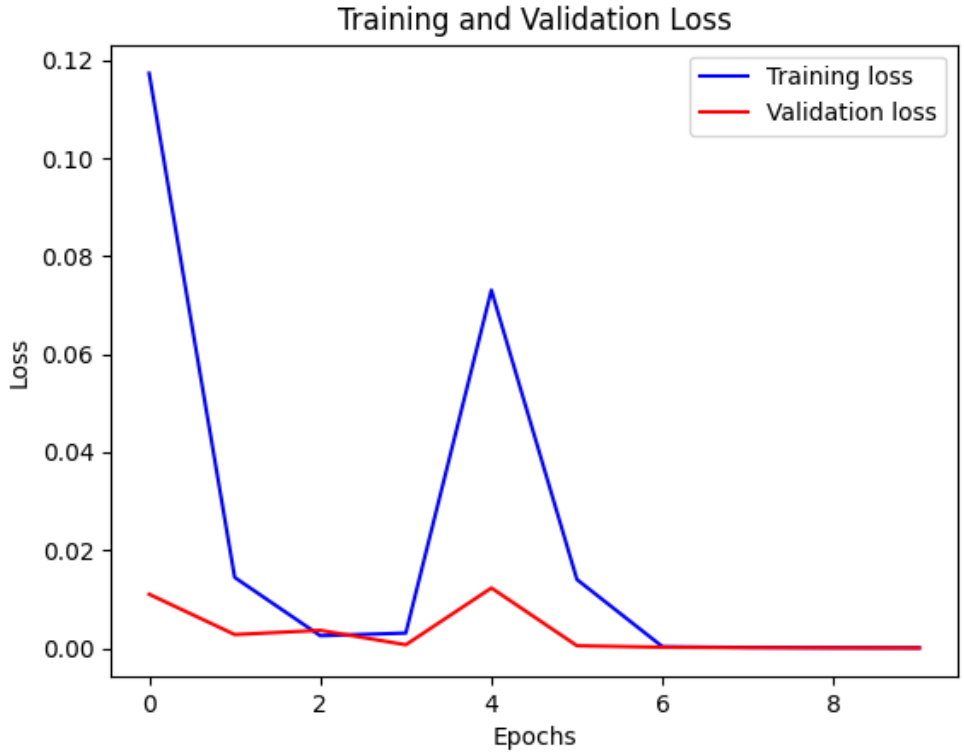
- Đầu vào: Mạng LSTM này chấp nhận đầu vào là một chuỗi thời gian có kích thước (batch_size, time_steps, input_dim), trong đó:
  - batch_size: Số lượng mẫu trong mỗi lần đào tạo.
  - time_steps: Số lượng bước thời gian trong mỗi mẫu, cụ thể trong kiến trúc trên là 5.
  - input_dim: Số lượng đặc trưng của mỗi bước thời gian, trong kiến trúc trên là 63, tương ứng với tọa độ của 21 điểm mốc trên bàn tay, mỗi điểm bao gồm 3 giá trị x, y, z (21*3=63).
- Lớp LSTM (128 units):
  - Lớp LSTM đầu tiên có 128 đơn vị (units) tương ứng với số lượng neural, và được cấu hình để trả về chuỗi kết quả (return_sequences = True) cho lớp LSTM tiếp theo.
  - Hàm kích hoạt của các cổng và tế bào nhớ trong LSTM được xác định mặc định là "tanh".
- Lặp lại lớp LSTM:
  - Lớp LSTM được lặp lại 4 lần với cùng số units, nhưng input sẽ nhận từ tầng LSTM phía trước đó.
- Lớp Dense và softmax:
  - Sau khi đã qua các lớp LSTM và Dropout, đầu ra được chuyển đến một lớp Dense với số lượng đơn vị bằng số lượng lớp là 6, tương ứng với 6 nhãn của dữ liệu, và hàm kích hoạt là softmax. Lớp này chuyển đổi các giá trị đầu ra thành xác suất của mỗi lớp.

3. Huấn luyện, đánh giá và tối ưu mô hình

Tiến hành huấn luyện mô hình với thông số như sau:

- Epochs: 10
- Batch_size: 32
- Optimizer: adam
- Loss: categorical_crossentropy

Kết quả huấn luyện mô hình được thể hiện thông qua biểu đồ accuracy và loss như **Hình 15**, **Hình 16**.



Hình 15: Biểu đồ loss của kết quả huấn luyện

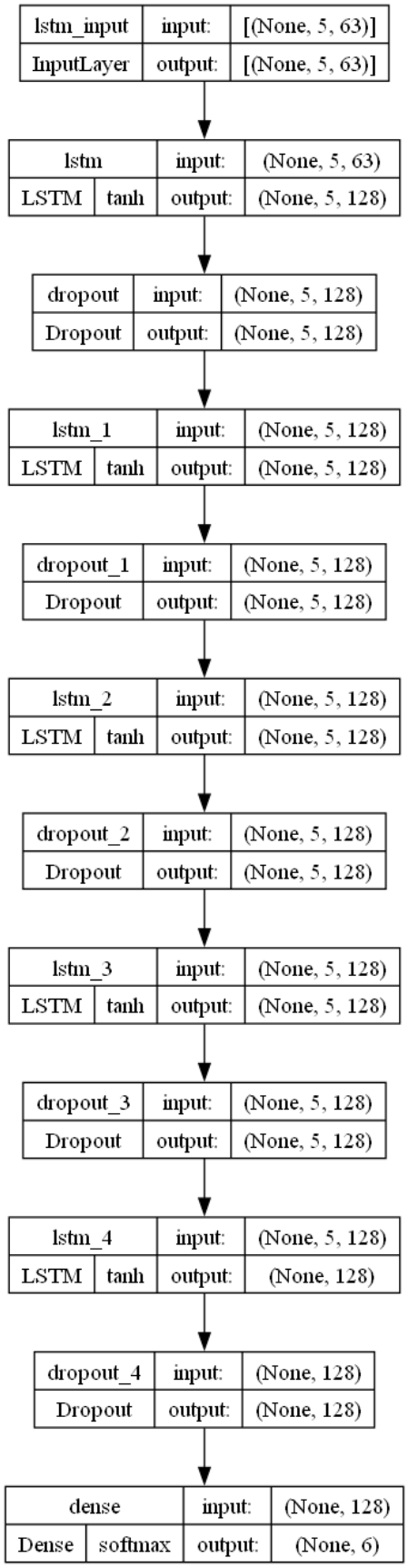
Hình 16: Biểu đồ accuracy của kết quả huấn luyện

Qua biểu đồ loss, nhận thấy rằng kết quả training loss và validation loss xảy ra tình trạng tăng giảm bất ổn định, để cải thiện tình trạng này cần tiến hành điều chỉnh các tham số, huấn luyện và đánh giá lại đến khi đạt kết quả tốt nhất đối với tập dữ liệu và mô hình này.

Sau nhiều lần điều chỉnh và thử nghiệm, để đạt được kết quả mô hình tốt nhất so với tập dữ liệu và mô hình này là giữ nguyên thông số huấn luyện bao gồm:

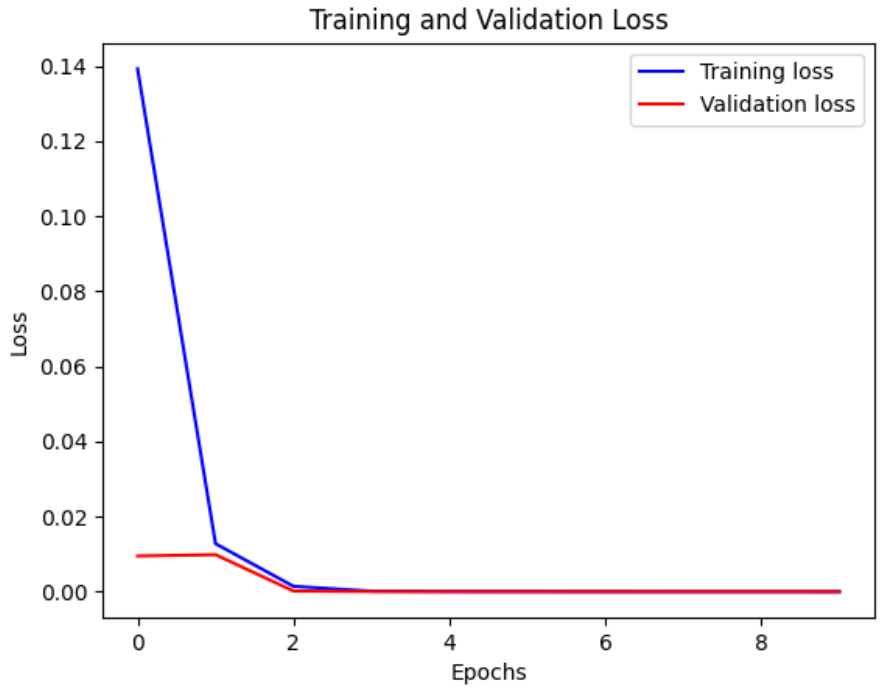
- Epochs: 10
- Batch_size: 32
- Optimizer: adam
- Loss: categorical_crossentropy

Nhưng đối với kiến trúc mô hình có sự thay đổi để cải thiện bằng cách thả xen kẽ giữa các tầng LSTM bằng các tầng Dropout với giá trị truyền vào là 0.2. Kiến trúc mô hình sau khi cải thiện được thể hiện như **Hình 17**.

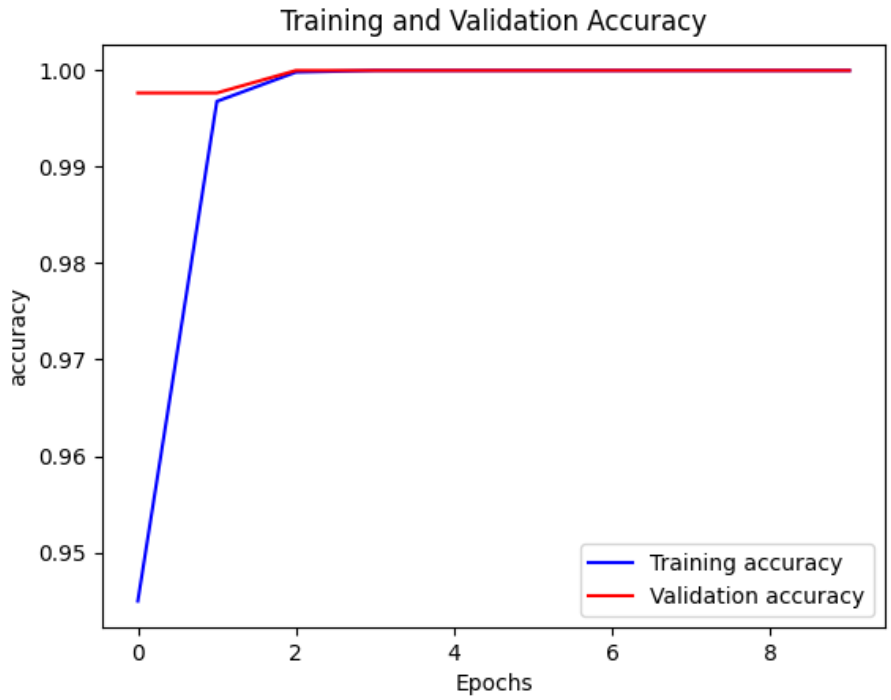


Hình 17: Kiến trúc mô hình sau khi cải tiến

Sau khi cải tiến mô hình, tình trạng giá trị của training loss và validation loss bất ổn định đã được giảm thiểu. Kết quả huấn luyện của mô hình sau khi cải tiến được thể hiện thông qua 2 biểu đồ loss và accuracy như **Hình 18**, **Hình 19**.



Hình 18: Biểu đồ loss của mô hình sau khi tối ưu



Hình 19: Biểu đồ accuracy của mô hình sau khi tối ưu

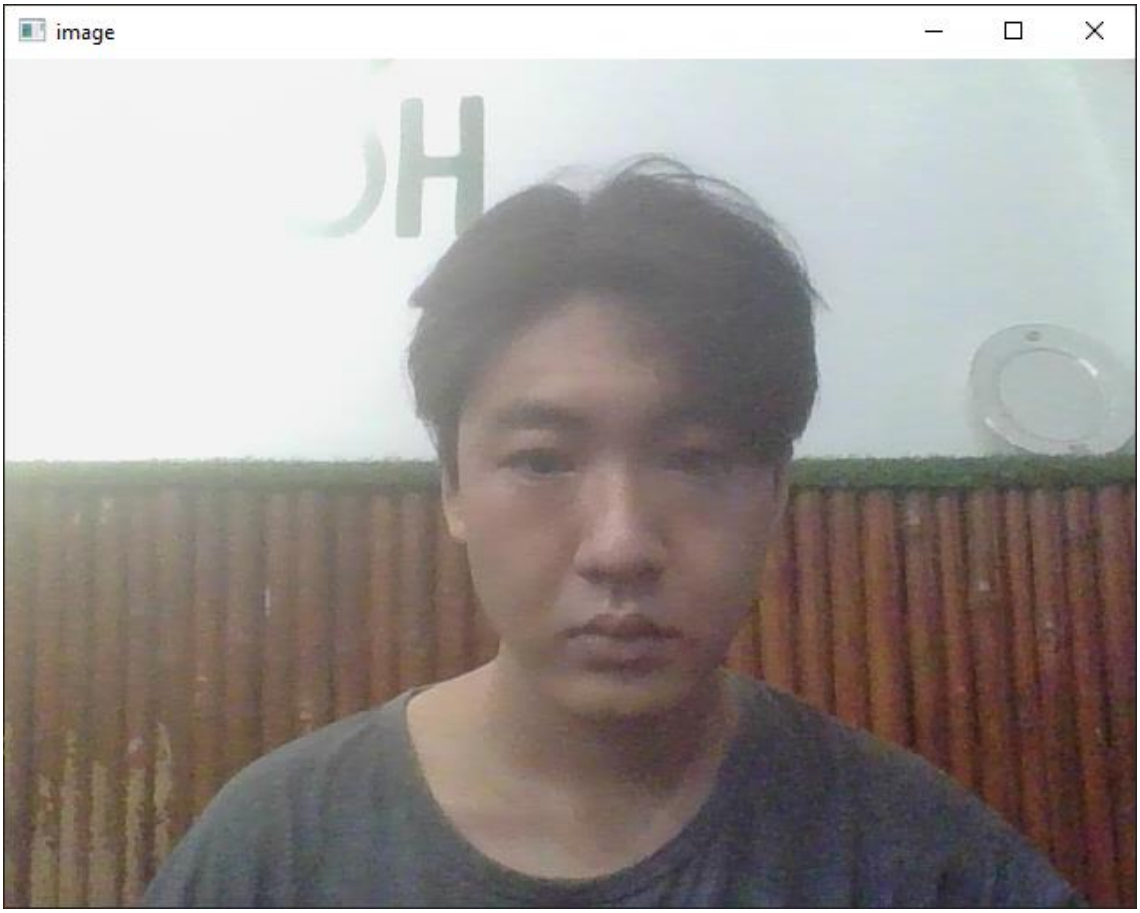
Qua biểu đồ Accuracy nhận thấy giá trị accuracy đã hội tụ đến 1.0, tình trạng này xảy ra do kích thước dữ liệu cho mô hình không quá lớn, nên mô hình dễ hội tụ và đạt được giá trị tối đa. Tuy nhiên, trong phạm vi bài này, kết quả này sẽ không làm ảnh hưởng đến kết quả tổng thể của đề tài.

4. Nhận diện cử chỉ

Sau khi huấn luyện và tối ưu mô hình, tiến hành thực nghiệm mô hình để kiểm thử độ hiệu quả của mô hình, xây dựng một chương trình đơn giản để nhận dạng cử chỉ/hành vi thông qua webcam.

4.1. Đọc thông tin từ webcam

Sử dụng thư viện OpenCV [5] để đọc và hiển thị thông tin từ webcam, kết quả như **Hình 20**.



Hình 20: Đọc và hiển thị thông tin từ webcam

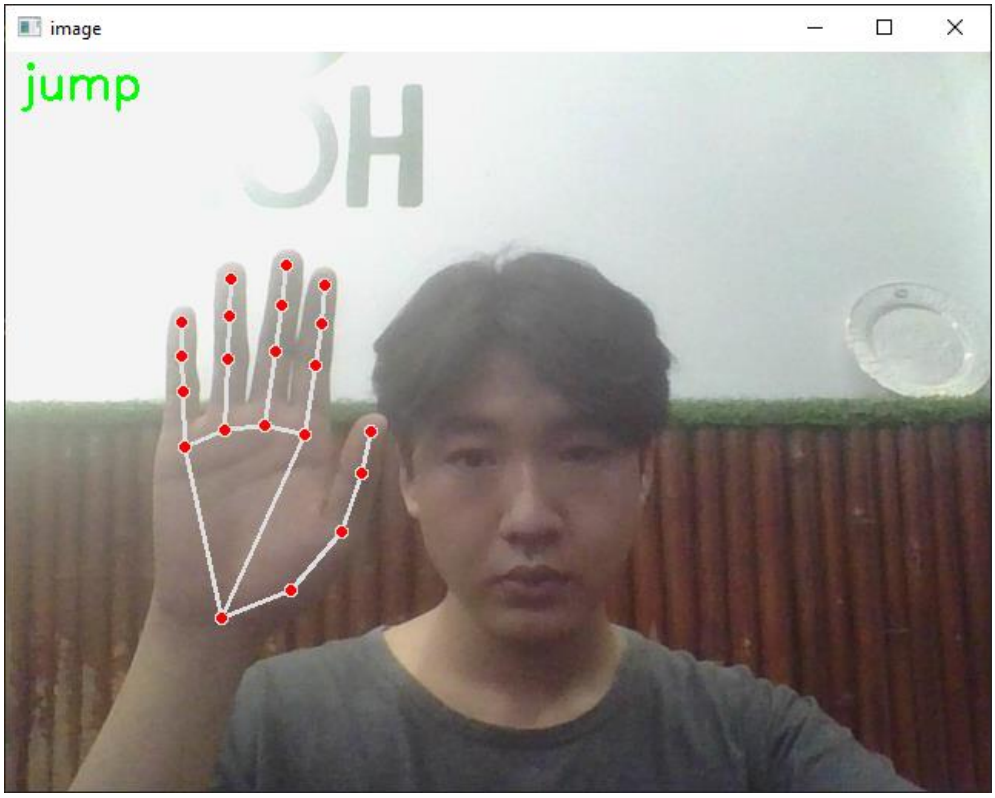
4.2. Phát hiện các điểm mốc và trích xuất đặc trưng

Vì đầu vào của mô hình là các đặc trưng được trích xuất dưới dạng các điểm mốc trên bàn tay, tiến hành sử dụng Hand landmarks detection [1] để phát hiện các điểm mốc, sau đó hiển thị lên kết quả như **Hình 21**.

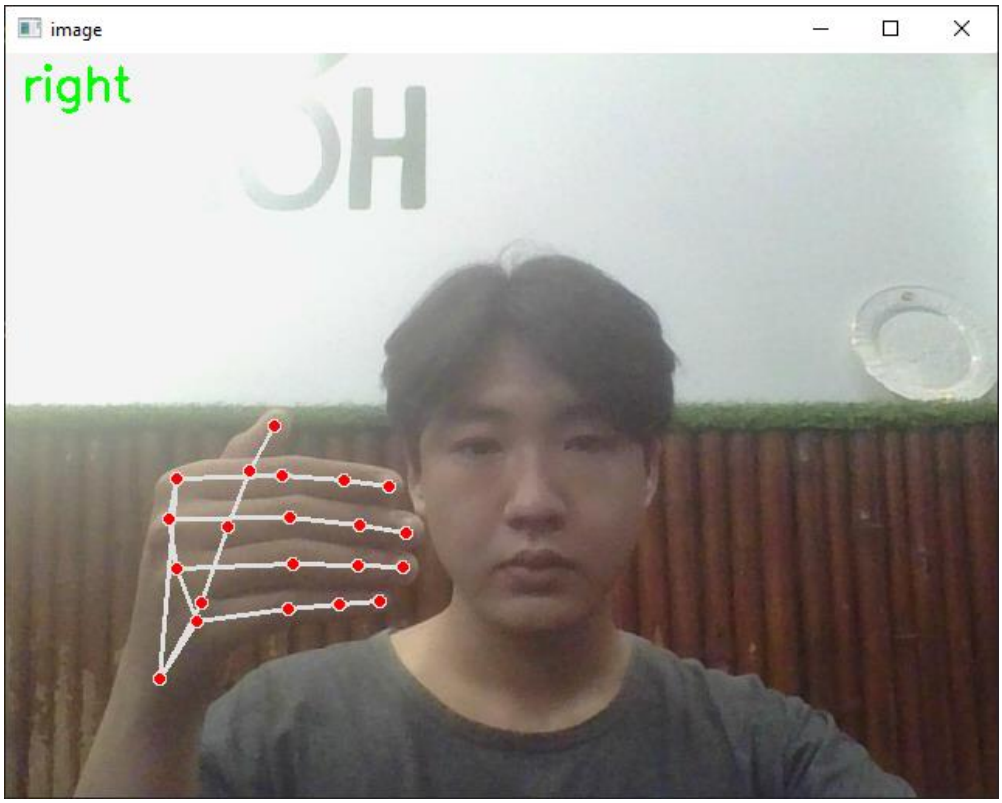
Hình 21: Vẽ các điểm mốc trên bàn tay

4.3. Dự đoán kết quả

Việc đọc dữ liệu từ webcam bằng OpenCV là việc đọc và hiển thị từng frame. Vì vậy, mỗi lần đọc qua một frame và trích xuất được các điểm mốc cơ thể, tiến hành chuẩn hóa theo 2 bước như cách chuẩn hóa dữ liệu ở Chương 2, sau khi chuẩn hóa lưu kết quả vào một danh sách. Lặp lại đến khi danh sách chứa số lượng bằng với số lượng timestep đầu vào của mô hình đã huấn luyện. Khi số lượng phần tử trong danh sách đã đủ, tiến hành đưa vào mô hình để dự đoán và hiển thị kết quả lên màn hình chương trình, việc dự đoán và hiển thị kết quả được thể hiện như **Hình 22**, **Hình 23.**



Hình 22: Dự đoán kết quả jump



Hình 23: Dự đoán kết quả right

CHƯƠNG 5: TÍCH HỢP ĐIỀU KHIỂN VÀO TRÒ CHƠI

Như đã đề cập, để giao tiếp giữa bộ nhận dạng cử chỉ/hành vi với trò chơi sẽ sử dụng tính năng UDP trong Unity.

1. Giao thức UDP (User Datagram Protocol)

UDP là một trong các giao thức truyền thông được sử dụng trong các ứng dụng mạng và là một phần của bộ giao thức Internet, bên cạnh TCP (Transmission Control Protocol). UDP là một giao thức không kết nối, có nghĩa là không yêu cầu thiết lập một kết nối giữa các thiết bị truyền và nhận trước khi gửi dữ liệu.

Đặc điểm của UDP:

- Không Kết Nối: Không giống như TCP, UDP không thiết lập một kết nối trước khi gửi dữ liệu. Điều này làm cho UDP nhanh hơn TCP vì nó không phải trải qua quá trình bắt tay và duy trì kết nối.
- Không Đảm Bảo: UDP không đảm bảo rằng gói tin sẽ đến đích một cách an toàn hay theo thứ tự. Nếu thứ tự hoặc tính toàn vẹn của gói tin là cực kỳ quan trọng, TCP có thể là một lựa chọn tốt hơn.
- Nhẹ Nhàng: Do thiếu các tính năng như kiểm soát lỗi và kiểm soát luồng, UDP yêu cầu ít tài nguyên hơn và có overhead (phần dư) thấp hơn trong gói tin.
- Hữu Ích cho Các Ứng Dụng Thời Gian Thực: UDP thường được sử dụng trong các ứng dụng cần phản hồi nhanh chóng như trò chơi trực tuyến, video trực tuyến, và VoIP (Voice over Internet Protocol), nơi độ trễ có thể là một vấn đề lớn và một vài gói tin bị mất không ảnh hưởng nghiêm trọng tới chất lượng.

Ứng dụng trong Unity:

Trong môi trường như Unity, UDP có thể được sử dụng để giao tiếp giữa các ứng dụng hoặc để truyền tải dữ liệu giữa các client trong một trò chơi mạng. Sử dụng UDP cho phép giảm thiểu độ trễ và cung cấp trải nghiệm người dùng mượt mà hơn trong các tình huống đòi hỏi phản hồi nhanh, như trong các trò chơi bắn súng nhanh hoặc trò chơi thể thao. Và trong phạm vi này chính là truyền tải tín hiệu điều khiển từ bộ phận nhận dạng cử chỉ đến trò chơi.

2. Tích hợp UDP vào trò chơi và bộ nhận dạng

2.1. Trò chơi

Để tích hợp UDP vào trò chơi được tạo bởi Unity, tiến hành tạo một script mới để thiết lập và quản lý việc nhận dữ liệu thông qua UDP. Sử dụng thư viện của bản của .NET là **System.Net.Sockets** để thiết lập việc nhận dữ liệu thông qua địa chỉ

IP 127.0.0.1 và cổng (port) 5052. Dữ liệu sẽ được nhận liên tục trong quá trình người dùng chơi trò chơi, dữ liệu nhận được sẽ có dạng chuỗi byte và được chuyển đổi thành chuỗi UTF-8. Sau đó được lưu trữ vào một biến toàn cục, dùng để tái sử dụng trong phần điều khiển trò chơi.

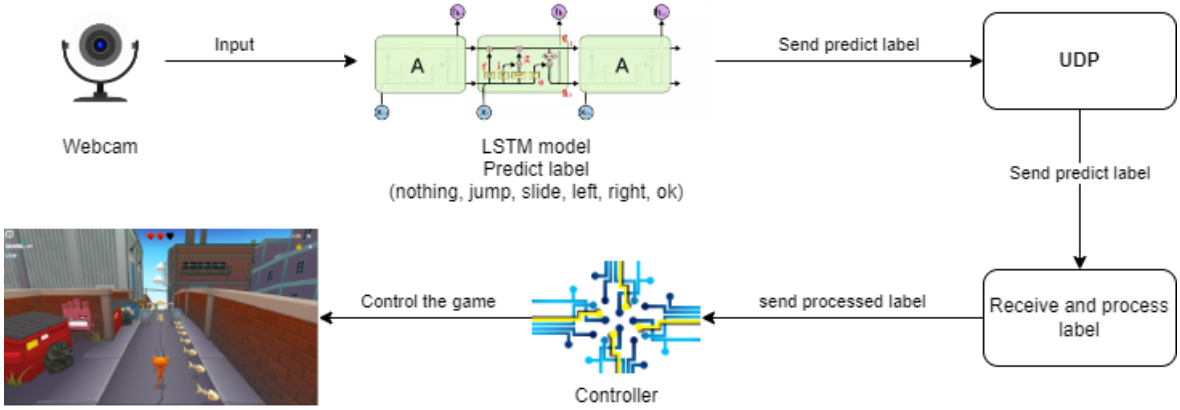
2.2. Bộ nhận dạng

Bộ nhận dạng được sử dụng để điều khiển trò chơi chính là chương trình nhận dạng đã được hoàn thiện như **Hình 22**. Tiến hành tích hợp việc gửi dữ liệu qua UDP vào trong chương trình này bằng cách sử dụng thư viện **socket** trong Python.

Tương tự như ở phía trò chơi, tiến hành thiết lập việc gửi tín hiệu điều khiển trong chương trình thông qua địa chỉ IP 127.0.0.1 và cổng (port) 5052. Khi trò chơi và chương trình bắt đầu. Mỗi khi nhận dạng được một cử chỉ, chương trình sẽ gửi thông tin nhận dạng qua UDP dưới dạng mảng byte để điều khiển trò chơi.

3. Tích hợp bộ nhận dạng vào trò chơi

Sau khi đã tích hợp tính năng UDP vào bộ nhận dạng và trò chơi, tiến hành tích hợp bộ nhận dạng thành một hệ thống hoàn chỉnh. Sơ đồ hoạt động tổng quan của hệ thống được trình bày như **Hình 24**.



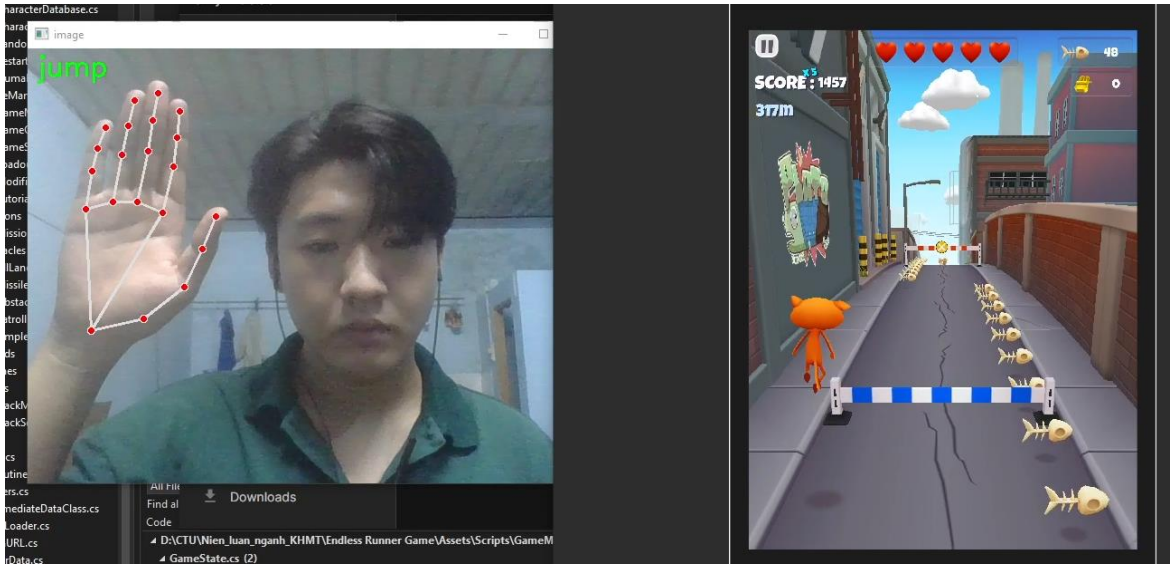
Hình 24: Sơ đồ hoạt động tổng quan

Cách hoạt động: bộ nhận dạng sẽ nhận hình ảnh trực tiếp từ webcam, với mỗi hình ảnh tiến hành trích xuất đặc trưng và lưu trữ gộp thành một lô với số lượng là 5 (tương ứng với số lượng timestep của mô hình). Khi đã đủ một lô tương đương với input đưa nào mô hình LSTM đã huấn luyện, mô hình sẽ dự đoán nhãn cho cử chỉ và gửi nhãn này qua UDP dưới dạng mảng byte. Bên phía trò chơi sẽ nhận nhãn và xử lý bằng cách chuyển về chuỗi UTF-8, sau đó lưu trữ và đưa vào bộ phận điều khiển của trò chơi để thực hiện điều khiển và thể hiện lên màn hình trò chơi.

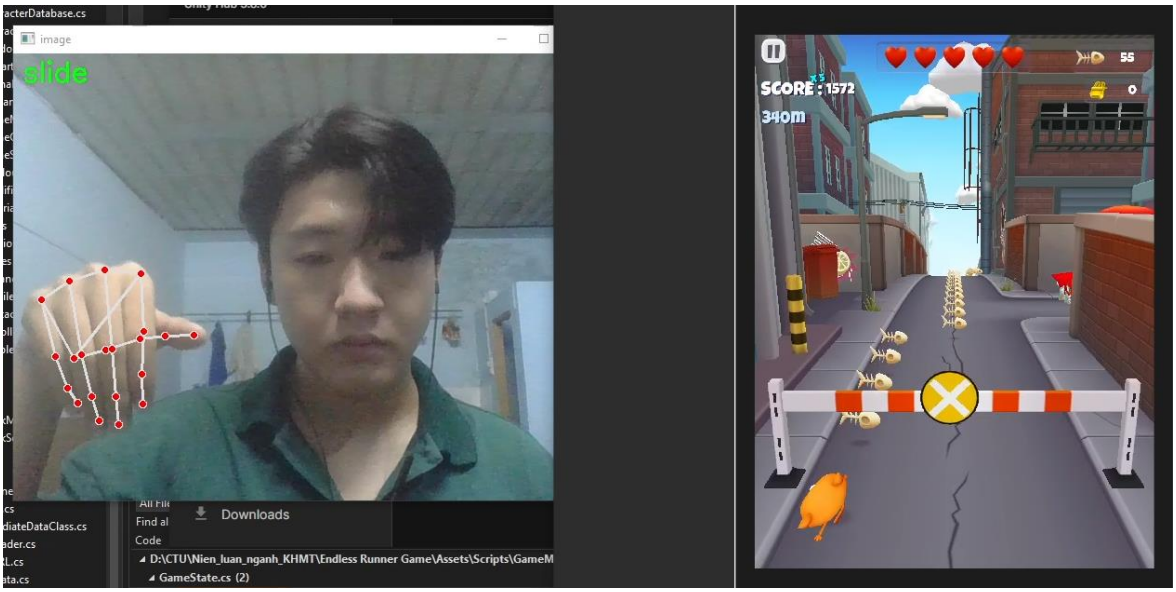
4. Kết quả thực nghiệm

Kết quả thực nghiệm được thể hiện như **Hình 25**, **Hình 26**. Trong 6 hành động được làm nhãn của mô hình, có 5 hành động dùng để điều khiển trong lúc chơi trò chơi (jump, slide, left, right, nothing) và 1 hành động (ok) dùng để xác nhận bắt đầu trò chơi khi đang ở màn hình chính, hoặc xác nhận chơi lại khi chơi thua.

Ví dụ, với **Hình 25**, khi người chơi vẫy tay lên trên, bộ nhận dạng sẽ nhận dạng thành nhãn jump và truyền kết quả tới trò chơi thông qua UDP, khi trò chơi nhận được sẽ xử lý và điều khiển nhân vật nhảy lên để vượt chướng ngại vật.



Hình 25: Kết quả thực nghiệm - điều khiển jump



Hình 26: Kết quả thực nghiệm - điều khiển slide

PHẦN KẾT LUẬN

1. Kết quả đạt được

- Cách xây dựng một trò chơi bằng Unity.
- Chuẩn hóa dữ liệu dạng video cho phù hợp với một mô hình LSTM.
- Giao tiếp giữa các ứng dụng để truyền tải thông tin cần thiết.
- Xây dựng được một trò chơi có thể điều khiển bằng cử chỉ của con người.

2. Hạn chế

- Phạm vi điều khiển còn hạn hẹp, chỉ mới sử dụng cử chỉ tay để điều khiển, thiếu việc thực tế ảo trong sự tương đồng cử chỉ điều khiển của người chơi và trong trò chơi.
- Việc nhận dạng vẫn còn xảy ra sai do dữ liệu còn ít, chưa đa dạng.

3. Hướng phát triển

- Cải tiến mô hình bằng cách thêm, tăng cường dữ liệu để giảm sai số. Đa dạng cử chỉ để thực tế ảo một cách giống nhất cho việc sử dụng cử chỉ để điều khiển trò chơi.
- Mở rộng thêm cho một số trò chơi liên quan tới thực tế ảo.
- Có thể áp dụng mô hình này trong đời sống như tích hợp vào camera giám sát để phát hiện và cảnh báo nếu xuất hiện hành vi lạ như trộm cướp, phá hoại,…

TÀI LIỆU THAM KHẢO

[1] Google, "Hand landmarks detection guide," 03 11 2023. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker. [Accessed 15 3 2024].

[2] N. T. Huyen, "Recurrent Neural Network: Từ RNN đến LSTM," 24 6 2021. [Online]. Available: https://viblo.asia/p/recurrent-neural-network-tu-rnn-den-lstm-gGJ597z1ZX2. [Accessed 16 3 2024].

[3] Wikipedia, "Endless runner," 12 03 2024. [Online]. Available: https://en.wikipedia.org/wiki/Endless_runner. [Accessed 17 03 2024].

[4] U. Technologies, "Endless Runner - Sample Game," 15 02 2023. [Online]. Available: https://assetstore.unity.com/packages/templates/tutorials/endless-runner-sample-game-87901. [Accessed 18 03 2024].

[5] OpenCV, "OpenCV - Open Computer Vision Library," [Online]. Available: https://opencv.org/. [Accessed 26 4 2024].