BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC DUY TÂN**

****

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**NHẬN DIỆN HÀNH ĐỘNG VỚI DEEP LEARNING**

Họ, tên sinh viên chịu trách nhiệm chính: Trương Hồng Ngân

Ngành học:  Big Data & Machine Learning - Khóa học: K24

Khoa: Khoa học máy tính

**Đà Nẵng, tháng 1 năm 2022**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC DUY TÂN**

****

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**NHẬN DIỆN HÀNH ĐỘNG VỚI DEEP LEARNING**

Thuộc nhóm ngành khoa học: Kỹ thuật

Họ và tên, ngành học, khóa học sinh viên/nhóm sinh viên thực hiện đề tài:

Trương Hồng Ngân - Big Data & Machine Learning - K24

Nguyễn Hoài Nam - Big Data & Machine Learning - K24

Trần Lê Thành Lợi - Big Data & Machine Learning - K24

Chức danh khoa học, học vị, họ và tên của giảng viên hướng dẫn:

Tiến sĩ - Đặng Việt Hùng

**Đà Nẵng, tháng 1 năm 2022**

# MỤC LỤC

[**MỤC LỤC**](#_heading=h.gjdgxs) **3**

[**DANH MỤC CÁC KÝ HIỆU**](#_heading=h.htqsa0bwfht7) **5**

[**DANH MỤC HÌNH ẢNH**](#_heading=h.nj1iau44t7ku) **6**

[**DANH MỤC TỪ VIẾT TẮT**](#_heading=h.65wpgciuc9le) **7**

[**MỞ ĐẦU**](#_heading=h.3y8ufqbtlzt0) **8**

[1. Lý do chọn đề tài](#_heading=h.jj7ivcvvicn4) 8

[2. Mục đích nghiên cứu](#_heading=h.4tgaf6jpb48i) 8

[3. Phương pháp nghiên cứu](#_heading=h.ennpjrhg5h6y) 9

[4. Đối tượng nghiên cứu](#_heading=h.4uxj6z2iuc8o) 9

[5. Phạm vi nghiên cứu](#_heading=h.wk2h41avam9m) 9

[**CHƯƠNG 1. TỔNG QUAN VỀ BÀI TOÁN NHẬN DẠNG HÀNH ĐỘNG**](#_heading=h.1fob9te) **10**

[1. 1. Giới thiệu về thị giác máy tính](#_heading=h.abwt42o5wbha) 10

[1. 2. Nhận diện hành động của con người](#_heading=h.9rybyooz0xqw) 10

[1. 2. 1. 3D Convolutional Neural Network[1]](#_heading=h.ineqpzos2tvx) 11

[1. 2. 2. Framework mediapipe và Long Short Term Memory [2][7]](#_heading=h.o8qqljdcwc0f) 11

[1. 2. 3. Kết luận](#_heading=h.lk1ioc2lna9) 12

[**CHƯƠNG 2. FRAMEWORK MEDIAPIPE VÀ LONG SHORT TERM MEMORY**](#_heading=h.821nq1kogw7t) **13**

[2. 1. Detection pose estimation](#_heading=h.75adibjte5jq) 13

[2. 2. Recurrent Neural Network và Long short-term memory](#_heading=h.vxh1vve22gf1) 14

[2. 2. 1. Recurrent Neural Network (RNN)[11]](#_heading=h.9rcmc8iilsk8) 14

[2. 2. 2. Long short-term memory (LSTM)[12]](#_heading=h.owaydtf1vhxb) 17

[**CHƯƠNG 3. THỰC NGHIỆM VÀ KẾT QUẢ**](#_heading=h.5l14ly9p30) **21**

[3. 1. Giới thiệu các công cụ](#_heading=h.ccztgfnp2mhl) 21

[3. 1. 1. Thư viện Numpy[3]](#_heading=h.d57f665n7w36) 21

[3. 1. 2. Thư viện PyQt5[8]](#_heading=h.2s8eyo1) 21

[3. 1. 3. Thư viện Scikit-learn (Sklearn)[6]](#_heading=h.93si7eobhhbg) 22

[3. 1. 4. Thư viện Matplotlib[5]](#_heading=h.17dp8vu) 23

[3. 1. 5. Thư viện Keras[10]](#_heading=h.x3y37tgwwpq5) 23

[3. 1. 6. Thư viện OpenCV[9]](#_heading=h.y1e3tvqtoykl) 24

[3. 1. 7. Thư viện Tensorflow[13]](#_heading=h.cy0m6o9w9ani) 25

[4. 1. 8. Thư viện Mediapipe[4]](#_heading=h.cfs9wgh71pip) 26

[3. 2. Phân tích bài toán](#_heading=h.yi07x487oyp4) 27

[3. 2. 1. Vấn đề về dữ liệu](#_heading=h.2b7w7sxywn58) 27

[3. 2. 2. Vấn đề lựa chọn thuật toán và mô hình](#_heading=h.iik058v902a) 27

[3. 3. Tiền xử lý dữ liệu](#_heading=h.kegsx8gha7vf) 28

[3. 4. Định nghĩa model](#_heading=h.bn0qehv5okma) 30

[3. 5. Tối ưu mô hình, tham số và lưu model](#_heading=h.c729kcy6pvf) 32

[3. 6. Giao diện bài toán.](#_heading=h.x3q9fc2sg3i2) 33

[3. 7. Đánh giá mô hình](#_heading=h.dtdmcr62vlrg) 35

[3. 7. 1. Độ chính xác của mô hình](#_heading=h.5i7f04fsu1j3) 35

[3. 7. 2. Giá trị hàm mất mát](#_heading=h.7um4is6xentk) 37

[3. 7. 3. Ma trận nhầm lẫn, Classification report](#_heading=h.qde0pvlqt087) 38

[**TỔNG KẾT**](#_heading=h.roy830nvmqw6) **43**

[1. Đánh giá kết quả đạt được](#_heading=h.3358gyslx0e7) 43

[2. Những hạn chế](#_heading=h.rl265if7tpwy) 43

[3. Hướng phát triển](#_heading=h.h6bq18mhmeod) 43

[**TÀI LIỆU THAM KHẢO**](#_heading=h.1cyirb207dsq) **44**

# DANH MỤC CÁC KÝ HIỆU

|  |  |
| --- | --- |
| **Ký hiệu** | **Ý nghĩa** |
| *x, y, N, k* | In nghiêng, thường hoặc hoa, các số vô hướng |
| **x, y** | In đậm, chữ thường, là các vector |
| **X, Y** | In đậm, chữ hoa, là các ma trận |
| R | Tập hợp các chữ số thực |
| N | Tập hợp các chữ số tự nhiên |
|  | Chuyển vị ma trận A |
|  | Hàm sigmoid |
| e | Cơ số e |
| w, theta | Bộ trọng số |
| argmax/argmin | Tìm chỉ số mà tại đó mảng một chiều đạt giá trị lớn nhất hay nhỏ nhất |
| J(w), L(w) | Biểu thị hàm mất mát |
| log | Logarit tự nhiên |
|  | Đạo hàm của hàm số L theo w |
|  | Tốc độ học (Learning rate) |
| n, m | Số lượng dữ liệu |
|  | Vận tốc thời điểm t |
|  | Thường được chọn là một giá trị khoảng 0. 9 |
| L(w) | Đạo hàm của hàm mất mát tại w |
|  | Ký hiệu của tích |

# DANH MỤC HÌNH ẢNH

|  |  |
| --- | --- |
| **Hình** | **Tên hình** |
| Hình 1.1 | Mô hình C3D cho bài toán nhận diện hành động |
| Hình 2.1 | Mô hình tổng quát cho bài toán sử dụng MediaPipe kết hợp LSTM để nhận diện video |
| Hình 2.2 | Các dạng bài toán RNN |
| Hình 2.3 | Mô hình RNN cho bài toán |
| Hình 2.4 | Mô hình RNN rút gọn |
| Hình 2.5 | Mô hình LSTM |
| Hình 2.6 | Mô hình LSTM cho bài toán nhận diện hành động |
| Hình 3.1 | Những lĩnh vực và ngôn ngữ MediaPipe hỗ trợ |
| Hình 3.2 | 33 keypoints trên cơ thể người |
| Hình 3.3 | Giao diện 1 |
| Hình 3.4 | Giao diện 2 |
| Hình 3.5 | Walking |
| Hình 3.6 | Running |
| Hình 3.7 | Standing |
| Hình 3.8 | Hand Clapping |
| Hình 3.9 | Fall Down |
| Hình 3.10 | Độ chính xác của mô hình |
| Hình 3.11 | Giá trị hàm mất mát |

# DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Ký hiệu viết tắt** | **Chữ viết đầy đủ** |
| 3D-CNN, C3D | 3D Convolutional Neural Network |
| LSTM | Long Short Term Memory |
| OPenCV | Open Source Computer Vision Library |
| MRI | Magnetic Resonance Imaging |
| CT Scan | A computerized tomography scan |
| AI | Artificial Intelligence |
| CPU | Central Processing Unit |
| GPU | Graphics Processing Unit |
| RNN | Recurrent Neural Network |
| CNN | Convolutional Neural Network |
| NN | Neural Network |
| SGD | Stochastic Gradient Descent |
| TP | True Positive |
| TN | True Negative |
| FP | False Positive |
| FN | False Negative |

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Nhờ vào sự tiến bộ của công nghệ và các nhu cầu ứng dụng trong những năm gần đây, nghiên cứu về nhận dạng hoạt động người đã có được sự phát triển mạnh mẽ. Sự tiến bộ của công nghệ đã thúc đẩy cộng đồng nghiên cứu chuyển từ truyền, thu nhận và xử lý dữ liệu mức thấp sang nghiên cứu tích hợp thông tin mức cao, xử lý ngữ cảnh, nhận dạng và suy diễn các hoạt động. Thêm vào đó, ngày càng có nhiều bài toán thực tế cần các giải pháp dựa vào nhận dạng hoạt động. Ví dụ như ứng dụng an ninh và theo dõi giám sát cần công nghệ nhận dạng hoạt động để xác định các mối đe dọa về khủng bố. Các ứng dụng trợ giúp con người trong cuộc sống hàng ngày cần công nghệ giám sát, nhận dạng hoạt động và hỗ trợ người sống một mình, đặc biệt là người già cô đơn. Hàng loạt ứng dụng mới khác như phòng họp thông minh, ngôi nhà thông minh, bệnh viện thông minh, hoặc ứng dụng trong việc chăm sóc sức khỏe,... Dựa vào nhận dạng hoạt động để cung cấp nhiều cách thức tương tác đa dạng, chủ động cung cấp các dịch vụ, trợ giúp người dùng hoàn thành công việc.

Nhận thấy được sự cần thiết của việc nhận diện hoạt động trong đời sống thực tế nên chúng tôi quyết định chọn đề tài này. Chúng tôi sẽ tập trung khai thác về nhận dạng một số hoạt động hằng ngày của con người.

## 2. Mục đích nghiên cứu

Mục đích của đồ án này là nhận dạng một số hành động thường xuyên gặp của con người như: chạy, đi bộ, đứng, ngã, vỗ tay,... Nhóm sẽ nghiên cứu và áp dụng một trong 2 cách sau: 3D Convolutional Neural Network (C3D), hoặc Frameworks mediapipe kết hợp với Long Short Term Memory (LSTM). Theo đó các mục tiêu mà nhóm đặt ra khi nghiên cứu và áp dụng là:

• Hiểu được nguyên lý hoạt động của thuật toán.

• Sử dụng Python và các thư viện liên quan đến bài toán.

• Áp dụng mô hình vào bài toán thực tế.

## 3. Phương pháp nghiên cứu

* Phương pháp lý thuyết: dựa vào lý thuyết các thuật toán 3D Convolutional Neural Network (C3D), framework mediapipe và Long Short Term Memory (LSTM) để sử dụng cho bài toán nhận diện hành động.
* Phương pháp thực nghiệm: áp dụng các lý thuyết vào bài toán thực tế, triển khai mô hình thực nghiệm, đánh giá mô hình.

## 4. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài này là bài toán nhận diện hành động của con người và các thuật toán Machine Learning.

## 5. Phạm vi nghiên cứu

Trong bài toán này, nhóm sẽ tập trung vào các hành động mà chúng ta thường xuyên gặp phải và có thể gặp ở mọi lúc, mọi nơi như: đi bộ, chạy, đứng, ngã, vỗ tay. Các thuật toán và framework được sử dụng là 3D Convolutional Neural Network (C3D), Long Short Term Memory (LSTM) và OpenCV để xử lý ảnh, Medipipe để vẽ các keypoint, PyQt5 để xây dựng giao diện.

# CHƯƠNG 1. TỔNG QUAN VỀ BÀI TOÁN NHẬN DẠNG HÀNH ĐỘNG

## 1. 1. Giới thiệu về thị giác máy tính

Thị giác máy tính (Computer Vision) là một lĩnh vực đã và đang rất phát triển. Khái niệm thị giác máy tính có liên quan tới nhiều ngành học và hướng nghiên cứu khác nhau. Từ những năm 1970, khi năng lực tính toán của máy tính ngày càng trở nên mạnh mẽ hơn, các máy tính lúc này có thể xử lý được những tập dữ liệu lớn như các hình ảnh, các đoạn phim thì khái niệm và kỹ thuật về thị giác máy tính ngày càng được nhắc đến và nghiên cứu nhiều hơn.

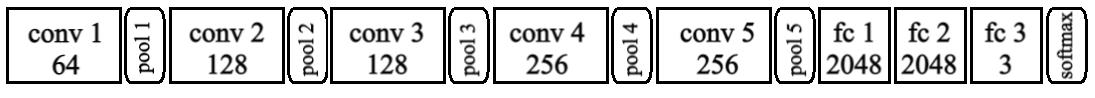
Thị giác máy tính bao gồm lý thuyết và các kỹ thuật liên quan nhằm mục đích tạo ra hệ thống nhân tạo có thể tiếp cận thông tin từ các hình ảnh thu được hoặc các tập dữ liệu đa chiều. Đối với mỗi con người chúng ta, quá trình nhận thức bên ngoài là một điều dễ dàng. Quá trình nhận thức đó được “học” thông qua quá trình sống của mỗi người. Tuy nhiên với các vật vô tri vô giác như máy tính, robot, … thì điều đó là một bước rất gian nan. Các thiết bị ngày nay không chỉ nhận thông tin ở dạng tín hiệu đơn lẻ mà còn có thể có cái “nhìn” thật với thế giới bên ngoài. Cái “nhìn” này qua quá trình phân tích, kết hợp với các mô hình như Machine learning, Neural network, … sẽ giúp cho thiết bị tiến dần tới một hệ thống nhân tạo có khả năng ra quyết định linh hoạt và đúng đắn hơn rất nhiều.

## 1. 2. Nhận diện hành động của con người

Nhận diện hành động của con người là một lĩnh vực nghiên cứu rộng rãi trong thị giác máy tính. Các ứng dụng thường được nhắc đến là các hệ thống giám sát, phân tích video, robot và một loạt các hệ thống có liên quan đến sự tương tác giữa người với người và các thiết bị điện tử.

### 1. 2. 1. 3D Convolutional Neural Network

Sử dụng thuật toán 3D-CNN (C3D) cho bài toán nhận diện hoạt động. Ý tưởng cơ bản, đầu vào là một hoặc nhiều video, trong mỗi video sẽ được tách ra thành nhiều frame, sau đó lấy lần lượt mỗi lần 16 frame để train và thay đổi kích thước tất cả các frame thành 112 × 112, như vậy kích thước đầu vào là (112, 112, 16, 3). Sau đó, tất cả các frame sẽ được đưa qua C3D để trích xuất các đặc trưng và qua softmax để phân loại. Mô hình sẽ có 5 lớp convolutional (conv), 5 lớp max pooling (pool), 3 lớp fully connected (fc) và cuối cùng là 1 lớp softmax như sau:



Hình 1. 1. Mô hình C3D cho bài toán nhận diện hành động

* Ưu điểm: Trong C3D, dữ liệu đầu vào và đầu ra của C3D là 4 chiều nên C3D thích hợp với dữ liệu hình ảnh 3D (MRI, CT Scan, Video,... ).
* Nhược điểm: Nếu mạng CNN có quá nhiều lớp thì quá trình huấn luyện sẽ tốn rất nhiều thời gian. Bên cạnh đó thì cũng cần tập dữ liệu đủ lớn để huấn luyện mới đem lại kết quả cao và cũng rất dễ gặp phải tình trạng overfitting. Ngoài những vấn đề trên thì việc dùng CNN ta chỉ nhận dạng hành động theo từng frame chứ không phải là nhận dạng cho cả video, nên việc sử dụng CNN sẽ rất dễ có sự nhầm lẫn giữa các hành động.

### 1. 2. 2. Framework mediapipe và Long Short Term Memory

Sử dụng framework Mediapipe và Long Short Term Memory (LSTM) cho bài toán nhận dạng hành động. Ý tưởng cơ bản, đầu vào là một hoặc nhiều video được vẽ các điểm keypoints. Cơ thể của một người sẽ được vẽ 33 keypoints và từ đó trích xuất các điểm tương ứng với mỗi hành động. Đây cũng là đầu vào mạng LSTM là các điểm tương ứng đó và kết quả khi qua mạng LSTM thì đưa ra phân loại cũng như nhận dạng hành đó.

* Ưu điểm: Do có frameworks Mediapipe nên dễ dàng vẽ được keypoints và dễ dàng triển khai mô hình.
* Nhược điểm: Nếu đầu vào là video không được rõ nét thì các điểm keypoints sẽ bị xáo trộn và nhận dạng hành động sai.

### 1. 2. 3. Kết luận

Dựa vào những ưu điểm, nhược điểm đã phân tích ở trên của cả hai thuật toán. Chúng tôi nhận thấy rằng, sử dụng Mediapipe và LSTM sẽ phù hợp với bài toán nhận dạng hành động hơn. Vì vậy, chúng tôi sẽ tập trung trình bày về thuật toán này ở những phần tiếp theo của báo cáo này.

# 

# CHƯƠNG 2. FRAMEWORK MEDIAPIPE VÀ LONG SHORT TERM MEMORY

Để phân loại một hành động, trước tiên cần xác định vị trí các bộ phận cơ thể khác nhau trong mỗi khung hình, sau đó phân tích chuyển động của các bộ phận cơ thể theo thời gian. Bước đầu tiên phải truyền video và sử dụng frameworks Mediapipe xuất ra được tư thế cơ thể (33 điểm chính) sau khi quan sát một khung duy nhất trong video. Bước thứ hai, sử dụng mạng LSTM để phân tích chuyển động của cơ thể theo thời gian và đưa ra dự đoán. Vì vậy, KeyPoint từ một chuỗi các khung được gửi đến LSTM để phân loại hành động.



Hình 2.1. Mô hình tổng quát cho bài toán sử dụng MediaPipe kết hợp LSTM để nhận diện video

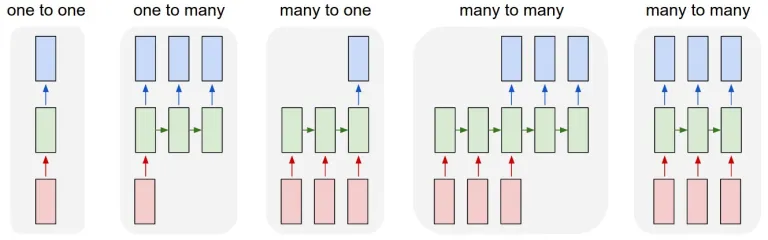
## 2. 1. Detection pose estimation

Về việc xác định vị trí các bộ phận cơ thể khác nhau trong mỗi khung hình thì chúng tôi sẽ sử dụng frameworks Mediapipe của google để xác định các keypoints. Mediapipe có hỗ trợ vẽ các điểm trên khuôn mặt, tay, tư thế. Tuy nhiên trong bài toán này, chúng tôi chỉ sử dụng để vẽ các điểm tư thế.

## 2. 2. Recurrent Neural Network và Long short-term memory

### 2. 2. 1. Recurrent Neural Network (RNN)

**\*Phân loại bài toán RNN:**

****

Hình 2.2. Các dạng bài toán RNN

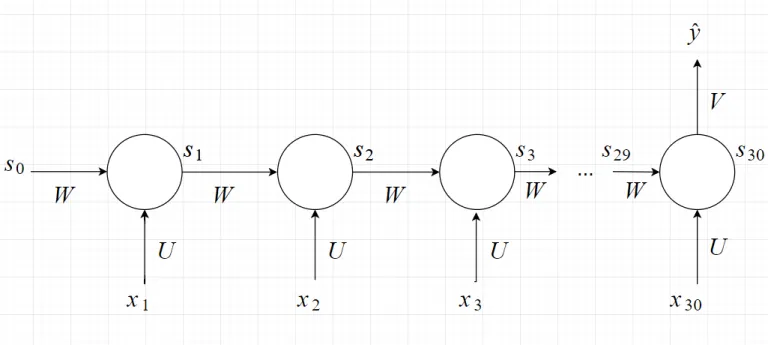
* One to one: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với CNN input là ảnh và output là ảnh được segment.
* One to many: bài toán có 1 input nhưng nhiều output, ví dụ: bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô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

**\*Ứng dụng bài toán RNN:**

* Speech to text: Chuyển giọng nói sang text.
* Sentiment classification: phân loại số sao cho các bình luận, ví dụ: input: “ứng dụng tốt”, output: 4 sao.
* Machine translation: Bài toán dịch tự động giữa các ngôn ngữ.
* Video recognition: Nhận diện hành động trong video.
* Heart attack: Dự đoán đột quỵ tim.
* ….

**\*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) thành các vector có kích thước n\*1. Vector tương ứng với ảnh ở giây thứ i là . Output là vector có kích thước d\*1, softmax function được sử dụng như trong bài phân loại ảnh.

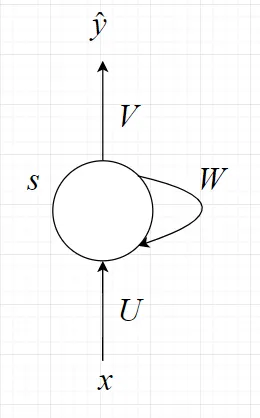


Hình 2.3. Mô hình RNN cho bài toán

Ta có:

* Mô hình có 30 input và 1 output, các input được cho vào model đúng với thứ tự ảnh trong video , , …, .
* Mỗi hình tròn được gọi là 1 state, state t có input là và (output của state trước); output là = f (U\* + W\*). f là activation function, thường là tanh hoặc ReLU.
* Có thể thấy mang cả thông tin từ state trước () và input của state hiện tại => giống như memory nhớ các đặc điểm của các input từ đến .
* 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 ​học được thông tin từ tất cả các input. ​ = g (V \* ). 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 2.4. Mô hình RNN rút gọn

### 2. 2. 2. Long short-term memory (LSTM)

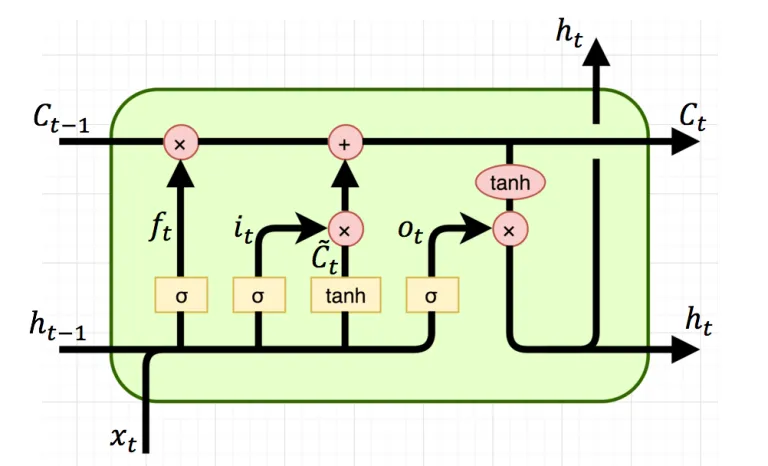
Về lý thuyết là RNN có thể mang thông tin từ các layer trước đến các layer sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng state nhất định, sau đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần đó => short term memory.

Ví dụ về short term memory, bài toán là dự đoán từ tiếp theo trong đoạn văn. Đoạn đầu tiên “Mặt trời mọc ở hướng …”, ta có thể chỉ sử dụng các từ trước trong câu để đoán là đông. Tuy nhiên, với đoạn, “Tôi là người Việt Nam. Tôi đang sống ở nước ngoài. Tôi có thể nói trôi chảy tiếng …” thì rõ ràng là chỉ sử dụng từ trong câu đấy hoặc câu trước là không thể dự đoán được từ cần điền là Việt. Ta cần các thông tin từ state ở trước đó rất xa => cần long term memory điều mà RNN không làm được => Cần một mô hình mới để giải quyết vấn đề này => Long short term memory (LSTM) ra đời.

**\*Mô hình LSTM:**

Ở state thứ t của mô hình LSTM:

* Output: ​, ​, ta gọi c là cell state, h là hidden state.
* Input: ​​, , ​. Trong đó là input ở state thứ t của model. ​, là output của layer trước. h đóng vai trò khá giống như s ở RNN, trong khi c là điểm mới của LSTM.



Hình 2.5. Mô hình LSTM

Cách đọc biểu đồ trên: σ, tanh là bước đấy dùng sigmoid, tanh activation function. Phép nhân ở đây là element-wise multiplication, phép cộng là cộng ma trận.

, , tương ứng với forget gate, input gate và output gate.

* Forget gate: =(\* + \* + )​
* Update gate: = (\* + \* + )​
* Candidate value: = tanh (\* + \* + )
* Output gate: = (\* + \* + )​

Forget gate :

* Giả sử chúng ta đọc các từ trong một đoạn văn bản tiếng anh và dự định sử dụng LSTM để theo dõi các cấu trúc ngữ pháp, chẳng hạn như chủ ngữ là số ít ("puppy") hay số nhiều ("puppies").
* Nếu chủ ngữ thay đổi trạng thái (từ số ít sang từ số nhiều), trí nhớ về trạng thái trước đó trở nên lỗi thời, vì vậy chúng ta “forget” đi trạng thái lỗi thời đó.
* Forget gate là 1 tensor chứa các giá trị nằm trong khoảng từ 0 đến 1.
* Nếu một đơn vị trong forget gate có giá trị gần bằng 0, LSTM sẽ "forget" trạng thái được lưu trữ trong đơn vị tương ứng của trạng thái ô trước đó.
* Nếu một đơn vị trong forget gate có giá trị gần bằng 1, LSTM chủ yếu sẽ ghi nhớ giá trị tương ứng trong trạng thái được lưu trữ.

Candidate value :

* Giá trị ứng cử viên là một tensor chứa thông tin từ bước thời gian hiện tại có thể được lưu trữ trong trạng thái ô hiện tại.
* Candidate value là một tensor chứa giá trị nằm trong khoảng 0 đến 1.

Update gate :

* Sử dụng update gate quyết định xem thêm candidate value vào trạng thái ẩn.
* Update gate là một tensor chứa các giá trị từ 0 đến 1.
* Khi một đơn vị trong update gate gần bằng 1, update gate cho phép giá trị của candidate value được chuyển sang trạng thái ẩn .
* Khi một đơn vị trong update gate gần bằng 0, update gate ngăn không cho giá trị tương ứng trong ứng viên được chuyển sang trạng thái ẩn.

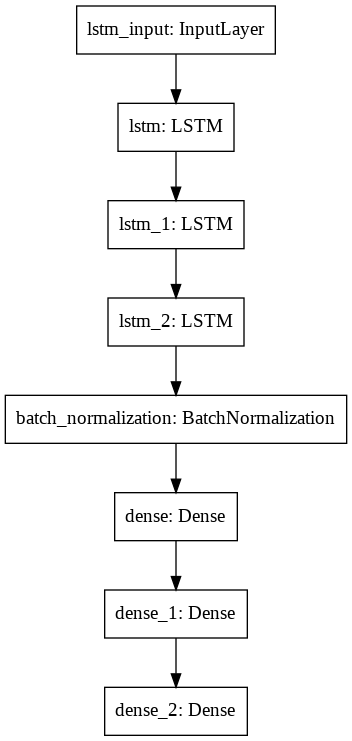
Output gate :

* Cổng đầu ra quyết định những gì được gửi dưới dạng dự đoán (đầu ra) của bước thời gian (time step).
* Cổng ra chứa các giá trị nằm trong khoảng từ 0 đến 1.

=> Nhận xét: 0 < , , < 1​; , , là các hệ số bias; hệ số W, U là trọng lượng tương ứng với ,

=> Nhận xét: ​, khá giống với RNN, nên model có short term memory. Trong khi đó 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.

Trong đề tài này, chúng tôi đã quyết định sẽ chọn mô hình LSTM như hình dưới. Ở chương sau, nội dung này sẽ trình bày chi tiết hơn.



Hình 2.6. Mô hình LSTM cho bài toán nhận diện hành động

# CHƯƠNG 3. THỰC NGHIỆM VÀ KẾT QUẢ

## 3. 1. Giới thiệu các công cụ

### 3. 1. 1. Thư viện Numpy

Numpy là một thư viện mã nguồn mở của Python. Numpy cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần. Numpy cung cấp các chức năng biên dịch nhanh cho các thao tác toán học và số. Hơn nữa, NumPy làm phong phú ngôn ngữ lập trình Python với các cấu trúc dữ liệu mạnh mẽ để tính toán hiệu quả các mảng và ma trận đa chiều. Việc thực hiện là nhằm vào ma trận và mảng khổng lồ. Bên cạnh đó các mô-đun cung cấp một thư viện lớn các chức năng toán học cao cấp để hoạt động trên các ma trận và mảng.

NumPy rất hữu ích cho việc tính toán các mảng nhiều chiều, có kích thước lớn với các hàm số đã được tối ưu áp dụng lên các mảng nhiều chiều đó, đặc biệt là các hàm số liên quan tới Đại Số Tuyến Tính.

Ưu điểm của việc sử dụng Numpy với Python:

* Tính toán theo mảng
* Triển khai hiệu quả các mảng đa chiều
* Được thiết kế để tính toán khoa học

### 3. 1. 2. Thư viện PyQt5

* Qt là một Application framework đa nền tảng viết trên ngôn ngữ C++, được dùng để phát triển các ứng dụng trên desktop, hệ thống nhúng và mobile. Hỗ trợ cho các platform bao gồm: Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS và một số platform khác. PyQt là Python interface của Qt, kết hợp của ngôn ngữ lập trình Python và thư viện Qt, là một thư viện bao gồm các thành phần giao diện điều khiển (widgets, graphical control elements).
* PyQt API bao gồm các module bao gồm số lượng lớn với các classes và functions hỗ trợ cho việc thiết kế ra các giao diện giao tiếp với người dùng của các phần mềm chức năng. Hỗ trợ với Python 2. x và 3. x.
* PyQt được phát triển bởi Riverbank Computing Limited
* Các class của PyQt5 được chia thành các module, bao gồm :

### 3. 1. 3. Thư viện Scikit-learn (Sklearn)

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Những thành phần gồm:

* Numpy: Gói thư viện xử lý dãy số và ma trận nhiều chiều
* SciPy: Gói các hàm tính toán logic khoa học
* Matplotlib: Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
* IPython: Notebook dùng để tương tác trực quan với Python
* SymPy: Gói thư viện các kí tự toán học
* Pandas: Xử lý, phân tích dữ liệu dưới dạng bảng

Những thư viện mở rộng của SciPy thường được đặt tên dạng SciKits. Như thư viện này là gói các lớp, hàm sử dụng trong thuật toán học máy thì được đặt tên là scikit-learn.

Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

Mặc dù được viết cho Python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của C để tăng hiệu suất làm việc. Ví dụ như: Numpy(Tính toán ma trận), LAPACK, LibSVM và Cython.

### 3. 1. 4. Thư viện Matplotlib

Matplotlib là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với Python và Numpy. Matplotlib như là một giải pháp để thực hiện các suy luận thống kê cần thiết.

### 3. 1. 5. Thư viện Keras

Keras là một open source cho Neural Network được viết bởi ngôn ngữ Python. Keras là một library được phát triển vào năm 2015bởi Francois Chollet, là một kỹ sư nghiên cứu Deep Learning. Keras có thể sử dụng chung với các thư viện nổi tiếng như Tensorflow, CNTK, Theano. Một số ưu điểm của Keras như:

* Dễ sử dụng, dùng đơn giản hơn Tensor, xây dựng model nhanh.
* Chạy được trên cả CPU và GPU.
* Hỗ trợ xây dựng CNN, RNN hoặc cả hai. Với những người mới tiếp cận đến Deep thì chọn sử dụng Keras để build model vì Keras rất đơn giản và dễ nắm bắt hơn các thư viện khác.

### 3. 1. 6. Thư viện OpenCV

OpenCV có trên các giao diện C++, C, Python và Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để hỗ trợ hiệu quả về tính toán và chuyên dùng cho các ứng dụng real-time (thời gian thực). Nếu được viết trên C/C++ tối ưu, thư viện này có thể tận dụng được bộ xử lý đa lõi (multi-core processing).

OpenCV có một cộng đồng người dùng khá hùng hậu hoạt động trên khắp thế giới bởi nhu cầu ngày càng tăng theo xu hướng chạy đua về sử dụng computer vision của các công ty công nghệ. OpenCV hiện được ứng dụng rộng rãi toàn cầu, với cộng đồng hơn 47. 000 người, với nhiều mục đích và tính năng khác nhau từ interactive art, đến khai thác mỏ, khai thác web map hoặc qua robotic cao cấp.

OpenCV được sử dụng cho đa dạng nhiều mục đích và ứng dụng khác nhau bao gồm:

* Hình ảnh street view
* Kiểm tra và giám sát tự động
* Robot và xe hơi tự lái
* Phân tích hình ảnh y học
* Tìm kiếm và phục hồi hình ảnh/video
* Phim – cấu trúc 3D từ chuyển động
* Nghệ thuật sắp đặt tương tác

Theo tính năng và ứng dụng của OpenCV, có thể chia thư viện này thánh các nhóm tính năng và module tương ứng như sau:

* Xử lý và hiển thị Hình ảnh/ Video/ I/O (core, imgproc, highgui)
* Phát hiện các vật thể (objdetect, features2d, nonfree)
* Geometry-based monocular hoặc stereo computer vision (calib3d, stitching, videostab)
* Computational photography (photo, video, superres)
* Machine learning & clustering (ml, flann)
* CUDA acceleration (gpu)

### 3. 1. 7. Thư viện Tensorflow

Với sự bùng nổ của lĩnh vực Trí Tuệ Nhân Tạo – AI trong thập kỷ vừa qua, machine learning và deep learning rõ ràng cũng phát triển theo cùng. Và ở thời điểm hiện tại, TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn.

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

Được viết bằng C++ và thao tác interface bằng Python nên phần thực thi của TensorFlow tốt. Đối tượng sử dụng cũng đa dạng không kém: từ các nhà nghiên cứu, nhà khoa học dữ liệu và dĩ nhiên không thể thiếu các lập trình viên.

### 4. 1. 8. Thư viện Mediapipe

Mediapipe là một frameworks chủ yếu được sử dụng để xây dựng âm thanh, video đa phương thức hoặc bất kỳ dữ liệu chuỗi thời gian nào. Với sự trợ giúp của frameworks MediaPipe, các mô hình suy luận như TensorFlow, TFLite và cả các chức năng xử lý phương tiện đều có thể được xây dựng.

MediaPipe cung cấp các giải pháp Python có thể tùy chỉnh dưới dạng một gói Python được tạo sẵn trên PyPI, có thể được cài đặt đơn giản với pip install Mediapipe. Mediapipe cũng cung cấp các công cụ để người dùng xây dựng các giải pháp của riêng họ.

Mediapipe hỗ trợ chúng ta trong việc phát hiện ra khuôn mặt, cử chỉ tay,... Cụ thể như sau:



Hình 3.1. Những lĩnh vực và ngôn ngữ MediaPipe hỗ trợ

## 3. 2. Phân tích bài toán

Bây giờ, bài toán cần đặt ra 2 vấn đề: một là vấn đề về dữ liệu, hai là vấn đề lựa chọn thuật toán và mô hình.

### 3. 2. 1. Vấn đề về dữ liệu

Để nhận dạng một hành động của con người thì cần số lượng dữ liệu rất lớn và dữ liệu ở đây chính các video hành động. Trong đồ án này, chúng tôi chỉ tập trung nhận dạng 5 hành động là đứng, đi bộ, chạy, ngã, vỗ tay. Hiện nay, trên các trang web cũng có rất nhiều bộ dữ liệu sẵn, nhưng những video đó thường có độ phân giải thấp, có các hành động dư thừa, hoặc là không có đủ các bộ phận của cơ thể con người, … Chính vì vậy nên phần lớn dữ liệu của chúng tôi đều là tự thu thập từ các trang web và tự quay. Bởi vì cần số lượng dữ liệu rất lớn nên chúng tôi đã gặp rất nhiều khó khăn trong việc tìm kiếm và thu thập dữ liệu.

### 3. 2. 2. Vấn đề lựa chọn thuật toán và mô hình

Với các bài toán Machine Learning, việc lựa chọn được một thuật toán, mô hình thích hợp với bộ dữ liệu là vô cùng quan trọng. Bên cạnh việc cần có bộ dữ liệu tốt thì thuật toán cũng là nhân tố ảnh hưởng rất lớn kết quả của bài toán. Một điều rõ ràng là với mỗi bài toán sẽ luôn có một cách giải quyết khác nhau. Không có thuật toán nào là thuốc chữa bách bệnh cho tất cả các vấn đề. Chỉ có thuật toán tốt nhất cho bài toán chứ không có thuật toán tốt nhất cho tất cả. Phương pháp được chọn sẽ ảnh hưởng đến độ chính xác, hiệu suất và kích thước của mô hình cuối cùng. Cùng một thuật toán nhưng mỗi bài toán thì lại cần những mô hình khác nhau tùy vào mục đích của bài toán. Ngoài ra, việc tinh chỉnh để tránh các trường hợp underfitting và overfitting xảy ra cũng rất cần thiết. Vì thế, việc chọn ra một thuật toán và mô hình thích hợp là vấn đề nan giải mà chúng tôi phải giải quyết. Như đã trình bày ở chương 1, chúng tôi đã đưa ra 2 thuật toán cho bài toán nhận dạng hành động. Sau khi phân tích những ưu, nhược điểm của các thuật toán khi áp dụng cho bài toán của mình, chúng tôi đã chọn sử dụng Frameworks Mediapipe kết hợp với Long Short Term Memory (LSTM).

## 3. 3. Tiền xử lý dữ liệu

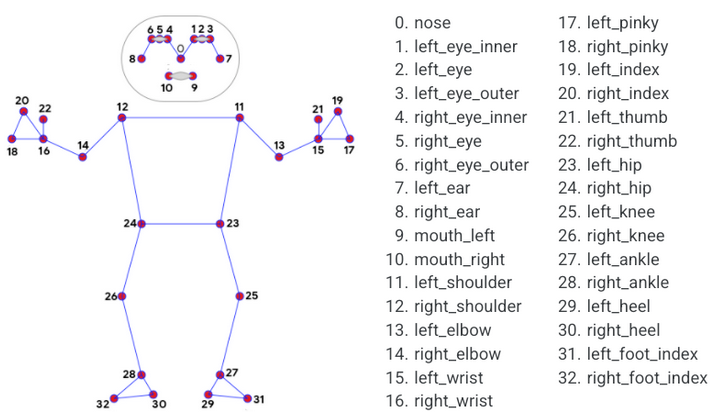
Trong phần phân tích bài toán, chúng tôi đã đề cập về việc gặp rất nhiều khó khăn trong quá trình chuẩn bị dữ liệu và thu thập dữ liệu. Ở phần này, chúng tôi sẽ nói rõ các tiêu chí để lựa chọn 1 tập dữ liệu phù hợp cho bài toán này:

* Dữ liệu phần lớn ở dạng video nên chất lượng video càng sắc nét các tốt thì kết quả bài toán cũng phát triển theo chiều hướng tốt.
* Như đã đề cập trước đó thì các hành động nhận diện dựa vào tư thế nên video cần phải bao quát hết cơ thể để vẽ đủ 33 keypoints.
* Lựa chọn các video ít các điểm outliers.

Khi đã có đủ dữ liệu thì cần xử lý dữ liệu sao cho phù hợp với thuật toán đã lựa chọn trước đó. Chúng tôi đã tiến hành xử lý như sau:

**\* Vẽ các điểm keypoints sử dụng Mediapipe:**

Trước hết cần qua bước xử lý đầu vào bằng cách chuyển ảnh về dạng RGB sau đó đưa qua phương thức *“process”* của Mediapipe để xử lý. Tiếp đến sẽ vẽ các điểm đó với phương thức *“draw\_landmarks”* vàcác điểm tương ứng như sau:



Hình 3.2. 33 keypoints trên cơ thể người

Trên hình 3.2 có thể cho thấy được các điểm tương ứng với các điểm khớp của con người. Có tổng cộng 33 điểm (cột mốc) cần được vẽ và tương ứng với mỗi điểm (cột mốc) thì ta sẽ có các tọa độ x, y, z, visibility. Tọa độ x và y: tọa độ mốc được chuẩn hóa thành [0, 1] theo chiều rộng và chiều cao hình ảnh tương ứng. Tọa độ z: đại diện cho độ sâu mốc với độ sâu ở điểm giữa của hông là nguồn gốc và giá trị càng nhỏ, mốc càng gần máy ảnh. Độ lớn của z sử dụng gần như cùng một tỷ lệ là x. Visibility: Một giá trị trong [0, 1] cho thấy khả năng nhìn thấy mốc có thể nhìn thấy (hiện tại và không bị tắc nghẽn) trong hình ảnh. Và dựa trên các tọa độ này ta sẽ trích xuất cột mốc thành mảng và làm phẳng mảng đó.

**\* Chuyển video sang hình ảnh:**

Trong bài toán này có 5 hành động thì sẽ tạo ra 5 thư mục mỗi thư mục chứa 1285 thư mục con và mỗi thư mục con sẽ chứa 30 frame ảnh. Từng video sẽ được load và vẽ các điểm keypoints. Sau đó, các cột mốc sẽ được trích xuất. Cuối cùng, các frame đó sẽ được lưu thành các file dạng *“. npy”.*

Bài toán này thuộc loại có học giám sát vì vậy cần cặp dữ liệu đầu vào (X, y). X ở đây sẽ là các điểm keypoints tương ứng từng hành động, và y sẽ là nhãn của từng hành động.

Đầu tiên, gắn nhãn cho từng hành động:

*Code tham khảo:*



Tiếp đến, load file đã được lưu dưới dạng “. npy” trước đó:

*Code tham khảo:*



Sau khi load dữ liệu và gắn nhãn xong thì sẽ có được cặp dữ liệu đầu vào tương ứng *(sequences1, labels)* và kích thước biến *“sequences1”* là (6425, 30, 165), còn *“labels”* có kích thước là (6425, 5). Chuyển cặp dữ liệu này về dạng mảng (array) và chuyển “labels” về one hot encoding. Sau đó, chia tập dữ liệu ra 2 phần training và testing.

## 3. 4. Định nghĩa model

Thực ra để lựa chọn model hiệu quả với bài toán thì phụ thuộc rất nhiều yếu tố. Lựa chọn model tốt nhất hay lựa chọn model phù hợp thì cần phải quan tâm tới chi phí cũng như máy móc thiết bị. Trong bài toán, nhóm chúng tôi chọn model phù hợp nhất. Ở những phần trước thì mạng LSTM đã được nhắc đến và ở phần này chúng tôi sẽ xây dựng mạng LSTM để train bài toán này.

Chúng tôi sử dụng các thư viện tensorflow để xây dựng mạng LSTM. Đầu vào của model sẽ là 30 frame và 165 điểm tương ứng các tọa độ của các cột mốc (tọa độ x, y, z, visibility). Chúng tôi sẽ sử dụng thư viện keras vì keras hỗ trợ cách dựng model, và đó chính là Sequential model. Model mà nhóm chúng tôi xây dựng sẽ có 7 layers. Đầu tiên sẽ có 3 lớp *“LSTM”* tương ứng số nơ-ron [64, 128, 128]. Sau đó sẽ có thêm một layer “*Batch Normalization*” để tăng tốc độ mô hình, Batch Normalization sẽ chuyển giá trị mean về gần 0 và độ lệch chuẩn về gần 1, và tính toán từng mini-batch. Tiếp đó sẽ là 2 layers *“Dense”* có số nơron tương ứng [128, 64]. *“Dense”* layers cuối có sẽ có 5 nơ-ron tương ứng với đầu ra của bài toán và một activation softmax để phân loại.

**\*Softmax function:**

Ở mỗi layer sẽ thực hiện 2 bước: tính tổng linear các node ở layer trước và thực hiện activation function (ví dụ sigmoid function, softmax function). Do sau bước tính tổng linear cho ra các giá trị thực nên cần dùng softmax function dùng để chuyển đổi giá trị thực trong các node ở output layer sang giá trị phần trăm.

**\*Loss Function:**

Để định nghĩa loss function, trước hết phải chuyển đổi label của ảnh về dạng one-hot encoding, chuyển label từ giá trị số sang vectơ cùng kích thước với output của model.

Giờ thì giá trị thật (label) dạng one-hot encoding và giá trị dự đoán ở output layer sau hàm softmax function sẽ có cùng kích thước 5\*1. Sau đó, cần phải định nghĩa hàm loss function để đánh giá độ tốt của model.

Hàm loss function trong keras gọi là *“categorical\_crossentropy”.* Với các bài toán classification (phân loại) thì nếu có 2 lớp thì hàm activation ở output layer là hàm sigmoid và hàm loss function là binary\_crossentropy, còn nhiều hơn 2 lớp thì hàm activation ở output layer là hàm softmax với loss function là hàm *“categorical\_crossentropy”*.

*Code tham khảo:*

model = Sequential()

model. add(LSTM(64, return\_sequences=True, activation='relu', input\_shape=(30, 165)))

model. add(LSTM(128, return\_sequences=True, activation='relu'))

model. add(LSTM(128, return\_sequences=False, activation='relu'))

model. add(BatchNormalization())

model. add(Dense(128, activation='relu'))

model. add(Dense(64, activation='relu'))

model. add(Dense(actions. shape[0], activation='softmax'))

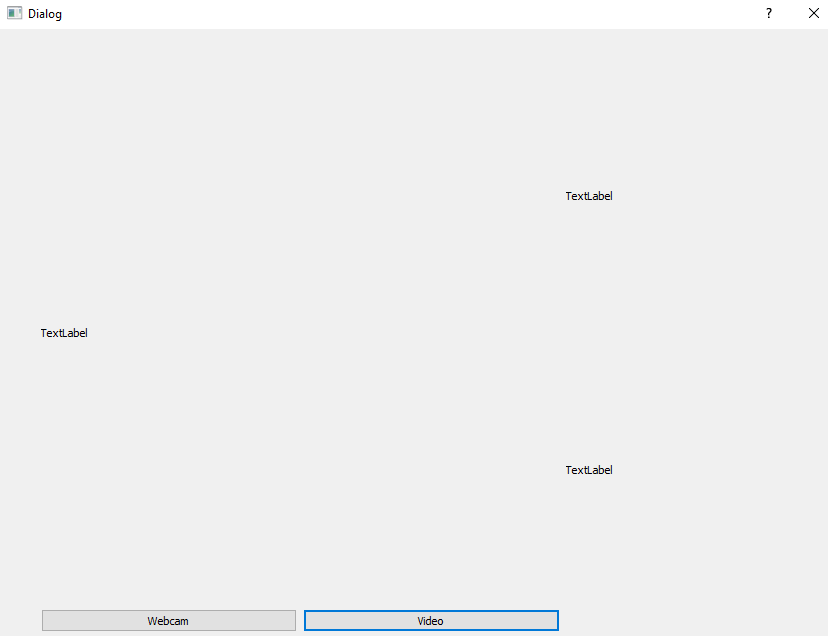
model. summary()

## 3. 5. Tối ưu mô hình, tham số và lưu model

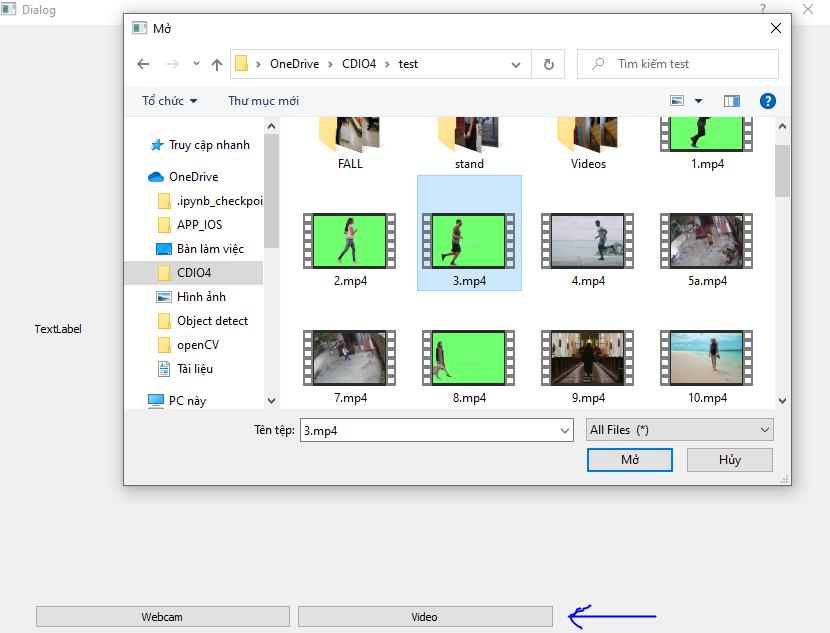
Xây dựng một mạng neuron network hay LSTM thì cần quan tâm đầu vào, đầu ra và số lớp ẩn, tuy nhiên cũng cần lựa chọn số layers sao cho phù hợp bài toán. Lựa chọn số layers không có một quy chuẩn nào cả nhưng để ra số layers phù hợp bài toán thì cần phải thay đổi, training với tập dữ liệu rồi quan sát xem sự thay đổi cũng như quan sát độ chính xác khi tăng hay giảm số layers. Từ đó, có thể chọn được số lớp layers. Để tối ưu các trọng số và bias thì nghĩ ngay đến gradient descent (VD: SGD, Momentum, RMSProp), tuy nhiên trong bài toán nhóm chúng tôi sẽ sử dụng thuật toán tối ưu Adam ,thật ra thuật toán Adam chính là sự kết hợp của Momentum + RMSProp, một thuật toán tối ưu hóa có thể được sử dụng thay cho quy trình giảm độ dốc ngẫu nhiên cổ điển (Stochastic Gradient Descent), để cập nhật lặp đi lặp lại trọng số mạng dựa trên dữ liệu huấn luyện. Còn tham số learning rate là một tham số vô cùng quan trọng nên lựa chọn một cách thật phù hợp, chỉ có một cách chọn learning rate phù hợp đó chúng ta thử nhiều giá trị khác nhau. Và thêm yếu tố weight decay, giảm learning rate theo vòng lặp.

Model sẽ được lưu lại ở định dạng file “.h5” và sử dụng lại cho những lần sau không phải mất công chờ đợi training model.

## 3. 6. Giao diện bài toán.



Hình 3.3. Giao diện 1

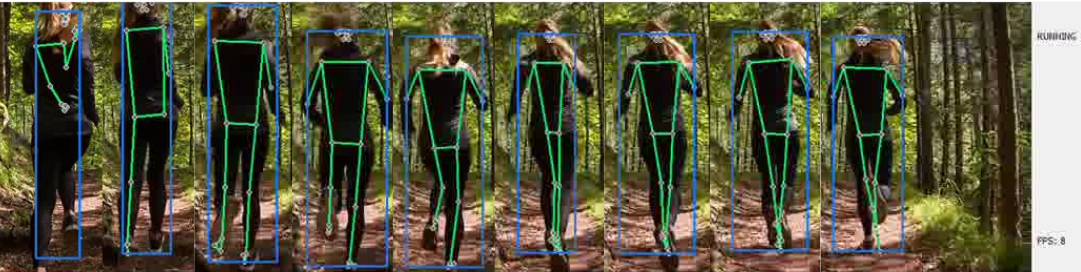


Hình 3.4. Giao diện 2

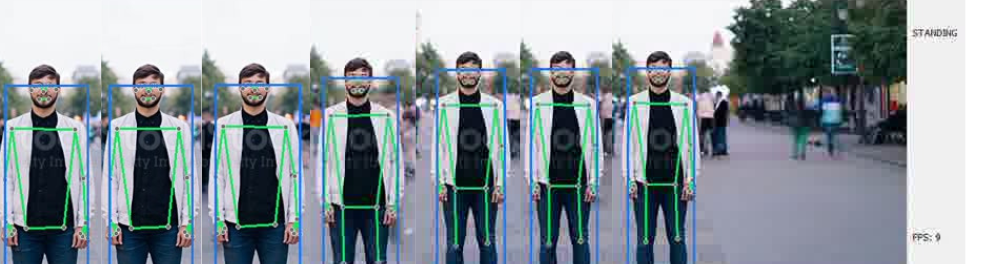
**\* *Một số kết quả thu được:***



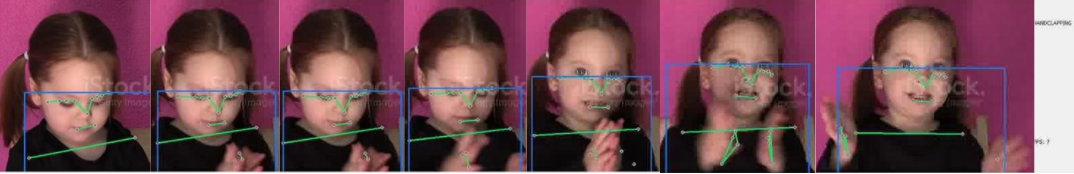
Hình 3.5. Walking



Hình 3.6. Running



Hình 3.7. Standing



Hình 3.8. Hand Clapping

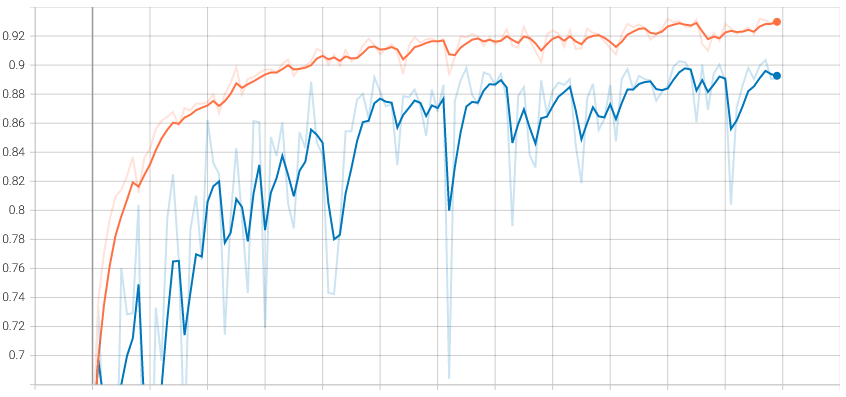


Hình 3.9. Fall down

## 3. 7. Đánh giá mô hình

### 3. 7. 1. Độ chính xác của mô hình

Độ chính xác của mô hình cung cấp độ chính xác tổng thể của mô hình, có nghĩa là phần nhỏ của tổng số mẫu được phân loại chính xác bởi bộ phân loại. Ở đồ thị hình 3.10, trục hoành đại diện cho epoch, càng về bên phải thì epoch càng lớn. Trục trung của đồ thị chỉ độ chính xác của mô hình. Đường màu cam đại diện cho dữ liệu training và đường màu xanh đại diện cho dữ liệu testing. Còn 2 đường màu da cam nhạt và màu xanh nhạt thể hiện sự dự đoán đường đó có thể lên hoặc xuống. Các đường được vẽ ở hình 3.10 dao động lên xuống như vậy bởi vì dữ liệu truyền vào là từng mini-batch chứ không phải từng batch. Các đường này mà có xu hướng đi lên thì mô hình đang hoạt động rất hiệu quả.

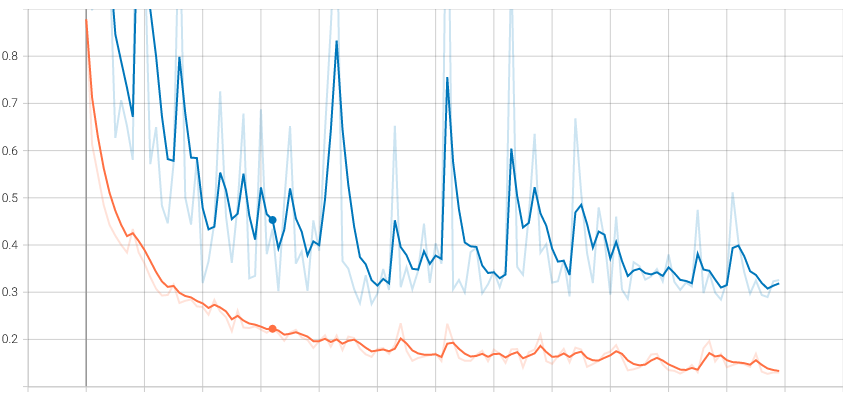
Độ chính xác cao nhất của mô hình với dữ liệu training là 93% và dữ liệu testing là 89%. Chúng tôi đã để giá trị epoch cho bài toán này là 500 epoch. Chúng tôi đã thử nghiệm rất nhiều lần và nhận thấy rằng: nếu số epoch quá thấp thì mô hình chưa khái quát tốt dữ liệu training thì đồng nghĩa với việc mô hình cũng sẽ không khái quát tốt trên tập dữ liệu test. Các epoch các tăng thì độ chính xác cũng có xu hướng tăng, có thể hiểu đơn giản mô hình có thể đang tới gần điểm hội tụ. Có thể thấy thì độ chính xác không tăng hẳn lên mà lên xuống thất thường bời vì dữ liệu được truyền vào theo mini-batch, ở mỗi bước đạo hàm thì nó chỉ đạo hàm tại một mini-batch. Mô hình lúc này sẽ bị underfitting vì vậy chúng tôi để số epoch lớn như vậy và theo dõi. 

Hình 3.10. Độ chính xác của mô hình

### 3. 7. 2. Giá trị hàm mất mát

Giá trị hàm loss có thể hiểu là do sự sai khác giữa giá trị thực tế và giá trị dự đoán. Loss function giống như một hình thức để bắt model đóng phạt mỗi lần dự đoán sai, và số mức phạt tỉ lệ thuận với độ trầm trọng của sai sót. Trong mọi bài toán supervised learning, mục tiêu của bài toán luôn luôn bao gồm cả việc giảm thiểu tổng mức phạt phải đóng. Trong trường hợp lý tưởng y\_hat = y, loss function sẽ trả về giá trị cực tiểu bằng 0. Ở hình 3.11, trục hoành chỉ số epoch và trục tung chỉ giá trị hàm loss, các đường màu cam và màu xanh thì giống đồ thị mô tả độ chính xác. Còn 2 đường màu da cam nhạt và màu xanh nhạt thể hiện sự dự đoán đường đó có thể lên hoặc xuống. Ở hình 3.11, các đường chi phí lỗi không giảm dần theo số vòng lặp như đã gặp ở nhiều bài toán khác. Bởi vì, nhóm chúng tôi sử dụng biến thể gradient descent đó là mini-batch gradient descent, thay vì tính toán toàn bộ dữ liệu trong mỗi lần đạo hàm chúng tôi lấy ngẫu nhiên n lô để tính toán, như vậy sẽ giúp mô hình hoạt động hiệu quả hơn và giảm được thời gian tính toán.

Như phần 3.7.1, chúng tôi đã đề cập tới việc để số lượng epoch lớn và quan sát theo dõi. Tuy nhiên, chúng tôi sẽ không theo dõi từng epoch vì vậy thì rất mất thời gian. Vì vậy, chúng tôi sử dụng kĩ thuật regularization early-stopping. Kỹ thuật này sẽ giúp mô hình không bị overfitting, mô hình sẽ được dừng trước khi đến điểm hội tụ và chúng ta cần dừng thế nào cho hợp lý. Chúng tôi sẽ đặt ngưỡng giá trị và đánh giá chi phí lỗi trên tập training và tập test, nếu như training error đang giảm dần mà test error có xuống hướng tăng và vượt qua giá trị ngưỡng chúng ta sẽ dừng lại. Chúng ta sẽ thấy rằng bài toán chưa thực sự tối ưu, tuy nhiên vẫn đạt được kết quả tốt thay vì mô hình overfitting. Trong hình vẽ ở bên dưới ta có thể thấy training error = 0.1 còn test error = 0.3 . Đây là kết quả tốt, nhưng chưa thật sự tối ưu.



Hình 3.11. Giá trị hàm mất mát

### 3. 7. 3. Ma trận nhầm lẫn, Classification report

Đánh giá độ chính xác mô hình tổng thể như phần ở trên sẽ khó để kiểm soát và đánh giá từng lớp. Vì vậy, chúng tôi có đánh giá mô hình qua ma trận nhầm lẫn, precision, recall.

Ma trận nhầm lẫn là một cách dạng bảng để trực quan hóa hiệu suất của mô hình dự đoán. Mỗi mục trong một ma trận nhầm lẫn biểu thị số lượng dự đoán được thực hiện bởi mô hình nơi mà mô hình phân loại các lớp một cách chính xác hoặc không chính xác.

Bất cứ ai đã quen thuộc với ma trận nhầm lẫn đều biết rằng ma trận nhầm lẫn thường được sử dụng nhiều trong phân loại nhị phân. Trong bài toán này chúng tôi đang làm là bài toán phân loại nhiều lớp nên chúng tôi sẽ tìm hiểu ma trận nhầm lẫn nhiều lớp.

**Các ký hiệu của ma trận nhầm lẫn:**

TP: Định nghĩa cho True Positive cũng giống như trong ma trận nhầm lẫn 2 lớp. Tuy nhiên, ở đây chúng tôi tính các khẳng định đúng cho mỗi lớp trong ma trận nhầm lẫn không giống như các khẳng định đúng tuyệt đối hoặc tổng quát trong ma trận nhầm lẫn 2 lớp.

Số khẳng định đúng là số dự đoán trong đó dữ liệu được gắn nhãn thuộc về một lớp cụ thể đã được phân loại chính xác thành lớp đã nói.

TN: Định nghĩa của True Negative cũng giống như trong ma trận nhầm lẫn 2 lớp. Ở đây chúng tôi tính toán các phủ định thực sự cho mỗi lớp trong ma trận nhầm lẫn không giống như các phủ định đúng nói chung hoặc tuyệt đối trong ma trận nhầm lẫn 2 lớp.

Các phủ định thực sự cho một lớp cụ thể được tính bằng cách lấy tổng các giá trị trong mỗi hàng và cột ngoại trừ hàng và cột của lớp mà chúng tôi đang cố gắng tìm.

FP: Định nghĩa của khẳng định sai giống như trong ma trận nhầm lẫn 2 lớp. Ở đây chúng tôi tính toán các khẳng định sai cho mỗi lớp trong ma trận nhầm lẫn không giống như các khẳng định sai nói chung hoặc tuyệt đối trong ma trận nhầm lẫn 2 lớp.

Giá trị khẳng định sai cho một lớp cụ thể có thể được tính bằng cách lấy tổng của tất cả các giá trị trong cột tương ứng với lớp đó ngoại trừ giá trị khẳng định đúng.

FN: Định nghĩa của phủ định sai giống như trong ma trận nhầm lẫn 2 lớp. Ở đây chúng tôi tính toán phủ định sai cho mỗi lớp trong ma trận nhầm lẫn không giống như vị tướng hoặc tuyệt đối dương tính giả trong ma trận nhầm lẫn 2 lớp.

Các phủ định sai cho một lớp cụ thể có thể được tính bằng cách lấy tổng của tất cả các giá trị trong hàng tương ứng với lớp đó ngoại trừ giá trị phủ định đúng.

Không giống như phân loại nhị phân, không có lớp tích cực hoặc tiêu cực ở đây. Lúc đầu, có thể hơi khó tìm TP, TN, FP và FN vì không có lớp tích cực hay tiêu cực, nhưng điều đó thực sự rất dễ dàng. Những gì phải làm ở đây là tìm TP, TN, FP và FN cho từng lớp riêng lẻ.

**Ma trận nhầm lẫn**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Action | Fall Down  (Thực tế) | Hand Clapping  (Thực tế) | Running  (Thực tế) | Standing  (Thực tế) | Walking  (Thực tế) |
| Fall Down  (Dự đoán) | 243 | 0 | 15 | 0 | 2 |
| Hand Clapping  (Dự đoán) | 1 | 252 | 1 | 1 | 0 |
| Running  (Dự đoán) | 2 | 1 | 248 | 0 | 8 |
| Standing  (Dự đoán) | 2 | 0 | 1 | 249 | 1 |
| Walking  (Dự đoán) | 3 | 0 | 74 | 7 | 174 |

Dựa vào ma trận nhầm lẫn có thể dễ dàng nhận thấy rằng lớp được dự đoán dễ rơi vào lớp nào so với lớp thực. Từ cái nhìn khái quát như vậy chúng ta cần điều chỉnh mô hình sao cho phù hợp để đưa ra một kết quả tốt nhất. Có thể suy ra rằng, hành động running và walking rất dễ có sự nhầm lẫn bởi vì cách hành động của 2 hoạt động này khá giống nhau, các vị trí keypoints chuyển động rất giống nhau và một phần mô hình chưa khái quát tốt với 2 hành động này. Ngược lại thì hành động hand clapping lại có rất ít sự nhầm lẫn vì hành động này chủ yếu dựa vào tư thế cử động của các khớp tay, và cử động tay của hành động này khác hẳn với 4 hành động còn lại.

Bảng phía trên thực chất là 1 ma trận nhầm lẫn 5 \* 5 tương ứng với 5 hành động các giá trị hàng ngang là có thể gọi *“Actual class”* và các giá trị hàng dọc là có thể gọi là *“Predicted class”.*

Ví dụ:

TP = 243

FN = 0 + 15 + 0 + 2 = 17

FP = 1 + 2 + 2 + 3 = 8

Từ ma trận nhầm lẫn ở trên chúng ta có thể đi tính toán precision, recall, f1-score.

**Classification report**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | f1-score | Support |
| Fall Down | 0. 97 | 0. 93 | 0. 95 | 260 |
| Hand Clapping | 1. 00 | 0. 99 | 0. 99 | 255 |
| Running | 0. 73 | 0. 96 | 0. 83 | 259 |
| Standing | 0. 97 | 0. 98 | 0. 98 | 253 |
| Walking | 0. 94 | 0. 67 | 0. 79 | 258 |

Precision: là tỉ lệ của số trường hợp dự đoán đúng thật sự so với tổng số trường hợp của một hành động trong thực tế. Precision càng lớn thì độ chính xác của điểm cần tìm càng cao, mục tiêu lý tưởng là precision là 1, công thức:

TP / (TP + FP).

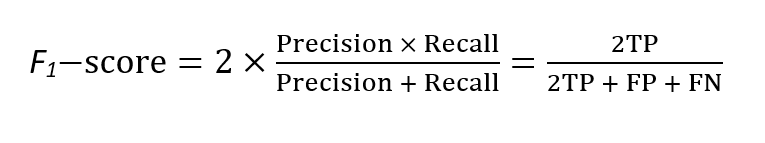
Ví dụ: Fall Down có 243 trường hợp dự đoán đúng và 8 trường hợp là Fall Down nhưng model dự đoán là hành động khác. Vậy precision của Fall Down là 243/(243+8) = 0.97, nghĩa là có 97% hành động ngã là đúng còn 3% là hành động khác.

Recall: là tỉ lệ của số trường hợp dự đoán đúng thật sự so với tổng số trường hợp model dự đoán của một hành động. Recall càng cao thì tỷ lệ bỏ sót càng ít, mục tiêu lý tưởng là recall là 1, công thức:

TP / (TP + FN).

Ví dụ: Fall Down có 243 trường hợp dự đoán đúng và 17 trường hợp là hành động khác nhưng model lại dự đoán là Fall Down. Vậy recall của Fall Down là 243/(243+17) = 0.93, nghĩa là có 93% hành động được mô hình dự đoán đúng và 7% là dự đoán sai.

F1-score: Kết hợp precision và recall thành một biện pháp duy nhất. Để dung hòa precision và recall . Về mặt toán học, đó là ý nghĩa hài hòa của độ chính xác và thu hồi.



Ví dụ: Fall Down (0.95), chính là sự dung hòa giữa precision và recall, precision hay recall chúng ta đều muốn cả 2 đều đạt được mục tiêu lý tưởng là 1, tuy nhiên trong các bài toán thì nó không như thế. Ở đây, 95% có thể phân tách hành động là ngã.

Support: Fall Down (260), chính là số hành động được đánh giá, có tất cả 260 hành động ngã.

Như vậy, từ ma trận nhầm lẫn có thể thấy sự phân bố giữa các lớp. Từ đó, có thể đánh giá số lớp để tinh chỉnh mô hình đạt hiệu quả nhất.

# TỔNG KẾT

## 1. Đánh giá kết quả đạt được

Chúng tôi đã lên ý tưởng, thiết kế và triển khai đề tài theo 2 phương pháp khác nhau, mỗi phương pháp đều cho ra kết quả theo hướng rất tích cực. Kết quả bài toán đã vượt qua sự mong đợi mà chúng tôi đã đặt ra. Mặc dù kết quả đó không phải là tối ưu nhất, nhưng có thể chấp nhận được. Độ chính xác trung bình được đánh giá trên mỗi hành động là 89% và tỷ lệ sai sót là 11%. Bởi vì mô hình phụ thuộc nhiều tố nên kết quả nhận được vẫn còn sai sót.

## 2. Những hạn chế

Đề tài triển khai trong phạm vi thời gian có hạn nên số lượng hành động dự đoán còn hơi ít (chỉ dừng lại ở 5 hành động). Hạn chế về phần cứng cũng là vấn đề rất lớn nên cũng ảnh hưởng lớn tới việc lựa chọn mô hình cho phù hợp chứ không phải mô hình tốt nhất. Chúng tôi đã tốn rất nhiều thời gian để tìm số layers cho phù hợp, cũng tốn nhiều thời gian tìm kiếm dữ liệu, vẫn chưa thể tối ưu bài toán cho cụ thể hơn và vẫn còn nhiều điểm gây nhiễu.

## 3. Hướng phát triển

Chúng tôi sẽ cố gắng triển khai đề tài này vào thực tế để giải quyết vấn đề về theo dõi người già và sẽ tích hợp ứng dụng này vào webcam. Thêm vào đó, chúng tôi sẽ đầu tư thêm phần cứng để có thể lựa chọn được mô hình tốt hơn.

# TÀI LIỆU THAM KHẢO

[1] DuTran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani (12/2015): “Learning Spatiotemporal Features with 3D Convolutional Networks”, ResearchGate

[https: //www. researchgate. net/publication/300408292\_Learning\_Spatiotemporal\_Features\_with\_3D\_Convolutional\_Networks](https://www.researchgate.net/publication/300408292_Learning_Spatiotemporal_Features_with_3D_Convolutional_Networks)

[2] Bibin Sebastian (26/7/2021): “Human Action Recognition using Detectron 2 and LSTM”, LearnOpenCV

[https: //learnopencv. com/human-action-recognition-using-detectron2-and-lstm/](https://learnopencv.com/human-action-recognition-using-detectron2-and-lstm/)

[3] Cuong Tran: “Giới thiệu Numpy”, Vimentor

[https: //vimentor. com/vi/lesson/22-gioi-thieu-numpy](https://vimentor.com/vi/lesson/22-gioi-thieu-numpy)

[4] Google: “MediaPipe Pose”, MediaPipe

[https: //google. github. io/mediapipe/solutions/pose. html](https://google.github.io/mediapipe/solutions/pose.html)

[5] “Giới thiệu Matplotlib - Matplotlib Cơ Bản”, VNCoder

[https: //vncoder. vn/bai-hoc/gioi-thieu-matplotlib-492](https://vncoder.vn/bai-hoc/gioi-thieu-matplotlib-492)

[6] hungdhv97@gmail. com (02/06/2020): “Thư Viện Scikit-learn Trong Python Là Gì”, CODELEARN

[https: //codelearn. io/sharing/scikit-learn-trong-python-la-gi](https://codelearn.io/sharing/scikit-learn-trong-python-la-gi)

[7]JasonBrownlee (24/9/2018): “ LSTM for Human Activity Recognition Time Series Classification”, Machine Learning Mastery

[https: //machinelearningmastery. com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/](https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/)

[8] “Lập trình giao diện với PyQt5 cho RaspberryPi - Phần 1”, mlap

[https: //mlab. vn/index. php?\_route\_=17161-bai-7-lap-trinh-giao-dien-voi-pyqt5-cho-raspberrypi-phan-1. html](https://mlab.vn/index.php?_route_=17161-bai-7-lap-trinh-giao-dien-voi-pyqt5-cho-raspberrypi-phan-1.html)

[9] “OpenCV là gì? Học Computer Vision không khó!”, TOPDev

[https: //topdev. vn/blog/opencv-la-gi-hoc-computer-vision-khong-kho/](https://topdev.vn/blog/opencv-la-gi-hoc-computer-vision-khong-kho/)

[10] To Duc Thang (16/3/2020): “Làm quen với Keras”, VIBLO

[https: //viblo. asia/p/lam-quen-voi-keras-gGJ59mxJ5X2](https://viblo.asia/p/lam-quen-voi-keras-gGJ59mxJ5X2)

[11] Thanh Tuan (25/5/2019): “Recurrent neural network”, Deep Learning cơ bản

[https: //nttuan8. com/bai-13-recurrent-neural-network/](https://nttuan8.com/bai-13-recurrent-neural-network/)

[12] Thanh Tuan (2/6/2019): “Long short term memory”, Deep Learning cơ bản

[https: //nttuan8. com/bai-14-long-short-term-memory-lstm/](https://nttuan8.com/bai-14-long-short-term-memory-lstm/)

[13] “TensorFlow là gì? Tìm hiểu về TensorFlow từ A đến Z”, TOPDev

[https: //topdev. vn/blog/tensorflow-la-gi/](https://topdev.vn/blog/tensorflow-la-gi/)