ĐẠI HỌC QUỐC GIA TP. HCM

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

--------------------

LÊ TRẦN QUANG HUY

**PHÁT HIỆN TÉ NGÃ DỰA VÀO TƯ THẾ CỦA CON NGƯỜI**

**FALL DETECTION BASED ON HUMAN POSE**

Chuyên ngành: Kỹ thuật Điều khiển và Tự động hóa

Mã số: 8520216

**LUẬN VĂN THẠC SĨ**

TP. HỒ CHÍ MINH, tháng 12 năm 2023

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC BÁCH KHOA –ĐHQG -HCM

Cán bộ hướng dẫn Khóa luận tốt nghiệp: TS. Phạm Việt Cường

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 1:

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 2:

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Khóa luận tốt nghiệp được bảo vệ tại Trường Đại học Bách Khoa, ĐHQG Tp.HCM, ngày tháng năm 2024

Thành phần Hội đồng đánh giá khoá luận tốt nghiệp gồm:

(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ khóa luận tốt nghiệp)

1. ... - Chủ tịch

2. ... - Phản biện 1

3. ... - Phản biện 2

4. ... - Thư ký

5. ... - Ủy viên

Xác nhận của Chủ tịch Hội đồng đánh giá khóa luận tốt nghiệp và Chủ nhiệm Bộ môn sau khi luận văn đã được sửa chữa (nếu có).

**CHỦ TỊCH HỘI ĐỒNG TRƯỞNG KHOA ĐIỆN – ĐIỆN TỬ**

| TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH  **KHOA ĐIỆN – ĐIỆN TỬ**  **BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG** | CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập - Tự do - Hạnh phúc |
| --- | --- |
|  | *TP. HCM, ngày….tháng…..năm……..* |

**NHIỆM VỤ LUẬN VĂN THẠC SĨ**

| Họ tên học viên: LÊ TRẦN QUANG HUY | MSHV: 2170710 |
| --- | --- |
| Ngày, tháng, năm sinh: 02/04/1999 | Nơi sinh: Tiền Giang |
| Chuyên ngành: Kỹ thuật Điều khiển và Tự động hóa | Mã số: 8520216 |

**I. TÊN ĐỀ TÀI:**

Phát hiện té ngã dựa vào tư thế của con người

Fall detection based on human pose

**II. NHIỆM VỤ VÀ NỘI DUNG:**

Sử dụng mạng YOLOv8 nano để phát hiện vùng ảnh nào có người. Tiến hành thu thập dữ liệu con người ở nhiều góc độ khác nhau đặc biệt là các tư thế khi té ngã cho việc huấn luyện.

Sử dụng mô hình theo dõi đối tượng DeepSORT để theo dõi và gán ID cho từng người đã được phát hiện bằng YOLOv8 nano trước đó, ID của mỗi người cần đảm bảo luôn không đổi qua các frame, để đảm bảo cho bước tiếp theo là phát hiện tư thế và ghép các tư thế lại thành một chuỗi hành động là đúng với từng người.

Sử dụng Movenet để phát hiện tư thế cho từng người đã được gán id trước đó, thông tin tư thế người ở đây là các điểm nút trên cơ thể người như đầu, vai, khớp tay, khớp chân,.. và ghép các tư thế này lại tạo thành thông tin của một chuỗi hành động ngắn.

Sử dụng mạng LSTM để phát hiện người ngã dựa vào chuỗi hành động ngắn. Tiến hành lấy dữ liệu huấn luyện bằng mô hình Movenet.

Triển khai, kết hợp các mô hình lại với nhau để có thể dự đoán té ngã cho nhiều người. Tối ưu và tăng tốc hệ thống.

**III. NGÀY GIAO NHIỆM VỤ:** 04/09/2023

**IV. NGÀY HOÀN THÀNH NHIỆM VỤ:** 18/12/2023

**V. CÁN BỘ HƯỚNG DẪN:** TS. Phạm Việt Cường

| **CÁN BỘ HƯỚNG DẪN**  (Họ tên và chữ ký) | *Tp. HCM, ngày . . . . tháng .. . . năm 20....*  **CHỦ NHIỆM BỘ MÔN ĐÀO TẠO**  (Họ tên và chữ ký) |
| --- | --- |
| **TRƯỞNG KHOA ĐIỆN – ĐIỆN TỬ**  (Họ tên và chữ ký) | |

**LỜI CẢM ƠN**

Tôi muốn bày tỏ lòng biết ơn sâu sắc đến Tiến sĩ Phạm Việt Cường, người đã đồng hành và hướng dẫn tôi suốt quá trình nghiên cứu và hoàn thiện luận văn. Sự kiên nhẫn và sự chỉ dạy tận tâm của Thầy, cùng với những góp ý quý báu, không chỉ giúp tôi hoàn thành luận văn một cách xuất sắc mà còn là nguồn động viên lớn để phát triển khả năng nghiên cứu và sáng tạo của bản thân.

Tôi cũng muốn bày tỏ lòng biết ơn đến các thầy cô trong bộ môn Điều khiển và Tự động hóa, cũng như các giảng viên tại Đại học Bách Khoa Thành phố Hồ Chí Minh. Kiến thức và kỹ năng mà tôi học được từ các thầy cô là nền tảng quan trọng giúp tôi hiểu rõ hơn về ngành và thực hiện luận văn của mình một cách xuất sắc. Sự tận tâm và sẻ chia không ngần ngại của các thầy cô đã góp phần quan trọng vào sự phát triển và thành công của tôi.

Cuối cùng, tôi muốn gửi lời cảm ơn chân thành đến gia đình, người thân và bạn bè, nguồn động viên vô cùng quý báu suốt chặng đường học tập và nghiên cứu của tôi. Sự ấm áp và tình yêu thương từ các bạn đã là nguồn động viên lớn giúp tôi vượt qua khó khăn và đạt được những thành tựu quan trọng.

Tôi rất biết ơn và hy vọng sẽ tiếp tục nhận được sự hỗ trợ và động viên từ thầy cô, gia đình và bạn bè trong những bước tiếp theo của cuộc sống và sự nghiệp. Xin chân thành cảm ơn và kính chúc mọi người luôn dồi dào sức khỏe và thành công trong mọi công việc.

|  | TP. Hồ Chí Minh, ngày 24 tháng 05 năm 2023  Học viên thực hiện  Lê Trần Quang Huy |
| --- | --- |

**TÓM TẮT**

Té ngã là một trong những tai nạn rất thường hay xảy ra trong đời sống, nó tiềm ẩn rất nhiều nguy cơ như gãy xương, vỡ đốt sống, hay thậm chí là tử vong. Đặc biệt, té ngã càng nguy hiểm hơn đối với người cao tuổi bởi sự suy yếu của hệ thống miễn dịch cũng như sức khỏe của độ tuổi.

Việc phát hiện sớm các trường hợp té ngã, đặc biệt là ở người cao tuổi, có thể giúp các thành viên trong gia đình, người chăm sóc hoặc bác sĩ đến bên nạn nhân kịp thời và có thể giúp hạn chế hậu quả của việc té ngã. Do đó nhu cầu về hệ thống phát hiện té ngã và có khả năng thông báo sớm cho người chăm sóc hoặc gia đình là rất cấp thiết.

Hiện nay camera an ninh đã dần được phổ biến không chỉ ở nơi công cộng mà còn ở nhà riêng. Chính vì vậy, tôi đề xuất một phương pháp phát hiện té ngã bằng thị giác máy dựa trên tư thế của con người. Luận văn dùng camera để phát hiện và theo dõi hành vi người trong khung hình, trích xuất các đặc trưng về tư thế và ghép các đặc trưng này lại thành các chuỗi hành động ngắn, từ đó sẽ dự đoán việc té ngã thông qua các chuỗi hành động này.

Kết quả thử nghiệm cho thấy hệ thống cơ bản có thể nhận phát hiện theo dõi và dự doán về trạng thái của một người có té ngã hay không, tốc độ khung hình khi chỉ có 1 người đủ để đáp ứng trên thời gian thực. Bên cạnh đó, hệ thống còn hạn chế khi số người trong khung hình tăng thì hệ thống sẽ bị chậm lại, do quá trình dự đoán sẽ tăng thêm.

**ABSTRACT**

Falling is a common accident in daily life, carrying various risks such as fractures, spinal injuries, or even fatalities. It poses greater danger, particularly to the elderly due to weakened immune systems and age-related health vulnerabilities.

Early detection of falls, especially in the elderly, can facilitate timely assistance from family members, caregivers, or medical professionals, potentially mitigating the aftermath of falls. Thus, the need for a fall detection system capable of early notification to caregivers or families is crucial.

Currently, security cameras have become increasingly prevalent, not only in public spaces but also in private homes. Hence, I propose a computer vision-based fall detection method utilizing human poses. The thesis employs cameras to detect and track human behavior within frames, extracting pose features and assembling them into short action sequences to predict falls based on these sequences.

Experimental results demonstrate that the basic system can detect, track, and predict whether a person has fallen or not, meeting real-time frame rates when monitoring a single individual. However, limitations arise when multiple individuals are present, causing system slowdowns due to increased prediction processe.

**LỜI CAM ĐOAN**

Tôi xin cam đoan rằng đề tài "Phát hiện té ngã dựa vào tư thế của con người" là công trình nghiên cứu của bản thân. Tất cả các nội dung, phân tích, và kết quả được trình bày trong đề tài này đều là thành quả của công việc nghiên cứu cá nhân của tôi.

Tất cả các thông tin, số liệu, và hình ảnh từ các nguồn tài liệu tham khảo đều được trích dẫn và thể hiện rõ ràng trong đề tài. Tôi cam đoan rằng không có sự vi phạm bản quyền hoặc vi phạm quyền sở hữu trí tuệ nào trong quá trình thực hiện nghiên cứu và viết luận văn này.

Tôi chịu trách nhiệm hoàn toàn trước bộ môn, khoa, và nhà trường về sự cam đoan này. Nếu có bất kỳ sự không trung thực nào trong luận văn này, tôi sẵn sàng chịu hoàn toàn trách nhiệm và chấp nhận mọi kỷ luật và hình phạt theo quy định của bộ môn và nhà trường.

|  | HỌ TÊN HỌC VIÊN  Lê Trần Quang Huy |
| --- | --- |

Mục lục

[CHƯƠNG 1.](#_heading=h.30j0zll) TỔNG QUAN ĐỀ TÀI 1

[1.1](#_heading=h.1fob9te) Lý do chọn đề tài 1

[1.2](#_heading=h.2et92p0) Những công trình nghiên cứu gần đây 2

[1.3](#_heading=h.tyjcwt) Mục tiêu đề tài 3

[1.4](#_heading=h.3dy6vkm) Nhiệm vụ đề tài 3

[CHƯƠNG 2.](#_heading=h.1t3h5sf) CƠ SỞ LÝ THUYẾT 5

[2.1](#_heading=h.4d34og8) Nhận dạng đối tượng 5

[2.1.1](#_heading=h.2s8eyo1) Giới thiệu YOLOv8 5

[2.1.2](#_heading=h.3rdcrjn) Kiến trúc YOLOv8 7

[2.1.3](#_heading=h.2jxsxqh) Anchor-Free Detection 10

[2.1.4](#_heading=h.z337ya) Training trên Google Colab 11

[2.2](#_heading=h.1y810tw) Thuật toán theo dõi Deep SORT 11

[2.2.1](#_heading=h.4i7ojhp) Giới thiệu Deep SORT 11

[2.2.2](#_heading=h.1ci93xb) Luồng xử lý của Deep SORT 13

[2.2.3](#_heading=h.3whwml4) Data Association 14

[2.2.4](#_heading=h.3as4poj) Track Life Cycle Management 17

[2.3](#_heading=h.1pxezwc) Movenet 17

[2.3.1](#_heading=h.49x2ik5) Giới thiệu Movenet 18

[2.3.2](#_heading=h.3o7alnk) Ước lượng tư thế người bằng Movenet 19

[2.4](#_heading=h.ihv636) LSTM - Long short term memory 19

[2.4.1](#_heading=h.32hioqz) Giới thiệu LSTM 19

[2.4.2](#_heading=h.vx1227) Thứ tự các bước của LSTM 21

[CHƯƠNG 3.](#_heading=h.19c6y18) THỰC HIỆN HỆ THỐNG 23

[3.1](#_heading=h.3tbugp1) Xây dựng mô hình YOLOv8 nano phát hiện người 23

[3.1.1](#_heading=h.nmf14n) Mô hình tổng quát 23

[3.1.2](#_heading=h.37m2jsg) Bộ dữ liệu của YOLOv8 nano 24

[3.1.3](#_heading=h.206ipza) Huấn luyện mô hình YOLOv8 nano trên môi trường colab 26

[3.2](#_heading=h.2zbgiuw) Theo dõi đối tượng 27

[3.3](#_heading=h.3ygebqi) Xây dựng mô hình LSTM nhận diện té ngã 28

[3.3.1](#_heading=h.2dlolyb) Mô hình tổng quát 29

[3.3.2](#_heading=h.3cqmetx) Bộ dữ liệu của LSTM 29

[3.3.3](#_heading=h.2r0uhxc) Huấn luyện mô hình LSTM trên môi trường colab 31

[3.4](#_heading=h.kgcv8k) Triển khai và tối ưu hóa hệ thống 34

[3.4.1](#_heading=h.34g0dwd) Tối ưu cho quá trình nhận diện người 34

[3.4.2](#_heading=h.1jlao46) Tối ưu hóa cho quá trình nhận dạng té ngã 35

[3.4.3](#_heading=h.43ky6rz) Triển khai hệ thống 37

[CHƯƠNG 4.](#_heading=h.xvir7l) KẾT QUẢ 38

[4.1](#_heading=h.3hv69ve) Mô hình phát hiện người: 38

[4.1.1](#_heading=h.1x0gk37) Đánh giá kết quả huấn luyện mô hình YOLOv8 nano 38

[4.1.2](#_heading=h.2afmg28) Đánh giá kết trên thực nghiệm 39

[4.2](#_heading=h.48pi1tg) Mô hình nhận diện người té ngã: 40

[4.2.1](#_heading=h.3mzq4wv) Đánh giá kết quả huấn luyện mô hình LSTM 40

[4.2.2](#_heading=h.1gf8i83) Đánh giá kết quả dựa trên thực nghiệm 41

[4.3](#_heading=h.2fk6b3p) Tổng hợp hệ thống: 41

[4.3.1](#_heading=h.4iylrwe) Đánh giá thực nghiệm 41

[4.3.2](#_heading=h.2y3w247) Khảo sát tốc độ xử lý 41

[CHƯƠNG 5.](#_heading=h.meukdy) KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 43

[5.1](#_heading=h.1d96cc0) Những kết quả đạt được 43

[5.2](#_heading=h.2koq656) Những hạn chế còn tồn tại 43

[5.3](#_heading=h.zu0gcz) Hướng phát triển 44

[TÀI LIỆU THAM KHẢO 45](#_heading=h.3jtnz0s)

DANH MỤC HÌNH VẼ

[Hình 2.1 So sánh tốc độ của các phiên bản YOLO trên tập dữ liệu MS COCO 5](#_heading=h.17dp8vu)

[Hình 2.2 Mô hình chung của YOLO 6](#_heading=h.26in1rg)

[Hình 2.3 Kiến trúc của YOLOv8 7](#_heading=h.lnxbz9)

[Hình 2.4 So sánh block C3 của YOLOv5 và C2f của YOLOv8 8](#_heading=h.35nkun2)

[Hình 2.5 So sánh phần cổ của YOLOv5 và YOLOv8 8](#_heading=h.1ksv4uv)

[Hình 2.6 So sánh phần đầu của YOLOv5 và YOLOv8 9](#_heading=h.44sinio)

[Hình 2.7 Logo của Google Colaboratory 10](#_heading=h.3j2qqm3)

[Hình 2.8 Mô hình tổng quát thuật toán Deep SORT kết hợp với một Object Detetor là YOLOv8 nano 11](#_heading=h.2xcytpi)

[Hình 2.9 Minh họa IOU 13](#_heading=h.2bn6wsx)

[Hình 2.10 Tổng quan kiến trúc CNN của Deep SORT sử dụng 15](#_heading=h.qsh70q)

[Hình 2.11 một vài hình ảnh demo của Movenet 17](#_heading=h.147n2zr)

[Hình 2.12 Đầu ra các điểm đặc trưng mà Movenet trả về 18](#_heading=h.23ckvvd)

[Hình 2.13 Minh họa RNN 19](#_heading=h.1hmsyys)

[Hình 2.14 Minh họa LSTM 19](#_heading=h.41mghml)

[Hình 2.15 Ô trạng thái của mạng LSTM 20](#_heading=h.2grqrue)

[Hình 2.16 Cổng quên của mạng LSTM 20](#_heading=h.3fwokq0)

[Hình 2.17 Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cổng vào và tẩng ẩn hàm tanh của mạng LSTM 21](#_heading=h.1v1yuxt)

[Hình 2.18 Ô trạng thái mới của mạng LSTM 22](#_heading=h.4f1mdlm)

[Hình 2.19 Điều chỉnh thông tin ở đầu ra thông qua hàm tanh của mạng LSTM 22](#_heading=h.2u6wntf)

[Hình 3.1 Tổng quan hệ thống phát hiện người té ngã 23](#_heading=h.28h4qwu)

[Hình 3.2 Tổng quan mô hình nhận diện người té ngã 24](#_heading=h.37m2jsg)

[Hình 3.3 Một vài ảnh trong tập dữ liệu nhận diện người 25](#_heading=h.46r0co2)

[Hình 3.4 Phần mềm LabelImg 25](#_heading=h.2lwamvv)

[Hình 3.5 Thông tin label được lưu vào file txt 26](#_heading=h.111kx3o)

[Hình 3.6 Cấu trúc tập dữ liệu của YOLOv8 nano 26](#_heading=h.3l18frh)

[Hình 3.7 CLI hiển thị quá trình huấn luyện 28](#_heading=h.4k668n3)

[Hình 3.8 Kết quả nhận diện và theo dõi sản phẩm của YOLOv8 và DeepSORT 29](#_heading=h.1egqt2p)

[Hình 3.9 Tổng quan mô hình nhận nhận diện té ngã 30](#_heading=h.sqyw64)

[Hình 3.10 Một vài ảnh từ video dữ liệu té ngã ở nhiều góc độ khác nhau 30](#_heading=h.1rvwp1q)

[Hình 3.11 Chuỗi 8 frame ảnh liên tiếp 31](#_heading=h.4bvk7pj)

[Hình 3.12 Thông tin tham số của mạng LSTM 34](#_heading=h.1664s55)

[Hình 3.13 CLI quá trình huấn luyện 34](#_heading=h.25b2l0r)

[Hình 3.14 Lưu đồ giải thuật của hệ thống 37](#_heading=h.2iq8gzs)

[Hình 4.1 Đồ thị quá trình huấn luyện phát hiện người 39](#_heading=h.1baon6m)

[Hình 4.2 Kết quả thực nghiệm phát hiện người kết hợp theo dõi thứ nhất 40](#_heading=h.pkwqa1)

[Hình 4.3 Kết quả thực nghiệm phát hiện người kết hợp theo dõi thứ hai 40](#_heading=h.39kk8xu)

[Hình 4.4 Kết quả thực nghiệm phát hiện người kết hợp theo dõi thứ ba 41](#_heading=h.1opuj5n)

[Hình 4.5 Đồ thị đánh giá mô hình LSTM 42](#_heading=h.2250f4o)

[Hình 4.6 Precision mỗi class của mô hình nhận diện té ngã 43](#_heading=h.haapch)

[Hình 4.7 Recall mỗi class của mô hình nhận diện té ngã 44](#_heading=h.319y80a)

[Hình 4.8 F1 score mỗi class của mô hình nhận diện té ngã 45](#_heading=h.40ew0vw)

[Hình 4.9 Lần thử thứ nhất của mẫu 1 46](#_heading=h.upglbi)

[Hình 4.10 Lần thử thứ hai của mẫu 1 46](#_heading=h.3ep43zb)

[Hình 4.11 Lần thử thứ nhất của mẫu 2 47](#_heading=h.1tuee74)

[Hình 4.12 Lần thử thứ hai của mẫu 2 47](#_heading=h.4du1wux)

[Hình 4.13 Lần thử thứ nhất của mẫu 3 48](#_heading=h.2szc72q)

[Hình 4.14 Lần thử thứ hai của mẫu 3 48](#_heading=h.184mhaj)

[Hình 4.15 Ba người cùng ngã 51](#_heading=h.36ei31r)

[Hình 4.16 Hai người cùng ngã cùng thời điểm 52](#_heading=h.1ljsd9k)

[Hình 4.17 Hai người ngã hai thời điểm liên tiếp nhau 52](#_heading=h.45jfvxd)

DANH MỤC BẢNG

[Bảng 4.1 Kết quả sau khi huấn luyện phát hiện người với tập validation 38](#_heading=h.2w5ecyt)

[Bảng 4.2 Kết quả sau khi huấn luyện phát hiện người với tập test 39](#_heading=h.3vac5uf)

[Bảng 4.3 Thời gian thực thi trên mỗi khung hình của mô hình phát hiện người 41](#_heading=h.2nusc19)

[Bảng 4.4 Thời gian thực thi trên mỗi khung hình của mô hình nhận diện té ngã 49](#_heading=h.3s49zyc)

[Bảng 4.5 Thời gian thực thi trên mỗi khung hình của hệ thống phát hiện người té ngã dựa vào tư thế 50](#_heading=h.279ka65)

**DANH MỤC TỪ VIẾT TẮT**

| **Số thứ tự** | **Tên viết tắt** | **Tên đầy đủ** |
| --- | --- | --- |
| 1 | RNN | Recurrent neural network |
| 2 | LSTM | Long Short Term Memory |
| 3 | ANN | Artificial Neural Network |
| 4 | CNN | Convolutional Neural Networks |
| 5 | ReLU | Rectified Linear Unit |
| 6 | IOU | Intersection Over Union |

# TỔNG QUAN ĐỀ TÀI

## Lý do chọn đề tài

Té ngã là một trong những tai nạn rất thường hay xảy ra trong đời sống, nó tiềm ẩn rất nhiều nguy cơ như gãy xương, vỡ đốt sống, hay thậm chí là tử vong. Đặc biệt, té ngã càng nguy hiểm hơn đối với người cao tuổi bởi sự suy yếu của hệ thống miễn dịch cũng như sức khỏe của độ tuổi.

Theo báo cáo vào năm 2008 của Tổ chức y tế thế giới (WHO), cho thấy tỉ lệ người cao tuổi bị té ngã mỗi năm là khoảng 28% - 35% đối với những người có tuổi từ 65 tuổi trở lên và 32% - 42% đối với những người trên 75 tuổi [12]. Theo một báo cáo mới hơn vào năm 2018 của Trung tâm kiểm soát và phòng ngừa dịch bệnh Hoa Kỳ (CDC), trong một năm tại Mỹ, cứ 4 người từ 65 tuổi trở lên thì có 1 người bị té ngã và khoảng 37% người té ngã cho biết họ bị chấn thương cần được điều trị y tế. [13]

Tại Việt Nam, theo bài viết vào năm 2020 của báo Sức khỏe và Đời sống (Cơ quan ngôn luận của bộ y tế Việt Nam), tính có khoảng 1,5 - 1,9 triệu người cao tuổi bị té ngã mỗi năm, 5% trong số đó phải nhập viện vì các chấn thương. Tại Khoa Lão - Chăm sóc giảm nhẹ Bệnh viện Đại học Y Dược TP.HCM (BV ĐHYD TP.HCM), mỗi tháng có khoảng 17% người bệnh nhập viện do té ngã hoặc do các biến cố liên quan tới té ngã. Theo TS BS. Thân Hà Ngọc Thể - Trưởng khoa Lão - Chăm sóc giảm nhẹ BV ĐHYD TPHCM, té ngã có thể dẫn đến nhiều hệ lụy nghiêm trọng đối với sức khỏe NCT. Đây là nguyên nhân hàng đầu gây chấn thương với hơn 50% trường hợp bị té ngã tại nhà và là nguyên nhân thứ 2 gây chấn thương não và tủy sống ở NCT. Tỉ lệ chấn thương do té ngã ở NCT là 10 - 25%, trong đó thương tích nghiêm trọng chiếm khoảng 5 - 15%. Trong các chấn thương do té ngã thì gãy xương chiếm tỉ lệ cao nhất với 87%, trong đó hơn 95% trường hợp bị gãy xương hông. Té ngã cũng nằm trong 5 nguyên nhân gây tử vong ở người trên 65 tuổi với tỉ lệ tử vong tăng theo tuổi. [11]

Việc phát hiện sớm các trường hợp té ngã, đặc biệt là ở người cao tuổi, có thể giúp các thành viên trong gia đình, người chăm sóc hoặc bác sĩ đến bên nạn nhân kịp thời và có thể giúp hạn chế hậu quả của việc té ngã. Do đó nhu cầu về hệ thống phát hiện té ngã và có khả năng thông báo sớm cho người chăm sóc hoặc gia đình là rất cấp thiết. Hiện nay camera an ninh đã dần được phổ biến không chỉ ở nơi công cộng mà còn ở nhà riêng. Chính vì vậy, tôi đề xuất một phương pháp phát hiện té ngã bằng thị giác máy dựa trên tư thế của con người, phương pháp này sẽ dự đoán việc té ngã thông qua 1 chuỗi tư thế của con người.

## Những công trình nghiên cứu gần đây

Bài viết **Vision-Based Fall Detection with Convolutional Neural Networks -** Adrián Núñez-Marcos, Gorka Azkune, and Ignacio Arganda-Carreras(2017 Dec)**[3]** Tác giả đưa ra giải pháp dựa trên thị giác máy bằng cách sử dụng một mạng Nơ ron tích chập (Convolutional Neural Networks) để xác định xem trong 1 chuỗi ảnh liên tiếp thì có người nào ngã không. Tác giả sử dụng hình ảnh luồng quang (optical flow images) làm đầu vào cho mạng nơ ron tích chập, sau đó là giai đoạn đào tạo ba bước mới. Sau khi thực hiện thì tác giả tiến hành đánh giá với 3 tập dữ liệu công khai với kết quả khá tốt là URFD (accuracy 95%), Multicam (accuracy 97%) và FDD (accuracy 97%).

Mặc dù kết quả của tác giả đưa ra là khá tốt và ý tưởng dự đoán một chuỗi ảnh liên tiếp rất hay, nhưng phần dữ liệu đầu vào của mạng CNN là ảnh optical flow, vì vậy cần phải có một hệ thống camera tương đối tốt để có thể chuyển từ ảnh thường sang ảnh optical flow. Ngoài ra, mạng CNN của tác giả chỉ dự doán trong 1 chuỗi ảnh liên tiếp có người ngã không, mà không biết trước được trong ảnh có người hay không, môi trường phải tĩnh dễ tránh việc nhiễu do có vật chuyển động trong khung ảnh.

Bài viết **Human-skeleton based Fall-Detection Method using LSTM for Manufacturing Industries -** Sungjoo Kang (2019 Jun) **[1]** Tác giả đưa ra giải pháp dựa trên thị giác máy bằng cách sử dụng một mạng LSTM để phát hiện việc té ngã. Tác giả cho đầu vào của LSTM là một chuỗi dữ liệu khung xương của con người (hay có thể nói là tư thế của người), được trích xuất từ thư viện OpenPose. LSTM là một kiến trúc học sâu mạnh mẽ trong việc xử lý việc phân loại dữ liệu chuỗi thời gian. Ngoài ra tác giả còn kết hợp thêm một số phương pháp tăng cường khi trích xuất đặc trưng, như HCLC (tọa độ trung tâm cơ tể người) và SHCLC (tốc độ của khung xương ở 2 khung hình liên tiếp) để tăng đặc trưng cho dữ liệu. Kết quả đạt được với tập dữ liệu công khai URFD rất tốt, accuracy cao nhất lên đến 98.83%.

Tuy nhiên việc trích xuất đặc trưng bằng thư viện OpenPose khá chậm, nên không khả thi với các hệ thống thời gian thực (realtime), tác giả cũng không đề cập đến thời gian thực thi của 1 lần dự đoán, và trong 1 khung hình tác giả chỉ dự đoán được 1 người có đang ngã hay không.

## Mục tiêu đề tài

Sau khi tham khảo các bài báo và nghiên cứu , hầu hết các bài nghiên cứu trước đó chỉ tập trung vào việc dự đoán trạng thái té ngã của một người tại một thời điểm, chưa có bài viết nào đề cập đến việc chạy trên thời gian thực. Nên mục tiêu của đề tài này là dựa trên ý tưởng nghiên cứu của Sungjoo Kang về việc dự đoán việc té ngã bằng LSTM, xây dựng một hệ thống có thể phát hiện và dự đoán té ngã cho nhiều người trong cùng 1 khung ảnh. Cố gắng tối ưu hệ thống để có thể chạy có thể chạy chạy trên thời gian thực.

Hệ thống sẽ phát hiện những người có trong khung ảnh và đưa ra dự đoán về trạng thái của từng người:

- Phát hiện từng người trong khung hình.

- Theo dõi và đánh số thứ tự cho từng người.

- Trích xuất đặc trưng tư thế của từng người người dựa vào việc phát các điểm nút trên cơ thể từng người.

- Ghép các đặc trưng tư thế lại thành một chuỗi hành động ngắn.

- Sau khi có được các chuỗi hành động ngắn của từng người, tiến hành dự đoán trạng thái của từng người từ đó phát hiện việc té ngã.

## Nhiệm vụ đề tài

Sử dụng mạng YOLOv8 nano để phát hiện vùng ảnh nào có người. Tiến hành thu thập dữ liệu con người ở nhiều góc độ khác nhau đặc biệt là các tư thế khi té ngã cho việc huấn luyện.

Sử dụng mô hình theo dõi đối tượng DeepSORT để theo dõi và gán ID cho từng người đã được phát hiện bằng YOLOv8 nano trước đó, ID của mỗi người cần đảm bảo luôn không đổi qua các frame, để đảm bảo cho bước tiếp theo là phát hiện tư thế và ghép các tư thế lại thành một chuỗi hành động là đúng với từng người.

Sử dụng Movenet để phát hiện tư thế cho từng người đã được gán id trước đó, thông tin tư thế người ở đây là các điểm nút trên cơ thể người như đầu, vai, khớp tay, khớp chân,.. và ghép các tư thế này lại tạo thành thông tin của một chuỗi hành động ngắn.

Sử dụng mạng LSTM để phát hiện người ngã dựa vào chuỗi hành động ngắn. Tiến hành lấy dữ liệu huấn luyện bằng mô hình Movenet.

Triển khai, kết hợp các mô hình lại với nhau để có thể dự đoán té ngã cho nhiều người. Tối ưu và tăng tốc hệ thống.

Khảo sát thực nghiệm và đánh giá.

# CƠ SỞ LÝ THUYẾT

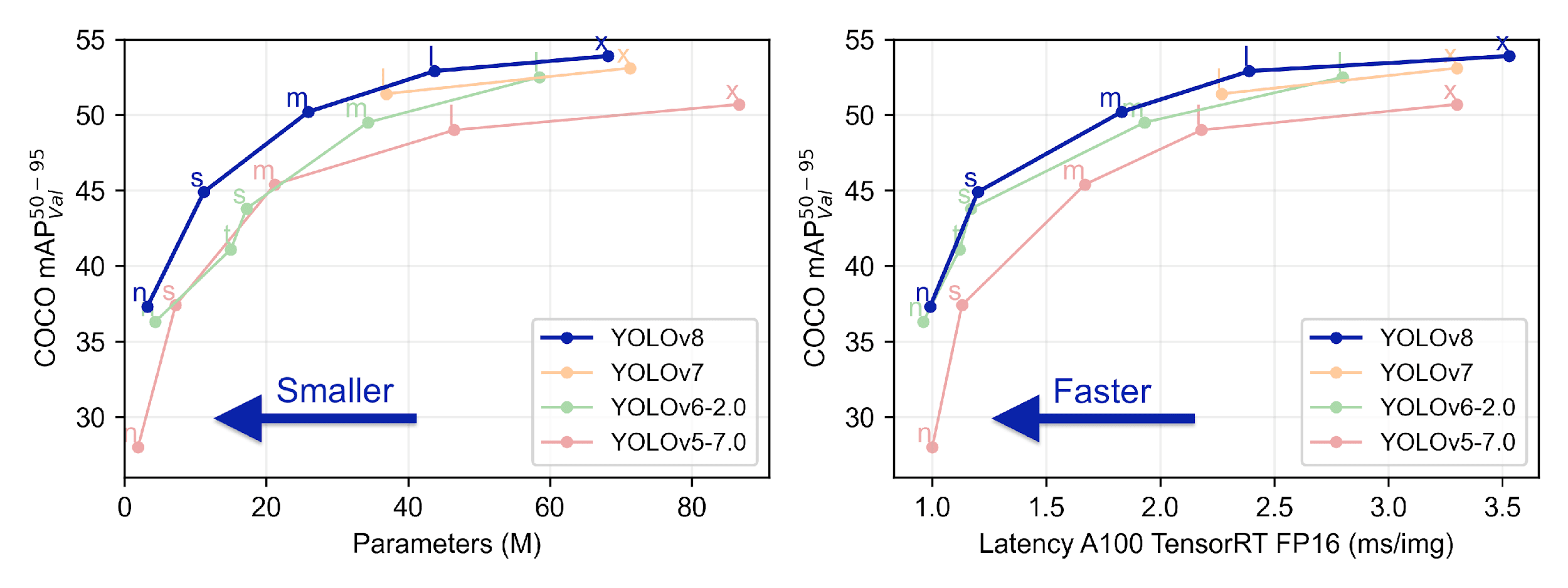
## Nhận dạng đối tượng

### Giới thiệu YOLOv8

YOLO - You Only Look Once là một trong những mô hình phát hiện vật tốt nhất ở thời điểm hiện tại. Dù đều được gọi là YOLO, các phiên bản của mô hình này đều có những cải tiến rất đáng kể sau mỗi phiên bản. Sau 3 phiên bản của tác giả chính Joseph Redmon là YOLOv1 đến v3, tính đến thời điểm hiện tại đã có thêm những phiên bản khác như: YOLOv4 của Alexey Bochkovskiy, YOLOv5 của Glenn Jocher, YOLOv6 của nhóm nghiên cứu Meituan, YOLOv7 Chien-Yao Wang, Alexey và Bochkovskiy và gần đây nhất là YOOLOv8 của nhóm Ultralytics.

YOLOv8 là mô hình one stage detectors (cùng lúc sẽ thực hiện cả việc tìm kiếm bounding box và classification cho một vật thể), chính vì đặc điểm này mà YOLO có tốc độ khá nhanh và độ chính xác cao.

YOLOv8 đặc biệt hơn so với các bản YOLO trước đó vì nó được ra mắt kèm theo framework Ultralytics, giúp cho việc huấn luyện và triển khai mạng dễ dàng hơn.

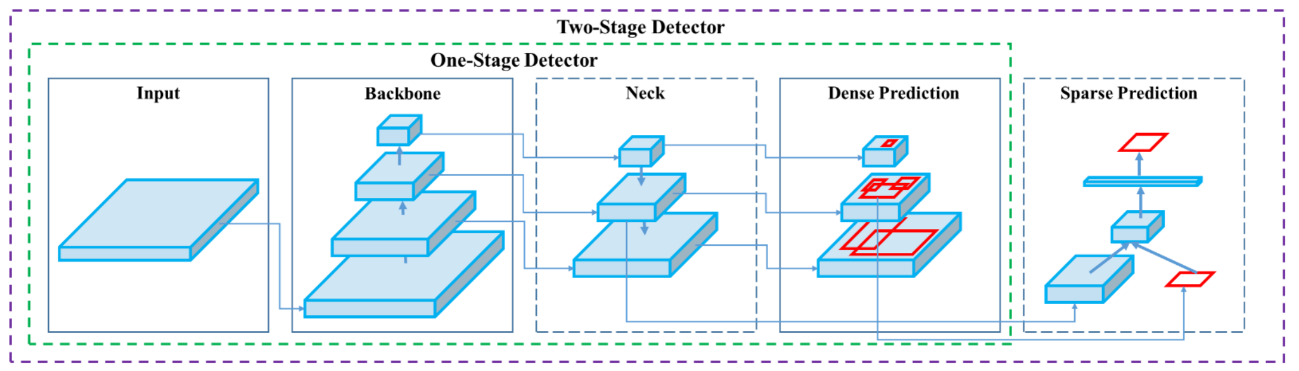


*Hình 2.1 So sánh tốc độ của các phiên bản YOLO trên tập dữ liệu MS COCO*

Có nhiều biến thể của YOLOv8, bao gồm YOLOv8x, YOLOv8l, YOLOv8m, YOLOv8s và YOLOv8xn. Trong luận văn này, phiên bản YOLOv8n, phiên bản nhỏ gọn của YOLOv8, được chọn để thực hiện quá trình transfer learning. Việc này nhằm đảm bảo rằng hệ thống có khả năng hoạt động ở tốc độ thời gian thực.

### Kiến trúc YOLOv8

Kiến trúc của YOLO nói chung:



*Hình 2.2 Mô hình chung của YOLO*

Kiến trúc của một mạng YOLO nói chung sẽ thường được chia làm ba phần: xương sống (backbone), cổ (neck) và đầu (head).

Trong đó phần xương sống được sử dụng để trích xuất các đặc trưng từ ảnh đầu vào. Backbone thường bao gồm các mạng nơ-ron tích chập (CNNs) như Darknet, ResNet, hoặc EfficientNet. Nhiệm vụ chính của backbone là chuyển đổi ảnh đầu vào thành biểu diễn có chất lượng cao hơn, chứa thông tin đặc trưng về các đối tượng có trong ảnh.

Phần cổ đặt giữa backbone và head và thường chịu trách nhiệm để kết hợp các đặc trưng từ các tầng khác nhau của backbone. Có thể sử dụng các kỹ thuật như Feature Pyramid Network (FPN) hoặc Path Aggregation Network (PAN) để tăng cường thông tin đa quy mô, giúp mô hình có khả năng dự đoán đối tượng ở các tỷ lệ và kích thước khác nhau trong ảnh.

Và phần cuối cùng của mô hình là đầu Head thường là nơi thực hiện các dự đoán về vị trí và lớp của các đối tượng trong ảnh. Đối với YOLO, phần đầu thường gồm các lớp tích chập để dự đoán các hộp giới hạn (bounding boxes) và lớp xác suất để phân loại đối tượng. Phần đầu cũng thường chịu trách nhiệm điều chỉnh kích thước và định dạng của các bounding boxes để chứa chính xác các đối tượng được dự đoán

Các mô hình YOLO ban đầu được phát triển bởi Joseph Redmon, một nhà khoa học máy tính. Ông đã tạo ra ba phiên bản YOLO, trong đó phiên bản cuối cùng của ông là YOLOv3, được viết bằng kiến trúc Darknet. Glenn Jocher đã triển khai phiên bản YOLOv3 trên PyTorch với một số thay đổi nhỏ và đặt tên nó là YOLOv5. Sau đó, kiến trúc của YOLOv5 đã được tinh chỉnh để phát triển thành YOLOv8 bởi Ultralytics, vì vậy cấu trúc của nó gần như YOLOv5 nhưng được hiệu chỉnh ở một số điểm nhằm mang lại kết quả tốt hơn.

A screenshot of a computer screen

Description automatically generated

*Hình 2.3 Kiến trúc của YOLOv8*

YOLOv8 sử dụng mạng CSPDarknet53 được tùy biến lại để làm phần xương sống. CSPDarknet53 sử dụng mô hình mạng thần kinh tích chập (CNN) để trích xuất các đặc trưng từ hình ảnh. Phần xương sống hoàn chỉnh của YOLOv8 là tập hợp của các C2F Block, các Conv block và SPPF block.

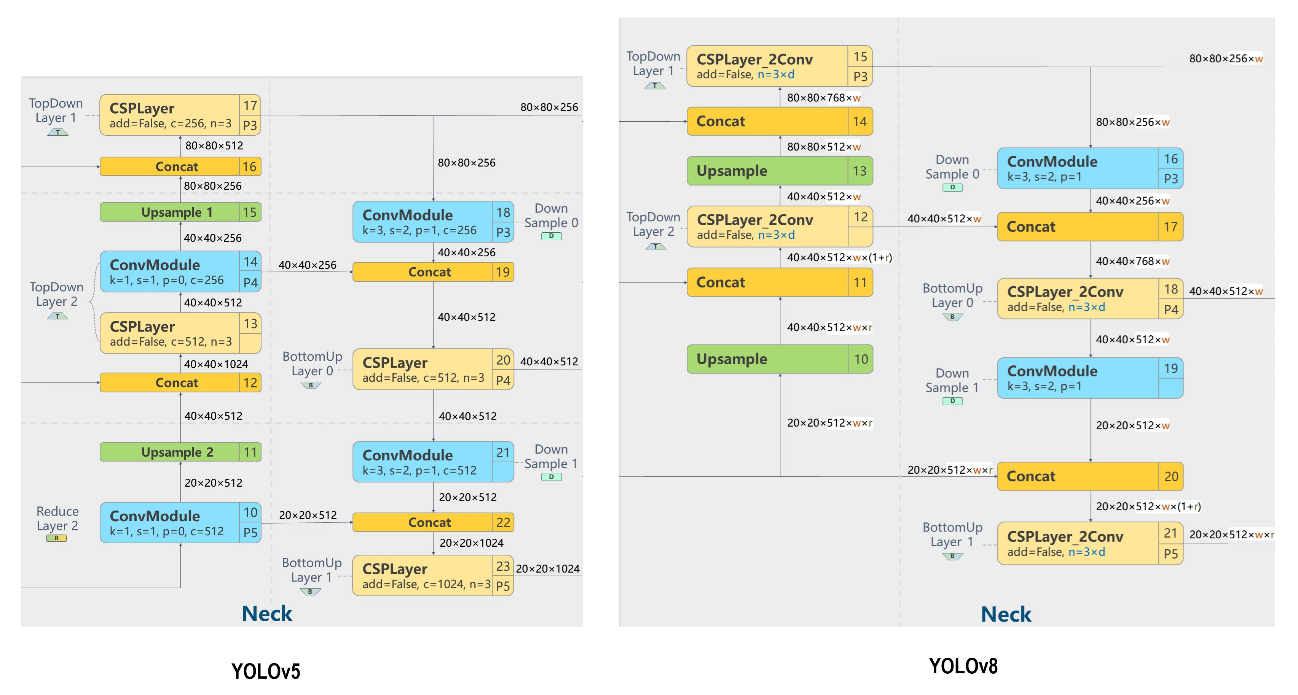
Trong phần xương sống thì YOLOv8 thay thế các C3 block của YOLOv5 thành các C2F block hoàn toàn mới.

A screenshot of a computer program

Description automatically generated

*Hình 2.4 So sánh block C3 của YOLOv5 và C2f của YOLOv8*

Phần cổ của YOLOv8 sử dụng kiến trúc FPN (Feature Pyramid Network). FPN sử dụng một mạng CNN để kết hợp các đặc trưng từ các cấp độ khác nhau của phần xương sống. Điều này giúp mô hình có thể phát hiện các đối tượng ở nhiều kích cỡ khác nhau. Phần cổ của Yolov8 thay thế các C3 block của YOLOv5 thành các C2F block và đồng thời loại bỏ 2 Conv block ở đầu.



*Hình 2.5 So sánh phần cổ của YOLOv5 và YOLOv8*

Phần đầu của YOLOv8 cũng là phần thay đổi đáng kể nhất, nó đã được thay đổi từ cấu trúc coupled head sang cấu trúc decoupled head. Đồng thời chuyển từ phương pháp Anchor-based sang Anchor-Free.

A diagram of a diagram of a head

Description automatically generated with medium confidence

*Hình 2.6 So sánh phần đầu của YOLOv5 và YOLOv8*

### Anchor-Free Detection

Đối với các phiên bản YOLO kể từ YOLOv2, để tìm được chính xác boundingbox cho vật thể thì YOLO đã sử dụng phương pháp Anchor-based detection. Anchor-based detection sử dụng các anchor boxes có kích thước và tỷ lệ đa dạng được đặt trên các vị trí cố định trên ảnh. Các anchor boxes giúp dự đoán các tọa độ và kích thước của bounding boxes bằng cách ánh xạ từ các offset (độ lệch) so với anchor boxes đã được định nghĩa trước.

Tuy nhiên việc sử dụng anchor boxes cũng có một số hạn chế sau:

* Huấn luyện với các anchor boxes có sẵn làm cho mô hình trở không tổng quát vì các anchor boxes sẽ quá cứng nhắc.
* Anchor boxes không thể hiệu quả khi đối tượng quá nhỏ hoặc quá lớn so với kích thước của anchor boxes. Điều này có thể dẫn đến việc mô hình không thể dự đoán chính xác vị trí hoặc kích thước của đối tượng.
* Tốn nhiều tài nguyên hơn để xử lý.

Vì vậy, YOLOv8 sử dụng phương pháp Anchor-Free Detection. Trong YOLOv8, mô hình được huấn luyện để dự đoán các khung bounding box của các đối tượng trong hình ảnh bằng cách sử dụng các hàm kích thước cố định (fixed-size functions). Các hàm này giúp cho mô hình có thể phát hiện các đối tượng ở nhiều kích cỡ và tỷ lệ khác nhau.

### Training trên Google Colab

Colaboratory, còn được gọi là Google Colab, là một sản phẩm từ Google Research. Nó cung cấp một môi trường hoạt động dựa trên trình duyệt cho việc chạy mã Python, đặc biệt thích hợp cho việc phân tích dữ liệu, học máy và giáo dục. Colab không đòi hỏi việc cài đặt hoặc cấu hình máy tính, mọi thứ có thể chạy thông qua trình duyệt. Bạn có thể sử dụng tài nguyên máy tính với CPU tốc độ cao cũng như GPU và TPU được cung cấp. Colab cung cấp nhiều loại GPU, thường là Nvidia K80s, T4s, P4s và P100s. Tuy nhiên, người dùng không thể chọn loại GPU trong Colab, vì GPU trong Colab thay đổi theo thời gian. Vì là một dịch vụ miễn phí, Colab có các ưu tiên và giới hạn về việc sử dụng tài nguyên hệ thống. Thời gian sử dụng tối đa là 12 giờ. Để khắc phục giới hạn này, tôi tải dữ liệu lên Google Drive để kết nối với Colab. Với khả năng kết hợp với Google Drive, Google Colab cho phép đọc và lưu dữ liệu nhanh chóng và tiện lợi hơn.



*Hình 2.7 Logo của Google Colaboratory*

## Thuật toán theo dõi Deep SORT

### Giới thiệu Deep SORT

Trong lĩnh vực xử lý ảnh bằng học máy, việc theo dõi và phân loại nhiều đối tượng qua các khung hình là một thách thức quan trọng. Điều này đặc biệt khó khăn khi gặp phải các tình huống như thay đổi góc quan sát hoặc sự chồng chất của các đối tượng. Trong bối cảnh này, Deep SORT - một thuật toán được phát triển bởi Nicolai Wojke và Alex Bewley - nổi bật với khả năng giải quyết những vấn đề này. Deep SORT là một phiên bản nâng cấp của thuật toán SORT ra đời trước đó không lâu. Ý tưởng của Deep SORT là dựa trên việc sử dụng deep learning để trích xuất ra các đặc trưng của từng đối tượng, từ đó việc đánh số ID cho đối tượng và liên kết giữa việc xác định (detection) và theo dõi (track) đối tượng qua các khung hình hiệu quả hơn, có độ chính xác cao hơn. Điều này đặc biệt hữu ích trong việc xử lý số lượng lớn ID, đây vấn đề mà thuật toán SORT trước đó còn rất hạn chế. Đáng chú ý, Deep SORT linh hoạt trong việc sử dụng các mô hình nhận diện như, tùy thuộc vào nhu cầu cụ thể của ứng dụng.

A diagram of a deep-sort

Description automatically generated

*Hình 2.8 Mô hình tổng quát thuật toán Deep SORT kết hợp với một Object Detetor là YOLOv8 nano*

Trong bài toán multiple object tracking, đặc biệt là đối với các thuật toán tracking-by-detection, nghĩa là phần detection và phần tracking được tách thành 2 bài toán độc lập, thì có 2 vấn đề quan trọng cần được xử lý để bài toán có thể đạt được hiệu suất tốt nhất:

* Data Association: quá trình kết hợp các detection mới với các track tương ứng đã được lưu trữ trước đó. Quá trình này phải có các tiêu chí đánh giá tốt để có thể tracking đối tượng qua các khung hình liên tiếp.
* Track Life Cycle Management: quá trình quản lý vòng đời của các track theo thời gian. Nó bao gồm việc tạo, cập nhật và loại bỏ các track dựa trên thông tin mới nhất từ việc detection và data association. Quản lý vòng đời của track giúp theo dõi đối tượng qua các khung hình video, bao gồm việc xác định khi nào một track nên được bắt đầu, cập nhật track với các detection mới và khi nào track nên kết thúc hoặc bị loại bỏ nếu đối tượng không còn trong khung hình nữa.

Nhóm tác giả của Deep SORT cũng sử dụng thuật toán Hungarian để giải quyết phần data association như thuật toán SORT trước đó, tuy nhiên họ đã mở rộng phương pháp này và khiến nó trở nên vượt trội hơn. Thay vì chỉ dựa vào IOU (Intersection over Union), họ còn xem xét các yếu tố khác như khoảng cách của detection và track trong không gian vector và khoảng cách cosine giữa hai vector đặc trưng được trích xuất từ detection và track tương ứng. Việc này giúp tăng cường sự chính xác trong việc gán các detections vào các track, đặc biệt là khi xét đến đặc trưng của cùng một đối tượng.

### Luồng xử lý của Deep SORT

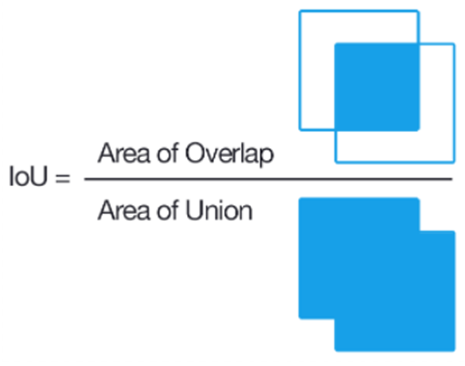
* Phát hiện đối tượng: Deep sort sẽ kết hợp với một mô hình nhận diện vật thể khác (ở luận văn này là YOLOv8 nano), để có thể phát hiện đối tượng (detection).
* Trích xuất đặc trưng: Điểm khác biệt lớn nhất là Deep SORT sẽ sử một mạng convolutional neural networks để trích xuất đặc trưng. Các đặc trưng này thể hiện các đối tượng trong không gian nhiều chiều, giúp việc so sánh và liên kết giữa các detection và các track hiệu quả hơn.
* Dự đoán chuyển động: Deep SORT sẽ sử dụng Kalman Filter để đưa ra dự đoán về vị trí mới của các đối tượng dựa vào thông tin của các frame trước đó được lưu ở các track.
* Liên kết dữ liệu: Sau khi có được các đặc trưng của từng đối tượng, cùng với dự đoán về vị trí mới của các đối tượng (track predicted) và thông tin về vị trí mới của các dối tượng từ các detection, Deep SORT sẽ tiến hành liên kết các detetction với các track dựa trên thuật toán Hungarian đã được mở rộng sao cho sai số của việc liên kết là nhỏ nhất. Cuối cùng là trả về kết quả id của từng đối tượng.

### Data Association

**Thuật toán Hungarian**

Thuật toán Hungarian ra mắt vào năm 1955, với mục đích là để giải quyết bài toán phân công công việc sao cho chi phí tổng cộng là nhỏ nhất. Ở đây nó được dùng để giải quyết sự tương quan giữa cặp trạng thái gồm: trạng thái dự đoán ước lượng bởi bộ lọc Kalman (track) và trạng thái đo được từ bộ phát hiện (detection). Chi phí ở đây là chỉ số được sử dụng để đánh giá ở đây là giá trị IOU giữa các bouding box của track và detection.

IoU (Intersection over union) : là tỉ lệ giữa đo lường mức độ giao nhau giữa hai bouding box (ở bài toán này là bouding box của track và detection). Tỷ lệ này được tính dựa trên phần diện tích giao nhau giữa 2 đường bao với phần tổng diện tích giao nhau và không giao nhau giữa chúng.

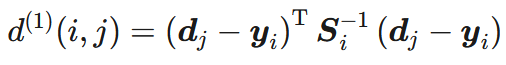


*Hình 2.9 Minh họa IOU*

Nhóm tác giả của Deep SORT đã mở rộng thuật toán Hungarian bằng việc bổ sung thêm hai chỉ số khác được sử dụng để đạt được sự tương quan là thông tin chuyển động (tính theo khoảng cách Mahalanobis) và đặc điểm ngoại hình (tính theo khoảng cách cosine). Cuối cùng, kết hợp 2 chỉ số với trọng số phù hợp, deep SORT tạo ra một chỉ mới.

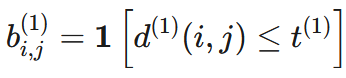
*Khoảng cách Mahalanobis tương quan chuyển động:*

Khoảng cách Mahalanobis, là một phương pháp hiệu quả để tính toán độ tương quan giữa hai tập hợp mẫu chưa biết, đo lường mức độ phù hợp của dự đoán và phát hiện, hay đo lường khoảng cách giữa track và detection:



Trong đó (yi,Si) là giá trị kỳ vọng và ma trận covariance của biến ngẫu nhiên track thứ i, và dj là giá trị của detection thứ j.

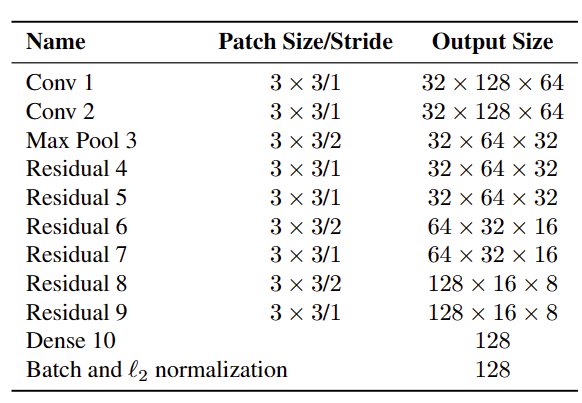
Khoảng cách Mahalanobis còn được sử dùng để loại bỏ các liên kết không chắc chắn bằng cách lập ngưỡng khoảng cách Mahalanobis ở khoảng tin cậy 95% được tính từ phân phối .



Với =9.4877

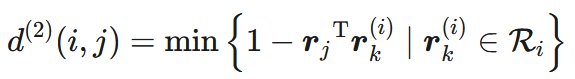
*Khoảng cách Cosine tương quan đặc điểm ngoại hình:*

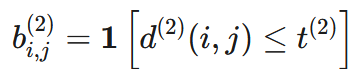
Deep SORT sử dụng một phương pháp đo lường khác về đặc trưng của đối tượng để đảm bảo khả năng liên kết chính xác ngay cả khi đối tượng biến mất và sau đó tái xuất hiện trong khung hình. Điều quan trọng đó là các đặc trưng. Với mỗi detection, các đặc trưng này được học từ mạng ReID network. Kiến trúc này được thiết kế chỉ với số lượng lớp rất nhỏ (16 lớp), nhưng vẫn mang lại hiệu suất vượt trội hơn so với các kiến trúc có hàng nghìn lớp khác. Điều đặc biệt là thời gian huấn luyện và dự đoán cũng nhanh hơn đáng kể.



*Hình 2.10 Tổng quan kiến trúc CNN của Deep SORT sử dụng*

Với mỗi detection, đặc trưng rj được trích xuất với ||rj ||=1. Với mỗi track, một danh sách với độ dài khoảng 100 được sử dụng để lưu trữ đặc trưng của 100 track gần nhất: . Khi đó, khoảng cách cosine được tính như sau:





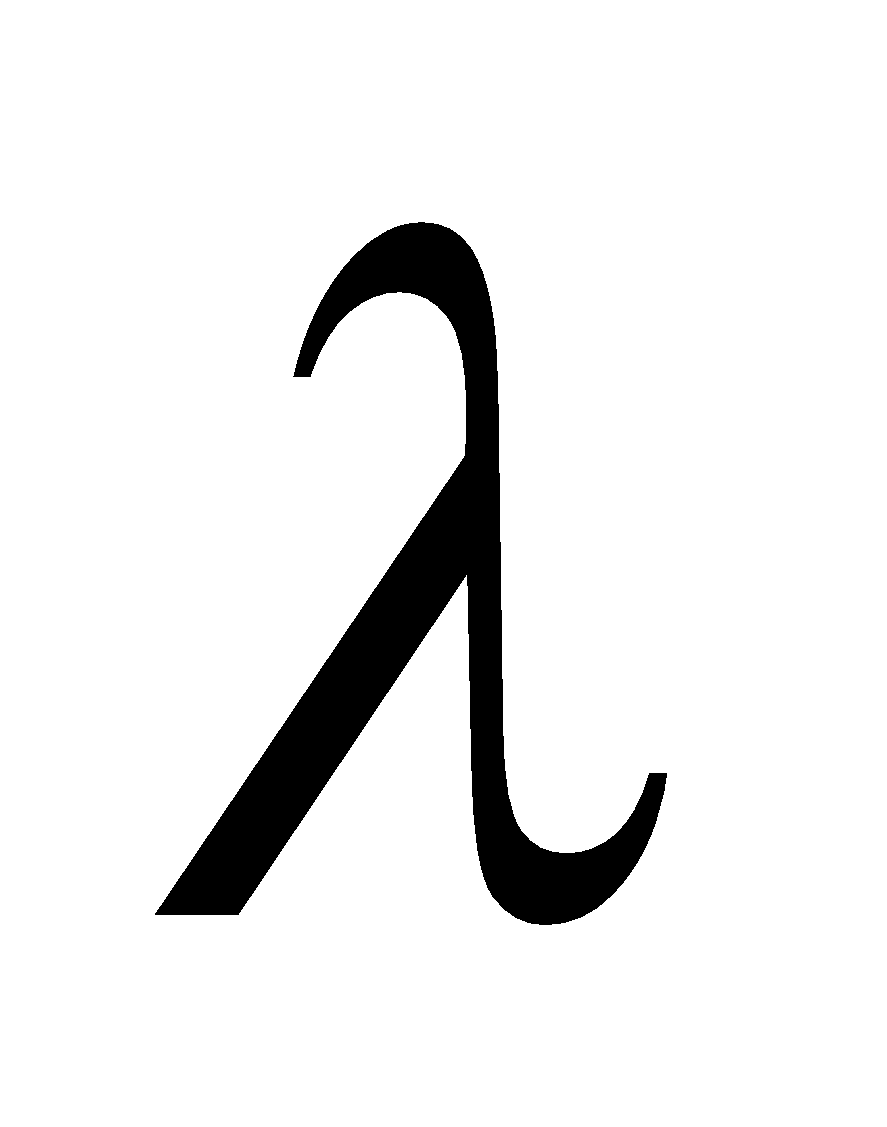
Với Với được chọn tùy vào tập dữ liệu

Nếu khoảng cách Cosine nhỏ hơn mức ngưỡng t, nó được xem là khớp.

*Tổng hợp hai khoảng cách Mahalanobis và khoảng các Cosine:*

Để thiết lập tương quan, Deep SORT sử dụng tổng có trọng số để kết hợp hai chỉ số:



Có thể sử dụng trọng số  để kiểm soát tác động của từng chỉ số đối với chỉ số tổng.

**Cascade matching – Chiến lược đối sánh theo tầng**

Để cải thiện độ chính xác của việc liên kết, chủ yếu là do khi đối tượng biến mất trong một khoảng thời gian dài, độ không chắc chắn của bộ lọc Kalman sẽ gia tăng và có thể dẫn đến sự phân tán của xác suất dự đoán. Khi dự đoán liên tục không được cập nhật, phương sai của phân phối chuẩn sẽ tăng lên. Khi đó, khoảng cách Mahalanobis giữa các điểm xa giá trị kỳ vọng và các điểm gần giá trị kỳ vọng có thể trở nên bằng nhau.

Chiến lược đối sánh theo tầng tiến hành lấy lần lượt từng track ở các frame trước đó, mục đích để xây dựng ma trận chi phí và giải bài toán phân công theo từng tầng.

A white text with black text

Description automatically generated

### Track Life Cycle Management

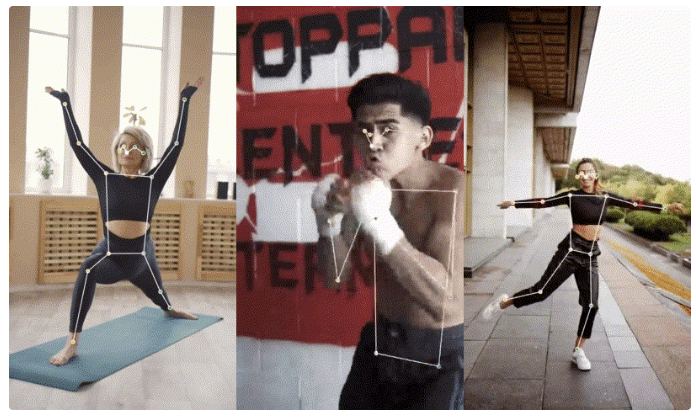
Trong Deep SORT, việc quản lý vòng đời của một track được thực hiện thông qua việc sử dụng một biến trạng thái với ba giá trị khác nhau: tentative, confirmed, và deleted:

* Ban đầu, khi một track mới được tạo, nó sẽ được gán trạng thái là tentative để thực hiện quá trình thăm dò.
* Nếu track duy trì được trong ít nhất 3 frame tiếp theo, trạng thái của track sẽ chuyển sang confirmed.
* Các track có trạng thái confirmed sẽ được duy trì trong 30 frame tiếp theo ngay cả khi chúng mất dấu.
* Nếu track mất dấu trước khi đạt được 3 frame, trạng thái sẽ chuyển sang deleted và track sẽ bị xóa khỏi trình theo dõi.

## Movenet

### Giới thiệu Movenet

Mô hình MoveNet là một mô hình học sâu được phát triển bởi Google được ra mắt vào tháng 5 năm 2021, chuyên về việc nhận diện và theo dõi các điểm khung cơ bản trên cơ thể con người trong video. Đặc điểm nổi bật của MoveNet là khả năng nhận diện chính xác các đặc trưng cơ bản như các điểm mốc (landmark) trên cơ thể, như đầu, vai, cổ, khửu tay, gối, và các điểm khung khác.



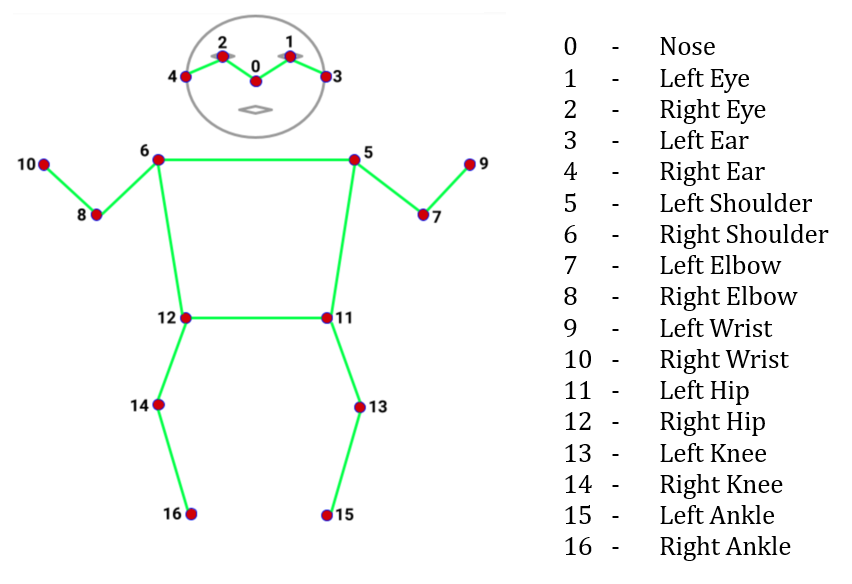
*Hình 2.11 một vài hình ảnh demo của Movenet*

Mô hình này được thiết kế để hoạt động hiệu quả trên các điều kiện ánh sáng khác nhau, phức tạp và trong thời gian thực, giúp xác định và theo dõi vị trí và chuyển động của cơ thể người trong video một cách chính xác. MoveNet thường được sử dụng trong ứng dụng thể dục, theo dõi tư thế, ứng dụng y tế, nhận dạng động tác, và các ứng dụng tương tự đòi hỏi khả năng nhận diện và theo dõi chuyển động của cơ thể con người. Điểm mạnh của MoveNet là khả năng làm việc tốt trên nhiều thiết bị với hiệu suất ổn định, đồng thời có khả năng tích hợp và triển khai linh hoạt trong các ứng dụng thực tế.

Hiện nay, có hai phiên bản chính của MoveNet đã được phát triển và công bố: MoveNet Thunder và MoveNet Lightning. Phiên bản Thunder được ưu tiên sử dụng trong các bài toán đòi hỏi độ chính xác cao, trong khi phiên bản Lightning được tối ưu hóa cho các ứng dụng cần độ trễ thấp. Tuy nhiên, cả hai mô hình, theo thông tin được công bố từ Google, đều có khả năng chạy ở tốc độ thời gian thực, vượt qua mức 30fps trên hầu hết các cấu hình máy tính khi chúng được giới thiệu.

### Ước lượng tư thế người bằng Movenet

Movenet sẽ trả về 17 landmark tương ứng với 17 trên cơ thể người dựa vào ảnh mà ta cung cấp.



*Hình 2.12 Đầu ra các điểm đặc trưng mà Movenet trả về*

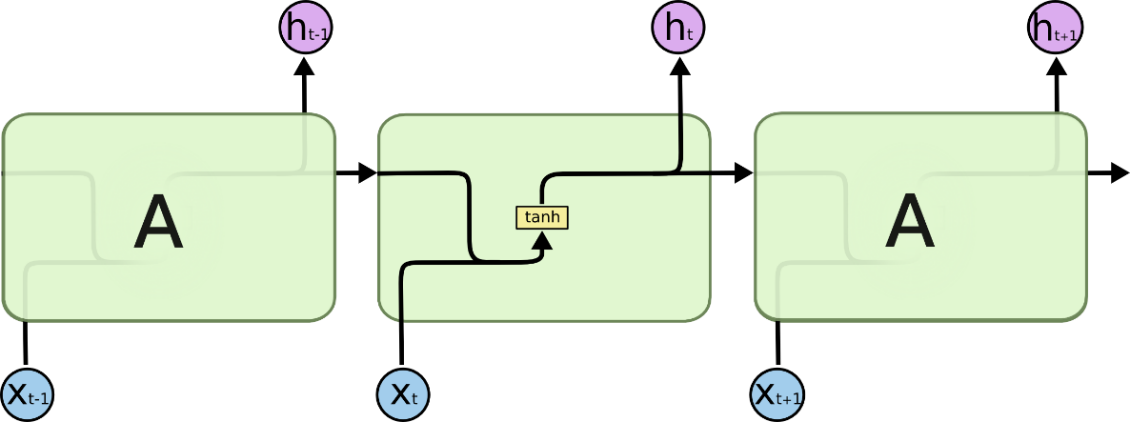
Mỗi landmark mà movenet trả về sẽ chưa thông tin về vị trí của landmark đó, độ chính xác.

## LSTM - Long short term memory

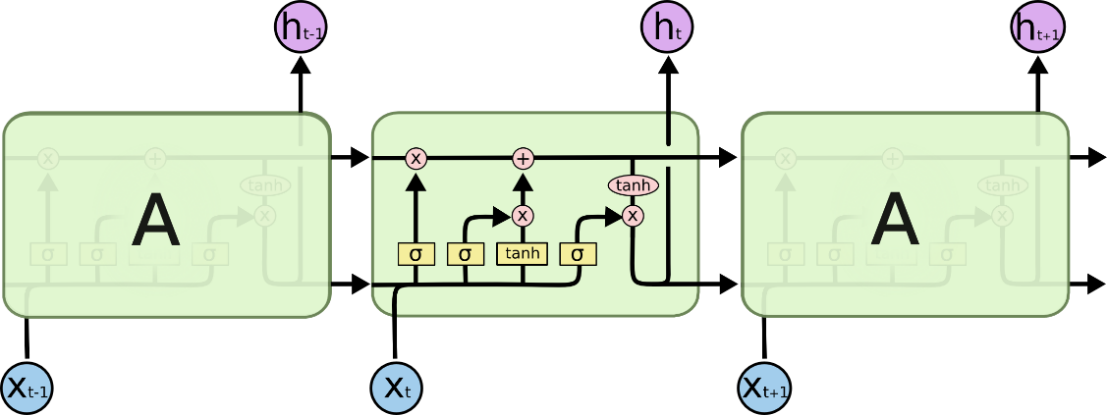
### Giới thiệu LSTM

Mạng trí nhớ ngắn hạn định hướng dài hạn (LSTM) là một kiến trúc đặc biệt của RNN có khả năng học được sự phục thuộc trong dài hạn (long-term dependencies) được giới thiệu bởi Hochreiter & Schmidhuber (1997). Kiến trúc này đã được phổ biến và sử dụng rộng rãi cho tới ngày nay. LSTM đã tỏ ra khắc phục được rất nhiều những hạn chế của RNN trước đây về triệt tiêu đạo hàm. Tuy nhiên cấu trúc của chúng có phần phức tạp hơn mặc dù vẫn dữ được tư tưởng chính của RNN là sự sao chép các kiến trúc theo dạng chuỗi. [9]

So sánh giữa RNN và LSTM:

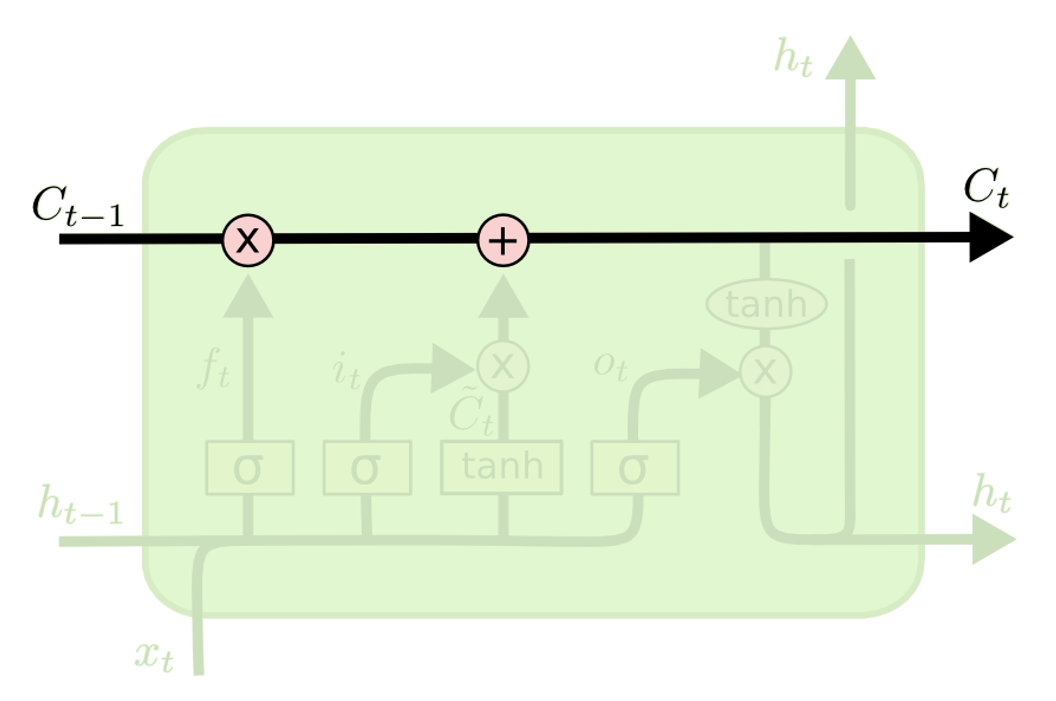


*Hình 2.13 Minh họa RNN*



*Hình 2.14 Minh họa LSTM*

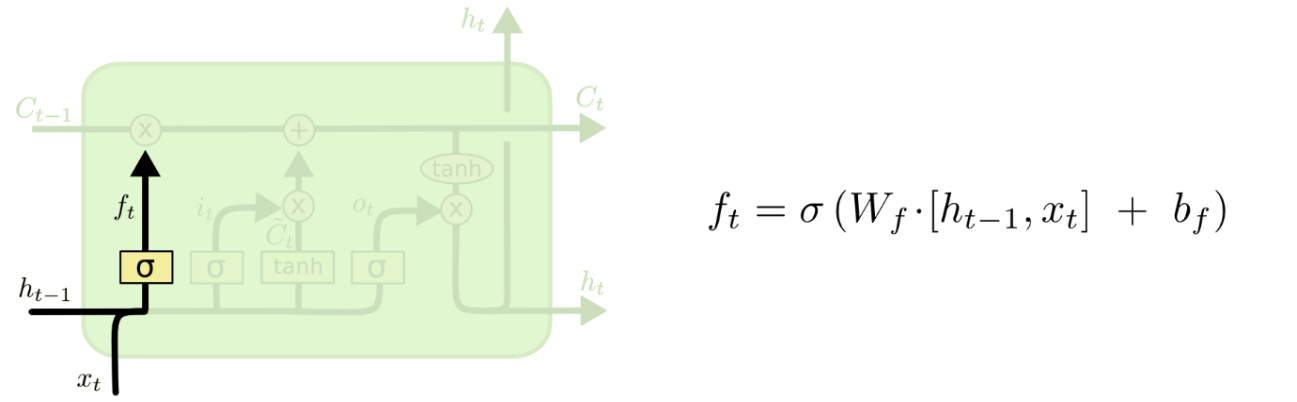
Ý tưởng chính của LSTM là thành phần ô trạng thái (cell state). Ô trạng thái là một dạng băng chuyền chạy thẳng xuyên suốt toàn bộ chuỗi với chỉ một vài tương tác tuyến tính nhỏ giúp cho thông tin có thể truyền dọc theo đồ thị mạng nơ ron ổn định. Ô trạng thái được thể hiện qua đường chạy ngang qua đỉnh đồ thị như hình vẽ bên dưới:



*Hình 2.15 Ô trạng thái của mạng LSTM*

### Thứ tự các bước của LSTM

Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Đầu tiên nó nhận đầu vào là 2 giá trị và  và trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái . Nếu giá trị bằng 1 thể hiện ‘giữ toàn bộ thông tin’ và bằng 0 thể hiện ‘bỏ qua toàn bộ chúng’:



*Hình 2.16 Cổng quên của mạng LSTM*

Bước tiếp theo chúng ta sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần.

Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Quyết định bằng cách nào, thông qua đặc điểm của hàm sigmoid (đầu ra nằm trong khoảng [0,1]), như vậy khi một vector thông tin đi qua đây, nếu nhân với 0, vector sẽ bị triệt tiêu hoàn toàn. Nếu nhân với 1, hầu hết thông tin sẽ được giữ lại.

Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một véc tơ của một giá trị trạng thái mới mà có thể được thêm vào trạng thái. Cuối cùng, kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái

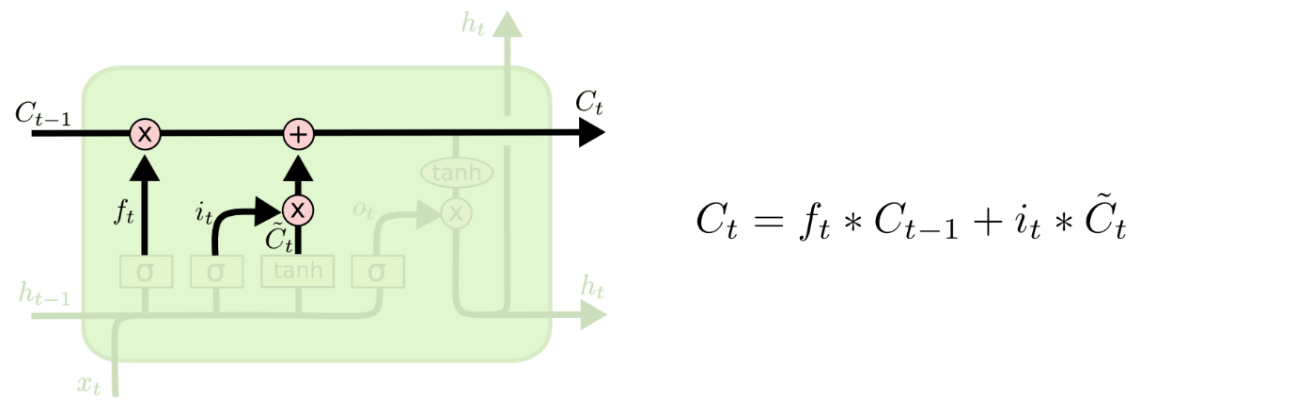


*Hình 2.17 Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cổng vào và tẩng ẩn hàm tanh của mạng LSTM*

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

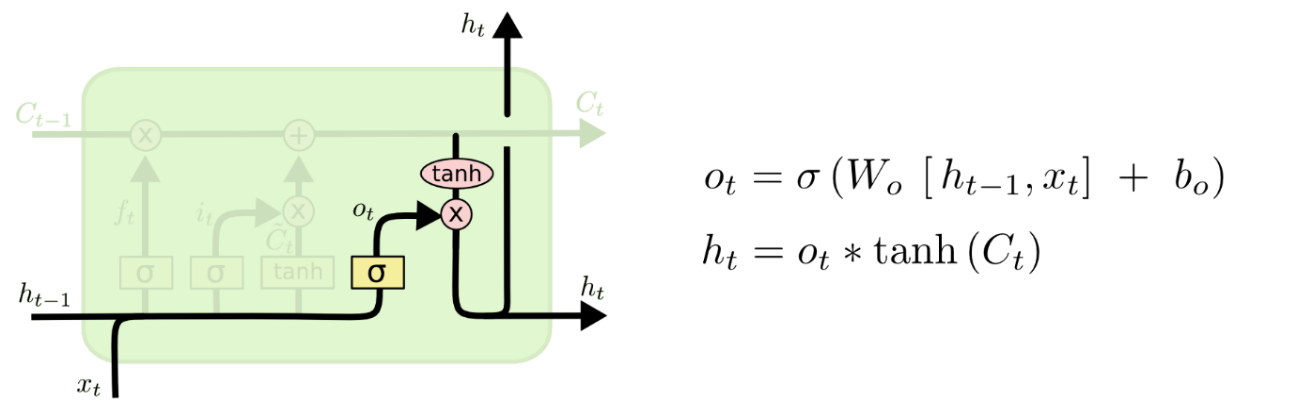
Chúng ta nhân trạng thái cũ với  tương ứng với việc quên những thứ quyết định được phép quên sớm và cộng với trạng thái ẩn chứa quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái mới . Ô trạng thái sẽ mang thông tin nào quan trọng truyền đi xa hơn và sẽ được dùng khi cần. Đây chính là long term memory.

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



*Hình 2.18 Ô trạng thái mới của mạng LSTM*

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



*Hình 2.19 Điều chỉnh thông tin ở đầu ra thông qua hàm tanh của mạng LSTM*

Nếu nhìn kỹ một chút, ta có thể thấy RNN truyền thống là dạng đặc biệt của LSTM. Nếu thay giá trị đầu ra của input gate là 1 và đầu ra forget gate là 0 (không nhớ trạng thái trước), ta được RNN thuần.

# THỰC HIỆN HỆ THỐNG

## Hệ thống phát hiện người té ngã

A diagram of a computer program

Description automatically generated

*Hình 3.1 Tổng quan hệ thống phát hiện người té ngã*

Phương pháp mà tôi tiếp cận trong luận văn này là dùng ảnh đầu vào được lấy từ camera và đưa qua lần lượt các mô hình. Đầu tiên là mô hình YOLOv8 nano sẽ phát hiện vị trí có người. Sau đó Deep SORT sẽ theo dõi và gán id cho từng người. Khi đã có id tiến hành trích xuất đặc trưng tư thế bằng mô hình Movenet và nhóm các đặc trưng vào cùng đúng ID đã gán trước đó. Cuối cùng khi chuỗi đăc trưng có độ dài bằng 8, tức 8 tư thế liên tiếp, sẽ tien61hanh2 nhận diện xem có người ngã không bằng mô hình LSTM.

Cấu hình máy tính được sử dụng trong luận văn: GPU là NVIDIA RTX 3070Ti Laptop và CPU là Intel® Core™ i7-12700Hw

## Xây dựng mô hình YOLOv8 nano phát hiện người

### Mô hình tổng quát

Để xây dựng mô hình YOLOv8 nano phát hiện người tôi triển khai qua hai giai đoạn:

* Giai đoạn 1: Tiến hành thu thập dữ liệu và huấn luyện mô hình.
* Giai đoạn 2: Triển khai mô hình. Điều chỉnh và tối ưu.

A diagram of a training set

Description automatically generated

*Hình 3.2* *Tổng quan mô hình nhận diện người té ngã*

### Bộ dữ liệu của YOLOv8 nano

#### Thu thập dữ liệu

Tôi thu thập dữ liệu bằng cách sử dụng webcam, và camera của điện thoại để chụp lại ảnh có người và đồng thời sử dụng nhũng dữ liệu có sẵn trên youtube để làm dữ liệu huấn luyện được phong phú hơn. Các dữ liệu được thu thập sẽ bao gồm các dữ liệu có tư thế té ngã của con người nhằm tăng khả năng dự doán cho bài toán phát hiện người té ngã của luận văn.

Tổng số lượng ảnh thu thập được là 6891 ảnh

A collage of several people

Description automatically generated

*Hình 3.3 Một vài ảnh trong tập dữ liệu nhận diện người*

#### Gán nhãn cho dữ liệu

Tiến hành gắn nhãn cho các dữ liệu bằng phần mềm LabelImg. Vì dối tượng ở đây chỉ là con người nên toàn bộ dữ liệu sẽ được gắn cùng 1 nhãn là “Person”.

A person lying on a bed in a room with clothes and a person standing

Description automatically generated

*Hình 3.4 Phần mềm LabelImg*

Sau khi label bằng phần mềm LabelImg, file thông tin gán nhãn trả về có định dạng là txt chứa thông tin về id của nhãn, tọa độ tâm, chiều rộng và chiều cao của bounding box dưới định dạng như sau : <id-class> <center-x> <center-y> <bbox-width> <bbox-height>. Trong trường hợp một ảnh có nhiều bounding box thì file annotation sẽ gồm nhiều dòng, mỗi một bounding box là một dòng.

A screenshot of a computer

Description automatically generated

*Hình 3.5 Thông tin label được lưu vào file txt*

#### Phân chia dữ liệu

Sau khi được gán nhãn, tập dữ liệu sẽ được phân chia thành ba phần với tỷ lệ lần lượt là 60% cho việc huấn luyện (training), 20% cho việc đánh giá mô hình trong quá trình huấn luyện (validation) và 20% cho việc đánh giá mô hình sau khi huấn luyện (test). Kết quả là tập training sẽ bao gồm tổng cộng 4699 hình ảnh, tập validation và tập test cùng sẽ có tổng cộng 1566 hình ảnh. Cấu trúc của bộ dữ liệu sẽ được tổ chức như sau:

A screenshot of a computer

Description automatically generated

*Hình 3.6 Cấu trúc tập dữ liệu của YOLOv8 nano*

### Huấn luyện mô hình YOLOv8 nano trên môi trường colab

Sau khi chuẩn bị xong dữ liệu, tôi huấn luyện mô hình YOLOv8 nano trên môi trường Google Colab.

Bởi vì Google Colab là một dịch vụ miễn phí, nó có những hạn chế về việc sử dụng tài nguyên hệ thống. Thời gian sử dụng tối đa cho mỗi phiên là 12 giờ và sữ liệu sau mỗi phiên sẽ bị xóa. Để vượt qua hạn chế này, tôi sẽ tải dữ liệu lên Google Drive dưới dạng file nén .zip và kết nối với Colab thông qua nó. Khả năng kết hợp với Google Drive cho phép Colab thao tác với dữ liệu một cách nhanh chóng và thuận tiện hơn.

**Set up môi trường:**

Tải những package python cần thiết cho quá trình training

| !pip install ultralytics==8.0.202 |
| --- |

Giải nén file dữ liệu

| !unzip /MyDrive/Yolo\_V8/Training/data.zip |
| --- |

Tạo file môi trường mydataset.yaml chứa thông số dữ liệu huấn luyện, file môi trường sẽ được lưu ở trong folder data:

| !echo 'train: /MyDrive/Yolo\_v8/data/train' >> data/mydataset.yaml  !echo 'val: /MyDrive/Yolo\_v8/data/val' >> data/mydataset.yaml  !echo 'nc: 1' >> data/mydataset.yaml  !echo "names: ['person']" >> data/mydataset.yaml |
| --- |

Tải file pretrain model:

| !wget https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n.pt |
| --- |

**Huấn luyện:**

Sử dụng CLI do Ultralytics cung cấp, bắt đầu huấn luyện model:

| !yolo task=detect mode=train model=Pretrained/yolov8n.pt data= /data/mydataset.yaml epochs=200 batch=128 imgsz=640 pretrained=True |
| --- |

**A white background with black text

Description automatically generated**

*Hình 3.7 CLI hiển thị quá trình huấn luyện*

## Theo dõi đối tượng

Sau khi đã có được thông tin các bounding của các dối tượng được nhận diện từ mạng YOLOv8 nano, ta sẽ tiến hành sử dụng mô hình Deep SORT để liên tục theo dõi và đánh số ID cho từng đối tượng. Hạn chế để ID bị thay đổi khi đối tượng ra khỏi phạm vi khung hình và sau đó xuất hiện trở lại và giảm được hiện tượng ID bị gán nhầm khi đối tượng có quỹ đạo chuyển động chồng chéo lên nhau.

Việc đánh số ID cho mỗi đối tượng và duy trì tính nhất quán của ID trong từng khung hình là rất quan trọng, vì điều này đảm bảo rằng các đặc trưng được trích xuất từ cùng một đối tượng sau đó sẽ được nhóm lại với nhau.

|  |  |
| --- | --- |
|  |  |

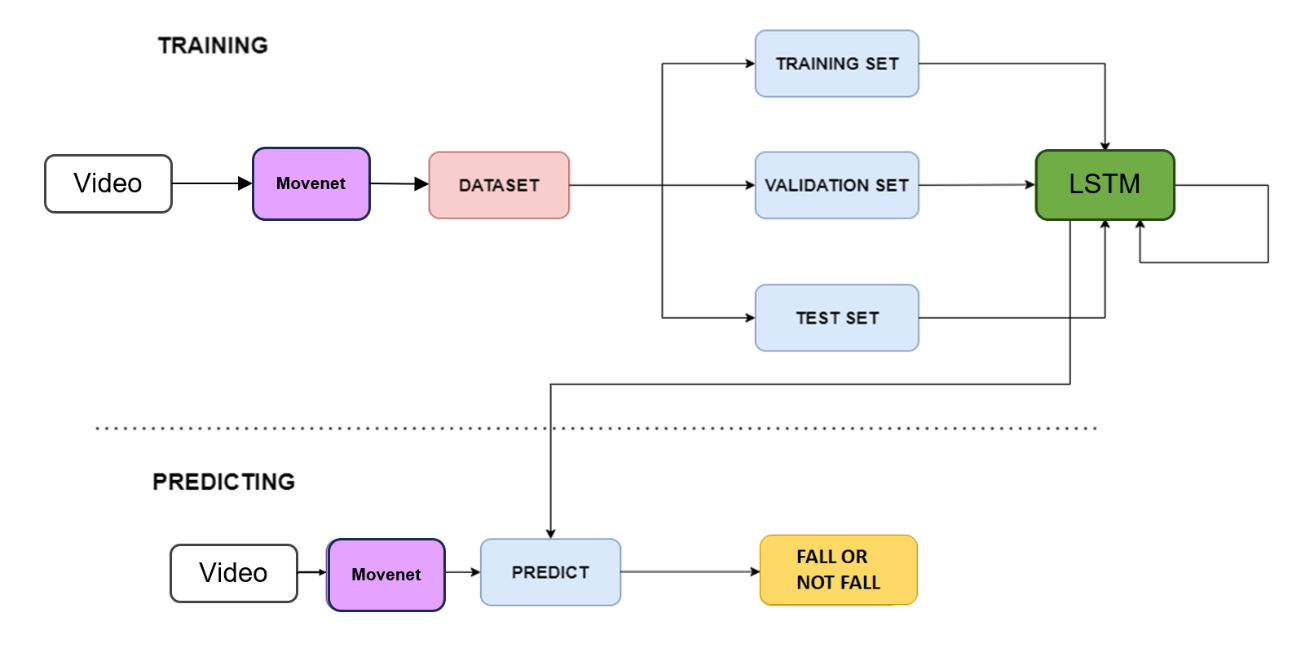
*Hình 3.8 Kết quả nhận diện và theo dõi sản phẩm của YOLOv8 và DeepSORT*

## Xây dựng mô hình LSTM nhận diện té ngã

### Mô hình tổng quát

Để xây dựng mô hình LSTM nhận diện té ngã cũng sẽ triển khai qua hai giai đoạn:

* Giai đoạn 1: Tiến hành thu thập dữ liệu và huấn luyện mô hình.
* Giai đoạn 2: Triển khai mô hình. Điều chỉnh và tối ưu.



*Hình 3.9 Tổng quan mô hình nhận nhận diện té ngã*

### Bộ dữ liệu của LSTM

Để tiến hành huấn luyện mô hình dự đoán té ngã, tôi sử dụng những video dữ liệu thô về các hành vi thông thường và những tình huống té ngã ở nhiều góc độ khác nhau, nhầm làm cho dữ liệu đa dạng hơn. Vì khi người bị ngã sẽ chuyễn từ trạng thái bình thường sang ngã và cuối cùng là nằm nên tập dữ liệu sẽ bao gồm 3 trạng thái là không ngã (NOT FALL), ngã (FALL) và nằm (LIE\_DOWN).

A collage of people in different poses

Description automatically generated

*Hình 3.10 Một vài ảnh từ video dữ liệu té ngã ở nhiều góc độ khác nhau*

Thu thập dữ liệu video thô là bước quan trọng đầu tiên. Tiếp theo, tôi sử dụng thư viện Movenet để trích xuất đặc trưng từ những video này. Đầu ra của Movenet cho một người ở mỗi frame ảnh là một vector 1x34 chứa thông tin về vị trí của 17 điểm nút trên cơ thể người.

Theo nghiên cứu của Konrad Schindler và Luc Van Gool về 'How many frames does human action recognition require?' [8], để tạo ra chuỗi chuyển động có ý nghĩa, chúng ta cần ít nhất 8 frame ảnh liên tiếp. Do đó, trong luận văn này, sau khi trích xuất đặc trưng từ từng frame ảnh, thì cứ 8 đặc trưng liên tiếp sẽ được ghép thành một ma trận có kích thước là 8x34. Đây sẽ là dữ liệu đầu vào cho mạng LSTM.

Khi thu thập dữ liệu, chúng tôi đảm bảo tổng số frame ảnh của mỗi video chia hết cho 8. Trong trường hợp không chia hết, chúng tôi loại bỏ vài frame ảnh cuối để đảm bảo tính nhất quán của dữ liệu.



*Hình 3.11 Chuỗi 8 frame ảnh liên tiếp*

Cuối cùng, sau khi đã xử lý xong thì tôi thu được 13792 dữ liệu cho tập FALL, 15104 dữ liệu cho tập NOT FALL và 13504 dữ liệu cho tập LIE\_DOWN. Ta sẽ lưu toàn bộ tập dữ liệu vào 2 file text là FALL.csv, NOT\_FALL.csv và LIE\_DOWN.csv, để có thể dễ dàng mang lên Google Colab huấn luyện mạng.

### Huấn luyện mô hình LSTM trên môi trường colab

Sau khi chuẩn bị xong dữ liệu, tôi huấn luyện mô hình LSTM trên môi trường Google Colab. Và cũng như YOLOv8 nano đã trình bày dữ liệu training sẽ được đưa lên Google Drive.

**Set up môi trường:**

Tải những package python cần thiết cho quá trình training.

| !pip install tensorflow==2.10.0 |
| --- |

Giải nén file data

| not\_fall\_list = pd.read\_csv("/MyDrive /LSTM/data/NOT\_FALL.csv")  fall\_list = pd.read\_csv("/MyDrive/ LSTM/data/FALL.csv")  lie\_list = pd.read\_csv("/MyDrive /LSTM/data/LIE.csv") |
| --- |

Chia dữ liệu thành các tập training, validation và test theo tỉ lệ 7/2/1, nghĩa là 70% dữ liệu sẽ được dùng cho việc huấn luyện (training), 20% được dùng cho việc đánh gái trong lúc huấn luyện (validation) và 10% còn lại dùng để đánh giá kết quả của mô hình sau khi huấn luyện (test)

| not\_fall\_list\_train, not\_fall\_list\_val, not\_fall\_labels\_train, not\_fall\_labels\_val = train\_test\_split(not\_fall\_list, not\_fall\_labels, test\_size=0.2, shuffle=True, random\_state=0)  not\_fall\_list\_train, not\_fall\_list\_test, not\_fall\_labels\_train, not\_fall\_labels\_test = train\_test\_split(not\_fall\_list\_train, not\_fall\_labels\_train, test\_size=0.125, shuffle=True, random\_state=0)  fall\_list\_train, fall\_list\_val, fall\_labels\_train, fall\_labels\_val = train\_test\_split(fall\_list, fall\_labels, test\_size=0.2, shuffle=True, random\_state=0)  fall\_list\_train, fall\_list\_test, fall\_labels\_train, fall\_labels\_test = train\_test\_split(fall\_list\_train, fall\_labels\_train, test\_size=0.125, shuffle=True, random\_state=0)  lie\_list\_train, lie\_list\_val, lie\_labels\_train, lie\_labels\_val = train\_test\_split(lie\_list, lie\_labels, test\_size=0.2, shuffle=True, random\_state=0)  lie\_list\_train, lie\_list\_test, lie\_labels\_train, lie\_labels\_test = train\_test\_split(lie\_list\_train, lie\_labels\_train, test\_size=0.125, shuffle=True, random\_state=0) |
| --- |

**Thiết kế mạng**

Tôi sẽ xây dựng mạng LSTM bằng thư viện Keras được tích hợp sẵn trong framework TensorFlow 2x.

Các thông số cấu hình cho mạng như sau:

* Learning rate: 0.0001
* Hàm mất mát: Crossentropy
* Hàm kích hoạt ở ngõ ra: Softmax

| def custom\_optimizer\_init\_fn():  learning\_rate = 1e-4  return tf.keras.optimizers.Adam(learning\_rate)  model = Sequential()  model.add(LSTM(units = 256, return\_sequences = True, input\_shape = (datas.shape[1], datas.shape[2])))  model.add(Dropout(0.2))  model.add(LSTM(units = 128, return\_sequences = True))  model.add(Dropout(0.2))  model.add(LSTM(units = 64, return\_sequences = False))  model.add(Dropout(0.2))  model.add(Dense(units = 3, activation="softmax"))  optimize = custom\_optimizer\_init\_fn()  model.compile(optimizer=optimize, metrics = METRICS, loss = "binary\_crossentropy")  model.summary ()  batch\_size = 256  epochs = 300  history = model.fit(datas\_train, labels\_train, batch\_size=batch\_size, epochs=epochs, validation\_data=(datas\_val, labels\_val)) |
| --- |

A screenshot of a computer

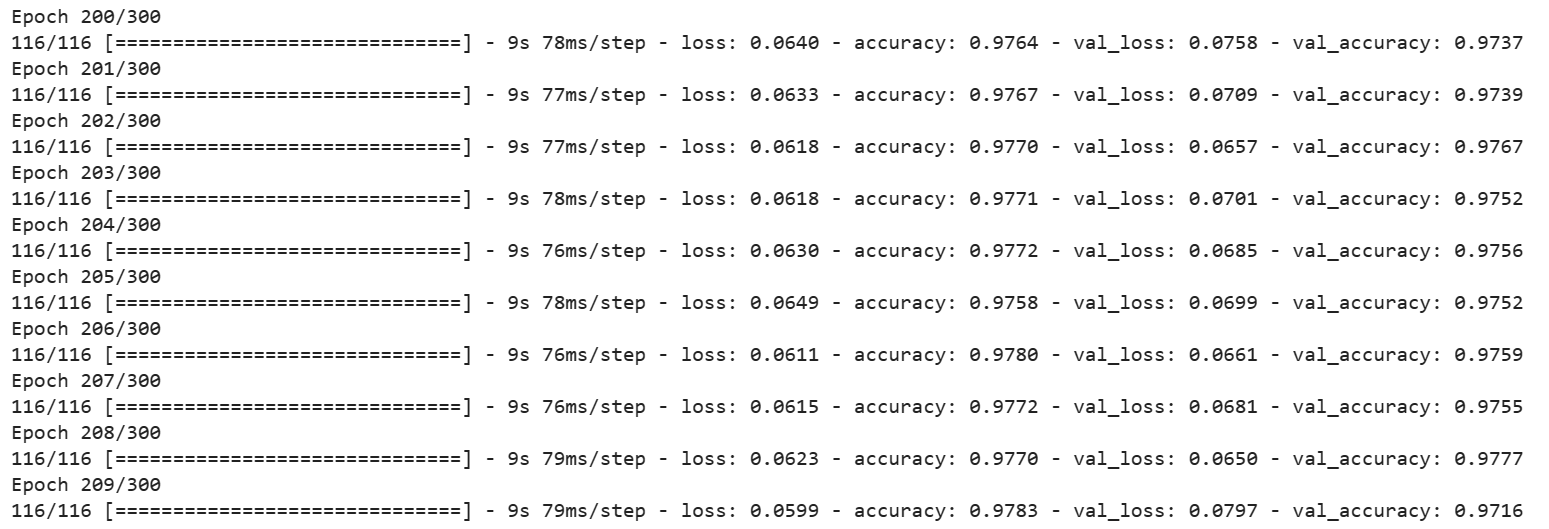
Description automatically generated

*Hình 3.12 Thông tin tham số của mạng LSTM*

**Huấn luyện:**

Sau khi đã có mô hình mạng ta tiến hành huấn luyện mạng với:

* Batch size = 256 (số lượng dữ liệu được đưa vào trong mỗi vòng huấn luyện)
* Epochs = 300 (số lần huấn luyện hết toàn bộ tập dữ liệu)

****

*Hình 3.13 CLI quá trình huấn luyện*

## Triển khai và tối ưu hóa hệ thống

### Tối ưu cho quá trình nhận diện người

Hai mô hình YOLOv8 nano và LSTM chiếm khá nhiều tài nguyên khi triển khai cùng một lúc 2 mô hình trên máy tính là rất lâu, tốc độ xử lý để cho ra kết quả khá chậm vì mô hình YOLOv8 nano có lượng thông số lên đến hàng triệu dẫn tới việc xử lý tính toán rất nhiều.

Với lý do đó, tôi đã tối ưu hóa mô hình YOLOv8 Nano bằng công cụ hỗ trợ TensorRT của NVIDIA. Nhóm tác giả từ Ultralytics [2] khi giới thiệu YOLOv8 cũng cung cấp framework mã nguồn mở cùng tên Ultralytics, một công cụ mạnh mẽ hỗ trợ rất đáng kể cho việc phát triển và triển khai các mô hình YOLO. Framework này cung cấp các hàm sẵn có để chuyển đổi từ file trọng số và triển khai mô hình YOLO bằng TensorRT.

Ta sẽ chuyển file trọng số của YOLOv8 nano thành file TensorRT engine:

| from ultralytics import YOLO  # Load a model  model = YOLO('D:/HCMUT/Ths/Thesis/yolov8/tensorrt/best.pt')  # Export the model  model.export(format='engine', half=True, workspace=4) |
| --- |

Trong đó có 2 thông số quan trọng cần cấu hình:

* workspace: là dung lượng bộ nhớ tạm thời tối đa của GPU cái mà Cuda có thể dùng trong quá trình hoạt động.
* half: cấu hình precision là FP16 (float 16)

### Tối ưu hóa cho quá trình nhận dạng té ngã

Mô hình LSTM được dùng để nhận diện té ngã trong luận văn này, mô hình được huấn luyện bằng thư viện Keras tích hợp trong framework TensorFlow 2.10. Keras cung cấp môi trường thuận tiện để thiết kế và huấn luyện, với giao diện thân thiện, tuy nhiên, khi sử dụng file trọng số .h5 sau quá trình huấn luyện, quá trình dự đoán có thể trở nên chậm khi áp dụng cho triển khai thực tế. Điều này đặc biệt quan trọng khi chuyển từ môi trường nghiên cứu sang triển khai thực tế. Chính vì lý do này, tôi tiến hành sử dụng frozen graph tối ưu hóa mô hình và để chuyển đổi file trọng số thành định dạng .pb dựa trên bài viết của tác giả Lei Mao [9], giúp tăng hiệu suất quá trình dự đoán và triển khai mô hình trong các ứng dụng thực tế.

Frozen graph là một thuật ngữ trong Tensorflow. Có thể hiểu nó là phiên bản tối ưu hóa của mô hình, nơi các trọng số của mạng đã được đóng băng và biến thành các hằng số, đồng thời loại bỏ các biến không cần thiết để dự đoán. Việc đóng băng trọng số giúp mô hình trở nên ổn định và không thay đổi trong quá trình dự đoán, đồng thời giảm bộ nhớ và tài nguyên tính toán cần thiết.

| from tensorflow import keras  import tensorflow as tf  from tensorflow.python.framework.convert\_to\_constants import convert\_variables\_to\_constants\_v2  import numpy as np  # Convert Keras model to ConcreteFunction  full\_model = tf.function(lambda x: model(x))  full\_model = full\_model.get\_concrete\_function(  tf.TensorSpec(model.inputs[0].shape, model.inputs[0].dtype))  # Get frozen ConcreteFunction  frozen\_func = convert\_variables\_to\_constants\_v2(full\_model)  frozen\_func.graph.as\_graph\_def()  # Save frozen graph from frozen ConcreteFunction to hard drive  tf.io.write\_graph(graph\_or\_graph\_def=frozen\_func.graph,  logdir="/MyDrive/ LSTM/output",  name=f"frozen\_graphpb",  as\_text=False) |
| --- |

### Triển khai hệ thống

A diagram of a flowchart

Description automatically generated

*Hình 3.14 Lưu đồ giải thuật của hệ thống*

Mỗi khung ảnh được xử lý theo từng bước: trước hết, khung ảnh được đưa qua YOLOv8 nano để phát hiện sự hiện diện của người trong khung ảnh. Nếu có người, quá trình tiếp theo là gán ID cho mỗi đối tượng trong khung ảnh bằng thuật toán Deep SORT. Khi đã có ID của từng người, MovNet được sử dụng để trích xuất đặc trưng về tư thế của mỗi người, và các đặc trưng này được nhóm vào ID tương ứng đã được gán trước đó. Nếu ID của một người có đủ 8 đặc trưng liên tiếp, quá trình nhận diện sẽ xác định trạng thái của người đó, ví dụ như việc họ có đang bị té ngã không.

# KẾT QUẢ

Trong chương này, tôi thực hiện đánh giá mô hình phát hiện người và mô hình nhận diện té ngã qua các tập dữ liệu validation, test và bằng hình ảnh thực nghiệm. Sau đó, tôi thực hiện khảo sát để kiểm tra khả năng hoạt động của hệ thống. Quá trình phát hiện này bao gồm việc phát hiện người, theo dõi trích xuất đặc trưng và dự đoán về hành vi té ngã.

## Mô hình phát hiện người:

### Đánh giá kết quả huấn luyện mô hình YOLOv8 nano

Sau khi huấn luyện xong mô hình YOLOv8 nano, chúng tôi đánh giá kết quả của quá trình huấn luyện bằng việc tính chỉ số mAP trên tập validation trong quá trình huấn luyện và tập test đã chia trước đó.

#### Tập validation

* Kết quả:

*Bảng 4.1 Kết quả sau khi huấn luyện phát hiện người với tập validation*

| Precision | Recall | mAP50 | mAP@50-95 |
| --- | --- | --- | --- |
| 0.967 | 0.945 | 0.982 | 0.885 |

A group of graphs showing the size of a number

Description automatically generated with medium confidence

*Hình 4.1 Đồ thị quá trình huấn luyện phát hiện người*

* Nhận xét: Precision ở mức 96.7% và recall đạt 94.5%. Điều này chứng tỏ mô hình có khả năng dự đoán chính xác và ít bỏ sót các đối tượng người trong ảnh. Mặt khác, mAP@50 đạt 98.2% và mAP@50-95 đạt 88.5%. Cả hai chỉ số này đều vượt qua ngưỡng 0.8, cho thấy độ chính xác và sự ổn định của mô hình trong việc phát hiện đối tượng, đặc biệt ở mức ngưỡng IOU lớn. Các đồ thị ở tập train đều hội tụ chứng tỏ mô hình không bị tình trạng underfitting. Đồng thời thì các đồ thị ở tập validation cũng hội tụ chứng tỏ mô hình không bị overfitting.

#### Tập test

*Bảng 4.2 Kết quả sau khi huấn luyện phát hiện người với tập test*

| Precision | Recall | mAP50 | mAP@50-95 |
| --- | --- | --- | --- |
| 0.977 | 0.958 | 0.99 | 0.902 |

Nhận xét: tương tự với tập validation, các chỉ số đánh giá ở tập test đều cao, trên mức 0.8. Có thể thấy mô hình không bị overffitting và có khả năng nhận diện tốt với những dữ liệu mới

### Kết quả thực nghiệm mô hình YOLOv8 kết hợp thuật toán Deep SORT

Sau khi đã huấn luyện và đánh giá mô hình dựa trên các chỉ số. Tôi tiến hành kết hợp mô hình với thuật toán DeepSORT để theo dõi đối tượng và đánh giá thực nghiệm.



*Hình 4.2 Kết quả thực nghiệm phát hiện người kết hợp theo dõi thứ nhất*



*Hình 4.3 Kết quả thực nghiệm phát hiện người kết hợp theo dõi thứ hai*



*Hình 4.4 Kết quả thực nghiệm phát hiện người kết hợp theo dõi thứ ba*

Nhận xét: Mô hình YOLOv8 nano sau khi huấn luyện có thể nhận diện nhiều người trong cùng 1 khung hình khá tốt. Thuật toán Deep SORT đã có kết quả ổn định trong việc theo dõi và đánh ID cho các đối tượng qua các khung hình liên tiếp. Tuy nhiên ở vài trường hợp, khi đối tượng di chuyển quá nhanh hoặc bị che khuất một cách tạm thời, thì hiện tượng gán lại ID khá vẫn xảy ra, nhưng sau đó thì ID mới vẫn được duy trì cho đối tượng, nên có hiện tượng ID bị thay đổi liên tục.

### Kết quả tối ưu mô hình

Sau khi sử dụng TensorRT để tối ưu mô hình tôi tiến hành so sánh với mô hình khi chưa tối ưu, do mỗi khung hình sẽ xử lý với 1 thời gian khác nhau, tùy vào số lượng object trong khung hình đó, nên tôi sẽ tính trưng bình thời gian 10 khung hình liên tiếp cho 2 trường hợp khi chưa tối ưu và khi tối ưu bằng TensorRT:

*Bảng 4.3 Thời gian thực thi trên mỗi khung hình của mô hình phát hiện người*

| Khung hình | Thời gian xử lý trước khi TensorRT (giây) | Thời gian xử lý sau khi TensorRT (giây) |
| --- | --- | --- |
| 1 | 0.031 | 0.012 |
| 2 | 0.030 | 0.0135 |
| 3 | 0.028 | 0.012 |
| 4 | 0.029 | 0.013 |
| 5 | 0.028 | 0.014 |
| 6 | 0.028 | 0.015 |
| 7 | 0.028 | 0.014 |
| 8 | 0.028 | 0.014 |
| 9 | 0.026 | 0.025 |
| 10 | 0.026 | 0.026 |
| Trung bình | 0.028 | 0.016 |

Từ bảng trên ta tính ra được trung bình fps của mô hình khi chưa tối ưu là 35.7 và sau khi tối ưu là 62.5. Có thể thấy tốc độ xử lý sau khi tối ưu bằng TensorRT tăng gấp 1.75 lần.

## Mô hình nhận diện người té ngã:

### Đánh giá kết quả huấn luyện mô hình LSTM

Sau khi huấn luyện xong ta sẽ thu được đồ thị đánh giá như sau:

A graph of a graph of a number of objects

Description automatically generated with medium confidence

*Hình 4.5 Đồ thị đánh giá mô hình LSTM*

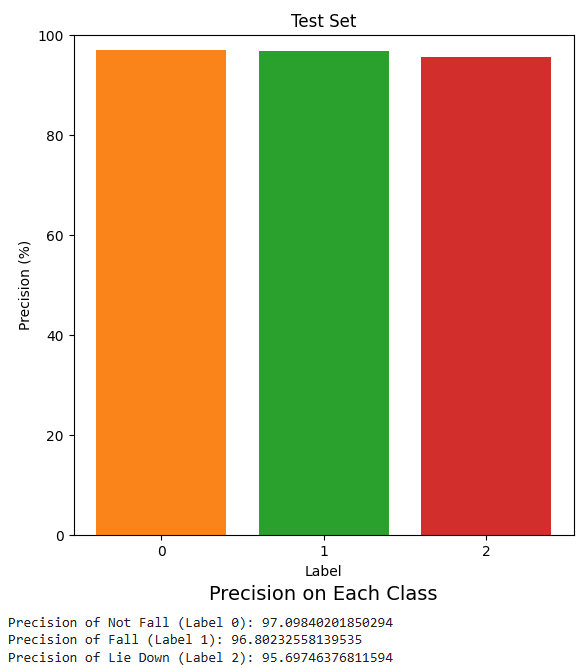
Nhận xét: Hiệu suất của mô hình được đánh giá thông qua hai đồ thị chính: accuracy và loss trên cả tập train và tập validation. Đồ thị accuracy thể hiện mức độ chính xác của mô hình, tăng dần và hội tụ ổn định trên cả hai tập dữ liệu. Đồng thời, đồ thị loss giảm dần, cho thấy mô hình đang học được thông tin và hội tụ đến mức chấp nhận được. Nhìn chung, các biểu đồ cho thấy mô hình đạt được sự hội tụ mong muốn và có khả năng áp dụng vào thực tế.

**Đánh giá dựa vào tập test:**

Tiếp theo, tôi tiến hành đánh giá mô hình trên tập test đã phân chia trước đó. Sau khi có kết quả dự đoán của tập test ta tiến hành đánh giá mô hình bằng các chỉ số Precision, Recall và F1 score.

* Precision và Recall

**Precision** là tỉ lệ giữa những điểm được dự đoán và đó thật sự là Positive (True Positive) so với tất cả các điểm được dự đoán là Positive (True Positive + False Positive).



*Hình 4.6 Precision mỗi class của mô hình nhận diện té ngã*

**Recall** là tỉ lệ giữa những điểm được dự đoán và những điểm đó thực sự là Positive (True Positive) so với những điểm thực sự là Postitive (True Positive + False Negative).

A graph of a bar chart

Description automatically generated

*Hình 4.7 Recall mỗi class của mô hình nhận diện té ngã*

**Nhận xét:** Precision và Recall của từng lớp đều khá cao. Precision cao có nghĩa là mô hình ít khi phân loại các dữ liệu Negative thành kết quả Positive sai, trong khi Recall cao cho thấy tỉ lệ bỏ sót các dữ liệu thực sự là Positive là thấp. Điều này chứng tỏ mô hình đã được huấn luyện tốt và có khả năng đưa ra các dự đoán chính xác cho từng lớp.

* F1 Score

F1 score là một thước đo kết hợp giữa precision và recall trong bài toán phân loại. Nó được sử dụng để đánh giá hiệu suất của một mô hình phân loại dựa trên số lượng dự đoán đúng (true positives), false positives và false negatives. Một F1 score cao (gần 1) cho thấy mô hình có độ chính xác và độ nhớ tốt.

A graph of a bar chart

Description automatically generated with medium confidence

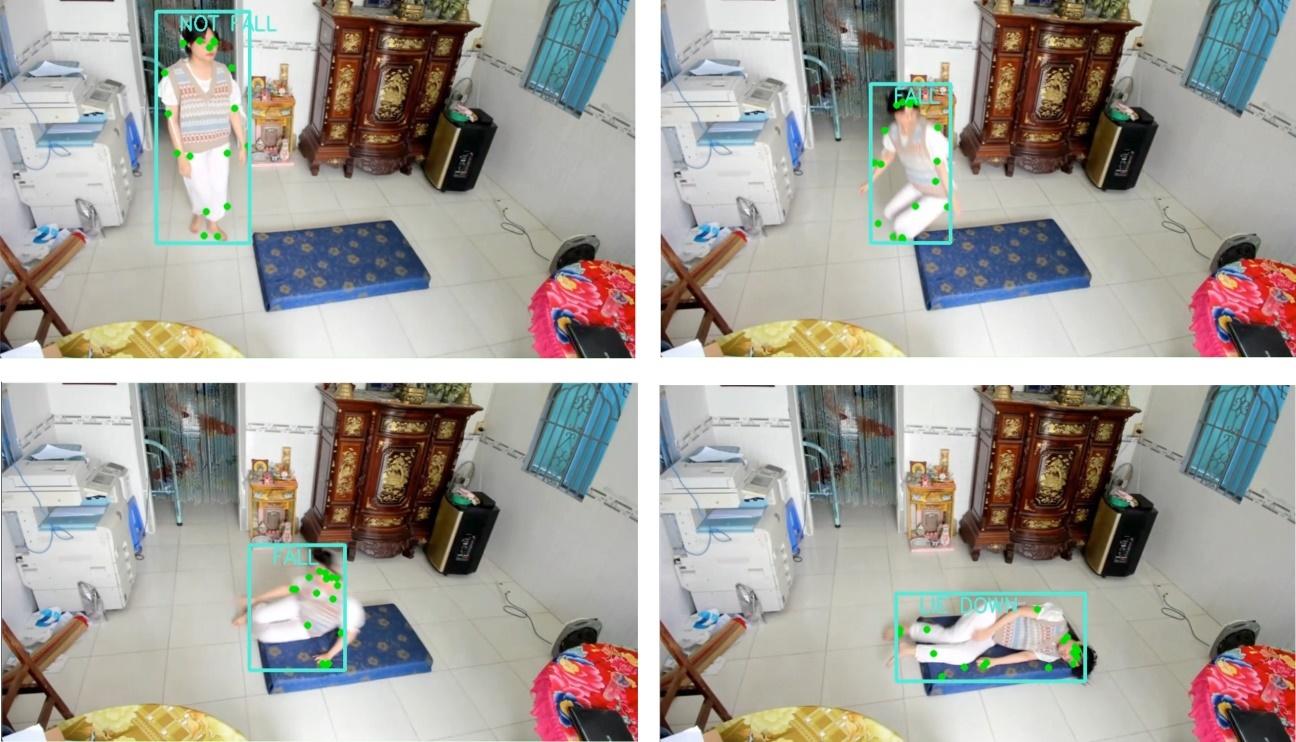
*Hình 4.8 F1 score mỗi class của mô hình nhận diện té ngã*

Nhận xét: Mô hình cho ra kết quả F1 Score khá cao. Điều này chỉ ra rằng mô hình có khả năng dự đoán chính xác và cân bằng giữa precision và recall cho mỗi lớp.

### Kết quả thực nghiệm

Sau khi huấn luyện và đánh giá mô hình LSTM với tập test, tôi tiến hành kết hợp mạng LSTM với mô hình YOLOv8 và Movenet để đánh giá thực nghiệm mô hình nhận diện té ngã. Tôi sẽ thực nghiệm lần lượt với 3 đối tượng khác nhau.

* Kết quả thực nghiệm với mẫu thứ nhất



*Hình 4.9 Lần thử thứ nhất của mẫu 1*



*Hình 4.10 Lần thử thứ hai của mẫu 1*

* Kết quả thực nghiệm với mẫu thứ hai



*Hình 4.11 Lần thử thứ nhất của mẫu 2*



*Hình 4.12 Lần thử thứ hai của mẫu 2*

* Kết quả thực nghiệm với mẫu thứ ba



*Hình 4.13 Lần thử thứ nhất của mẫu 3*



*Hình 4.14 Lần thử thứ hai của mẫu 3*

Nhận xét: Kết quả thực nghiệm cho thấy mạng LSTM có thể phát hiện trạng thái ngã của 1 người rất tốt hầu hết các trạng thái ngã đều có thể dự đoán được. Tuy nhiên, trong lần test của mẫu 2 lần thứ 2 mạng đưa ra dự đoán sai về trạng thái nằm của đối tượng thành fall, đều này là do môi trường ánh sáng yếu dẫn đến việc trích xuất đặc trưng của Movenet bị nhiễu, làm cho kết quả dự đoán bị sai.

### Kết quả sau khi tối ưu mô hình

Sau khi tối ưu mô hình LSTM bằng frozen graph, tôi tiến hành đánh giá khả năng tối ưu. Tôi sử dụng 10 mẫu bất kỳ ở tập test để tiến hành đo thời gian thực thi của mạng trước và sau khi tối ưu, sau đó sẽ so sánh thời gian thực thi trung bình.

*Bảng 4.4 Thời gian thực thi trên mỗi khung hình của mô hình nhận diện té ngã*

| Mẫu dữ liệu | Thời gian xử lý trước khi tối ưu (giây) | Thời gian xử lý sau khi tối ưu (giây) |
| --- | --- | --- |
| 1 | 0.07 | 0.009 |
| 2 | 0.057 | 0.009 |
| 3 | 0.055 | 0.012 |
| 4 | 0.062 | 0.011 |
| 5 | 0.059 | 0.011 |
| 6 | 0.063 | 0.01 |
| 7 | 0.062 | 0.013 |
| 8 | 0.061 | 0.011 |
| 9 | 0.064 | 0.01 |
| 10 | 0.061 | 0.009 |
| Trung bình | 0.0614 | 0.0105 |

Từ bảng trên ta tính ra được thời gian thực thi của mô hình LSTM sau khi tối ưu bằng frozen graph tăng gấp 5.8 lần sao với trước khi tối ưu. Như vậy tốc độ đã được tối ưu rất nhiều tăng hiệu suất của hệ thống phát hiện người té ngã.

## Tổng hợp hệ thống:

Sau khi kết hợp các phần của hệ thống lại với nhau tối tiến hành kiểm tra thực nghiệm của hệ thống. Mục tiêu đánh giá tốc độ thực thi trong bình của hệ thống và khả năng phát hiện có người té ngã khi trong khung hình xuất hiện nhiều người.

### Đánh giá thực nghiệm tốc độ xử lý của hệ thống

**Mục tiêu đánh giá:**

Xác định tốc FPS trung bình của hệ thống qua nhiều lần đo. Từ đó đánh giá về khả năng hoạt động trên thời gian thực của hệ thống.

**Cách thực hiện:**

* Thực hiện động tác ngã khi trong khung hình có 1 người, 2 người và 3 người.
* Tiến hành do và tính toán thời gian trung bình xử lý một khung hình lần lượt khi trong khung hình.
* Do hệ thống được kết hợp từ nhiều phần nên ta sẽ thực hiện nhiều lần bước trên, từ đó mới tính trung bình thời gian xử lý 1 khung hình và FPS trung bình của hệ thống.

**Kết quả:**

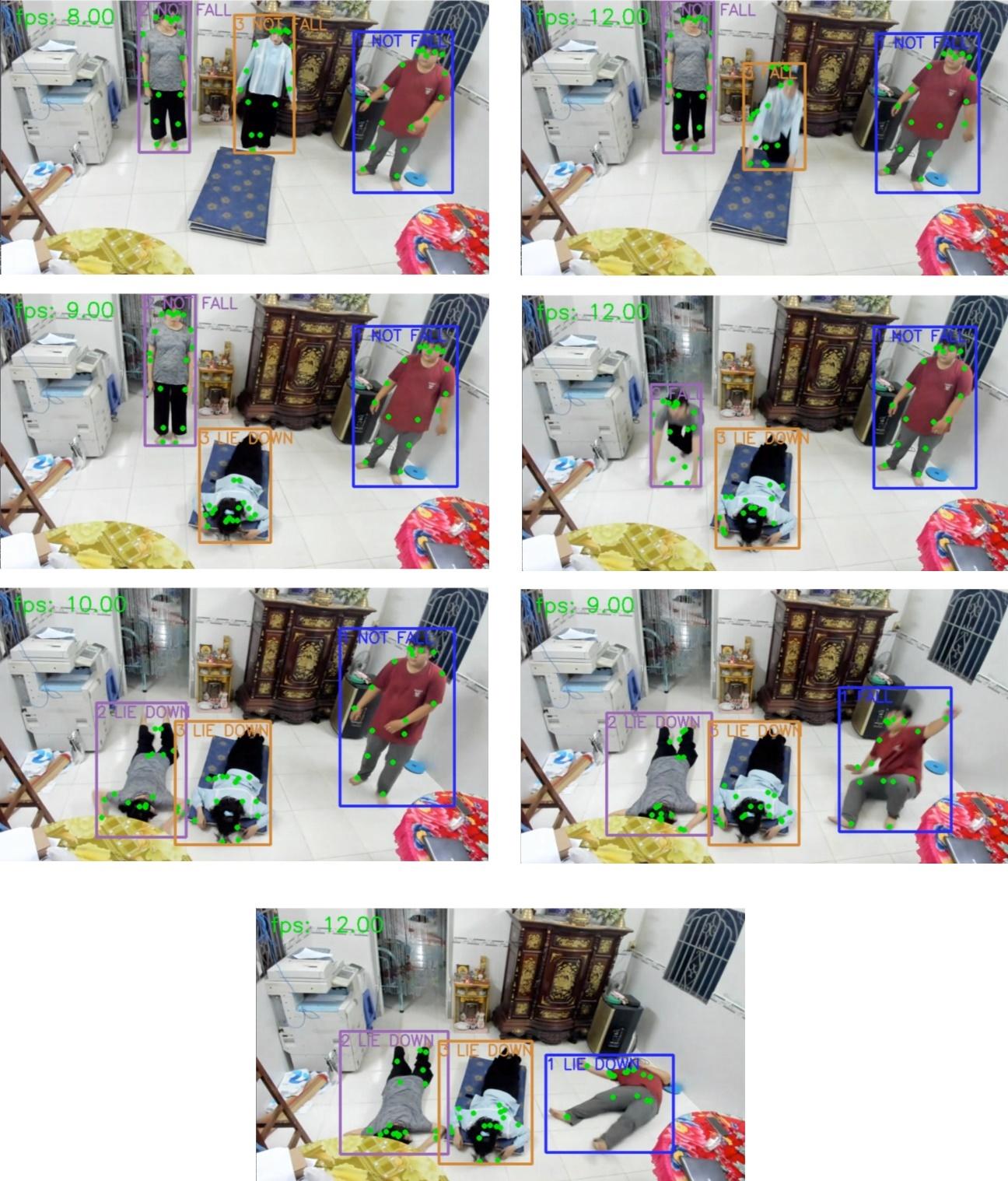
*Bảng 4.5 Thời gian thực thi trên mỗi khung hình của hệ thống phát hiện người té ngã dựa vào tư thế*

| Lần đo | Khung hình có 1 người | Khung hình có 2 người | Khung hình có 3 người |
| --- | --- | --- | --- |
| 1 | 0.0406 | 0.0704 | 0.1053 |
| 2 | 0.0464 | 0.072 | 0.1061 |
| 3 | 0.0384 | 0.0708 | 0.0944 |
| 4 | 0.0364 | 0.0744 | 0.1032 |
| 5 | 0.0379 | 0.073 | 0.0931 |
| Trung bình thời gian xử lý 1 khung hình (giây) | 0.04 | 0.07212 | 0.1 |
| FPS trung bình | 25 | 13.87 | 10 |

Nhận xét: hệ thống đạt tốc độ rất tốt khi khung hình có 1 người là 25 FPS, đây là một mức độ xử lý khá ổn định cho ứng dụng thời gian thực. Tuy nhiên khi số người tăng lên thì tốc độ xử lý giảm xuống dưới 20 FPS, tốc độ xử lý này chưa thể đáp ứng các ứng dụng thời gian thực. Vì vậy, hệ thống chỉ phù hợp với các ứng dụng khi khung ảnh trong video chỉ có 1 người.

### Đánh giá thực nghiệm khi trong khung hình có nhiều người

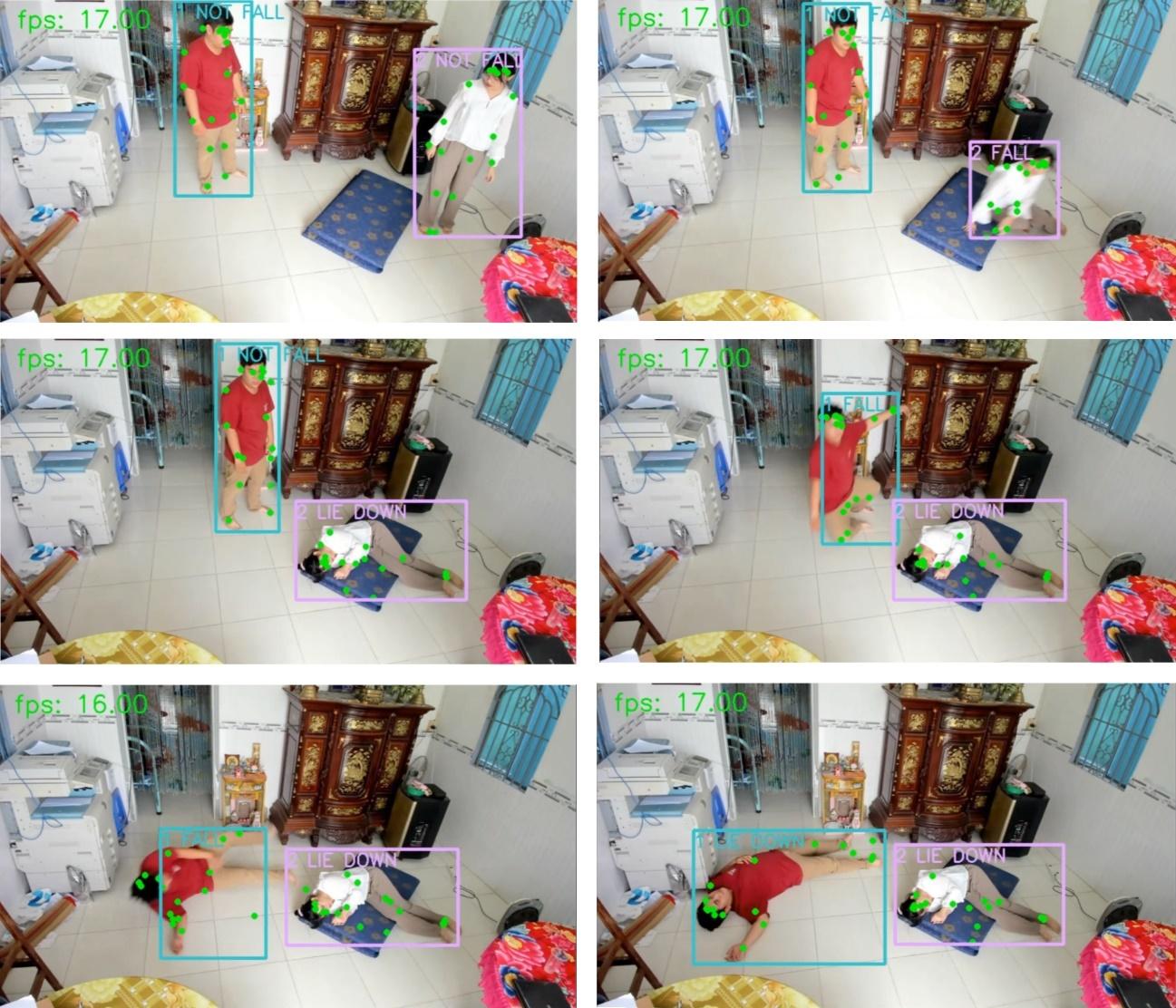
* Kết quả thực nghiệm



*Hình 4.15 Ba người cùng ngã*



*Hình 4.16 Hai người cùng ngã cùng thời điểm*

**

*Hình 4.17 Hai người ngã hai thời điểm liên tiếp nhau*

* Nhận xét:
  + Trong hầu hết các tình huống, mô hình phát hiện người YOLOv8 nano kết hợp với việc theo dõi bằng Deep SORT hoạt động hoạt động rất hiệu quả. Tuy nhiên, khi ánh sáng trở nên yếu hoặc khi con người di chuyển nhanh, hình ảnh từ camera thu về có thể trở nên mờ đi, gây khó khăn cho mạng YOLOv8 nano trong việc nhận diện. Tuy vậy, trong những trường hợp thử nghiệm tổng thể, hiệu suất vẫn duy trì ở mức khá tốt.
  + Mô hình nhận dạng té ngã hoạt động khá tốt với khả năng nhận diện chính xác khi có người bị té ngã. Tuy nhiên, trong một số trường hợp, lớp FALL dễ bị nhầm lẫn với hai lớp khác, đặc biệt là trong điều kiện ánh sáng yếu hoặc khi đối tượng thực hiện các tư thế chuyển động nhanh, dẫn đến mô hình Movenet trích xuất tư thế của người bị dao động và các điểm các điểm đặc trưng trên cơ thể (keypoint) bị dịch chuyển liên tục. Điều này tạo ra sự tương đồng với việc đối tượng đang bị ngã. Kết quả, thông tin đầu vào cho mạng LSTM trở nên nhiễu, gây ra các dự đoán không chính xác trong vài trường hợp.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Với những mục tiêu ban đầu được đề ra, sau quá trình tìm hiểu và nghiên cứu nhiều phương án khác nhau, luận văn đã thực hiện cơ bản một hệ thống có khả năng phát hiện té ngã dựa vào tư thế của con người, với kết quả khá ổn. Tuy nhiên vẫn còn một số hạn chế.

### Kết quả đạt được

* Luận văn đề xuất một phương pháp nhằm theo dõi và dự đoán việc xác định liệu có người bị té ngã trong khung hình chứa một hoặc nhiều người. Ảnh đầu vào sẽ trải qua hai giai đoạn chính để nhận diện và theo dõi đối tượng. Đầu tiên, qua mô hình phát hiện đối tượng YOLOv8 nano để xác định vị trí của các đối tượng trong khung ảnh. Tiếp theo, thông qua thuật toán theo dõi DeepSORT để theo dõi và gán các ID cho từng đối tượng trong khung ảnh. Qua hai giai đoạn này, hệ thống sẽ có khả năng theo dõi các đối tượng trong hình ảnh một cách chính xác và liên tục.
* Trích xuất được đặc trưng về tư thế thông qua các điểm các điểm đặc trưng trên cơ thể người bằng mô hình Movenet và nhóm các đặc trưng này vào đúng đối tượng đã được gán ID trước đó.
* Xây dựng được mô hình LSTM nhận diện té ngã với đầu vào là chuỗi đặc trưng về tư thế người của các khung hình liên tiếp nhau.
* Tối ưu mô hình YOLOv8 nano bằng công cụ TensorRT giúp cho việc phát hiện người được nhanh hơn.
* Tối ưu thời gian thực thi của mạng LSTM bằng frozen graph, giúp cho thời gian dự đoán trạng thái té ngã của từng người trong từng khung hình giảm xuống đáng kể.
* Kết hợp các thành phần để tạo thành một hệ thống có khả năng phát hiện, theo dõi và nhận diện xác định liệu có người đang bị ngã hay không.
* Tốc độ xử lý khi có một người trong khung hình đạt 25 FPS sau khi thực hiện tối ưu hóa các mô hình.

### Những hạn chế còn tồn tại

* Mô hình LSTM phát hiện người té ngã thi thoảng cho ra kết quả dự đoán chưa chính xác. Điều này là do quá trình thu thập dữ liệu huấn luyện chưa thực sự tốt, chưa thu thập đủ những dữ liệu trong các điều kiện ánh sáng khác nhau hoặc khi đối tượng di chuyển nhanh. Đặc trưng trích xuất từ Movenet cũng không thu.
* Việc kết hợp các mô hình như YOLOv8 nano, Deep SORT, Movenet và LSTM đòi hỏi tài nguyên tính toán lớn dù đã được tối ưu hóa, gây ra sự chậm trễ trong thực thi khi xử lý video thời gian thực, chưa thể xử lý thời gian thực khi có nhiều người trong khung hình.
* Hệ thống chưa có giao diện hoàn chỉnh để gửi thông tin về việc phát hiện té ngã cho người dùng.

## Hướng phát triển

Từ những hạn chế vừa nêu trên, tôi có những hướng phát triển cho đề tài như sau:

* Thay thế các mô hình hiện tại bằng các mô hình tối ưu, hiệu quả hơn.
* Lên thêm nhiều kịch bản về các trường hợp khi ngã và khi không ngã để thu thập thêm dữ liệu trong nhiều điều kiện môi trường khác nhau để mô hình có thể nhận diện người té ngã trong mọi tình huống.
* Kết hợp thêm vài đặc trưng khác để việc nhận diện té ngã hiệu quả hơn, các đặc trưng có thể là tọa độ tâm cơ thể và tốc độ di chuyển của tọa độ tâm.
* Những yếu tố ảnh hưởng đến camera có thể được khắc phục bằng cách thay camera có độ phân giải cao hơn, hoạt động ổn định hơn trong các điều kiện ánh sáng khác nhau.
* Xây dựng thêm giao diện, api để thông báo và gửi thông tin liên quan đến việc phát hiện người té ngã cho người dùng cuối.

# TÀI LIỆU THAM KHẢO

| [1] | Jeong, Sungil & Kang, Sungjoo & Chun, Ingeol. (2019). Human-skeleton based Fall-Detection Method using LSTM for Manufacturing Industries. 1-4. 10.1109/ITC-CSCC.2019.8793342. |
| --- | --- |
| [2] | Ultralytics. Ultralytics document, [Online]. Available: https://docs.ultralytics.com/#where-to-start. |
| [3] | Adrián Núñez-Marcos, Gorka Azkune, and Ignacio Arganda-Carreras- “Vision-Based Fall Detection with Convolutional Neural Networks” (2017 Dec) |
| [4] | NVIDIA. NVIDIA TensorRT, [Online]. Available: https://developer. nvidia.com/tensorrt. |
| [5] | A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "Real-time object detection method based on improved YOLOv4-tiny," ArXiv Preprint arXiv:2011.04244, 2022. |
| [6] | K. Shailender, S. Pranav, Vishal and P. Nitin, "Object tracking and counting in a zone using YOLOv4, DeepSORT and TensorFlow," *Proceedings of the International Conference on Artificial Intelligence and Smart Systems (ICAIS),* pp. 1017-1022, 2021. |
| [7] | N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," *2017 IEEE International Conference on Image Processing (ICIP),* pp. 3645-3649, 21017. |
| [8] | K. Schindler and L. van Gool, "Action snippets: How many frames does human action recognition require?," 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 2008, pp. 1-8, doi: 10.1109/CVPR.2008.4587730. |
| [9] | Lei Mao, " Save, Load and Inference From TensorFlow 2.x Frozen Graph," [Online]. Available: https://leimao.github.io/blog/Save-Load-Inference-From-TF2-Frozen-Graph/. |
| [10] | Tzutalin, "LabelImg," [Online]. Available: https://github.com/tzutalin/labelImg, 2015. |

| [11] | Phương Thu Nguyễn. (26/03/2020). “Phòng chống té ngã ở người cao tuổi”, [Online]. Available: https://suckhoedoisong.vn/phong-chong-te-nga-o-nguoi-cao-tuoi-169176004.htm. |
| --- | --- |
| [12] | "Older Adult Falls Reported by State", [Online]. Available: https://www.who.int/publications/i/item/9789241563536 |
| [13] | “WHO global report on falls prevention in older age”, [Online]. Available: https://www.cdc.gov/falls/data/falls-by-state.html |

**LÝ LỊCH TRÍCH NGANG**

| Họ và tên: Lê Trần Quang Huy | Giới tính: Nam |
| --- | --- |
| Ngày, tháng, năm sinh: 02/04/1999 | Nơi sinh: Tiền Giang |
| Điện thoại liên hệ: 0854688879 | Email: letranquanghuy0204@gmail.com |
| Địa chỉ liên hệ: 45/27-29 Lê Văn Huân, phường 13, quận Tân Bình, TP.HCM | |

**Quá trình đào tạo**

- Từ tháng 9/2017 đến tháng 11/2021: học kỹ sư ngành Kỹ Thuật Điều Khiển Và Tự Động Hóa tại trường Đại học Bách Khoa, ĐHQG-HCM.

- Từ tháng 3/2022 đến nay: học thạc sĩ ngành Kỹ Thuật Điều Khiển Và Tự Động Hóa tại trường Đại học Bách Khoa, ĐHQG-HCM.

**Quá trình công tác**

- Từ tháng 12/2021 đến nay: Kĩ sư phần mềm tại Công ty Bosch Global Software Technologies Vietnam. Tòa nhà Etown 1, số 364, Đường Cộng Hòa, Phường 13, Quận Tân Bình, Thành phố Hồ Chí.