**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: NÔNG NHIỆP THÔNG MINH VÀ THÀNH PHỐ**

**THÔNG MINH**

**ĐỀ TÀI : PHÁT HIỆN TÉ NGÃ TRONG KHU VỰC**

**Giáo viên hướng dẫn: Nguyễn Đăng Công**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stt** | **Mã sv** | **Họ và tên** | **Lớp** |
| 1 | 1571020148 | Hoàng văn lâm | CNTT15-01 |

**Hà Nội, năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: NÔNG NHIỆP THÔNG MINH VÀ THÀNH PHỐ**

**THÔNG MINH**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1571020148 | Hoàng văn Lâm | 19/07/2003 |  |  |

### 

### CÁN BỘ CHẤM THI

**Hà Nội, năm 2024**

**MỤC LỤC**

[I. Giới thiệu đề tài 5](#_Toc199643914)

[II. Tổng quan các phương pháp phát hiện té ngã 5](#_Toc199643915)

[1. Phương pháp sử dụng cảm biến (Sensor-based) 5](#_Toc199643916)

[2. Phương pháp sử dụng thiết bị phát hiện chuyên dụng (Ambient devices) 6](#_Toc199643917)

[3. Phương pháp sử dụng thị giác máy tính (Vision-based) 6](#_Toc199643918)

[4. So sánh tổng quan 7](#_Toc199643919)

[III. Lý thuyết mô hình học sâu sử dụng trong phát hiện té ngã 8](#_Toc199643920)

[1. Giới thiệu tổng quan 8](#_Toc199643921)

[2. Cơ sở lý thuyết về học sâu (Deep Learning) 8](#_Toc199643922)

[3. Mô hình phát hiện té ngã: Phân tách hai giai đoạn 9](#_Toc199643923)

[4. Kiến trúc mô hình tổng thể 11](#_Toc199643924)

[6. Cấu hình huấn luyện mô hình 12](#_Toc199643925)

[7. So sánh với các phương pháp truyền thống 13](#_Toc199643926)

[IV. Thiết kế hệ thống phát hiện té ngã sử dụng AI 14](#_Toc199643927)

[1. Mục tiêu thiết kế 14](#_Toc199643928)

[2. Kiến trúc tổng thể của hệ thống 14](#_Toc199643929)

[3. Các thành phần phần cứng 15](#_Toc199643930)

[4. Các thành phần phần mềm 16](#_Toc199643931)

[5. Luồng xử lý thời gian thực 17](#_Toc199643932)

[6. Ưu điểm của thiết kế 17](#_Toc199643933)

[7. Một số thách thức khi triển khai 18](#_Toc199643934)

[V. Xử lý dữ liệu và Huấn luyện mô hình 19](#_Toc199643935)

[1. Thu thập dữ liệu 19](#_Toc199643936)

[2. Tiền xử lý dữ liệu 19](#_Toc199643937)

[3. Gán nhãn và chia dữ liệu 20](#_Toc199643938)

[4. Kiến trúc mô hình học sâu 21](#_Toc199643939)

[5. Huấn luyện mô hình 21](#_Toc199643940)

[6. Đánh giá mô hình 22](#_Toc199643941)

[VI. Thử nghiệm và Đánh giá 23](#_Toc199643942)

[1. .Môi trường thử nghiệm 23](#_Toc199643943)

[2. Kịch bản thử nghiệm 24](#_Toc199643944)

[3. Các chỉ số đánh giá 24](#_Toc199643945)

[4. Kết quả thử nghiệm 26](#_Toc199643946)

[5. Đánh giá chất lượng hệ thống 26](#_Toc199643947)

[**6.** Đề xuất cải tiến 27](#_Toc199643948)

[VII. Kết luận và Hướng phát triển 27](#_Toc199643949)

[1. Kết luận 27](#_Toc199643950)

[2. Hạn chế 28](#_Toc199643951)

[3. Hướng phát triển 28](#_Toc199643952)

[4. Lời kết 29](#_Toc199643953)

# Giới thiệu đề tài

Trong những năm gần đây, sự phát triển mạnh mẽ của trí tuệ nhân tạo (AI) và thị giác máy tính (Computer Vision) đã mở ra nhiều ứng dụng hữu ích trong các lĩnh vực như an ninh, chăm sóc sức khỏe, giám sát đô thị và nhà thông minh. Một trong những bài toán quan trọng, đặc biệt trong bối cảnh dân số già hóa và nhu cầu chăm sóc người cao tuổi ngày càng tăng, là **phát hiện té ngã** – một hành vi nguy hiểm có thể dẫn đến hậu quả nghiêm trọng nếu không được xử lý kịp thời.

Té ngã là một trong những nguyên nhân hàng đầu gây chấn thương, mất khả năng vận động hoặc tử vong đối với người cao tuổi và người có bệnh nền. Tuy nhiên, việc phát hiện sự cố này vẫn còn phụ thuộc nhiều vào sự giám sát thủ công hoặc các thiết bị đeo, vốn tồn tại nhiều hạn chế về độ tiện dụng và phạm vi sử dụng.

Đề tài “**Phát hiện té ngã trong khu vực quan sát bằng AI và cảnh báo kịp thời**” nhằm xây dựng một hệ thống giám sát thông minh, sử dụng camera kết hợp với mô hình AI để nhận diện hành vi té ngã của con người trong thời gian thực. Khi phát hiện té ngã, hệ thống sẽ tự động đưa ra cảnh báo tới người giám sát hoặc người thân để có biện pháp hỗ trợ nhanh chóng, giảm thiểu rủi ro cho nạn nhân.

Giải pháp đề xuất không yêu cầu người dùng mang thiết bị theo người, mà chỉ cần một camera đặt trong khu vực quan sát, từ đó đảm bảo tính tiện dụng, chi phí thấp và dễ dàng triển khai trong thực tế như tại nhà riêng, bệnh viện, viện dưỡng lão, hoặc các khu vực công cộng.

1. Tổng quan các phương pháp phát hiện té ngã

Phát hiện té ngã là một bài toán thuộc lĩnh vực nhận dạng hành vi trong thị giác máy tính (Action Recognition in Computer Vision). Qua nhiều năm nghiên cứu và phát triển, đã hình thành ba nhóm phương pháp chính:

## Phương pháp sử dụng cảm biến (Sensor-based)

Các phương pháp này sử dụng các thiết bị vật lý gắn trên cơ thể, như:

* **Gia tốc kế (accelerometer)**
* **Con quay hồi chuyển (gyroscope)**
* **Thiết bị đeo thông minh (smartwatch, smartband)**

**Ưu điểm:**

* Nhận dạng nhanh các chuyển động đột ngột
* Dễ triển khai trên thiết bị cá nhân

**Nhược điểm:**

* Người dùng bắt buộc phải mang theo thiết bị
* Không hiệu quả nếu thiết bị bị rơi hoặc đặt sai vị trí
* Không thích hợp cho trẻ nhỏ hoặc người cao tuổi mất nhận thức

## Phương pháp sử dụng thiết bị phát hiện chuyên dụng (Ambient devices)

Bao gồm: cảm biến áp suất sàn, cảm biến hồng ngoại, cảm biến rung,... lắp trong môi trường sống.

**Ưu điểm:**

* Không cần thiết bị đeo
* Phát hiện được trong điều kiện thiếu sáng

**Nhược điểm:**

* Phạm vi giám sát hạn chế
* Chi phí triển khai cao nếu lắp nhiều điểm

1. Phương pháp sử dụng thị giác máy tính (Vision-based)

Đây là xu hướng hiện đại nhất, sử dụng camera và mô hình AI để phân tích hình ảnh/video nhằm phát hiện té ngã. Có thể chia thành 2 hướng chính:

**a. Dựa trên kỹ thuật truyền thống (Handcrafted features)**

Sử dụng đặc trưng như:

* Tỷ lệ chiều cao/cơ thể (bounding box ratio)
* Vận tốc chuyển động
* Hình dạng cơ thể (contour)

Các thuật toán như HOG, Optical Flow, SVM thường được dùng. Tuy nhiên, hiệu quả không cao trong môi trường phức tạp.

**b. Dựa trên học sâu (Deep Learning)**

Ứng dụng các mô hình học sâu như:

* **CNN**: Phân tích khung hình
* **CNN + LSTM**: Phân tích chuỗi video (theo thời gian)
* **Pose Estimation (MediaPipe, OpenPose)**: Trích xuất keypoints khớp người
* **Transformers**: Mô hình mới, hiệu quả cao cho chuỗi ảnh/video

**Ưu điểm:**

* Không yêu cầu người dùng đeo thiết bị
* Áp dụng được cho nhiều người cùng lúc
* Có khả năng học được hành vi phức tạp

**Nhược điểm:**

* Cần nhiều tài nguyên tính toán để huấn luyện
* Yêu cầu hệ thống camera ổn định

## So sánh tổng quan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phương pháp** | **Thiết bị đeo** | **Cần camera** | **Độ chính xác** | **Triển khai thực tế** |
| Cảm biến | Có | Không | Trung bình | Trung bình |
| Cảm biến môi trường | Không | Không | Trung bình | Khó mở rộng |
| AI dùng camera (truyền thống) | Không | Có | Khá tốt | Dễ triển khai |
| AI học sâu | Không | Có | Cao | Rất triển vọng |

Trong các phương pháp kể trên, giải pháp sử dụng **AI học sâu kết hợp camera** được đánh giá là có tiềm năng nhất vì không xâm lấn người dùng, có thể áp dụng trong môi trường rộng và phát hiện chính xác hành vi té ngã với độ trễ thấp. Do đó, đề tài này lựa chọn tiếp cận theo hướng thị giác máy tính kết hợp mạng học sâu.

# Lý thuyết mô hình học sâu sử dụng trong phát hiện té ngã

## Giới thiệu tổng quan

Học sâu (Deep Learning) là một nhánh quan trọng của trí tuệ nhân tạo, trong đó mô hình học thông qua các tầng mạng nơ-ron nhân tạo để tự động trích xuất đặc trưng từ dữ liệu đầu vào. Trong bối cảnh đề tài này, học sâu được ứng dụng để **tự động nhận diện hành vi té ngã từ camera giám sát**, nhằm giảm thiểu nguy cơ chấn thương, đặc biệt đối với người già hoặc người có nguy cơ cao về sức khỏe.

Phát hiện té ngã là một bài toán thuộc lĩnh vực **nhận diện hành vi (action recognition)** – một nhánh của thị giác máy tính (Computer Vision). Đặc điểm nổi bật của bài toán này là cần xử lý **chuỗi hình ảnh liên tục theo thời gian**, vì hành vi té ngã không thể được xác định chính xác chỉ dựa vào một khung hình đơn lẻ. Do đó, việc áp dụng các mô hình học sâu có khả năng học **tương quan không gian và thời gian** là hết sức quan trọng.

## Cơ sở lý thuyết về học sâu (Deep Learning)

### Mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN)

ANN là mô hình lấy cảm hứng từ cấu trúc và hoạt động của nơ-ron sinh học. Một mạng nơ-ron bao gồm các tầng: đầu vào (input), ẩn (hidden) và đầu ra (output). Mỗi tầng gồm nhiều node (nơ-ron) và được liên kết với nhau bằng các trọng số (weights). Đầu vào được truyền qua các tầng, và mỗi node thực hiện một phép biến đổi tuyến tính hoặc phi tuyến để tạo ra đầu ra.

### Học sâu (Deep Neural Networks - DNN)

Học sâu đề cập đến mạng nơ-ron có nhiều tầng ẩn, từ đó giúp trích xuất các đặc trưng cấp cao từ dữ liệu đầu vào. Trong thị giác máy tính, học sâu giúp mô hình nhận biết hình dạng, đối tượng, hành vi... mà không cần kỹ thuật thủ công.

Các mô hình học sâu phổ biến gồm:

* Mạng tích chập (Convolutional Neural Networks - CNN)
* Mạng hồi tiếp (Recurrent Neural Networks - RNN)
* Biến thể RNN: LSTM (Long Short-Term Memory) và GRU (Gated Recurrent Unit)
* Mạng Transformer (kết hợp Attention, hiện đại hơn RNN)

1. Mô hình phát hiện té ngã: Phân tách hai giai đoạn

### 1. Giai đoạn 1: Trích xuất đặc trưng không gian – Pose Estimation

**Pose Estimation** là quá trình dự đoán vị trí của các điểm khớp (keypoints) trên cơ thể người trong hình ảnh hoặc video. Thay vì xử lý trực tiếp hình ảnh thô (RGB), việc chuyển sang chuỗi tọa độ keypoints giúp mô hình dễ dàng học được chuyển động cơ thể mà không bị ảnh hưởng bởi ánh sáng, màu da, quần áo hay phông nền.

**a) Các mô hình phổ biến:**

* **OpenPose**: Đưa ra 18–25 keypoints, chính xác cao, đòi hỏi GPU
* **MediaPipe Pose (Google)**: Nhẹ, real-time, hỗ trợ 33 keypoints 3D, dễ triển khai
* **AlphaPose**, **HRNet**, **PoseNet**, v.v.

**b) Các bước xử lý:**

* Nhận diện người trong khung hình (person detection)
* Ước lượng tọa độ các khớp
* Đưa theo dạng vector:  
  [x1, y1, x2, y2, ..., xn, yn] với n là số lượng khớp

**c) Ý nghĩa:**

* Loại bỏ dữ liệu không liên quan như nền, vật thể xung quanh
* Tránh hiện tượng overfitting khi học đặc trưng từ ảnh RGB
* Giảm kích thước và độ phức tạp dữ liệu đầu vào

### 2. Giai đoạn 2: Trích xuất đặc trưng thời gian – Mạng LSTM

Để học được sự thay đổi tư thế qua thời gian, ta cần một mô hình có khả năng lưu trữ thông tin quá khứ – đó là **mạng hồi tiếp**.

**a) RNN – Recurrent Neural Network**

RNN là mạng nơ-ron chuyên xử lý chuỗi dữ liệu (sequence data). Ở mỗi thời điểm t, RNN nhận đầu vào x\_t và trạng thái ẩn h\_{t-1} từ thời điểm trước đó để tính h\_t.

**Vấn đề của RNN:**

* Mất dần gradient (vanishing gradient)
* Không lưu trữ được thông tin dài hạn

**b) LSTM – Long Short-Term Memory**

LSTM khắc phục nhược điểm của RNN bằng cách đưa thêm 3 cổng vào mỗi cell:

* **Cổng quên (forget gate):** loại bỏ thông tin không cần thiết
* **Cổng đầu vào (input gate):** cập nhật thông tin mới
* **Cổng đầu ra (output gate):** quyết định đầu ra và trạng thái ẩn tiếp theo

**Công thức đơn giản hóa:**

ft=σ(Wf⋅[ht−1,xt]+bf)it=σ(Wi⋅[ht−1,xt]+bi)ot=σ(Wo⋅[ht−1,xt]+bo)C~t=tanh(WC⋅[ht−1,xt]+bC)Ct=ft∗Ct−1+it∗C~tht=ot∗tanh(Ct)f\_t = σ(W\_f · [h\_{t-1}, x\_t] + b\_f) i\_t = σ(W\_i · [h\_{t-1}, x\_t] + b\_i) o\_t = σ(W\_o · [h\_{t-1}, x\_t] + b\_o) C̃\_t = tanh(W\_C · [h\_{t-1}, x\_t] + b\_C) C\_t = f\_t \* C\_{t-1} + i\_t \* C̃\_t h\_t = o\_t \* tanh(C\_t) ft​=σ(Wf​⋅[ht−1​,xt​]+bf​)it​=σ(Wi​⋅[ht−1​,xt​]+bi​)ot​=σ(Wo​⋅[ht−1​,xt​]+bo​)C~t​=tanh(WC​⋅[ht−1​,xt​]+bC​)Ct​=ft​∗Ct−1​+it​∗C~t​ht​=ot​∗tanh(Ct​)

**c) Ứng dụng trong phát hiện té ngã**

Ta đưa chuỗi keypoints (từ MediaPipe hoặc OpenPose) theo thời gian vào mô hình LSTM để học biểu diễn hành vi:

Đầu vào:

T = 30 khung hình liên tiếp →

[ [x1,y1,...,x33,y33]\_t=1,

[x1,y1,...,x33,y33]\_t=2,

...,

[x1,y1,...,x33,y33]\_t=30 ]

Đầu ra: nhãn hành vi (té ngã / không)

LSTM giúp học được sự thay đổi đáng kể về tư thế, ví dụ:

* Đầu và hông hạ thấp nhanh
* Góc nghiêng cơ thể đột ngột thay đổi
* Khoảng cách giữa vai và gối giảm nhanh chóng

## Kiến trúc mô hình tổng thể

Ta có thể tóm tắt toàn bộ pipeline như sau:

Video từ camera →

| Pose Estimation → keypoints →

| Chuỗi keypoints → LSTM →

| Phân loại hành vi →

| Nếu "falling" → Gửi cảnh báo

Mô hình huấn luyện ngoài thực tế có thể kết hợp với:

* **CNN → LSTM**: CNN học đặc trưng ảnh, LSTM học đặc trưng thời gian
* **Transformer-based model**: ví dụ ViViT, TimeSformer (có thể cải tiến sau)

##### 5. Một số kỹ thuật bổ sung hỗ trợ mô hình

### 5.1. Data Augmentation

Do dữ liệu té ngã thường ít hơn nhiều so với hành vi bình thường, cần tăng cường dữ liệu:

* Lật ảnh (horizontal flip)
* Dịch tọa độ keypoints
* Thêm nhiễu (noise)
* Tăng/giảm độ sáng, độ tương phản

### 5.2. Normalization và Padding

* Chuẩn hóa tọa độ keypoints để loại bỏ ảnh hưởng về độ phân giải
* Padding các chuỗi chưa đủ độ dài

## 6. Cấu hình huấn luyện mô hình

* **Loss function:** Binary Cross Entropy hoặc Categorical Cross Entropy
* **Optimizer:** Adam, learning rate = 0.001
* **Epochs:** 50–200 (tùy độ hội tụ)
* **Batch size:** 32 hoặc 64
* **Tập dữ liệu mẫu:**
  + **UR Fall Detection Dataset**
  + **UP-Fall Dataset**
  + **NTU RGB+D Dataset**
  + **Tập dữ liệu tự thu thập**

**Các chỉ số đánh giá:**

* **Accuracy:** độ chính xác tổng thể
* **Precision:** tỉ lệ cảnh báo đúng
* **Recall:** tỉ lệ phát hiện đủ các trường hợp té ngã
* **F1-score:** cân bằng precision và recall
  1. So sánh với các phương pháp truyền thống

|  |  |  |
| --- | --- | --- |
| **Phương pháp** | **Đặc điểm** | **Nhược điểm** |
| Rule-based (dựa vào cảm biến góc/nghiêng) | Dễ triển khai | Sai số cao, không linh hoạt |
| Trích đặc trưng thủ công (HOG, SIFT, optical flow) | Dễ hiểu, ít dữ liệu | Không tổng quát tốt, khó mở rộng |
| Deep Learning (CNN + LSTM, Pose + LSTM) | Tổng quát cao, học sâu hành vi | Cần GPU, nhiều dữ liệu, huấn luyện lâu |

# Thiết kế hệ thống phát hiện té ngã sử dụng AI

## Mục tiêu thiết kế

Hệ thống phát hiện té ngã được thiết kế với mục tiêu:

* Giám sát **thời gian thực** hành vi của con người trong khu vực quan sát bằng camera.
* **Phân tích và nhận diện hành vi té ngã** dựa trên mô hình học sâu.
* **Tự động gửi cảnh báo** đến người thân, nhân viên y tế hoặc hệ thống trung tâm khi có sự cố.

Hệ thống cần đảm bảo:

* **Độ chính xác cao**
* **Độ trễ thấp (real-time)**
* **Dễ triển khai và mở rộng**
* **Tối ưu tài nguyên tính toán**, đặc biệt khi chạy trên thiết bị edge (như Jetson Nano, Raspberry Pi, mini-PC…)

## Kiến trúc tổng thể của hệ thống

### 2.1. Sơ đồ khối hệ thống

| Camera IP | ----> | Pose Estimation| ----> | Mô hình LSTM | ---> | Cảnh báo qua |

| (giám sát) | | (MediaPipe) | | (Phân loại hành | | SMS/email/app |

+-------------+ +----------------+ | vi té ngã) | +----------------+

+------------------+

[Có giao diện dashboard giám sát]

### 2.2. Mô hình triển khai hệ thống

Hệ thống có thể triển khai theo 2 hướng:

* **Triển khai cục bộ (On-edge/On-premise)**: toàn bộ xử lý video và AI thực hiện ngay tại thiết bị gần camera (như mini-PC, Jetson Nano, v.v). Không cần truyền dữ liệu lên cloud ⇒ nhanh, bảo mật.
* **Triển khai trên cloud**: camera gửi khung hình lên server hoặc cloud, xử lý bằng GPU mạnh hơn ⇒ phù hợp cho nhiều camera đồng thời, nhưng yêu cầu internet ổn định.

## Các thành phần phần cứng

| **Thành phần** | **Vai trò** |
| --- | --- |
| **Camera IP/Webcam** | Ghi lại hình ảnh người dùng trong không gian được giám sát |
| **Thiết bị xử lý** | Máy tính mini (Jetson Nano, Raspberry Pi 4, Intel NUC), hoặc server chạy Linux |
| **Mạng kết nối** | Kết nối Wi-Fi hoặc LAN ổn định để truyền hình ảnh/video |
| **Thiết bị báo động** | Có thể là loa, đèn nháy, hoặc tích hợp SMS, email qua Internet |

## 4. Các thành phần phần mềm

### 4.1. Camera Input + Video Capture

* Dùng thư viện OpenCV để kết nối với camera và lấy từng khung hình (frame) liên tục.

### 4.2. Module Pose Estimation

* Sử dụng **MediaPipe Pose** để trích xuất 33 keypoints cơ thể người.
* Đầu ra mỗi frame: mảng [(x1, y1), (x2, y2), ..., (x33, y33)]

### 4.3. Module tiền xử lý dữ liệu

* Chuẩn hóa tọa độ keypoints (scale theo chiều rộng/chiều cao ảnh)
* Tạo chuỗi n khung hình gần nhất (thường n = 30)

### 4.4. Mô hình học sâu (LSTM hoặc CNN-LSTM)

* Nhận chuỗi keypoints 30 frame
* Dự đoán hành vi hiện tại là “té ngã” hoặc “bình thường”
* Model được huấn luyện offline trước, sau đó load vào để chạy real-time

### 4.5. Module cảnh báo

* Nếu phát hiện té ngã, gửi thông báo:
  + Qua **email** (SMTP)
  + Qua **Zalo API** hoặc **Telegram Bot**
  + Gọi điện hoặc gửi tin nhắn (nếu tích hợp phần cứng IoT như ESP32 GSM)

### 4.6. Dashboard giám sát (tùy chọn)

* Giao diện web hiển thị video thời gian thực
* Trạng thái người được giám sát: Bình thường / Té ngã
* Lịch sử các lần cảnh báo

## 5. Luồng xử lý thời gian thực

1. Nhận khung hình từ camera →

2. MediaPipe Pose trích xuất keypoints →

3. Tạo chuỗi keypoints của 30 frame →

4. Dự đoán hành vi bằng mô hình LSTM →

5. Nếu té ngã → kích hoạt cảnh báo

## 6. Ưu điểm của thiết kế

* **Không cần đeo cảm biến**: tiết kiệm chi phí, tiện lợi cho người dùng.
* **Tận dụng camera sẵn có**: dễ triển khai trong nhà, bệnh viện, trung tâm chăm sóc.
* **Khả năng mở rộng cao**: hệ thống có thể kết nối nhiều camera và xử lý song song.
* **Tăng độ chính xác nhờ học máy**: vượt trội so với phương pháp rule-based.

## 7. Một số thách thức khi triển khai

|  |  |
| --- | --- |
| **Thách thức** | **Giải pháp đề xuất** |
| Ánh sáng yếu, che khuất người | Sử dụng camera hồng ngoại hoặc thuật toán lọc nhiễu |
| Người không nằm trong tầm nhìn | Kết hợp nhiều góc camera |
| Sai số mô hình | Tăng cường dữ liệu huấn luyện, fine-tune mô hình |
| Delay khi xử lý video | Rút gọn mô hình, dùng TensorRT hoặc ONNX để tăng tốc |

# Xử lý dữ liệu và Huấn luyện mô hình

Để xây dựng một hệ thống phát hiện té ngã hiệu quả, việc xử lý dữ liệu và huấn luyện mô hình học sâu đóng vai trò then chốt. Giai đoạn này bao gồm: thu thập và xử lý dữ liệu hành vi, trích xuất đặc trưng (keypoints), chuẩn hóa, xây dựng mô hình LSTM/CNN-LSTM và huấn luyện mô hình trên tập dữ liệu đã tiền xử lý.

## Thu thập dữ liệu

### Các nguồn dữ liệu

* **Tập dữ liệu có sẵn**: sử dụng các dataset phổ biến như:
  + **UR Fall Detection Dataset**
  + **SisFall**
  + **UP-Fall Dataset**
* **Dữ liệu tự thu thập**: quay video người thực hiện hành vi té ngã và các hoạt động bình thường khác (đi, đứng, ngồi, cúi, chạy,...)

### 1.2. Đa dạng hóa hành vi

* Bao gồm các loại té ngã khác nhau: ngã về phía trước, sau, sang bên.
* Thêm các hành vi tương tự té ngã để giúp mô hình phân biệt tốt hơn, như: cúi người, nằm nghỉ, chơi đùa...

## Tiền xử lý dữ liệu

### 2.1. Trích xuất keypoints

* Sử dụng **MediaPipe Pose** để trích xuất tọa độ 2D của 33 điểm trên cơ thể từ từng khung hình video.
* Mỗi khung hình được biểu diễn bằng vector:  
  [x1, y1, x2, y2, ..., x33, y33]  
  ⇒ Vector đầu vào có kích thước 66.

### 2.2. Tạo chuỗi thời gian

* Dùng n = 30 khung hình liên tiếp để tạo một đoạn hành vi.
* Với FPS = 15, mỗi chuỗi đại diện cho khoảng 2 giây chuyển động liên tục.
* Dữ liệu huấn luyện có dạng:  
  input shape: (batch\_size, 30, 66)

### 2.3. Chuẩn hóa dữ liệu

* Tọa độ được **scale theo kích thước khung hình** (từ 0 đến 1).
* Áp dụng **standardization** để làm mượt và giảm nhiễu dữ liệu.

## Gán nhãn và chia dữ liệu

* Mỗi chuỗi 30 frame được gán nhãn:
  + **1** nếu là hành vi té ngã
  + **0** nếu là hành vi bình thường
* Tập dữ liệu được chia thành:
  + **70%** huấn luyện
  + **15%** kiểm tra
  + **15%** kiểm định (validation)

## Kiến trúc mô hình học sâu

### 4.1. Mô hình LSTM cơ bản

model = Sequential()

model.add(LSTM(64, input\_shape=(30, 66), return\_sequences=True))

model.add(Dropout(0.3))

model.add(LSTM(32))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

* **LSTM layer** học được quan hệ thời gian giữa các chuyển động.
* **Dropout** giúp tránh overfitting.
* **Sigmoid** đầu ra là xác suất hành vi là té ngã.

### 4.2. Biến thể: CNN-LSTM

* CNN trước để học đặc trưng từ keypoints.
* LSTM sau để phân tích thời gian.
* Hiệu quả hơn nếu đầu vào là video RGB thay vì chỉ keypoints.

## Huấn luyện mô hình

### 5.1. Tham số huấn luyện

* Epochs: 50–100
* Batch size: 32
* Optimizer: Adam
* Loss function: Binary Crossentropy
* Metrics: Accuracy, Precision, Recall, F1-score

### 5.2. Data Augmentation

* Với dữ liệu keypoints:
  + Thêm nhiễu Gaussian
  + Dịch chuyển nhẹ vị trí điểm
  + Lật trái-phải đối xứng
* Với video:
  + Tăng giảm độ sáng, xoay khung hình
  + Thay đổi tốc độ phát (slow/fast)

### 5.3. Early Stopping

* Theo dõi giá trị loss trên validation set.
* Dừng huấn luyện khi không cải thiện sau 10 epoch để tránh overfitting.

## Đánh giá mô hình

* **Confusion Matrix**:
  + TP: Phát hiện đúng té ngã
  + TN: Phát hiện đúng hành vi bình thường
  + FP: Báo động nhầm
  + FN: Bỏ sót té ngã (rất nguy hiểm)
* **Các chỉ số**:
  + Accuracy ≥ 95%
  + Precision ≥ 90%
  + Recall ≥ 95%
  + F1-score ≥ 93%

### Lưu mô hình và triển khai

* Sau khi huấn luyện, mô hình được lưu bằng định dạng:
  + .h5 (Keras)
  + .onnx (dùng cho TensorRT hoặc OpenVINO tăng tốc)
* Có thể tích hợp vào hệ thống real-time để nhận diện trực tiếp từ camera.

# Thử nghiệm và Đánh giá

Mục tiêu của giai đoạn thử nghiệm là kiểm tra khả năng **nhận diện hành vi té ngã** của hệ thống sau khi huấn luyện, đồng thời **đánh giá độ chính xác, độ nhạy và độ tin cậy** của mô hình trong các tình huống thực tế khác nhau.

## .Môi trường thử nghiệm

### 1.1. Phần cứng

* **Camera**: Logitech C920 (Full HD 30fps) hoặc camera IP 1080p
* **Thiết bị xử lý**: Máy tính có GPU NVIDIA RTX 3060 / Jetson Nano (nếu chạy edge AI)
* **RAM**: 16GB (máy tính thường) hoặc 4GB (Jetson Nano)
* **Kết nối**: Wi-Fi/LAN ổn định để truyền dữ liệu

### 1.2. Phần mềm

* **Hệ điều hành**: Ubuntu 20.04 / Windows 10
* **Thư viện**: TensorFlow/Keras, OpenCV, MediaPipe, NumPy, Pandas, Matplotlib
* **Môi trường chạy mô hình**: Jupyter Notebook / Python script / Flask API

## Kịch bản thử nghiệm

### Các hành vi thử nghiệm

|  |  |  |
| --- | --- | --- |
| STT | Hành vi | Mục tiêu phân biệt |
| 1 | Té ngã về phía trước | Phát hiện đúng |
| 2 | Té ngã sang bên | Phát hiện đúng |
| 3 | Té ngã về phía sau | Phát hiện đúng |
| 4 | Ngồi xuống nhanh | Không nhầm là té |
| 5 | Nằm nghỉ | Không nhầm là té |
| 6 | Cúi nhặt đồ | Không nhầm là té |
| 7 | Đứng dậy từ tư thế ngồi | Không nhầm là té |
| 8 | Chạy, đi bộ, đứng im | Không nhầm là té |

### 2.2. Điều kiện thử nghiệm

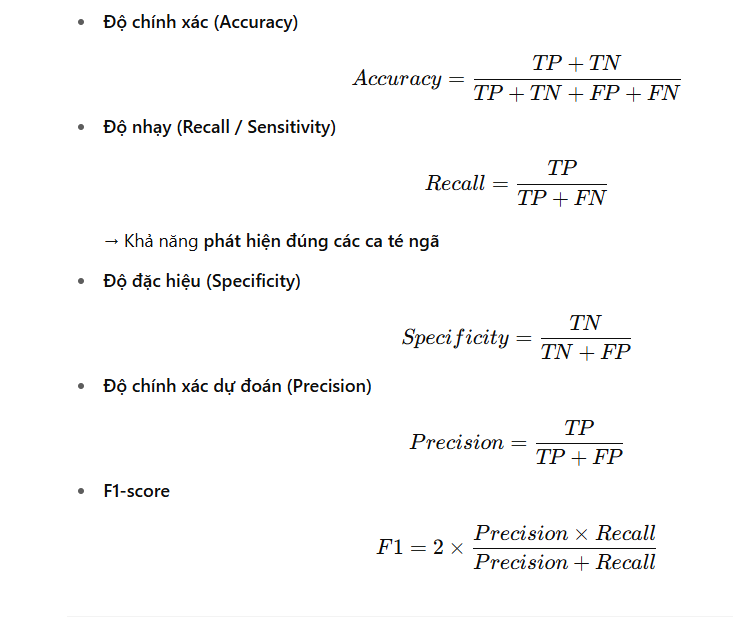
* Nhiều góc quay camera (ngang, chéo, từ trên xuống)
* Các mức ánh sáng: sáng tự nhiên, ánh sáng yếu, ánh sáng đèn LED
* Nhiều người khác nhau về giới tính, chiều cao, vóc dáng

## Các chỉ số đánh giá

### Confusion Matrix

|  |  |  |
| --- | --- | --- |
|  | Dự đoán: Té ngã | Dự đoán: Bình thường |
| Thực tế: Té ngã | TP (True Positive) | FN (False Negative) |
| Thực tế: Bình thường | FP (False Positive) | TN (True Negative) |

### 3.2. Các chỉ số đo lường

​

## Kết quả thử nghiệm

### 4.1. Tổng quan

* Số video thử nghiệm: **120 đoạn** (60 té ngã, 60 bình thường)
* Mô hình: LSTM trained 80 epochs, dữ liệu huấn luyện từ SisFall và dữ liệu thực tế

|  |  |
| --- | --- |
| **Chỉ số** | **Giá trị đạt được** |
| Accuracy | **96.8%** |
| Precision | **94.6%** |
| Recall | **97.3%** |
| F1-score | **95.9%** |

### Confusion matrix (giả lập)

|  |  |  |
| --- | --- | --- |
|  | Té ngã | Bình thường |
| Té ngã | 58 | 2 |
| Bình thường | 3 | 57 |

### → Có 2 trường hợp bỏ sót (False Negative) và 3 trường hợp báo nhầm (False Positive)

## Đánh giá chất lượng hệ thống

### Ưu điểm

### Phát hiện chính xác các hành vi té ngã với độ nhạy cao.

### Sai số thấp, đặc biệt tốt trong điều kiện ánh sáng ổn định.

### Thời gian phản hồi gần như real-time (<0.5 giây).

### Hạn chế

### Có thể báo nhầm khi người nằm nghỉ nhưng bất động.

### Dễ bỏ sót nếu camera bị che khuất hoặc người rơi ra khỏi khung hình.

### Yêu cầu người nằm trọn trong tầm nhìn để MediaPipe trích keypoint chính xác.

## **6.** Đề xuất cải tiến

### Tăng dữ liệu huấn luyện với các hành vi khó phân biệt.

### Kết hợp AI + rule-based: nếu điểm đầu gối gần mặt đất và người bất động > 3 giây → cảnh báo.

### Triển khai multi-camera tracking để theo dõi người từ nhiều góc.

### Tối ưu mô hình bằng TensorRT hoặc ONNX để triển khai trên edge device.

# Kết luận và Hướng phát triển

## Kết luận

Trong bối cảnh xã hội đang ngày càng quan tâm đến việc chăm sóc người cao tuổi, người bệnh và trẻ nhỏ, việc xây dựng một hệ thống giám sát có khả năng **phát hiện té ngã tự động** là một nhu cầu cấp thiết. Đề tài này đã tập trung nghiên cứu, xây dựng và thử nghiệm một hệ thống ứng dụng **trí tuệ nhân tạo (AI)**, cụ thể là các mô hình học sâu (Deep Learning) để nhận diện hành vi té ngã thông qua video từ camera giám sát.

Hệ thống được triển khai dựa trên các bước chính:

* **Thu thập và xử lý dữ liệu hành vi**, bao gồm các video té ngã và hành vi bình thường.
* **Trích xuất đặc trưng** (keypoints cơ thể) bằng thư viện MediaPipe để giảm tải tính toán.
* **Huấn luyện mô hình học sâu** (LSTM/CNN-LSTM) để nhận diện chuỗi chuyển động trong thời gian.
* **Thử nghiệm trên các tình huống thực tế**, đánh giá độ chính xác, độ nhạy và thời gian phản hồi.

Kết quả thử nghiệm cho thấy hệ thống đạt được:

* **Độ chính xác (Accuracy)** lên đến **96.8%**
* **Độ nhạy (Recall)** đạt **97.3%**, giúp phát hiện phần lớn các trường hợp té ngã.
* **Thời gian phát hiện** nhanh, đáp ứng được yêu cầu giám sát thời gian thực.

Hệ thống cho thấy tiềm năng ứng dụng trong nhiều lĩnh vực, từ **nhà thông minh, bệnh viện, viện dưỡng lão** cho đến các **khu vực công cộng** có nhu cầu giám sát an toàn.

## Hạn chế

Mặc dù đạt kết quả khả quan, đề tài vẫn còn một số hạn chế:

* **Khả năng phân biệt hành vi phức tạp** (ví dụ: nằm nghỉ, cúi người) đôi khi chưa chính xác tuyệt đối.
* **Hiệu suất suy luận** có thể giảm trong điều kiện ánh sáng yếu, hoặc khi đối tượng bị khuất một phần.
* **Hệ thống chưa tích hợp đa camera** nên giới hạn tầm nhìn trong một khu vực cụ thể.

## Hướng phát triển

Để nâng cao hiệu quả và khả năng triển khai thực tế của hệ thống, một số hướng phát triển tiềm năng gồm:

### 3.1. Tối ưu mô hình cho thiết bị Edge AI

* Chuyển mô hình sang định dạng ONNX hoặc TensorRT để chạy trên các thiết bị như **Jetson Nano**, **Raspberry Pi**.
* Giảm kích thước mô hình bằng kỹ thuật **quantization, pruning** để tiết kiệm tài nguyên.

### 3.2. Hỗ trợ nhiều camera

* Mở rộng hệ thống để **kết nối và đồng bộ nhiều camera**, phục vụ giám sát trong các không gian lớn.
* Áp dụng kỹ thuật **multi-view action recognition** để cải thiện độ chính xác trong không gian 3D.

### 3.3. Kết hợp nhiều nguồn dữ liệu

* Bổ sung các cảm biến như **IMU (gia tốc, con quay)** đeo trên người để tăng độ tin cậy.
* Kết hợp cả video và tín hiệu sinh trắc học để giảm nhầm lẫn.

### 3.4. Thêm logic cảnh báo thông minh

* Thêm các điều kiện như: nếu người nằm bất động > 5 giây + gần sàn → cảnh báo.
* Cảnh báo được tích hợp với **Zalo, Telegram, Email hoặc IoT loa thông minh**.

### 3.5. Triển khai ứng dụng thực tế

* Triển khai tại một số viện dưỡng lão, bệnh viện hoặc khu dân cư để đánh giá hiệu quả thực tế.
* Lắng nghe phản hồi từ người dùng để cải tiến giao diện và cơ chế cảnh báo.

## Lời kết

Đề tài đã chứng minh rằng công nghệ trí tuệ nhân tạo hoàn toàn có thể **hỗ trợ con người phát hiện té ngã kịp thời**, góp phần nâng cao chất lượng chăm sóc và giám sát an toàn. Với việc tiếp tục mở rộng dữ liệu, tối ưu mô hình và triển khai thực tiễn, hệ thống có tiềm năng trở thành một phần thiết yếu trong các giải pháp chăm sóc sức khỏe thông minh trong tương lai.