

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA ĐIỆN – ĐIỆN TỬ  
BỘ MÔN VIỄN THÔNG

-----o0o-----



## LUẬN VĂN TỐT NGHIỆP

# MẠNG TÍCH CHẬP SÂU – NHẬN DẠNG HÀNH ĐỘNG CON NGƯỜI DEEP CONVOLUTIONAL NETWORK – HUMAN ACTIVITY RECOGNITION

SINH VIÊN THỰC HIỆN:  
**ĐẶNG LÊ ANH KHOA – 1511561**  
**NGUYỄN KHẮC TRUNG TÍN – 1513489**

GIẢNG VIÊN HƯỚNG DẪN:  
**PGS. TS. HÀ HOÀNG KHA**

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2019

Số: \_\_\_\_\_ /BKĐT  
Khoa: **Điện – Điện tử**  
Bộ Môn: **Viễn Thông**

-----☆-----

## **NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP**

1. Họ và tên: **Đặng Lê Anh Khoa** MSSV: 1511561  
**Nguyễn Khắc Trung Tín** MSSV: 1513489

2. Ngành: **Điện – Điện tử** Chuyên ngành: **Kỹ thuật Điện tử - Truyền thông**

3. Đề tài: **Mạng Tích Chập Sâu – Nhận Dạng Hành Động Con Người.**

4. Nhiệm vụ:

- Tìm hiểu về Deep Learning, Deep Convolutional Network.
- Tìm hiểu về các thuật toán để thực hiện bài toán nhận dạng hành động.
- Xây dựng mô hình nhận dạng hành động, huấn luyện mô hình trên bộ dữ liệu AVA.
- Xây dựng chương trình nhận dạng hành động thời gian thực trên mô hình xây dựng.
- Đánh giá mô hình, kết luận, viết cuốn luận văn.

5. Ngày giao nhiệm vụ luận văn: **24/08/2019**

6. Ngày hoàn thành nhiệm vụ: **23/12/2019**

7. Họ và tên người hướng dẫn: **PGS. TS. Hà Hoàng Kha** Phản hướng dẫn

**PGS. TS. Hà Hoàng Kha** **100%**  
BM Viễn Thông, Khoa Điện – Điện Tử

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

*TP.HCM, ngày 23 tháng 12 năm 2019*

**CHỦ NHIỆM BỘ MÔN**

**NGƯỜI HƯỚNG DẪN CHÍNH**

PGS. TS. Hà Hoàng Kha

PGS. TS. Hà Hoàng Kha

### **PHẦN DÀNH CHO KHOA, BỘ MÔN:**

Người duyệt (chấm sơ bộ): .....

Đơn vị: .....

Ngày bảo vệ : .....

Điểm tổng kết: .....

Nơi lưu trữ luận văn: .....

## Lời Cảm Ơn



Được học tập và rèn luyện tại trường Đại Học Bách Khoa TP.HCM trong hơn 4 năm qua là niềm vinh dự và tự hào đối với cá nhân tôi. Trong khoảng thời gian này, tôi đã được truyền đạt nhiều kiến thức, tham gia các hoạt động xã hội, gặp gỡ và sinh hoạt với bạn bè, thầy cô và nhận được rất nhiều sự giúp đỡ. Đây sẽ là những trải nghiệm quý báu cho tôi khi bước chân vào đời.

Vì vậy, tôi xin gửi lời cảm ơn đến tất cả các **thầy cô khoa Điện – Điện tử** nói riêng cũng như **ban lãnh đạo và toàn thể cán bộ, nhân viên, giảng viên** của Trường ĐH Bách Khoa nói chung.

Tiếp theo, tôi xin cảm ơn **thầy Hà Hoàng Kha**. Thầy không chỉ truyền đạt kiến thức mà con tận tình giúp đỡ tôi để tôi hoàn thành luận văn lần này.

Tôi cũng xin dành lời cảm ơn đến **gia đình, người thân** và đặc biệt là **bà mẹ tôi**. Những người đã tạo mọi điều kiện tốt nhất cho tôi để tôi có thể chuyên tâm học tập và hoàn tất luận văn tốt nghiệp.

Cuối cùng, tôi xin dành lời cảm ơn cho tất cả các  **bạn của tôi**. Những người bạn đã luôn ở bên động viên tôi trong suốt quá trình thực hiện. Giúp tôi có thêm động lực và niềm tin để thực hiện luận văn.

Xin chân thành cảm ơn.

Tp. Hồ Chí Minh, ngày 23 tháng 12 năm 2019

Sinh viên

Đặng Lê Anh Khoa

Nguyễn Khắc Trung Tín

## *Lời Cam Đoan*

Tôi tên: Đặng Lê Anh Khoa – Nguyễn Khắc Trung Tín, là sinh viên chuyên ngành Kỹ thuật Điện tử – Truyền thông, khóa 2015, tại Đại học Quốc gia thành phố Hồ Chí Minh – Trường Đại học Bách Khoa. Tôi xin cam đoan những nội dung sau đều là sự thật:

- i. Công trình nghiên cứu này hoàn toàn do chính tôi thực hiện.
- ii. Các tài liệu và trích dẫn trong luận văn này được tham khảo từ các nguồn thực tế, có uy tín và độ chính xác cao.
- iii. Các số liệu và kết quả của công trình này được tôi tự thực hiện một cách độc lập và trung thực.

*Tp. Hồ Chí Minh, ngày 23 tháng 12 năm 2019*

**Sinh viên**

*Đặng Lê Anh Khoa*

*Nguyễn Khắc Trung Tín*

## *Tóm tắt luận văn*

Trong luận văn này, chúng tôi tập trung vào xây dựng một hệ thống nhận dạng hành động của con người. Đây là một lĩnh vực mới và đang được tích cực nghiên cứu trong nhiều năm trở lại đây, với nhiều mô hình được phát triển cho việc nhận dạng đối tượng(object detection) và phân loại hành động (action detection). Việc nhận dạng hành động của con người có thể ứng dụng vào nhiều lĩnh vực như giám sát, cảnh báo ở các khu vực quan trọng như bệnh viện, quân đội, trường học, phát hiện đánh nhau, té ngã, vv.

Nhận dạng hành động con người là việc nhận ra hành động con người đang thực hiện là hành động nào trong một video clip hay video chạy trong thời gian thực. Khi quan sát một sự kiện phức tạp với nhiều đối tượng trong đó, con người không đánh giá từng đối tượng riêng lẻ mà suy ra từ bối cảnh. Bối cảnh xung quanh cung cấp thông tin cần thiết để hiểu hành động. Do đó, chúng tôi giới thiệu một phương pháp thay thế vùng quan tâm (RoI) bằng một module “chú ý” (attention module) để xếp hạng mức độ phù hợp của khu vực không-thời gian cho việc phát hiện nhân vật mà không phải cắt ảnh ra, phương pháp này được gọi là Actor Conditioned Attention Maps (ACAM).

Luận văn trình bày một mô hình nhận dạng hành động được huấn luyện trên tập AVA. Chương trình nhận dạng của chúng tôi sử dụng 2 model chạy song song là model object detection cho việc nhận dạng đối tượng và model action detection cho việc nhận dạng hành động. Chương trình nhận dạng hành động con người của chúng tôi được xử lý bằng máy tính, chạy trên hệ điều hành Ubuntu, framework Keras, backend Tensorflow, và được viết bằng ngôn ngữ Python. Chương trình có thể nhận dạng mười hành động khác nhau là: Đứng, Ngồi, Nói chuyện, Quỳ, Di bộ, Nằm, Cầm/nắm, Trả lời điện thoại, Đánh nhau, Té ngã. Chương trình chạy thời gian thực với 10 – 15 FPS.

## *Abstract*

In this thesis, we focus on building a system of recognizing human actions. Recognition human action is a new field and has been actively researched in recent years, with many models developed for object detection and action detection. The recognition of human activities can be applied in many areas such as surveillance and warnings in important areas such as hospitals, military, schools, detecting fights, falling, etc.

Human action recognition is the recognition of what action a human action is performing in a video clip or video run in real-time. When observing a complex event with multiple objects, humans do not assess each object separately but infer from the context. The surrounding context provides essential information for understanding actions. Therefore, we perform a method to replace region of interest (RoI) with an attention module, which ranks each Spatio-temporal region's relevance to a detected actor instead of cropping, this method is called Actor Conditioned Attention Maps (ACAM).

The thesis performs an action recognition model training on AVA dataset. Our program uses two models running in parallel, model object detection for object recognition and model action detection for action recognition. The human action recognition system is computerized, runs on Ubuntu OS, framework Keras, backend Tensorflow, and is written by Python. Our system can recognition ten different actions as: Stand, Sit, Talk, Knee, Walk, Lie, Carry/Hold, Answer Phone, Fight, Fall down. Our system run in real-time with 10-15 FPS.

## *Danh mục viết tắt*

Viết tắt	Tiếng Anh	Tiếng Việt
ACAM	Actor Conditioned Attention Maps	Bản đồ chú ý điều kiện “actor”
AI	Artificial Intelligence	Trí thông minh nhân tạo
AVA	Atomic Visual Actions	Trực quan hành động ‘atomic’
CNNs	Convolutional Neural Networks	Mạng neuron tích chập
DCNs	Deep Convolutional Networks	Mạng tích chập sâu
DL	Deep Learning	Học Sâu
FPS	Frame per second	Khung hình/ giây
GD	Gradient Descent	Trượt theo dốc
LSTM	Long short-term memory	Mạng bộ nhớ ngắn hạn hướng dài hạn
ML	Machine Learning	Học Máy hay Máy Học
ReLU	Rectified Linear Unit	Đơn vị tuyết tính chỉnh lưu
SGD	Stochastic Gradient Descent	Giảm độ dốc ngẫu nhiên
SSG	Single Shot Multibox Detector	Phát hiện đối tượng trong một lần

## *Danh mục định nghĩa*

**Actor:** Trong luận văn này có ý nghĩa là một người có hành động trong video.

**Atomic actions:** Một chuỗi các hoạt động đơn giản liền nhau mà không bị gián đoạn (hoặc được dự đoán sẽ thực hiện như vậy), cũng có thể xem là hành động xảy ra tức thời.

**Bounding box:** Khung hình bao chứa vật thể được xác định trong quá trình huấn luyện

**Class score:** Trong phân loại đa lớp, mạng neuron có cùng số lượng output node với số lượng lớp. Mỗi output node thuộc về một số lớp và đưa ra một số điểm cho lớp đó.

**Error:** Mức độ sai khác giữa 2 giá trị thực tế và mong muốn.

**Feedforward:** Là một mạng gồm một hay nhiều lớp neuron, trong đó các dây dẫn tín hiệu chỉ truyền theo một chiều từ input qua các lớp, cho đến output.

**Frame:** Khung hình. Được hiểu như một khung hình đầy đủ của Video.

**Hyponyms:** Những từ là ví dụ cụ thể của một từ tổng hợp. Ví dụ: Đỏ, Cam, Vàng là các hyponym của màu sắc.

**Optical Flow:** là khái niệm chỉ sự chuyển động tương đối của các điểm trên bề mặt của một đối tượng, vật thể nào đó gây ra dưới góc quan sát của một điểm mốc (ví dụ: mắt, camera, ...)

**Regularization:** là thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó.

**Tập validation:** Trích một tập con nhỏ từ tập training data và thực hiện việc đánh giá mô hình trên tập con này (khi xây dựng mô hình ta không dùng test data), tập con này gọi là tập validation.

## *Mục Lục*

<b>CHƯƠNG 1 GIỚI THIỆU ĐỀ TÀI .....</b>	<b>1</b>
1.1    Tính cấp thiết của đề tài.....	1
1.2    Mục tiêu của luận văn .....	4
1.3    Đối tượng và phạm vi nghiên cứu.....	4
1.3.1    Đối tượng .....	4
1.3.2    Phạm vi nghiên cứu.....	5
1.4    Nhiệm vụ của luận văn .....	5
<b>CHƯƠNG 2 TỔNG QUAN NHẬN DẠNG HÀNH ĐỘNG .....</b>	<b>6</b>
2.1    Các nghiên cứu liên quan.....	6
2.1.1    Theo dõi khung xương .....	6
2.1.2    Mạng tích chập 2 luồng .....	7
2.1.3    Ưu nhược điểm của từng phương pháp.....	8
2.2    Sự phát triển của mạng tích chập.....	8
2.3    Nhận dạng ảnh.....	10
2.4    Nhận dạng hành động con người trong Video .....	11
2.5    Hệ thống nhận dạng đối tượng .....	14
2.6    Kết luận chương 2.....	16
<b>CHƯƠNG 3 GIỚI THIỆU DEEP LEARNING.....</b>	<b>17</b>
3.1    Học Máy.....	17
3.1.1    Học có giám sát (Supervised Learning) .....	19
3.1.2    Học không giám sát (Unsupervised Learning) .....	19

3.1.3	Hồi quy (Regression) .....	19
3.1.4	Phân loại (Classification) .....	21
<b>3.2</b>	<b>Mạng tích chập sâu (Deep Convolutional Network) .....</b>	<b>23</b>
3.2.1	Lớp kết nối đầy đủ (Fully Connected Layers).....	24
3.2.2	Lớp chập (Convolutional Layers) .....	25
3.2.3	Batch Normalization.....	29
3.2.4	Cơ chế tắt ngẫu nhiên (Dropout) .....	29
<b>3.3</b>	<b>Tối ưu hóa tham số (Parameter Optimization) .....</b>	<b>30</b>
3.3.1	Categorical Crossentropy.....	30
3.3.2	Gradient Descent .....	31
3.3.3	Stochastic Gradient Descent.....	32
3.3.4	Weight Decay .....	33
3.3.5	Học chuyển tiếp (Transfer Learning) .....	34
<b>3.4</b>	<b>Kết luận chương 3.....</b>	<b>35</b>
<b>CHƯƠNG 4 NHẬN DẠNG HÀNH ĐỘNG CON NGƯỜI.....</b>		<b>36</b>
<b>4.1</b>	<b>Mạng tích chập 2 luồng .....</b>	<b>36</b>
4.1.1	VGG16 – A Very Deep Convolutional Network.....	37
4.1.2	Luồng không gian .....	37
4.1.3	Luồng thời gian .....	37
4.1.4	Optical flow .....	38
<b>4.2</b>	<b>Nhận dạng đối tượng .....</b>	<b>39</b>
4.2.1	YOLO – You Only Look Once .....	39

4.2.2	SSD – Single Shot MultiBox Detector.....	44
4.2.3	Deep SORT .....	47
<b>4.3</b>	<b>Luồng ngữ nghĩa (Semantic Stream) .....</b>	<b>48</b>
4.3.1	Vị trí kết nối với YOLO .....	48
4.3.2	Vị trí kết nối với SSD .....	49
4.3.3	Kiến trúc của luồng ngữ nghĩa .....	49
4.3.4	Sự kết hợp các luồng .....	50
<b>4.4</b>	<b>Tăng cường phân kỳ (Boosting Divergence).....</b>	<b>50</b>
4.4.1	Average Layer.....	51
4.4.2	Kullback-Leibler Divergence.....	51
4.4.3	Dot Product Divergence .....	52
4.4.4	Combining Average with Divergence.....	52
<b>4.5</b>	<b>Actor conditioned actor maps (ACAM) .....</b>	<b>53</b>
4.5.1	Giới thiệu về phương pháp ACAM.....	53
4.5.2	Kiến trúc ACAM .....	54
<b>4.6</b>	<b>Mạng tích chập 3 chiều (3D CNN).....</b>	<b>54</b>
<b>4.7</b>	<b>Bộ dữ liệu AVA .....</b>	<b>58</b>
4.7.1	Định nghĩa về Atomic Action.....	59
4.7.2	Chuỗi thời gian cho từng người hành động .....	59
4.7.3	Chú thích cho bộ dữ liệu .....	60
4.7.4	Ghi nhãn hành động toàn diện .....	60
<b>4.8</b>	<b>Kết luận chương 4.....</b>	<b>61</b>

<b>CHƯƠNG 5 XÂY DỰNG MÔ HÌNH NHẬN DẠNG HÀNH ĐỘNG .....</b>	<b>63</b>
<b>5.1 Kiến trúc hệ thống.....</b>	<b>63</b>
<b>5.2 Huấn luyện mô hình.....</b>	<b>64</b>
5.2.1 Chuẩn bị dữ liệu huấn luyện .....	64
5.2.2 Chú thích cho dữ liệu .....	66
5.2.3 Cài đặt các thông số huấn luyện .....	68
5.2.4 Google Colab .....	68
5.2.5 Huấn luyện mô hình .....	69
<b>5.3 Chương trình nhận dạng hành động.....</b>	<b>72</b>
5.3.1 Đầu vào .....	72
5.3.2 Nhận dạng đối tượng .....	73
5.3.3 Deep SORT .....	73
5.3.4 Đầu ra.....	73
<b>5.4 Kết luận chương 5.....</b>	<b>74</b>
<b>CHƯƠNG 6 KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ MÔ HÌNH .....</b>	<b>75</b>
<b>6.1 Kết quả thực hiện .....</b>	<b>75</b>
<b>6.2 Đánh giá.....</b>	<b>78</b>
6.2.1 Ma trận nhầm lẫn (confusion matrix) .....	78
6.2.2 Đánh giá mô hình .....	79
<b>CHƯƠNG 7 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>81</b>
<b>7.1 Kết luận .....</b>	<b>81</b>

7.2 Hướng phát triển.....	82
<b>THAM KHẢO .....</b>	<b>83</b>

## ***Danh Lục Hình Ảnh***

<i>Hình 1.1 Mối liên hệ giữa AI, ML, DL.</i> .....	2
<i>Hình 1.2 Thống kê té ngã ở người cao tuổi</i> .....	3
<i>Hình 2.1 Mô hình khung xương con người dưới dạng 2D và 3D [1]</i> .....	7
<i>Hình 2.2 Mạng tích chập 2 luồng [1]</i> .....	8
<i>Hình 2.3 Kiến trúc mô hình AlexNet</i> .....	9
<i>Hình 2.4 Bộ dữ liệu MNIST cho nhận dạng chữ số viết tay</i> .....	10
<i>Hình 2.5 Ví dụ về cách tiếp cận của sổ trượt để nhận dạng đối tượng</i> .....	14
<i>Hình 2.6 Nhận dạng con người với YOLO</i> .....	15
<i>Hình 3.1 Vị trí, tư thế, ánh sáng, ..., làm cho việc lập trình thông thường để nhận dạng một “con mèo” được xem là bất khả thi</i> .....	18
<i>Hình 3.2 Ví dụ về hồi quy tuyến tính của một biến</i> .....	20
<i>Hình 3.3 Hàm sigmoid</i> .....	22
<i>Hình 3.4 Hàm kích hoạt cho lớp Fully Connected sử dụng trong luận văn</i> ..	24
<i>Hình 3.5 Các tích chập khác nhau, được trực quan hóa</i> .....	25
<i>Hình 3.6: Phần màu xám chính là phần padding thêm nào input</i> .....	28
<i>Hình 3.7 Minh họa cho overfitting</i> .....	30
<i>Hình 3.8 Hình ảnh thể hiện sự khác nhau của GD khi learning rate khác nhau.</i> .....	32
<i>Hình 3.9 So sánh tương quan hiệu quả của mô hình train từ đầu và transferred model</i> .....	35
<i>Hình 4.1 Minh họa đầu ra của YOLOv2</i> .....	43

Hình 4.2 Mô hình SSD VGG19 model 1 ..... 44

Hình 4.3 Mô hình SSD VGG19 ..... 45

Hình 4.4 Luồng ngữ nghĩa ..... 48

Hình 4.5 Kiến trúc ‘three stream’ ..... 50

Hình 4.6 Attention module cho actor  $a$  tại một chỉ số không-thời gian duy nhất  $t, h, w$ . Trọng số chú ý duy nhất trong ACAM tại  $t, h, w$  được tạo từ đặc trưng actor  $\mathbf{r}_a$  và đặc trưng bối cảnh tại cùng chỉ số  $\mathbf{E}_{t,h,w}$  [12] ..... 53

Hình 4.7 Kiến trúc của ACAM. [12] ..... 54

Hình 4.8 So sánh giữa 2D convolution và 3D convolution [13] ..... 56

Hình 4.9 Mô hình cấu trúc mạng Inflated Inception-V1 (bên trái) và chi tiết của module Inception (bên phải) [14] ..... 57

Hình 4.10 Bộ dữ liệu AVA của Google ..... 58

Hình 5.1 Kiến trúc model nhận dạng hành động ..... 63

Hình 5.2 Tải xuống bộ dữ liệu AVA ..... 65

Hình 5.3 Bộ dữ liệu AVA sau khi tải về ..... 65

Hình 5.4 Quá trình cắt nhỏ video ..... 66

Hình 5.5 Video sau khi được xử lý ..... 66

Hình 5.6 File csv cho tập train sau khi được chúng tôi chỉnh sửa lại ..... 67

Hình 5.7 Tesla K80 GPU của Google Colab. Một GPU mạnh mẽ được Google cho sử dụng miễn phí ..... 69

Hình 5.8 Giao diện notebook của Google Colab ..... 70

Hình 5.9 Cài đặt để được sử dụng GPU của Google Colab ..... 70

Hình 5.10 Kết nối Google Drive và Google Colab ..... 70

<i>Hình 5.11 Chạy chương trình huấn luyện .....</i>	71
<i>Hình 5.12 Quá trình huấn luyện bằng Google Colab .....</i>	72
<i>Hình 5.13 Nguyên lý hoạt động chương trình chính.....</i>	72
<i>Hình 5.14 Đầu ra của chương trình, kết quả nhận dạng hành động được hiển thị đồng thời trên Terminal và màn hình máy tính.....</i>	74
<i>Hình 6.1 Kết quả nhận dạng hành động nhiều đối tượng cùng lúc.....</i>	75
<i>Hình 6.2 Kết quả nhận dạng hành động nhiều hành động khác nhau ở các góc máy khác nhau .....</i>	76
<i>Hình 6.3 Kết quả nhận dạng hành động ở điều kiện ánh sáng khác nhau... </i>	77
<i>Hình 6.4 Kết quả nhận dạng hành động ở điều kiện ánh sáng khác nhau... </i>	77
<i>Hình 6.5 Minh họa ma trận nhầm lẫn chưa chuẩn hóa (bên trái) và đã chuẩn hóa (bên phải) .....</i>	78
<i>Hình 6.6 Ma trận nhầm lẫn của hệ thống .....</i>	79

## *Danh Mục Bảng Biểu*

Bảng 4.1 Kiến trúc VGG16 .....	38
Bảng 4.2 Kiến trúc của YOLOv2.....	40
Bảng 4.3 Kiến trúc của YOLO9000 .....	42
Bảng 4.4 Kiến trúc chính của SSD .....	46

# CHƯƠNG 1

## GIỚI THIỆU ĐỀ TÀI

Chương 1 nhằm mục đích giới thiệu đề tài của luận văn. Qua chương này, chúng tôi muốn giới thiệu tính cấp thiết đề tài, mục tiêu, nhiệm vụ và đối tượng nghiên cứu của luận văn lần này.

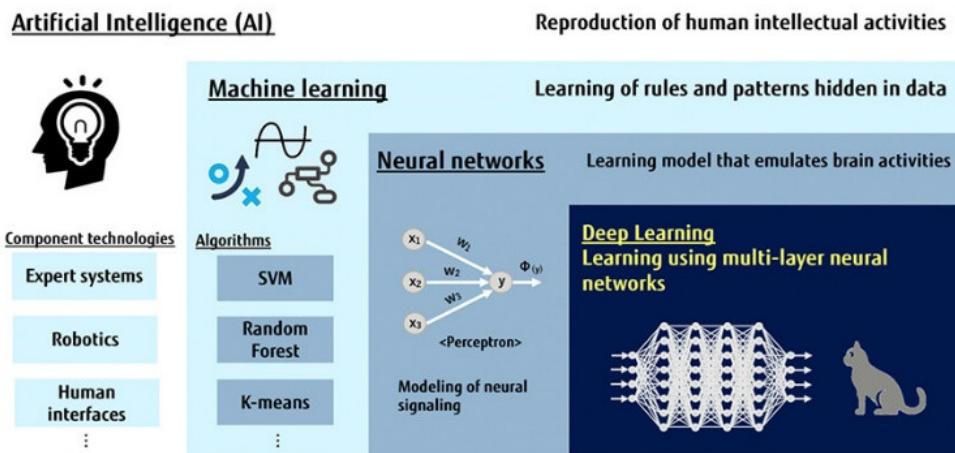
### 1.1 Tính cấp thiết của đề tài

Trí tuệ nhân tạo, một thuật ngữ vừa gần gũi vừa xa lạ, đã và đang len lỏi vào nhiều lĩnh vực trong đời sống hàng ngày của chúng ta. Mỗi ngày chúng ta được nghe/nhìn thấy các cụm từ như AI, Machine Learning hay Deep Learning ngày một nhiều hơn. Việc các cụm từ này ngày càng phổ biến chứng tỏ cho sự phát triển ngày càng nhanh của AI. Nguyên nhân là do sự xuất hiện của cuộc cách mạng công nghiệp lần thứ 4 đang diễn ra với phạm vi toàn cầu mà AI là một trong 3 động lực chính (AI, Robot & BigData, IOT). AI, ML và DL giờ đây đã trở thành xu hướng mới trên toàn thế giới. Mỗi quan hệ giữa AI, ML và DL được minh họa trong Hình 1.1. Trong đó, ML là một công nghệ thành phần của AI, Neural Network là một thuật toán của ML, còn DL một cấu trúc nhiều lớp của các Neural Networks.

Cùng với sự phát triển vũ bão của AI, thị giác máy tính đã có những bước phát triển mới đáng kinh ngạc. Trước đây, người ta cho rằng thị giác máy tính không thể cạnh tranh với thị giác của một em bé 1 tuổi. Điều đó đã không còn đúng, máy tính giờ đây có thể nhận dạng hầu hết các vật thể bằng hình ảnh giống hệt đa số người trưởng thành, các hệ thống máy tính được trang bị trên các xe ô tô tự hành có thể lái an toàn hơn cả một thanh niên 16 tuổi, các ứng dụng nhận dạng gương mặt, nhận dạng đối tượng có độ chính

## CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

xác gần đạt 100%, hệ thống nhận dạng ung thư da được phát triển tại Stanford có độ chính xác lên đến 96%<sup>1</sup>. Điều này dẫn đến sự tăng mạnh nhu cầu học tập, nghiên cứu AI nói chung, hay thị giác máy tính nói riêng, trong những năm gần đây trên toàn thế giới cũng như Việt Nam.



Hình 1.1 Mối liên hệ giữa AI, ML, DL.  
(Nguồn: <https://journal.jp.fujitsu.com/en/2018/11/29/01/>)

Ngoài ra, nhu cầu nhận dạng hành động cho các mục đích khác nhau đang trở nên cấp thiết hơn. Chẳng hạn như tỷ lệ bạo lực học đường đang ngày càng trở nên nghiêm trọng hơn, số liệu được Bộ Giáo dục và đào tạo (GD-ĐT) đưa ra gần đây nhất, trong một năm học, toàn quốc xảy ra gần 1.600 vụ việc học sinh đánh nhau ở trong và ngoài trường học (khoảng 5 vụ/ngày). Cũng theo thống kê của Bộ GD-ĐT, cứ khoảng trên 5.200 học sinh (HS) thì có một vụ đánh nhau; cứ hơn 11.000 HS thì có một em bị buộc thôi học vì đánh nhau; cứ 9 trường thì có một trường có học sinh đánh nhau<sup>2</sup>. Các thống kê cho thấy nhu cầu cần phải có các thiết bị giám sát ở trường học để phát hiện các vụ việc đánh nhau, từ đó kịp thời ngăn chặn. Một ví dụ khác là tình trạng té ngã ở người cao tuổi. Té ngã là một hiện tượng phổ biến ở người cao tuổi, nhưng lại gây ra những hậu quả hết sức nghiêm trọng. Tỷ lệ chấn thương 10

<sup>1</sup> <https://khoaahoc.tv/thuat-toan-ai-co-kha-nang-chan-doan-duoc-ung-thu-da-dung-den-96-78011>

<sup>2</sup> <https://baohatinh.vn/giao-duc/moi-ngay-co-5-vu-hoc-sinh-danh-nhau-trong-va-ngoai-truong-hoc/171362.htm>

– 25% tổng số ca ngã, 8% người có tuổi phải vào cấp cứu vì té ngã, là nguyên nhân thứ 2 gây tổn thương não và tỷ sống<sup>3</sup>. Việc phải có những thiết bị giám sát để kịp thời phát hiện, cứu giúp những người bị té ngã là rất cần thiết.



*Hình 1.2 Thống kê té ngã ở người cao tuổi  
(Nguồn: <https://goldage.com.au/preventing-falls-elderly/>)*

Để bắt kịp nhu cầu của xã hội và vì sự yêu thích nghiên cứu các vấn đề AI, thị giác máy tính, chúng tôi quyết định thực hiện luận văn **Nhận dạng hành động của con người**.

**Nhận dạng hành động con người** là một lĩnh vực được nghiên cứu rộng rãi trong thị giác máy tính. Các ứng dụng của nó bao gồm các hệ thống giám sát, phân tích video, robot,... và đã được nghiên cứu từ những năm 80. Hiện nay, các nghiên cứu chủ yếu tập trung vào nhận biết hành động từ chuỗi video. Có thể lập luận rằng một nhận thức thời gian là điều cần thiết để nhận ra hành động của con người. Ví dụ, một hình ảnh tĩnh của một người sắp ngồi sẽ không thể phân biệt được với hình ảnh tĩnh của một người đứng lên từ vị trí đang ngồi. Nhưng nếu chúng ta có thể thấy chuyển động của người đó trong nhiều frames liên tiếp, có lẽ chúng ta sẽ có thể suy luận nếu người đó ngồi xuống hoặc đứng lên. Mặc khác, một nhận thức không gian cũng có thể rất quan trọng, vì một số hành động thường được thực hiện trong môi trường nhất định. Ví dụ: một hành động có thể liên quan đến nấu ăn hoặc ăn nếu

---

<sup>3</sup> <http://benhvientinh.quangtri.gov.vn/vi/scientific-research/Nguy-co-te-nga-o-benh-nhan-cao-tuoi-dang-dieu-tri-tai-benh-vien-da-khoa-tinh-Quang-Tri.html>

chúng ta biết hành động được thực hiện trong nhà bếp. Công nghệ hiện nay để nhận dạng hành động con người là kEEP hợp thông tin không gian và thông tin thời gian của video. Các mô hình này bao gồm một luồng tin cho không gian và một luồng tin cho thời gian, do đó được gọi là mô hình hai luồng (two-stream). Bên cạnh đó, các hệ thống nhận dạng đối tượng ngày nay rất hiệu quả. Một số hệ thống nhanh nhất không chậm hơn nhiều so với các hệ thống nhận dạng hình ảnh. Vì vậy điều này cho phép chúng tôi sử dụng nhận dạng đối tượng vào hệ thống của chúng tôi. Tuy nhiên, việc chạy 2 module song song sẽ gặp nhiều khó khăn, và cần phải tìm hiểu rất nhiều. Chúng tôi sẽ cố gắng giải quyết các vấn đề đó trong luận văn này.

### 1.2 Mục tiêu của luận văn

Luận văn này tập trung vào mục tiêu xây dựng một hệ thống nhận dạng hành động con người. Gồm các phần sau:

- Tìm hiểu tổng quan về nhận dạng hành động
- Tìm hiểu tổng quan về Machine Learning, Deep Learning, mạng tích chập sâu.
- Tìm hiểu hệ thống nhận diện đối tượng và nhận dạng hành động áp dụng vào đề tài.
- Xây dựng được một mô hình nhận dạng hành động dựa trên bộ dữ liệu AVA.
- Đánh giá hệ thống, các phương pháp cải tiến, hướng phát triển của đề tài.

### 1.3 Đối tượng và phạm vi nghiên cứu

#### 1.3.1 Đối tượng

Luận văn nghiên cứu cấu trúc và hoạt động của CNN, ứng dụng CNN vào bài toán nhận dạng hành động con người.

### 1.3.2 Phạm vi nghiên cứu

Luận văn nghiên cứu, thiết kế một hệ thống nhận dạng hành động của con người bằng camera và được xử lý bằng máy tính. Luận văn tập trung nhận dạng 10 hành động là: Đứng, Ngồi, Nói chuyện, Quỳ, Đi bộ, Nằm, Cầm/nắm, Trả lời điện thoại, Đánh nhau, Té ngã.

## 1.4 Nhiệm vụ của luận văn

Các công việc cần thực hiện trong luận văn là:

- Tìm hiểu về Deep Learning, các thuật toán liên quan đến nhận dạng hành động.
- Thiết kế một hệ thống nhận dạng hành động con người.
- Đánh giá mô hình, ưu nhược điểm của mô hình.
- Đưa ra các phương pháp cải tiến và kết luận.

# CHƯƠNG 2

## TỔNG QUAN NHẬN DẠNG HÀNH ĐỘNG

Chương 2 cho một cái nhìn tổng quan về đề tài. Mục 2.1 cho biết các hướng nghiên cứu được thực hiện trong lĩnh vực nhận dạng hành động con người. Mục 2.2 nói về sự phát triển của CNN trở thành một thuật toán phổ biến cho việc giải quyết các bài toán phân loại. Các mục 2.3 2.4 2.5 nói về các bài toán nhận dạng và phân loại ảnh, nhận dạng đối tượng, nhận dạng hành động, và các bộ dữ liệu đã được phát triển cho việc nhận dạng hành động.

### 2.1 Các nghiên cứu liên quan

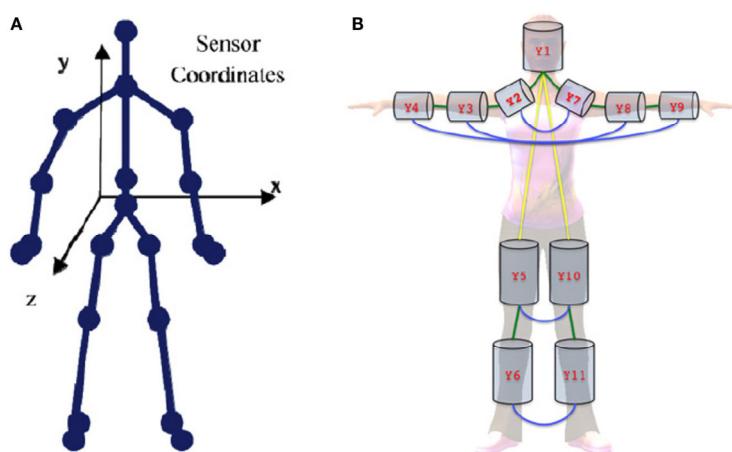
Nghiên cứu được thực hiện trong lĩnh vực nhận dạng hành động con người cho đến nay có thể được phân thành 2 cách tiếp cận là theo dõi (tracking) khung xương và sử dụng CNN 2 luồng [1]. Mỗi phương pháp đều có những ưu nhược điểm riêng, tuy nhiên, vì mục đích của chúng tôi là cố gắng xây dựng chương trình sử dụng hình ảnh từ các camera giám sát nên chúng tôi sẽ sử dụng phương pháp CNN 2 luồng.

#### 2.1.1 Theo dõi khung xương

Theo dõi khung xương con người đã nhận được rất nhiều phản hồi tốt từ các nhà nghiên cứu trong những thập kỷ vừa qua. Các bộ phận của con người được mô tả trong không gian 2D dưới dạng các điểm hình chữ nhật và dưới dạng hình thê tích (xem hình 2.1). Người ta biết rằng các thuật toán nhận dạng hoạt động dựa trên hình bóng con người đóng vai trò rất quan trọng

trong việc nhận ra hành động con người. Một hình bóng con người gồm các chi tiết liên kết với nhau, và điều quan trọng là phải có được các bộ phận cơ thể con người chính xác từ video. Vấn đề này được coi là một phần của quá trình nhận dạng hành động.

Trong phương pháp này, một luồng video được truyền qua một thuật toán tracking khung xương. Sau đó, dựa trên sự di chuyển giữa các khớp xương được lựa chọn và vận tốc góc tương ứng của chúng, nhận dạng hành động có thể được thực hiện.

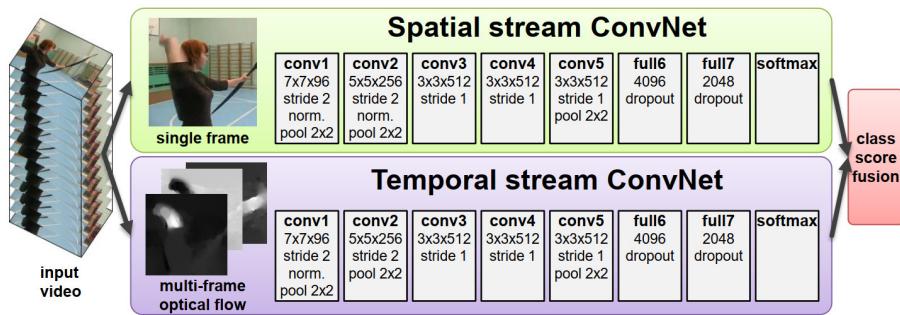


*Hình 2.1 Mô hình khung xương con người dưới dạng 2D và 3D [1]*

### 2.1.2 Mạng tích chập 2 luồng

Video có thể được phân tích thành các thành phần không gian và thời gian. Ở thành phần không gian, được phân tích trong một frame đơn lẻ mang thông tin về cảnh và vật thể được mô tả trong video. Phần thời gian, được phân tích dưới dạng chuyển động của các frames, chuyển tải chuyển động của đối tượng được quan sát từ camera.

Two-stream CNNs được huấn luyện chủ yếu trên mật độ đa khung (multiframe dense) optical flow. Trong đó luồng thời gian và không gian xử lý chuyển động dưới dạng mật độ optical flow và video frames tương ứng. Vấn đề này sẽ được chúng tôi làm rõ ở mục 4.1.



Hình 2.2 Mạng tích chập 2 luồng [1]

### 2.1.3 Ưu nhược điểm của từng phương pháp

Cả hai hướng nghiên cứu được chúng tôi nêu ra đã được sử dụng rộng rãi cho nhiệm vụ nhận dạng hành động. Mỗi cách tiếp cận này đều có nhược điểm riêng.

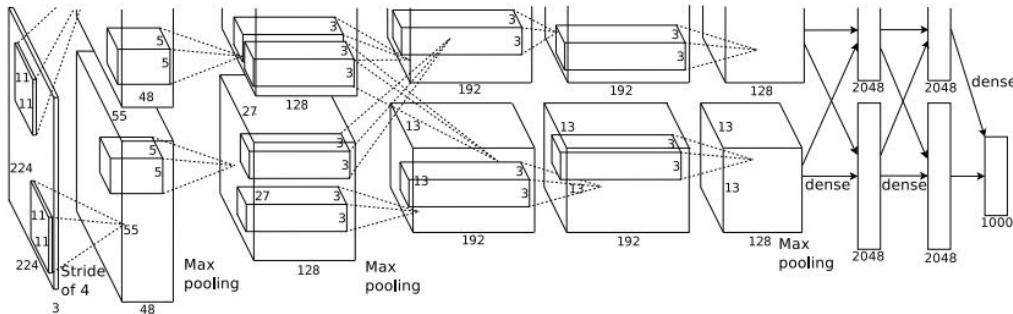
- Two-stream CNNs: Với việc thiếu các mô hình cơ thể người rõ ràng, việc nhận dạng hành động dựa trên video xử lý kém với sự lộn xộn nền và các chuyển động không có hành động.
- Theo dõi khung xương: Vì chỉ dựa vào việc theo dõi các tư thế cơ thể, các ngữ cảnh và tín hiệu chung trong video sẽ không được sử dụng. Ngoài ra, các góc khác nhau của camera sẽ gây ra khó khăn cho việc xây dựng và sử dụng theo dõi khung xương.

Vì chúng tôi muốn xây dựng một hệ thống có thể áp dụng được hình ảnh từ các camera giám sát, nên với đề tài lần này, chúng tôi sẽ sử dụng phương pháp two-stream CNN để giải quyết bài toán nhận dạng hành động.

## 2.2 Sự phát triển của mạng tích chập

Như đã nêu ở chương 1, thị giác máy tính đã thay đổi nhanh chóng trong vài năm qua. Nguyên nhân chính của sự thay đổi này là sự phát triển của AI hay cụ thể hơn là sự phát triển của CNN. Sự phổ biến của CNN bắt đầu khi nó được chứng minh là có hiệu quả trên các điểm chuẩn (benchmark) nhận dạng hình ảnh lớn như ImageNet. Kể từ thời điểm đó, CNNs đã được sử dụng để

giành chiến thắng trong nhiều cuộc thi và nổi tiếng nhất là chiến thắng của Alex Krizhevsky trong cuộc thi ImageNet LSVRC-2012 vào năm 2012. Kiến trúc mạng CNN được sử dụng vào năm 2012 được gọi là kiến trúc AlexNet.



*Hình 2.3 Kiến trúc mô hình AlexNet  
(nguồn: <https://en.wikipedia.org/wiki/AlexNet>)*

Ngoài việc CNN tạo ra bước đột phá với nhận dạng hình ảnh, các nhà nghiên cứu đã phát hiện CNN cũng rất hữu ích cho các ứng dụng thị giác máy tính khác. Ví dụ: CNNs đã được sử dụng để chú thích cho hình ảnh và video, tạo âm thanh cho video không tiếng và thậm chí tạo ra ảnh màu từ ảnh xám. CNN còn được sử dụng bên ngoài thị giác máy tính, chẳng hạn như học chơi cờ vây. Các khả năng của CNN dường như là vô tận.

CNN không phải là một phát minh mới. Nó đã được sử dụng rất sớm ngay từ những năm 1990 cho các vấn đề về nhận dạng hình ảnh. Lý do tại sao CNN trở nên phổ biến trong 10 năm gần đây là do sự ra đời của các bộ dữ liệu khổng lồ có gán nhãn được sử dụng miễn phí như ImageNet, cho phép việc huấn luyện DCNs. Ngoài ra, một lý do khác của sự phổ biến là việc sử dụng card đồ họa (GPU) để huấn luyện mô hình, GPU được tạo ra chủ yếu dành cho game thủ, với khả năng chạy song song nhiều lõi, đã trở thành một công cụ cực kỳ thích hợp với CNN, giúp tăng tốc độ thuật toán lên gấp nhiều lần so với CPU. [2]

## 2.3 Nhận dạng ảnh

Phân loại, hay nhận dạng ảnh, là việc gán nhãn cho ảnh theo nội dung của nó. Một ví dụ điển hình của bài toán nhận dạng hình ảnh đơn giản là nhận dạng chữ số trong tập dữ liệu các chữ số viết tay MNIST; quyết định chữ số nào được hiển thị trong một ảnh. Các bài toán nhận dạng ảnh phức tạp khác, chẳng hạn như nhiệm vụ phân loại ImageNET ILSVRC-2012, yêu cầu một hệ thống nhận dạng ảnh phải gán nhãn chính xác cho 50.000 ảnh thành 1.000 thể loại. Chúng ta biết rằng các giá trị pixel thô của ảnh thường rất cao. VD: không gian ảnh của một ảnh RGB vuông có kích thước 256 là  $256 \times 256 \times 3 = 196.608$  dim. Tuy nhiên, có nhiều bậc tự do (degrees of freedom) cho các đặc trưng ảnh có lẽ là không cần thiết cho một hệ thống phân loại ảnh. Một mẫu ngẫu nhiên của một không gian ảnh có khả năng chỉ là một nhiễu ngẫu nhiên mà không chứa thông tin giá trị.

*Hình 2.4 Bộ dữ liệu MNIST cho nhận dạng chữ số viết tay  
(Nguồn: [https://www.wikiwand.com/en/MNIST\\_database](https://www.wikiwand.com/en/MNIST_database))*

Do đó, một phương pháp phổ biến là chiểu các đặc trưng của hình ảnh vào các không gian chiều thấp hơn để nhận dạng hình ảnh. Không gian chiều thấp hơn là ý tưởng tốt để miêu tả các đặc trưng của ảnh và chứa càng nhiều thông tin có giá trị càng tốt. Các phương thức nhận dạng khuôn mặt truyền thống như Eigenfaces, chiểu đầu vào vào một không gian tuyến tính với Principal Component Analysis (phân tích thành phần chính).

Một phương pháp khác khả thi hơn trong việc phân loại ảnh là thay đổi mô tả của ảnh. Bộ mô tả RGB diễn hình rất tốt để thể hiện màu sắc của hình ảnh, nhưng không cần thiết cho việc miêu tả hình dạng hoặc cấu trúc của ảnh. Histogram of Oriented Gradients (HOG) là một mô tả thủ công, nó tốt hơn không trong việc đại diện cho tính chất cấu trúc của một hình ảnh. HOG đã được sử dụng cho nhiệm vụ nhận dạng hình ảnh truyền thống. Bộ mô tả HOG ánh xạ hình ảnh vào một không gian HOG nơi hiển thị độ dốc cục bộ của hình ảnh.

Nhưng CNN đã làm cho cả việc miêu tả và mô tả thủ công như HOG trở nên lỗi thời để nhận dạng hình ảnh. Thay vì sử dụng các miêu tả thủ công, các mạng CNN được sử dụng để học việc miêu tả trực tiếp từ các đặc trưng RGB của hình ảnh. Có vẻ như CNN đủ linh hoạt để tự học các miêu tả. Thậm chí người ta đã chứng minh rằng các tính năng thủ công như HOG có thể được hiểu là một phần của CNN.

### 2.4 Nhận dạng hành động con người trong Video

Nhận dạng hành động con người trong video là việc nhận ra những loại hành động mà con người thực hiện trong một video. Các hành động có thể nhận dạng đơn giản như vẫy tay, đi bộ, nhảy hoặc những hành động nâng cao hơn như chơi thể thao, nghe điện thoại, đánh nhau,... Ngay cả khi nhận dạng hành động con người có vẻ giống với nhận dạng hình ảnh, CNN cho đến nay vẫn không được hưởng lợi cho nhận dạng hành động con người đối với việc thể hiện video thủ công dưới dạng sub-stantially.

3D CNNs đã được sử dụng để kết hợp các tính năng không gian và thời gian của đầu vào trong cùng một mô hình để nhận dạng hành động của con người. Tuy nhiên, 3D CNNs chỉ dẫn đến một thành công nhỏ cho bài toán nhận dạng hành động. Một lý do cho sự thành công không lớn của 3D CNNs là do các bộ dữ liệu có sẵn để nhận dạng hành động của con người hiện quá nhỏ để có thể học chính xác 3D CNNs.

Một cách tiếp cận khác bằng việc kết hợp các dự đoán của 2D CNNs từ một chuỗi các frames cũng đã được thử. Trong những cách tiếp cận thành công nhất của việc sử dụng CNNs cho nhận dạng hành động con người là chia các đặc điểm không gian và thời gian thành hai mạng riêng biệt, thường được gọi là mạng hai luồng (two-stream networks). Các mạng hai luồng học riêng các đặc trưng của RGB frames và các Optical flow của video. Điều này được lấy cảm hứng thông qua việc quan sát cách não bộ nhận dạng hành động. Thông thường, các RGB frames và Optical flow của video cung cấp các dự đoán tổng hợp để nhận dạng hành động con người.

Mạng neuron hồi quy, thường được sử dụng cho dữ liệu tuần tự, cũng được xem xét để nhận dạng hành động. LSTM là một loại mạng hồi quy cụ thể được chứng minh là có khả năng học tập quy mô lớn về nhận dạng giọng nói và các vấn đề xử lý ngôn ngữ tự nhiên khác. Các tiếp cận two-stream ban đầu dự đoán trên cơ sở single-frame. Ở dạng tiêu chuẩn, two-stream không thể mô hình hóa các cấu trúc thời gian và có một số nỗ lực để giải quyết vấn đề này. LSTM đã được sử dụng để mở rộng cách tiếp cận two-stream để chấp nhận dữ liệu tuần tự, với sự cải thiện quan sát so với đường cơ sở single-frame.

Một cách khác để giải quyết việc mô hình hóa các cấu trúc thời gian với các mạng hai luồng bằng cách lấy sự đồng thuận giữa các đoạn được lấy mẫu của clip. Đối với mỗi đoạn mã, một hành động được dự đoán bằng cách sử dụng two-streams. Các dự đoán sau đó được kết hợp bằng cách có được sự đồng thuận giữa các dự đoán. Cách tiếp cận trên đạt được trên UCF-101 với 94.2%.

Gần đây, với việc Google đã cho ra đời bộ dữ liệu nhận dạng hành động AVA, việc nhận dạng hành động con người đã có những bước tiến mới. Bộ dữ liệu AVA chú thích các hành động một cách chi tiết và không gian của tất cả các diễn viên trong các cảnh phức tạp. Các phương pháp ban đầu trên bộ dữ liệu AVA đã mở rộng kiến trúc Faster-RCNN thành các cấu trúc 3D, trong đó các lớp ban đầu tạo các đặc trưng cho các actors và mỗi đặc trưng được phân

tích bởi các lớp tiếp theo. Chúng tôi giới thiệu một phương pháp mới là phương pháp ACAM, bản chất của ACAM là điều kiện các đặc trưng được trích xuất từ toàn bộ bối cảnh trên mỗi actors và động lực của hành động. Phương pháp này được chúng tôi sử dụng trong luận văn này và sẽ được nêu chi tiết hơn ở mục 4.6.

### Các bộ dữ liệu cho nhận dạng hành động con người

Vì việc nhận dạng hành động con người trong video đã phát triển, nhu cầu về các bộ dữ liệu phức tạp hơn cũng tăng lên. Các bộ dữ liệu ban đầu, như KTH và Weizmann đã nhỏ và trong môi trường rất hạn chế. Bộ dữ liệu KTH chứa 6 loại hành động khác nhau được thực hiện bởi 25 đối tượng trong 4 môi trường khác nhau. Máy ảnh tĩnh và được quay phim để có thể nhìn thấy toàn bộ cơ thể của các đối tượng. Bối cảnh của các video cũng không có bất kỳ sự phân chia nào. Bộ dữ liệu Weizmann rất giống với bộ dữ liệu KTH, nhưng ít bị ràng buộc hơn bằng cách để các đối tượng ở các môi trường phức tạp hơn. Tuy nhiên, các hành động vẫn rất đơn giản và hạn chế.

UCF-Sports là bộ dữ liệu với các hành động thể thao được thu thập từ các chương trình phát sóng trên TV khác nhau. Bộ dữ liệu này đa dạng hơn các bộ dữ liệu trước do ở trong một cài đặt ít ràng buộc hơn với các góc máy ảnh, ánh sáng, hình nền khác nhau. Tương tự, bộ dữ liệu Hollywood thu thập các hành động khác nhau của con người từ các bộ phim Hollywood. Bộ dữ liệu này cũng bao gồm các bức ảnh trong clip.

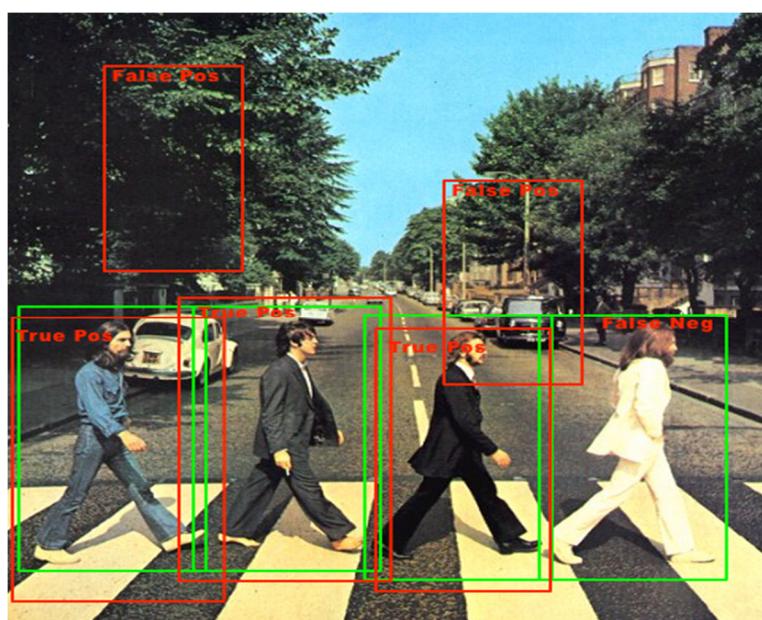
Trong khi nhiều bộ dữ liệu như HMDB-51, Kinetics, UCF-101 rất hữu ích cho việc tìm kiếm video và phân loại chúng. Thì gần đây **bộ dữ liệu AVA** (sẽ được chúng tôi đề cập chi tiết hơn ở mục 4.7) và JHMDB tập trung vào các atomic actions với các đoạn video ngắn. Atomic actions có khả năng tổng quát các bối cảnh khác nhau, trở thành building blocks cho nhiều hành động phức tạp hơn và cải thiện sự hiểu biết chung về các hành động của con người trong các video. Bộ dữ liệu AVA có thể phân lớp tới 80 hành động khác nhau,

đây là bộ dữ liệu mới và rất mạnh cho việc phát triển các mô hình nhận dạng hành động.

## 2.5 Hệ thống nhận dạng đối tượng

Trong khi nhận dạng hành động con người đã không đạt được bước nhảy vọt tương tự như nhận dạng hình ảnh, các hệ thống nhận dạng đối tượng đã được hỗ trợ rất tốt bởi CNNs.

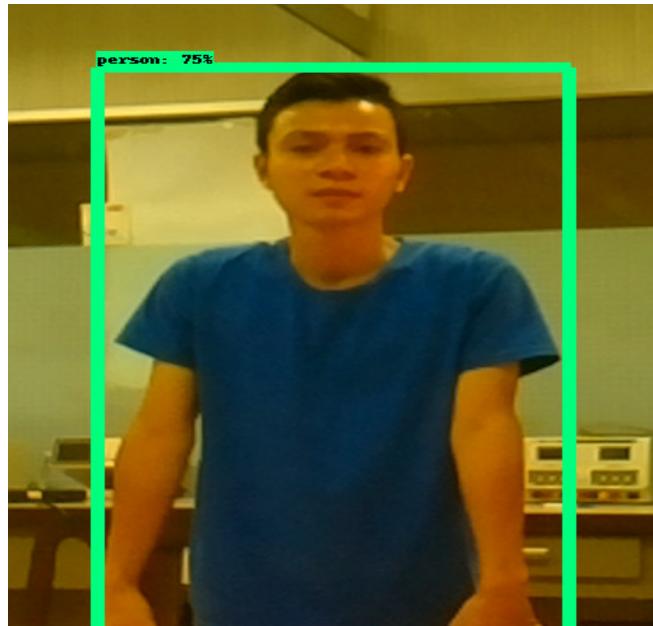
Các hệ thống nhận dạng đối tượng truyền thống, như tập hình các mô hình mẫu hoặc mô hình bộ phận biến dạng (Deformable parts models), đã sử dụng cửa sổ trượt (sliding windows) để sử dụng cho việc nhận dạng đối tượng. Các phương pháp này tìm thấy các đối tượng trong hình ảnh bằng cách chạy trình phân loại tại các vị trí cách đều nhau trên toàn bộ hình ảnh. Chỉ có một trình phân loại cần được học cho cách tiếp cận cửa sổ trượt. Tuy nhiên, sử dụng cửa sổ trượt rất tốn kém tài nguyên vì một bộ phân loại phải được sử dụng nhiều lần cho mỗi hình ảnh, điều này khiến cho việc nhận dạng thời gian thực rất khó khăn với phương pháp này.



*Hình 2.5 Ví dụ về cách tiếp cận cửa sổ trượt để nhận dạng đối tượng.  
(Nguồn: <https://intellipaat.com/community/22135/how-to-categorize-true-negatives-in-sliding-window-object-detection>)*

Các cách tiếp cận gần đây, như Faster R-CNN, tránh sử dụng cửa sổ trượt mà sử dụng CNNs để đề xuất các “khu vực” của đối tượng. Sau khi đề xuất các khu vực mà loại trừ các bản sao có thể có, một bộ phân loại được chạy qua từ đề xuất khu vực để có được các class score. Nhược điểm của các phương pháp như Faster R-CNN là cần sử dụng hai mạng để học. Một mạng cho việc đề xuất khu vực và một mạng khác để phân loại các khu vực. Faster R-CNN nhanh hơn cũng chậm hơn thời gian thực ngay cả trên các GPU mạnh mẽ như Geforce GTX Titan X.

Mặt khác, hệ thống nhận dạng đối tượng You Only Look Once (YOLO), đã chứng minh rằng có thể thống nhất đề xuất khu vực và dự đoán class scores trong một CNN duy nhất. Mạng đơn phương này tối ưu hóa đơn giản hơn nhiều so với Faster R-CNN do nhận dạng đối tượng có thể được đóng khung như một vấn đề hồi quy. Faster R-CNN yêu cầu nhiều thành phần trong một đường ống phức tạp được đào tạo riêng. Kiến trúc mạng CNN cũng cho phép YOLO đạt được hiệu suất thời gian thực.



*Hình 2.6 Nhận dạng con người với YOLO*

YOLO đã được cải thiện từ phiên bản đầu tiên khi giới thiệu phiên bản thứ 2, YOLOv2. YOLOv2 tương tự như phiên bản gốc, nhưng một số điều chỉnh

trong mô hình làm nó chính xác hơn và chạy nhanh hơn. **Single Shot Multibox Detector** (SSD) là một mạng khác tiếp nối thành công của YOLO bằng cách thực hiện phát hiện đối tượng trong một CNN duy nhất. Mô hình YOLO đã được cải tiến một lần nữa bởi Joseph Remodmon và Ali Farhadi trong bài báo năm 2018 với tiêu đề “YOLOv3: An Incremental Improvement”. Những cải tiến này khá nhỏ, chủ yếu thay đổi mô hình DCN trong trích xuất đặc trưng. Chúng tôi sẽ đi vào chi tiết 2 mô hình YOLO và SSD ở chương 4.

### 2.6 Kết luận chương 2

Chương 2 đã giới thiệu các nghiên cứu liên quan đến đề tài, ưu nhược điểm của chúng và lý do chúng tôi lựa chọn phương pháp two-stream CNNs vào luận văn lần này. Ở các mục tiếp theo, chúng tôi trình bày các nội dung liên quan đến sự phát triển của CNN và lý do CNN phát triển trong những năm gần đây. Sau đó chúng tôi giới thiệu tổng quát nhận dạng ảnh, nhận dạng hành động, các hệ thống nhận dạng đối tượng. Từ đó có được cái nhìn chung về đề tài nhận dạng hành động để tiếp tục đi sâu vào chi tiết trong các chương sau.

# CHƯƠNG 3

## GIỚI THIỆU DEEP LEARNING

Chương này giới thiệu nhanh về kỹ thuật DL được sử dụng trong luận văn này. Phần 3.1 thảo luận về các kỹ thuật ML quan trọng đối với đề tài. Để nhận dạng hành động người, chúng tôi muốn chương trình hiểu cách phân biệt các hành động khác nhau trong video, thường được thực hiện với ML. Phần 3.2, tiếp tục bằng cách mô tả các lớp phổ biến được sử dụng cho các mạng tích chập sâu. Cuối cùng, phần 3.3 thảo luận về cách các tham số của mạng DCNs có thể tối ưu hóa trong việc sắp xếp phân loại.

### 3.1 Học Máy

Học máy hay máy học (từ giờ sẽ được gọi là ML) là tên gọi chung cho các kỹ thuật học từ dữ liệu. Mục đích của ML là giải quyết các vấn đề rất khó để giải quyết trực tiếp bằng cách lập trình thông thường. Ví dụ, một bài toán yêu cầu lập trình bằng tay một hệ thống có nhiệm vụ nhận ra “con mèo” trong ảnh. Chúng ta thấy rằng yêu cầu này dường như không thể thực hiện, một “con mèo” có thể được nhìn theo nhiều cách khác nhau, với nhiều màu sắc và hình dạng lông khác nhau. Ngoài ra, chúng ta cần xem xét các điều kiện ánh sáng khác nhau và các hướng được nhìn khác nhau của con mèo. Bên cạnh đó, con mèo cũng có thể ở những tư thế khác nhau. Điều này làm cho việc giải bài toán có vẻ là bất khả thi nếu chỉ dùng những phương pháp lập trình thông thường.



*Hình 3.1 Vị trí, tư thế, ánh sáng,..., làm cho việc lập trình thông thường để nhận dạng một “con mèo” được xem là bất khả thi.  
(Ảnh trên Internet)*

Nhưng chúng ta vẫn có thể giải quyết bài toán bằng một phương pháp khác, được gọi là ML. Cách ML giải quyết vấn đề này là để cho chương trình học cách làm thế nào để nhận biết một “con mèo” từ dữ liệu có sẵn. Để làm được điều này, chúng ta cần chuẩn bị một số dữ liệu bằng hình ảnh, với các bức ảnh có “con mèo” và các bức ảnh không có “con mèo”, cho biết đầu ra mong muốn của ảnh tương ứng. Chương trình sau đó sẽ cố gắng tìm ảnh xạ tốt nhất có thể giữa đầu vào và đầu ra. Mục tiêu của chương trình là có thể nhận ra các “con mèo” có trong một bức ảnh, mà ảnh đó không có trong các ví dụ.

Nói một cách tổng quát hơn, ML bao gồm các vấn đề, mà chúng ta muốn học ánh xạ  $f$  “tốt nhất” giữa đầu vào  $X$  và đầu ra  $Y$ , bằng cách quan sát một tập hợp con  $X_{train} \subset X$ . Ý nghĩa của “tốt nhất” phụ thuộc vào vấn đề chúng ta muốn giải quyết, và liệu rằng đầu ra mong muốn  $Y$  có được biết hay không.

#### Lưu ý về ký hiệu

Trong Luận văn, chúng tôi sử dụng các chữ cái viết ở dạng in nghiêng (có thể là viết Hoa) để biểu diễn các số vô hướng, ví dụ  $x$ ,  $u$ ,  $y$ ,  $N$ . Các vector được biểu diễn bằng các chữ cái thường in đậm, ví dụ  $\mathbf{y}$ ,  $\mathbf{x}$ . Các ma trận được biểu diễn bởi các chữ viết hoa in đậm, ví dụ  $\mathbf{X}$ ,  $\mathbf{Y}$ ,  $\mathbf{W}$ .

### 3.1.1 Học có giám sát (Supervised Learning)

Học có giám sát là một phương thức học của ML khi chúng ta biết cả đầu vào  $X$  và đầu ra  $Y$  trong quá trình huấn luyện. Trong trường hợp này, chúng ta thường muốn giảm thiểu “error” giữa mục tiêu dự đoán  $\tilde{\mathbf{y}}^{(i)}$  và đầu ra mong muốn  $\mathbf{y}^{(i)} \in Y$  với  $\tilde{\mathbf{y}}^{(i)} = f(\mathbf{x}^{(i)})$  và  $\mathbf{x}^{(i)} \in X$ .

### 3.1.2 Học không giám sát (Unsupervised Learning)

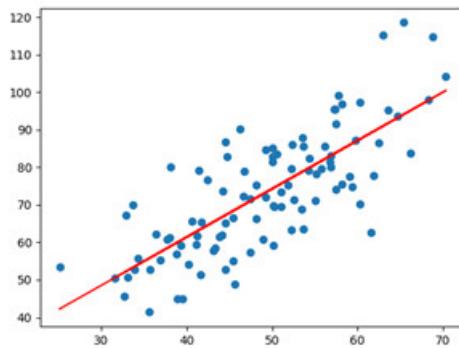
Học không giám sát là một phương thức học của ML khi chúng ta biết đầu vào  $X$ , nhưng không biết đầu ra  $Y$ . Trong trường hợp này chương trình phải học cả ánh xạ  $f$  và biểu diễn “tốt nhất” của  $Y$ . Một lần nữa, ý nghĩa “tốt nhất” phụ thuộc vào vấn đề cần giải quyết. Thông thường chúng ta muốn  $Y$  cung cấp càng nhiều thông tin có giá trị của  $X$  càng tốt.

Người ta cũng thường sử dụng học không giám sát là để tăng cường cho học có giám sát. Đầu vào  $X$  ở dạng thô có thể quá phức tạp đối với người sử dụng học có giám sát. Một người học không giám sát ở trường hợp này có thể được sử dụng để tìm biểu diễn của  $X$  một cách dễ dàng hơn. Một cách rõ ràng hơn, chúng ta muốn tìm hai ánh xạ  $f$  và  $g$  sao cho “error” giữa  $\tilde{\mathbf{y}}^{(i)} = f(g(\mathbf{x}^{(i)}))$  và đầu ra mong muốn  $\mathbf{y}^{(i)}$  là nhỏ nhất, trong đó  $f$  là người học có giám sát, và  $g$  là người học không giám sát.

Sự kết hợp giữa học không giám sát và học có giám sát là rất quan trọng đối với DCNs, đây là thuật toán sẽ được nhắc lại trong phần 3.2.

### 3.1.3 Hồi quy (Regression)

Hồi quy là bài toán trong việc tìm kiếm ánh xạ giá trị thực (real-valued mapping) tốt nhất  $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ , với  $n$  và  $m$  là số chiều của đầu vào và đầu ra. Khi xem nhận dạng hành động con người là một bài toán phân loại (chi tiết trong mục 3.1.4), thì hồi quy có vai trò quan trọng trong bài toán, do đó, nó cần phải bao quát nhiều chi tiết.



Hình 3.2 Ví dụ về hồi quy tuyến tính của một biến.

Trong mục nhỏ này, chúng tôi sẽ chỉ chi tiết hồi quy với hàm tuyến tính. Các phần sau sẽ mở rộng trên hàm quyến tính để hồi quy.

### Hồi quy tuyến tính (Linear Regression)

Hồi quy tuyến tính là bài toán tìm một hàm tuyến tính ánh xạ tốt nhất của một cặp đầu vào và đầu ra có giá trị thực. Trong trường hợp đơn biến, hồi quy tuyến tính được đưa về theo dạng sau:

$$wx + b = y \quad (3.1.1)$$

Với  $w$  và  $b$  là các thông số chúng ta cần học. Chúng ta gọi  $w$  là trọng số và  $b$  là bias. Tuy nhiên, thông thường chúng ta cần thực hiện hồi quy với nhiều biến đầu vào và đầu ra. Hồi quy tuyến tính với đầu vào đa biến được viết thành tổng của hồi quy tuyến đa biến đơn:

$$\sum_{i=0}^D w_i x_i + b_i = \mathbf{w}^T \mathbf{x} + \sum_{i=0}^D b_i = \mathbf{w}^T \mathbf{x} + b = y \quad (3.1.2)$$

Với  $b = \sum_{i=0}^D b_i$ . Hồi quy tuyến tính dễ dàng được mở rộng thành đầu ra đa biến bằng mô hình hồi quy tuyến tính một đầu ra cho nhiều giá trị đầu ra, được viết dưới dạng ma trận như sau:

$$\mathbf{W}^T \mathbf{x} + \mathbf{b} = \mathbf{y} \quad (3.1.3)$$

### 3.1.4 Phân loại (Classification)

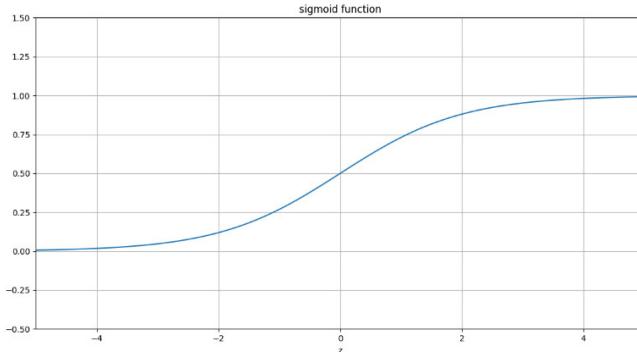
Phân loại là bài toán liên kết đầu vào đã cho với lớp thích hợp nhất. Các lớp được biểu diễn trong một vector nhị phân  $\mathbf{c}$  trong đó  $c_i = 1$  đại diện cho đầu vào thuộc về lớp thứ  $i$ . Chúng ta chỉ xem xét trường hợp đầu vào chỉ có thể thuộc về một lớp, vì vậy các thành phần khác của  $\mathbf{c}$  phải bằng 0.

Trong thị giác máy tính, loại bài toán này thường được gọi là bài toán nhận dạng hình ảnh. Nhận dạng hình ảnh là bài toán về xác định xem một mục tiêu nào đó có đang hiện diện trong một hình ảnh hay không. Lưu ý rằng chúng ta không quan tâm đến vị trí của mục tiêu trong ảnh. Vị trí của các mục tiêu được xem xét trong bài toán phát hiện (detection problem), mà chúng ta sẽ đề cập trong mục 4.2.

Khi mô hình hóa bài toán phân loại, sẽ dễ dàng hơn nếu coi đầu ra là một phân phối xác suất  $\bar{\mathbf{c}}$ . Thành phần  $\bar{c}_i$  đại diện cho xác suất đầu vào thuộc về lớp thứ  $i$ . Vậy  $c_i = 1$  nếu  $i = \text{argmax}_i c_i$ . Biểu diễn đầu ra dưới dạng phân phối xác suất cho phép chúng ta tiếp cận bài toán phân loại như là một bài toán hồi quy. Ngoài ra, phân phối xác suất cho phép chúng ta mô hình hóa tính chất xác thực của dự đoán. Một dự đoán được xem là đáng tin cậy hơn khi xác suất cho một lớp gần bằng 1.

#### Logistic Regression

Nhưng làm sao để chúng ta mô hình phân loại như là một phân phối xác suất? Chúng ta bắt đầu với trường hợp chỉ có một lớp mục tiêu  $c$ , trong đó  $c$  là xác suất đầu vào thuộc về lớp đã cho. Giá trị gần với 1 hơn thì thể hiện độ tin cậy cao rằng đầu vào thuộc về lớp đã cho, và ngược lại, giá trị gần 0 thể hiện một độ tin cậy thấp.



Hình 3.3 Hàm sigmoid

Logistic regression khá giống với hồi quy tuyến tính. Sự khác nhau là đầu ra được giới hạn trong khoảng từ 0 đến 1 (nếu không thì đó không phải là xác suất). Đặt  $z$  là phụ thuộc tuyến tính từ đầu vào  $x$  bởi:

$$z = \mathbf{w}^T \mathbf{x} + b \quad (3.1.4)$$

Để ràng buộc đầu ra ở giữa 2 giá trị 0 và 1, chúng ta sử dụng hàm sigmoid:

$$\sigma(z) = \frac{e^z}{e^z + 1} \quad (3.1.5)$$

Có thể thấy, hàm sigmoid có các thuộc tính sau:

$$\begin{aligned} \sigma(z) &\rightarrow 1 \text{ khi } z \rightarrow \infty \\ \sigma(z) &\rightarrow 0 \text{ khi } z \rightarrow -\infty \text{ và} \\ \sigma(0) &= 0.5 \end{aligned} \quad (3.1.6)$$

Điều này làm cho hàm sigmoid khả thi cho việc mô hình hóa một xác suất.

### Softmax

Logistic regression chỉ có thể áp dụng cho các bài toán phân loại chỉ với một lớp mục tiêu. Đối với trường hợp có nhiều lớp mục tiêu, chúng ta sử dụng hàm softmax để mô hình hóa phân phối xác suất. Hàm softmax hoạt động tương tự như logistic regression và được định nghĩa bởi:

$$\text{softmax}(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=0}^K e^{z_k}} \quad (3.1.7)$$

Với  $\mathbf{z}$  là phụ thuộc tuyến tính của  $x$  bởi:

$$\mathbf{z} = \mathbf{W}^T \mathbf{x} + \mathbf{b} \quad (3.1.8)$$

Điều làm cho softmax và logistic regression thích hợp cho DCNs là chúng có các đạo hàm dạng đóng, điều này rất quan trọng đối với việc tính toán gradients cho lan truyền ngược (back-propagation) trong DCNs. (xem mục 3.3)

### Beyond Linear Classification

Softmax là một ví dụ về hàm phân loại tuyến tính, có nghĩa là decision boundaries (decision boundary của hàm softmax cho lớp thứ  $j$  là tất cả  $\mathbf{x}$  sao cho  $\text{softmax}(\mathbf{x})_j = 0.5$ ) của mỗi lớp là một hàm tuyến tính. Một số bài toán phân loại là phân loại tuyến tính, nhưng đa số các bài toán phân loại khác thì không phải.

Tuy nhiên, nếu chúng ta có thể ánh xạ  $x_j$  đến một không gian đặc trưng, có thể sử dụng phân loại tuyến tính cho bài toán phân loại. DCNs cố gắng học một không gian đặc trưng tuyến tính riêng biệt của  $x_j$  và phân loại tuyến tính cùng một lúc, được đề cập trong phần sau của luận văn.

## 3.2 Mạng tích chập sâu (Deep Convolutional Network)

Deep learning, còn được gọi là học sâu có cấu trúc (deep structured learning) hay là học phân cấp (hierarchical learning), là một lớp các thuật toán Machine learning để học tập miêu tả ở nhiều cấp độ. Về mặt toán học, Deep learning có thể được xem như là kết hợp của việc có thể học  $f_0, f_1, \dots, f_n$  sao cho:

$$f_0 \circ f_1 \circ \dots \circ f_n(\mathbf{x}) = \tilde{\mathbf{y}} \quad (3.2.1)$$

Như có thể thấy ở công thức trên, DCNs là feedforward, đầu vào của hàm  $f_i$  chỉ phụ thuộc vào đầu ra của hàm  $f_j$  sao cho  $j > i$ .

Bằng cách nào để các hàm số của DCN được chọn dựa vào ứng dụng. Thông thường các hàm số rất đơn giản. Đối với bài toán phân loại  $f_0$ , thường được chọn là hàm softmax, được định nghĩa ở phần 3.1.4. Điều kiện duy nhất của

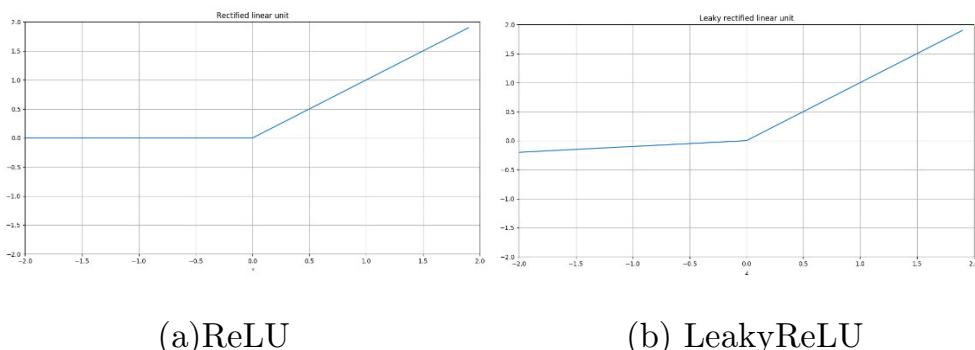
các hàm số là khả vi. Nếu không thì nó không thể tính toán gradient cho lan truyền ngược (xem phần 3.3). Từ giờ trở đi, chúng ta gọi các hàm chức năng là các lớp (layers) và  $f_1, f_2, \dots, f_n$  trong 3.2.1 được gọi là các lớp ẩn. Phần này giới thiệu các lớp được sử dụng trong luận văn này.

### 3.2.1 Lớp kết nối đầy đủ (Fully Connected Layers)

Lớp cơ bản nhất được sử dụng trong DCNs là các lớp kết nối đầy đủ (fully connected layer). Một lớp  $f_{fc}$  được kết nối đầy đủ nếu mỗi thành phần  $y_i \in \tilde{y} = f_{fc}(\mathbf{x})$  phụ thuộc vào tất cả các thành phần của  $\mathbf{x}$ .

Dạng cơ bản nhất của lớp kết nối đầy đủ là khi  $\tilde{y}$  và  $\mathbf{x}$  là phụ thuộc tuyến tính –  $f$  là hàm tuyến tính được định nghĩa trong công thức (3.1.3). Tuy nhiên, các hàm tuyến tính không thể mô hình hóa cho chế độ phi tuyến tính. Nếu  $f_1$  và  $f_2$  đều là tuyến tính, thì thành phần  $f_1 \circ f_2$  cũng là tuyến tính.

Mục tiêu của việc thêm các lớp ẩn vào mạng tích chập sâu (DCNs) là mô hình hóa các chế độ phi tuyến tính. Để tạo một lớp kết nối đầy đủ phi tuyến tính, chúng ta kích hoạt các thành phần của  $\tilde{y}$  bằng hàm kích hoạt phi tuyến tính. Dưới đây là mô tả các hàm kích hoạt được sử dụng trong luận văn này.



Hình 3.4 Hàm kích hoạt cho lớp Fully Connected sử dụng trong luận văn

#### Rectified Linear Unit (ReLU)

Một trong những lớp kích hoạt được sử dụng phổ biến nhất là ReLU. Định nghĩa của ReLU rất đơn giản; nó xác định nếu giá trị đầu vào dương, và bằng 0 nếu đầu vào âm. Về mặt toán học, ReLU được biểu diễn:

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.2.2)$$

Các quan sát trong khoa học thần kinh cho thấy rằng kích hoạt các tế bào thần kinh trong não có thể xấp xỉ bằng một bộ chỉnh lưu. Điều này đã truyền cảm hứng cho việc sử dụng ReLU trong các mạng tích chập sâu (DCNs).

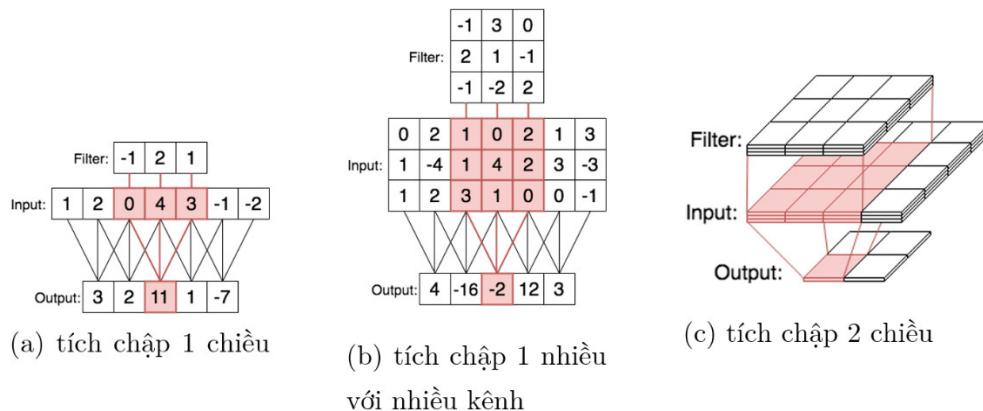
### Leaky Rectified Linear Unit (LeakyReLU)

Một biến thể của ReLU là LeakyReLU. Ở đây, đầu vào là âm được chia tỉ lệ với một phần nhỏ. Trong luận văn này, chúng tôi chia tỉ lệ ReLU với 0.1 nếu là âm.

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.1x & \text{otherwise} \end{cases} \quad (3.2.3)$$

Định nghĩa này cho phép một số thông tin đi qua hàm kích hoạt ngay cả khi đầu vào là âm. Trong bài này, LeakyReLU chỉ được sử dụng cho hệ thống phát hiện đối tượng YOLO.

### 3.2.2 Lớp chập (Convolutional Layers)



Hình 3.5 Các tích chập khác nhau, được trực quan hóa

Vấn đề của lớp kết nối đầy đủ là quá nhiều tham số được yêu cầu khi đầu vào lớn. Ví dụ, nếu cả đầu vào và đầu ra của một lớp kết nối đầy đủ đều có kích thước là 4096, thì số tham số được yêu cầu là  $4096 \times 4096 = 16.777.216$

trọng số tham số cho lớp kết nối đầy đủ này. Hình ảnh thường có số chiều rất cao, vì vậy các lớp kết nối đầy đủ thường không khả thi.

Một hạn chế khác của các lớp kết nối đầy đủ là thông tin không gian trong hình ảnh không được xem xét. Một lớp kết nối đầy đủ không biết liệu một thành phần của đầu vào đại diện cho các đặc trưng từ pixel ở góc trên bên phải hay góc dưới bên trái.

Các lớp chập giải quyết được cả hai hạn chế này bằng cách sử dụng các convolutional filter (hay còn gọi là kernel). Chúng ta sẽ bắt đầu với trường hợp 1 chiều của convolutional filters.

### Tích chập 1 chiều (1 Dimensional Convolution)

Trong trường hợp 1 chiều, đầu vào  $x$  là một vector có chiều dài  $W$ , và  $f_{conv}$  là lớp chập một chiều hoạt động trên  $x$ . Thì  $\tilde{y} = f_{conv}$  được định nghĩa cho mỗi  $\tilde{y}_i$  như sau:

$$\tilde{y}_i = w_0 x_{i-\lfloor K/2 \rfloor} + \dots + w_{\lfloor K/2 \rfloor} x_i + \dots + w_{K-1} x_{i+\lceil K/2 \rceil - 1} + b \quad (3.2.4)$$

Ở đây,  $K$  là chiều dài của convolution filter của  $f_{conv}$ . Như có thể thấy ở công thức 3.2.4, thành phần  $\tilde{y}_i$  chỉ phụ thuộc vào những thành phần  $K$  trong khoảng cách gần  $x_i$ . Điều này có nghĩa là một số thông tin không gian của  $x$  được duy trì trong  $\tilde{y}$ . Ngoài ra, các thành phần của  $\tilde{y}$  cùng chia sẻ trọng số  $w$  giống nhau. Do đó, chỉ có  $K$  thông số huấn luyện được yêu cầu bởi lớp này [3].

Mặt khác, chú ý rằng một bias kí hiệu  $b$  được thêm vào mỗi thành phần đầu ra. Phép chập 1 chiều được minh họa trong hình 3.4(a).

### Tích chập 2 chiều (2 Dimensional Convolution)

Đối với hình ảnh, chúng ta cần xem xét sự lân cận về cả chiều dọc và ngang của thành phần đầu vào. Điều này đòi hỏi phải sử dụng convolutional filter 2 chiều.

Với đầu vào  $\mathbf{x}$  có kích thước  $W \times H$ , và convolutional filter có kích thước  $K \times L$ . Để cho đơn giản, chúng ta chỉ xem xét trường hợp khi  $K = L = 3$ , nhưng việc mở rộng định nghĩa cho các giá trị khác của  $K$  và  $L$  là dễ dàng. Thành phần  $\tilde{y}_{i,j}$  được định nghĩa như sau:

$$\begin{aligned}\tilde{y}_{i,j} = & w_{0,0}x_{i-1,j-1} + w_{1,0}x_{i,j-1} + w_{2,0}x_{i+1,j-1} + w_{0,1}x_{i-1,j} \\ & + w_{1,1}x_{i,j} + w_{2,1}x_{i+1,j} + w_{0,2}x_{i-1,j+1} \\ & + w_{1,2}x_{i,j+1} + w_{2,2}x_{i+1,j+1} + b\end{aligned}\quad (3.2.5)$$

Tương tự như trường hợp 1 chiều, các tham số  $w$  được chia sẻ giữa tất cả các thành phần của  $\tilde{y}$ , do đó, chỉ có  $K \times L$  trọng số và 1 bias có khả năng huấn luyện được yêu cầu trong những lớp này.

### Tích chập 2 chiều với đa kênh (2 Dimensional Convolution with Channels)

Đến lúc này, chúng ta chưa xem xét rằng hình ảnh có thể sử dụng nhiều kênh. Ví dụ: hình ảnh RGB có 3 kênh – mỗi kênh là màu đỏ, xanh lục, xanh lam. Chúng ta hay xem xét chung với trường hợp có  $C$  kênh để đầu vào có kích thước là  $W \times H \times C$ . Vì thứ tự các kênh mang ý nghĩa – biểu diễn RGB cũng tương tự biểu diễn BGR – tất cả các kênh đều được xem xét trong tích tích chập. Đặt  $\mathbf{x}_{i,j,*}$  là vector của tất cả các kênh trong vị trí  $(i, j)$ . Thì thành phần  $\tilde{y}_{i,j}$  được định nghĩa là:

$$\begin{aligned}\tilde{y}_{i,j} = & \mathbf{w}_{0,0,*}^T \mathbf{x}_{i-1,j-1,*} + \mathbf{w}_{1,0,*}^T \mathbf{x}_{i,j-1,*} + \mathbf{w}_{2,0,*}^T \mathbf{x}_{i+1,j-1,*} + \mathbf{w}_{0,1,*}^T \mathbf{x}_{i-1,j,*} \\ & + \mathbf{w}_{1,1,*}^T \mathbf{x}_{i,j,*} + \mathbf{w}_{2,1,*}^T \mathbf{x}_{i+1,j,*} + \mathbf{w}_{0,2,*}^T \mathbf{x}_{i-1,j+1,*} \\ & + \mathbf{w}_{1,2,*}^T \mathbf{x}_{i,j+1,*} + \mathbf{w}_{2,2,*}^T \mathbf{x}_{i+1,j+1,*} + b\end{aligned}\quad (3.2.6)$$

Tương tự giống những trường hợp trước, các tham số  $w$  được chia sẻ. Vì vậy, số lượng trọng số có thể huấn luyện là  $K \times L \times C$  và một bias để huấn luyện [3].

Tích chập 2 chiều với nhiều kênh được minh họa ở hình 3.4(c).

## Đa Bộ Lọc Tích Chập (Multiple Convolutional Filters)

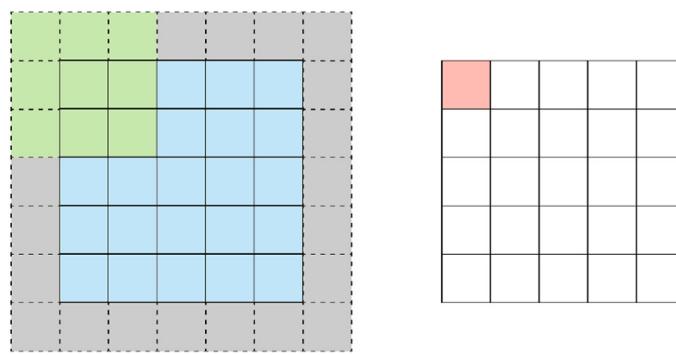
Cuối cùng, chúng ta hoàn thành định nghĩa của lớp chập. Định nghĩa trong công thức 3.2.6, cho đầu ra là biểu diễn với 1 kênh. Nếu chúng ta muốn có đầu ra là  $D$  kênh, chúng ta định nghĩa  $D$  convolution filters khác nhau và xếp chồng biểu diễn kết quả trên không gian kênh mở rộng.

Vì vậy, một lớp chập với  $D$  filters có chiều rộng  $K$ , chiều cao  $L$  được áp dụng biểu diễn cho  $C$  kênh có  $K \times L \times C \times D$  trọng số có thể huấn luyện và  $D$  bias có thể huấn luyện.

Để đạt được phi tuyến tính, các thành phần đại diện cho đầu ra  $y$  được kích hoạt bằng hàm kích hoạt phi tuyến tính, đó là hàm ReLU.

### Padding

Định nghĩa trên cho convolutional filters (bộ lọc tích chập) chỉ xác định rõ cho  $\tilde{y}_{i,j}$  tại  $\lfloor K/2 \rfloor < i < W - \lceil K/2 \rceil$  và  $\lfloor L/2 \rfloor < j < H - \lceil L/2 \rceil$ . Đối với các giá trị khác của  $i$  và  $j$ , tích chập sẽ bao gồm các phần tử biểu diễn bên ngoài. Để giải quyết vấn đề này, chúng ta đệm đại diện bằng 0 xung quanh các đường biên của nó để tích chập được xác định cho tất cả các phần tử có giá trị của đại diện [3].



Stride 1 with Padding

Feature Map

*Hình 3.6: Phần màu xám chính là phần padding thêm nào input  
(Nguồn: <https://towardsdatascience.com/convolutional-neural-network-cb0883dd6529>)*

### Max pooling

Max pooling là một cách để giảm kích thước không gian của sự biểu diễn để giảm lượng tham số và tính toán trong mạng, và do đó nó cũng có thể kiểm soát overfitting. Nó hoạt động bằng cách chia các đại diện thành các ô có kích thước  $N \times M$ . Trong mỗi ô, phần tử có giá trị cao nhất được trả về. Thông thường ta chọn  $N = M = 2$  [3].

### Strides

Một sự thay thế khác cho max pooling là striding. Striding cũng làm giảm kích thước không gian của một biểu diễn. Nếu stride là  $N \times M$  thì tích chập sẽ chuyển đến đầu vào thứ  $N$  tiếp theo trong trục x và đầu vào thứ  $M$  trong trục y [3].

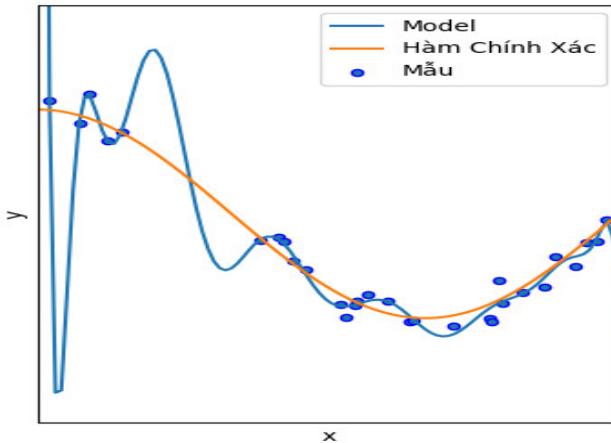
### 3.2.3 Batch Normalization

Một lớp batch normalization học những giá trị trung bình và phương sai của đầu vào. Lớp này sau đó chuẩn hóa đầu vào để đầu ra có giá trị trung bình là 0 (zero mean) và phương sai là 1 (unit-variance) theo giá trị trung bình và phương sai đã học.

Một lợi ích được nhận định của batch normalization là huấn luyện nhanh hơn. Trong luận văn này batch normalization được sử dụng cho hệ thống SSD object detection [4].

### 3.2.4 Cơ chế tắt ngẫu nhiên (Dropout)

Một điều không mong muốn với các mô hình huấn luyện là rất nhiều mô hình cho kết quả rất tốt với tập dữ liệu hiện tại, nhưng lại hoàn toàn không thể đem áp dụng thực tế vì kết quả dự đoán quá thấp. Hiện tượng mà mô hình dữ liệu làm việc tốt trên tập dữ liệu mẫu nhưng lại làm việc tệ trước việc dự đoán dữ liệu thực tế được gọi là **overfitting** [2].



*Hình 3.7 Minh họa cho overfitting*

Một cách đơn giản để ngăn chặn overfitting là sử dụng lớp dropout. Lớp dropout có một siêu tham số tỉ lệ dropout  $p$ , là một thành phần xác suất, lựa chọn ngẫu nhiên một số thành phần của đầu vào và loại bỏ nó trong quá trình huấn luyện. Nếu một thành phần bị loại bỏ, nó được thiết lập là 0. Trong validation set, lớp dropout không thực hiện.

Để tránh sự không nhất quán giữa training và validation, đầu ra của lớp dropout được chia theo tỉ lệ  $1/(1-p)$ .

### 3.3 Tối ưu hóa tham số (Parameter Optimization)

Trong các phần trước, chúng tôi chỉ mới giới thiệu về việc xây dựng các khôi cho DCNs, nhưng chúng tôi chưa nói về cách mà các tham số trong mạng được học. Trong phần này, chúng tôi sẽ thảo luận về cách tối ưu hóa các tham số cho dữ liệu huấn luyện.

#### 3.3.1 Categorical Crossentropy

Khi huấn luyện mô hình, chúng ta muốn giảm error giữa mục tiêu dự đoán  $\tilde{y}$  và mục tiêu thực tế  $y$  càng nhiều càng tốt. Đặt  $\mathcal{L}(\tilde{y}, y)$  là hàm mất mát (loss function), biểu thị error giữa  $\tilde{y}$  và  $y$ . Các tham số  $\theta_{optimal}$  của mô hình  $f$  là tối ưu khi:

$$\theta_{optimal} = \arg \min_{\theta} \sum_{i=0}^N \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}) \quad (3.3.1)$$

Để nhận dạng hành động của con người trong video, chúng ta muốn tối giảm các classification error. Một hàm phổ biến được sử dụng cho classification error là categorical crossentropy, được định nghĩa như sau:

$$\mathcal{L}_{cross}(\tilde{\mathbf{y}}, \mathbf{y}) = - \sum_{k=0}^K y_k \log \tilde{y}_k \quad (3.3.2)$$

Ở đây, một sự phân loại là tốt khi  $\mathcal{L}_{cross}(\tilde{\mathbf{y}}, \mathbf{y}) = 0$

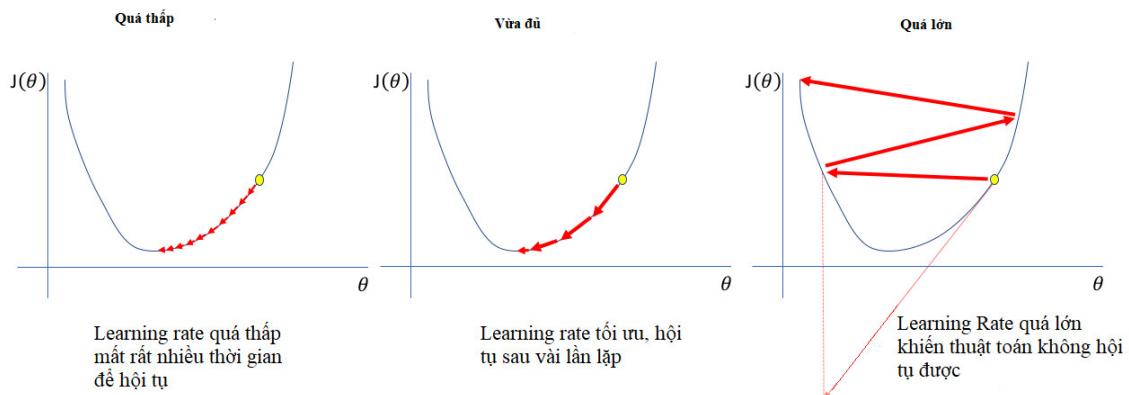
### 3.3.2 Gradient Descent

Gradient Descent (GD) là một kỹ thuật đơn giản để tối ưu hóa các mục tiêu phi tuyến. Bắt đầu từ một  $\theta_0$  giả định ban đầu, lặp đi lặp lại tính  $\theta_{t+1}$  bằng cách tính theo gradient của hàm mục tiêu  $g$  với các thông số  $\theta_t$ .

Cập nhật  $\theta_{t+1}$  được định nghĩa như sau:

$$\theta_{t+1} = \theta_t - \gamma_{theta} \nabla_{\theta_t} g(\mathbf{x}_i; \theta_t) \quad (3.3.3)$$

Trong đó,  $g$  là mục tiêu chúng ta muốn tối giảm (*minimize*) và  $\gamma$  là hệ số learning rate. Hệ số learning rate được sử dụng để kiểm soát độ lớn “bước nhảy” mà gradient sẽ thực hiện. Hệ số learning rate thấp hơn thường chính xác hơn nhưng sử dụng nhiều hơn gian hơn để hội tụ. Một cách thường được sử dụng khi tiến hành việc huấn luyện là bắt đầu với hệ số learning rate lớn và giảm hệ số learning rate theo một lịch trình định trước hoặc khi mục tiêu “bão hòa”.



*Hình 3.8 Hình ảnh thể hiện sự khác nhau của GD khi learning rate khác nhau.*

Trong trường hợp này, mục tiêu chúng ta muốn tối giảm là rủi ro thực nghiệm (empirical risk)  $E_N(\mathcal{L})$  – tốn thất trung bình trên tất cả đầu vào.

$$E_N(\mathcal{L}) = \frac{1}{N} \sum_{i=0}^N \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}) \quad (3.3.4)$$

GD cho categorical cross entropy được định nghĩa như sau:

$$\theta_{t+1} = \theta_t - \gamma \frac{1}{N} \sum_{i=0}^N \nabla_{\theta_i} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta_t), \mathbf{y}^{(i)}) \quad (3.3.5)$$

Một nhược điểm của GD là nó cần phải xem xét toàn bộ dữ liệu trước khi các thông số được cập nhật một bước. Điều này làm cho việc huấn luyện rất tốn kém tài nguyên và có vẻ không khả thi khi dữ liệu quá lớn. Do đó, GD hiếm khi được sử dụng cho một bộ dữ liệu lớn.

### 3.3.3 Stochastic Gradient Descent

Thay vì sử dụng toàn bộ tập dữ liệu để cập nhật tham số thì ta có thể sử dụng từng dữ liệu một để cập nhật. Phương pháp như vậy được gọi là Stochastic Gradient Descent (SGD). Về cơ bản ở mỗi lần cập nhật tham số, ta duyệt qua toàn bộ các cặp mẫu, và việc cập nhật tham số được định nghĩa như sau:

$$\theta_{t+1} = \theta_t - \gamma \frac{1}{|\mathcal{X}_t|} \sum_{i \in \mathcal{X}_t} \nabla_{\theta_t} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta_t), \mathbf{y}^{(i)}) \quad (3.3.6)$$

Vì sử dụng từng mẫu đơn một nên tốc độ tính đạo hàm của SGD sẽ nhanh hơn nhiều so với GD nhưng bù lại tốc độ hội tụ bị giảm đi. Một lưu ý khi thiết lập giải thuật này là mỗi bước cập nhật ta nên xáo trộn dữ liệu rồi sau đó mới lấy ra cập nhật. Việc này giúp giảm đi sự đi lòng vòng về đích của giải thuật vì ta trao khả năng cập nhật ngẫu nhiên cho SGD tức là sẽ có cơ hội nhảy được một bước xa hơn khi tính toán.

### Momentum

Một phần mở rộng của SGD là thay vì cập nhật trực tiếp các thông số, thì ta cộng dồn một vận tốc  $v_t$  cho mỗi vòng lặp  $t$  của việc cập nhật thông số. Ở mỗi vòng lặp, các thông số được cập nhật bởi vận tốc lũy kế  $v_t$ . Các gradient sẽ có một thành phần gia tốc trên các thông số.

SGD với momentum được định nghĩa như sau:

$$\begin{aligned} v_{t+1} &= \mu v_t - \gamma \frac{1}{|\mathcal{X}_t|} \sum_{i \in \mathcal{X}_t} \nabla_{\theta_t} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta_t), \mathbf{y}^{(i)}) \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned} \quad (3.3.7)$$

Với  $\mu \in [0, 1]$  là hệ số động lượng (đà). Hệ số động lượng (the momentum coefficient) yết định vận tốc lũy kế (the accumulated velocity) sẽ giảm chậm ra sao. Nếu  $\mu = 0$  có nghĩa là không có momentum được sử dụng, và tương ứng với SGD thông thường.

### 3.3.4 Weight Decay

Weight Decay (Decay có nghĩa tiêu biến) là một cách đơn giản để giảm khả năng bị overfitting trong DCNs bằng cách cộng thêm một đại lượng regularization vào hàm mất mát [2]. Overfitting có thể xảy ra khi ta có các mạng tích chập sâu học các cấu trúc phức tạp trong tập huấn luyện, nhưng thực tế thì có rất ít thông tin trong tập huấn luyện đó. Weight Decay sẽ khiến cho các thành phần của vector trọng số không quá lớn, thậm chí nhiều thành

phần bằng 0. Điều này khiến cho việc có nhiều hidden unit vẫn an toàn vì nhiều trong số chúng gần với 0.

Với dạng thông thường của gradient descent trong công thức 3.3.5, weight decay có dạng là:

$$\theta_{t+1} = \theta_t - \gamma \left( \frac{1}{N} \sum_{i=0}^N \nabla_{\theta_i} \mathcal{L}(f(\mathbf{x}^{(i)}; \theta_t), \mathbf{y}^{(i)}) - \alpha \theta_t \right) \quad (3.3.8)$$

Với  $\alpha$  là hệ số weight decay.當然, đại lượng weight decay có thể được bao gồm trong SGD và SGD với momentum.

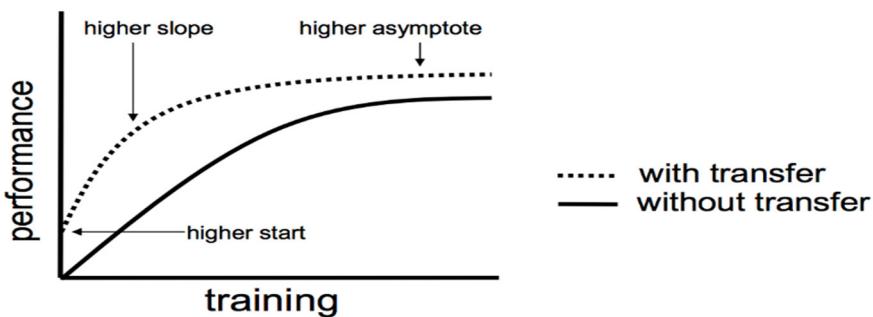
### 3.3.5 Học chuyển tiếp (Transfer Learning)

Các thuật toán ML truyền thống giả định rằng tập huấn luyện và validation cho một tác vụ được rút ra từ cùng một phân phối. Điều này nghĩa là các mô hình ML được đào tạo lại từ đầu cho mỗi một nhiệm vụ mới. Tuy nhiên, một phương pháp mới ngày càng phổ biến đó là chuyển các tham số đã được học từ các nhiệm vụ trước cho các nhiệm vụ mới. Điều này được gọi là học chuyển tiếp.

Học chuyển tiếp cho phép thu được các tham số từ một nhiệm vụ đã thực hiện được chuyển sang một nhiệm vụ mới. Việc chuyển tiếp này dễ dàng được thực hiện trên các DCNs. Một thực tế phổ biến là thay thế các lớp cuối cùng (thường là các lớp liên kết đầy đủ) bằng các lớp mới, chưa được huấn luyện, phù hợp hơn cho nhiệm vụ.

Ta có thể thấy được 3 lợi ích của việc học chuyển tiếp trong hình 3.9:

1. Higher star: Điểm bắt đầu (trước khi tinh chỉnh mô hình) trên mô hình học chuyển tiếp cao hơn so với mô hình thông thường.
2. Higher slope: Tốc độ cải thiện kỹ năng trong quá trình huấn luyện mô hình học chuyển tiếp có độ dốc cao hơn so với huấn luyện thông thường.
3. Higher asymptote: Mô hình học chuyển tiếp hội tụ tốt hơn so với mô hình thông thường.



*Hình 3.9 So sánh tương quan hiệu quả của mô hình train từ đầu và transferred model.*

(*Nguồn: <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>*)

Một ví dụ, nếu chúng ta có một mô hình được huấn luyện về một bài toán phân loại với 1000 mục tiêu và chúng ta muốn sử dụng cùng một mô hình cho một bài toán với 100 mục tiêu, chúng ta chỉ cần thay thế lớp kết nối đầy đủ trên cùng bằng một lớp kết nối đầy đủ với 100 mục tiêu.

Với bài toán nhận dạng hình ảnh, người ta thường sử dụng các mạng được huấn luyện sẵn trên các mạng được đào tạo trước trên các bộ dữ liệu lớn như ImageNet. Một số trong số này có thể tải về miễn phí để sử dụng.

### 3.4 Kết luận chương 3

Nhìn chung, chương 3 đã khái quát được một cách tổng quát về DL. Chúng tôi đã trình bày các thuật toán quan trọng trong ML: các phương pháp học ML, hồi quy và phân loại. Chương 3 tiếp tục trình bày về mạng tích chập sâu với việc trình bày các lớp quan trọng trong mạng và việc tránh overfitting. Sau đó, chúng tôi tiếp tục trình bày về việc tối ưu hóa tham số bằng các phương pháp như GD và SGD và tránh overfitting bằng weight decay. Cuối cùng chương 3 trình bày về học chuyển tiếp, một kỹ thuật quan trọng trong việc huấn luyện mô hình DL. Các kiến thức được trình bày là cơ sở cho việc xây dựng mô hình ở chương 5.

# CHƯƠNG 4

## NHẬN DẠNG HÀNH ĐỘNG CON NGƯỜI

Với những kiến thức Deep Learning cơ bản đã được giới thiệu trong chương 3, chúng tôi bây giờ có thể mô tả chi tiết các phương pháp nhận dạng hành động con người được sử dụng trong luận văn này. Các phương pháp được mô tả gồm có: Mạng 2 luồng, mô hình YOLO và SSD cho nhận dạng đối tượng, phương pháp ACAM, 3D CNN và bộ dữ liệu AVA.

### 4.1 Mạng tích chập 2 luồng

Mô hình CNN 2 luồng [5], được giới thiệu bởi Karen Simonyan và Andrew Zisserman, là một trong những cách tiếp cận thành công nhất để nhận dạng hành động của con người trong video. Ý tưởng của mạng tích chập 2 luồng là kết hợp các dự đoán của hai mạng riêng biệt được huấn luyện trên các thông tin khác nhau.

Mạng đầu tiên được huấn luyện về thông tin không gian - ảnh RGB thông thường - của các videos. Mạng này được gọi là luồng không gian và rất giống với các hệ thống nhận dạng hình ảnh. Luồng không gian chỉ nhìn thấy một frame tại một thời điểm, vì vậy nó không thể chụp bất kỳ chuyển động nào. Một cách để giải thích luồng không gian là nó nhìn thấy bối cảnh của videos. Ví dụ: nếu hành động trong môi trường như “nhà bếp”, thì hành động sẽ có thể liên quan đến “nấu ăn”.

Mạng thứ hai được huấn luyện về thông tin thời gian của các videos. Mạng này được gọi là luồng thời gian và ghi lại chuyển động của video. Đầu vào của luồng thời gian là luồng quan của nhiều frames liên tiếp của video.

Hai luồng này cung cấp dự đoán bổ sung. Các thí nghiệm cho thấy rằng dự đoán trung bình từ cả 2 luồng mang lại dự đoán chính xác hơn so với sử dụng các luồng riêng biệt.

### 4.1.1 VGG16 – A Very Deep Convolutional Network

VGG16 [6] được sử dụng làm cơ sở cho các luồng không gian và thời gian, đây là mô hình tiên tiến cho thử thách nhận dạng hình ảnh ImageNet ILSVRC-2012. Việc thực hiện chuyển giao học tập trên VGG16 sang các vấn đề nhận dạng hình ảnh khác là điều phổ biến. Trọng số được huấn luyện sẵn từ ImageNet được sử dụng là trọng số ban đầu khi huấn luyện cho vấn đề nhận dạng hành động con người. Kiến trúc của VGG16 được mô tả trong bảng 4.1.

### 4.1.2 Luồng không gian

Như đã đề cập trước đó, luồng không gian đưa ra dự đoán dựa trên các frames RGB tĩnh. Mạng dựa trên VGG16 trong đó lớp cuối cùng được thay thế bằng lớp được kết nối đầy đủ với số lớp đích chính xác cho tập dữ liệu. Đầu vào của luồng không gian được sub-tracted với trị trung bình của ImageNet.

### 4.1.3 Luồng thời gian

Luồng thời gian cũng dựa trên VGG16, nhưng với optical flow là đầu vào thay vì RGB frames. Một vấn đề với luồng thời gian là đầu vào sử dụng kênh  $2L$  thay vì  $3$  kênh cho RGB. Để giải quyết vấn đề này, lớp chập đầu tiên của luồng thời gian được thay thế bằng một lớp với các kênh đầu vào  $2L$ . Chúng tôi sử dụng  $L = 10$  cho luồng thời gian, có nghĩa là đầu vào của dòng thời gian là  $10$  hình ảnh optical flow liên tiếp.

Bảng 4.1 Kiến trúc VGG16

Type	Name	Filters	Kernel	Output shape
Input	input	3		224 × 224
Conv	conv1_1	64	3 × 3	224 × 224
Conv	conv1_2	64	3 × 3	224 × 224
Maxpool	pool1		2 × 2	112 × 112
Conv	conv2_1	128	3 × 3	112 × 112
Conv	conv2_2	128	3 × 3	112 × 112
Maxpool	pool2		2 × 2	56 × 56
Conv	conv3_1	256	3 × 3	56 × 56
Conv	conv3_2	256	3 × 3	56 × 56
Conv	conv3_3	256	3 × 3	56 × 56
Maxpool	pool3		2 × 2	28 × 28
Conv	conv4_1	512	3 × 3	28 × 28
Conv	conv4_2	512	3 × 3	28 × 28
Conv	conv4_3	512	3 × 3	28 × 28
Maxpool	pool4		2 × 2	14 × 14
Conv	conv5_1	512	3 × 3	14 × 14
Conv	conv5_2	512	3 × 3	14 × 14
Conv	conv5_3	512	3 × 3	14 × 14
Maxpool	pool5		2 × 2	7 × 7
FC-4096	fc1			4096
FC-4096	fc2			4096
FC-1000	fc3			1000
Softmax	predictions			1000

#### 4.1.4 Optical flow

Optical flow là một chuỗi các trường vector dịch chuyển  $d_t$ . Vector chuyển vị  $d_t(u,v)$  biểu diễn chuyển động của điểm  $(u,v)$  giữa hai khung liên tiếp  $t$  và  $t+1$ .

Vector dịch chuyển bao gồm một vector ngang  $d_t^x(u,v)$  và một vector dọc  $d_t^y(u,v)$ . Do đó optical flow giữa hai frames liên tiếp  $W \times H$  là, trong đó  $W$  và  $H$  là chiều rộng và chiều cao của hình ảnh. Nó có được xem như là một hình ảnh 2 kênh trong đó 2 kênh đại diện cho flow ngang và flow dọc.

Chuyển động trên một chuỗi các frames được thể hiện bằng cách xếp chồng các flow channels của các frames L liên tiếp. Do đó, đầu vào được biểu diễn dưới dạng một dung tích của các kênh  $2L$ .

Về mặt toán học, đầu vào  $I$  của frame cho luồng thời gian được xây dựng như sau:

$$\begin{aligned} I_\tau(u, v, 2k-1) &= d_{\tau+k-1}^x(u, v) \\ I_\tau(u, v, 2k) &= d_{\tau+k-1}^y(u, v) \\ u &= [1; w], v = [1; h], k = [1; L] \end{aligned} \quad (4.1.1)$$

## 4.2 Nhận dạng đối tượng

Nhận dạng đối tượng là một phần mở rộng đầy hứa hẹn của nhận dạng hình ảnh. Trong khi nhận dạng hình ảnh chỉ xem xét sự hiện hữu của một mục tiêu nhất định trong một ảnh, nhận dạng đối tượng còn xem xét vị trí và kích thước của mục tiêu. Thường có nhiều mục tiêu hiện diện trong cùng một hình ảnh để hệ thống nhận dạng đối tượng phát hiện.

### 4.2.1 YOLO – You Only Look Once

#### Tổng quan về YOLO

YOLO [7] có nghĩa là “bạn chỉ cần nhìn một lần”, là một hệ thống phát hiện đối tượng nhanh và chính xác. Hệ thống chỉ có duy nhất một mạng CNN, điều này có nghĩa là YOLO đề xuất các bounding boxes và class scores tương ứng một cách đồng thời. Chúng tôi chỉ giới thiệu về YOLOv2 và YOLO9000. YOLOv2 được huấn luyện trên tập dữ liệu COCO, là tập dữ liệu để phát hiện đối tượng với 80 lớp mục tiêu. Mặt khác, YOLO9000 được huấn luyện trên ImageNet object detection challenge với 200 lớp mục tiêu được chia thành hơn 9000 đối tượng. Đó là lý do nó được gọi là YOLO 9000.

#### Kiến trúc của mô hình YOLOv2

Kiến trúc của YOLOv2 [8] được mô tả trong bảng 4.2.

Bảng 4.2 Kiến trúc của YOLOv2

#	Type	Name	Filters	Kernel	Output Shape
1	Input	input	3		$224 \times 224$
2	Conv	conv1-1	32	$3 \times 3$	$416 \times 416$
3	Maxpool	pool1		$2 \times 2$	$208 \times 208$
4	Conv	conv2-1	64	$3 \times 3$	$208 \times 208$
5	Maxpool	pool2		$2 \times 2$	$104 \times 104$
6	Conv	conv3-1	128	$3 \times 3$	$104 \times 104$
7	Conv	conv3-2	64	$1 \times 1$	$104 \times 104$
8	Conv	conv3-3	128	$3 \times 3$	$104 \times 104$
9	Maxpool	pool3		$2 \times 2$	$28 \times 28$
10	Conv	conv4-1	256	$3 \times 3$	$52 \times 52$
11	Conv	conv4-2	128	$1 \times 1$	$52 \times 52$
12	Conv	conv4-3	256	$3 \times 3$	$52 \times 52$
13	Maxpool	pool4		$2 \times 2$	$26 \times 26$
14	Conv	conv5-1	512	$3 \times 3$	$26 \times 26$
15	Conv	conv5-2	256	$1 \times 1$	$26 \times 26$
16	Conv	conv5-3	512	$3 \times 3$	$26 \times 26$
17	Conv	conv5-4	256	$1 \times 1$	$26 \times 26$
18	Conv	conv5-5	512	$3 \times 3$	$26 \times 26$
19	Maxpool	pool5		$2 \times 2$	$13 \times 13$
20	Conv	conv6-1	1024	$3 \times 3$	$13 \times 13$
21	Conv	conv6-2	512	$1 \times 1$	$13 \times 13$
22	Conv	conv6-3	1024	$3 \times 3$	$13 \times 13$
23	Conv	conv6-4	512	$1 \times 1$	$13 \times 13$
24	Conv	conv6-5	1024	$3 \times 3$	$13 \times 13$
25	Conv	conv6-6	1024	$3 \times 3$	$13 \times 13$
26	Conv	conv6-7	1024	$3 \times 3$	$13 \times 13$
27	Reorganize conv5-5 to 2048 filters				
28	Concatenate above with conv6-7				
29	Conv	conv7-1	1024	$3 \times 3$	$13 \times 13$
30	Conv (Linear)	conv7-2	425	$1 \times 1$	$13 \times 13$
31	Region	region			$13 \times 13$

YOLOv2 chủ yếu bao gồm các lớp thông thường cho CNN, nhưng có một vài lưu ý quan trọng cần phải chú ý.

Ở lớp 24, các kênh của đầu ra ‘conv5-3’ (từ lớp thứ 16) được xếp chồng lên nhau trên các kênh của đầu ra ‘conv6-5’ (từ lớp thứ 22). Hoạt động này được gọi là sự kết nối (concatenation). Nhưng hoạt động này đòi hỏi tất cả các kênh có không gian giống nhau. Trong trường hợp này không thể áp dụng được với ‘conv5-3’ và ‘conv6-5’. Không gian của ‘conv5-3’ là  $26 \times 26$ , trong khi không gian của ‘conv6-5’ là  $13 \times 13$ .

Để giải quyết vấn đề này, các kênh của ‘conv5-3’ được tổ chức lại ở lớp 23 sao cho không gian mới của ‘conv5-3’ là  $13 \times 13$ . Điều này có nghĩa là số lượng kênh của ‘conv5-3’ sẽ tăng từ 512 đến 2048. Do đó, phần kết nối sẽ có 3062 kênh.

Sự kết nối sau đó được theo sau bởi 2 lớp chập là lớp 25 và 26 trước khi các regions được dự đoán ở lớp 27. Phần ngoài của mạng là một mảng có không gian  $13 \times 13 \times 425$ .

Mỗi lớp chập được theo sau bởi một lớp batch normalization. Các lớp batch normalization lần lượt được kích hoạt bởi Leaky ReLU (được giới thiệu ở mục 3.2.1). Bias của mỗi lớp chập cũng được áp dụng sau lớp batch normalization tương ứng.

### Kiến trúc của YOLO9000

Kiến trúc của YOLO9000 [8] được thể hiện ở bảng 4.3, rất giống với YOLOv2. Các lớp giống hệt nhau giữa YOLOv2 và YOLO9000 cho tới lớp thứ 25, ‘conv6-6’. Đối với YOLO9000, lớp ‘conv6-6’ là lớp tích chập cuối cùng trước lớp region. YOLO 9000 hỗ trợ 9418 đối tượng khác nhau với 3 dự đoán cho mỗi ô, đó là lý do tại sao ‘conv6-6’ có 28269 kênh đầu ra ( $3 \times (9418 + 5) = 28269$ ).

Bảng 4.3 Kiến trúc của YOLO9000

#	Type	Name	Filters	Kernel	Output shape
1	Input	input			$224 \times 224$
2	Conv	conv1-1		$3 \times 3$	$416 \times 416$
3	Maxpool	pool1		$2 \times 2$	$208 \times 208$
4	Conv	conv2-1	3	$3 \times 3$	$208 \times 208$
5	Maxpool	pool2	32	$2 \times 2$	$104 \times 104$
6	Conv	conv3-1	64	$3 \times 3$	$104 \times 104$
7	Conv	conv3-2	128	$1 \times 1$	$104 \times 104$
8	Conv	conv3-3	64	$3 \times 3$	$104 \times 104$
9	Maxpool	pool3	128	$2 \times 2$	$28 \times 28$
10	Conv	conv4-1	256	$3 \times 3$	$52 \times 52$
11	Conv	conv4-2	128	$1 \times 1$	$52 \times 52$
12	Conv	conv4-3	256	$3 \times 3$	$52 \times 52$
13	Maxpool	pool4	512	$2 \times 2$	$26 \times 26$
14	Conv	conv5-1	256	$3 \times 3$	$26 \times 26$
15	Conv	conv5-2	512	$1 \times 1$	$26 \times 26$
16	Conv	conv5-3	256	$3 \times 3$	$26 \times 26$
17	Conv	conv5-4	512	$1 \times 1$	$26 \times 26$
18	Conv	conv5-5	1024	$3 \times 3$	$26 \times 26$
19	Maxpool	pool5	512	$2 \times 2$	$13 \times 13$
20	Conv	conv6-1	1024	$3 \times 3$	$13 \times 13$
21	Conv	conv6-2	512	$1 \times 1$	$13 \times 13$
22	Conv	conv6-3	1024	$3 \times 3$	$13 \times 13$
23	Conv	conv6-4	28269	$1 \times 1$	$13 \times 13$
24	Conv	conv6-5		$3 \times 3$	$13 \times 13$
25	Conv	conv6-6		$1 \times 1$	$13 \times 13$
26	Region	region			$13 \times 13$

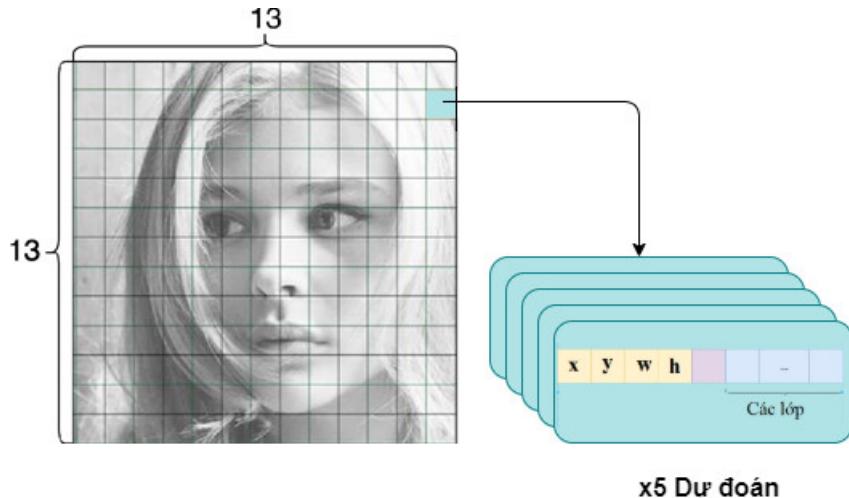
### Định dạng đầu vào của YOLO

Không giống như VGG16, không có phép trừ trung bình nào được thực hiện trên các hình ảnh đầu vào của các mô hình YOLO. Thay vào đó, đầu vào của các mô hình YOLOv2 được chuẩn hóa theo hệ số 1/255.

### Định dạng đầu ra của YOLOv2

Đầu ra của YOLOv2 có kích thước cố định, nó luôn luôn là một mảng có khong gian  $13 \times 13 \times 425$  cho dù có phát hiện bao nhiêu đối tượng. Đầu ra biểu

điển bởi một lưới  $13 \times 13$  của hình ảnh đầu vào, trong đó mỗi ô của lưới chứa 5 dự đoán bounding boxes. Điều này có nghĩa YOLO sẽ luôn dự đoán  $13 \times 13 \times 5 = 845$  bounding boxes.



Hình 4.1 Minh họa đầu ra của YOLOv2.

Hình ảnh được di chuyển vào lưới  $13 \times 13$ . Mỗi ô của lưới chứa 5 dự đoán bounding boxes với tọa độ tương ứng  $x, y, w, h$ , sự tự tin của dự đoán c và class scores.

Mỗi bounding boxes chứa 4 giá trị cho tọa độ của chúng trong một lưới, 1 giá trị cho độ tin cậy của dự đoán và vector class score. Ở YOLOv2, vì được huấn luyện trên bộ dữ liệu COCO, nên vector class score có 80 giá trị, một giá trị cho mỗi lớp. Điều này có nghĩa bounding box được biểu diễn bằng một vector gồm 85 giá trị. Giá trị cho độ tin cậy được sử dụng để loại trừ các dự đoán không chắc chắn hoặc dự đoán nền.

Vì trong mỗi ô lưới chứa 5 dự đoán, nên mỗi ô chứa  $85 \times 5 = 425$  giá trị.

### **Định dạng đầu ra của YOLO9000**

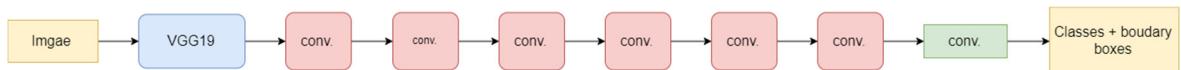
Định dạng đầu ra của YOLO9000 tương tự như YOLOv2. Điểm khác biệt duy nhất là cách điểm số của lớp được thể hiện. YOLO9000 sử dụng hệ thống phân cấp WordNet cho class scores. WordNet là một cơ sở dữ liệu ngôn ngữ

mô hình các kết nối giữa các từ liên quan trong biểu đồ có hướng. Ví dụ, trong WordNet, “hunting dog” là một hyponym của “dog”.

Class scores của YOLO9000 sử dụng các hyponyms của nhãn các lớp để mô hình hóa một hệ thống phân cấp của các lớp. Điều này cho phép YOLO9000 được huấn luyện trên cả COCO và ImageNet. Ví dụ, COCO có lớp mục tiêu “airplane”, trong khi ImageNet có lớp mục tiêu “jet”, “biplane”, “airbus” và “stealth fighter”, tất cả đều là hyponym của “máy bay”. Sử dụng hệ thống phân cấp hyponym, YOLO9000 có thể suy ra mối quan hệ giữa “airplane” và “jet”.

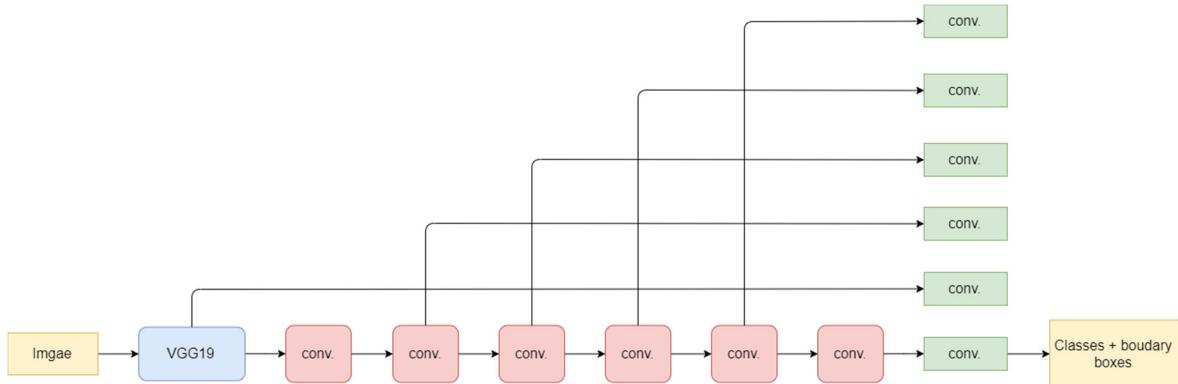
#### 4.2.2 SSD – Single Shot MultiBox Detector

SSD [9] là một hệ thống phát hiện đối tượng tiếp nối thành công của YOLO. SSD là mô hình single shot detector sử dụng mạng VGG19 để rút trích đặc trưng. Mô hình SSD được miêu tả trong hình 4.2. Trong đó, những convolution có màu hồng là những custom convolution layer (những lớp tích chập có thể điều chỉnh). Convolution filter layer (khối màu xanh lá cây) có nhiệm vụ tổng hợp các thông tin để đưa ra quyết định.



Hình 4.2 Mô hình SSD VGG19 model 1

Khi sử dụng mô hình trên, chúng ta thấy rằng các custom convolution layer có nhiệm vụ làm giảm chiều và giảm độ phân giải của bức ảnh. Cho nên, mô hình chỉ có khả năng nhận ra các đối tượng có kích thước lớn. Để giải quyết vấn đề này, chúng ta sẽ sử dụng các object detector khác nhau trên mỗi feature maps (xem output của mỗi custom convolution layer là một feature map).



Hình 4.3 Mô hình SSD VGG19

Tương tự như YOLO, SSD cũng được đào tạo trên bộ dữ liệu COCO. Do đó, SSD có thể dự đoán 80 loại đối tượng khác nhau. SSD cũng có một lớp được chỉ định cho nền. Lớp nền được sử dụng để nói rằng không có đối tượng nào được phát hiện tại một khu vực cụ thể. Do đó, SSD dự đoán class scores cho 81 lớp khác nhau cho mỗi detection box khác nhau.

### Kiến trúc mô hình SSD

SSD có kiến trúc phức tạp hơn một chút so với YOLO. Đầu ra của SSD liên quan đến dự đoán ở nhiều cấp độ khác nhau. Trong khi YOLO các detection boxes trong một lưới có không gian  $13 \times 13$ , SSD xuất ra các detection boxes ở các lưới có kích cỡ khác nhau.

Các detection boxes được dự đoán từ các feature của các lớp ‘conv4-3’, ‘fc7’, ‘conv6-2’, ‘conv7-2’, ‘conv8-2’, ‘conv9-2’, mỗi lớp đại diện cho các lưới khác nhau ở các cấp độ khác nhau cho các detection boxes (kích thước lưới lần lượt là  $38 \times 38$ ,  $19 \times 19$ ,  $10 \times 10$ ,  $5 \times 5$ , và  $3 \times 3$ ).

Mỗi ô của lưới chứa các bounding boxes và class scores của chúng. Các bounding boxes và class scores được dự đoán bởi các lớp chập được kết nối với từng mentioned layers. Mỗi lớp chập có kích thước kernel là  $3 \times 3$  với  $4 \times (4 + 81) = 340$  filters. (4 bounding boxes với 4 giá trị cho bounding boxes và 81 lớp).

Không như YOLO, SSD không thực hiện batch normalization sau mỗi lớp.

Kiến trúc chính của SSD được mô tả trong bảng 4.4.

Bảng 4.4 Kiến trúc chính của SSD

Type	Name	Filters	Kernel	Strides	Output shape
Input	input	3			$300 \times 300$
Conv	conv1-1	64	$3 \times 3$		$300 \times 300$
Conv	conv1-2	64	$3 \times 3$		$300 \times 300$
Maxpool	pool1		$2 \times 2$		$150 \times 150$
Conv	conv2-1	128	$3 \times 3$		$150 \times 150$
Conv	conv2-2	128	$3 \times 3$		$150 \times 150$
Maxpool	pool2		$2 \times 2$	$2 \times 2$	$75 \times 75$
Conv	conv3-1	256	$3 \times 3$		$75 \times 75$
Conv	conv3-2	256	$3 \times 3$		$75 \times 75$
Conv	conv3-3	256	$3 \times 3$		$75 \times 75$
Maxpool	pool3		$2 \times 2$	$2 \times 2$	$38 \times 38$
Conv	conv4-1	512	$3 \times 3$		$38 \times 38$
Conv	conv4-2	512	$3 \times 3$		$38 \times 38$
Conv	conv4-3	512	$3 \times 3$		$38 \times 38$
Maxpool	pool4		$2 \times 2$	$2 \times 2$	$19 \times 19$
Conv	conv5-1	512	$3 \times 3$		$19 \times 19$
Conv	conv5-2	512	$3 \times 3$		$19 \times 19$
Conv	conv5-3	512	$3 \times 3$		$19 \times 19$
Maxpool	pool5		$3 \times 3$	$1 \times 1$	$19 \times 19$
Conv	fc6*	1024	$3 \times 3$		$19 \times 19$
Conv	fc7	1024	$1 \times 1$		$19 \times 19$
Conv	conv6-1	256	$1 \times 1$		$19 \times 19$
Conv	conv6-2	512	$3 \times 3$	$2 \times 2$	$10 \times 10$
Conv	conv7-1	128	$1 \times 1$		$10 \times 10$
Conv	conv7-2	256	$3 \times 3$	$2 \times 2$	$5 \times 5$
Conv	conv8-1	128	$1 \times 1$		$5 \times 5$
Conv	conv8-2	256	$3 \times 3$	$2 \times 2$	$3 \times 3$
Conv	conv9-1	128	$1 \times 1$		$3 \times 3$
Conv	conv9-2	128	$3 \times 3$		$3 \times 3$

### Định dạng đầu vào của SSD

Định dạng đầu vào của SSD giống hệt VGG16. Các hình ảnh đầu vào được khử bởi ImageNet mean.

Các chi tiết của mô hình SSD có thể tìm hiểu thêm tại [9].

#### 4.2.3 Deep SORT

Ngoài việc phát hiện các đối tượng, thì còn một công việc cần được thực hiện để giải quyết bài toán nhận dạng hành động, đó là theo dõi các đối tượng trong Video. Điều này đòi hỏi chúng ta phải hợp nhất các kiến thức object detection với phân tích thông tin thời gian và sử dụng nó để dự đoán quỹ đạo tốt nhất. Phổ biến nhất và là một trong những phương pháp theo dõi đối tượng được sử dụng nhiều nhất là Deep SORT, một phần mở rộng của SORT (Simple Online and Realtime Tracking).

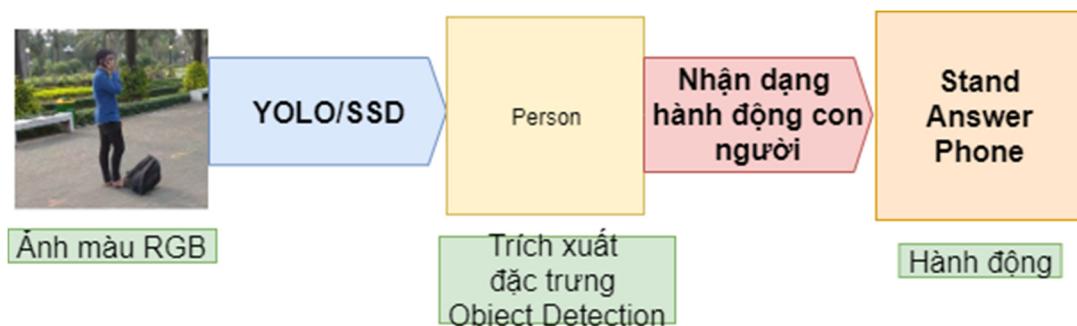
SORT sử dụng bộ lọc Kalman để tạo ra một “track” cho mỗi lần phát hiện đối tượng chứa các thông tin trạng thái cần thiết. Bộ lọc Kalman cũng chứa một tham số để theo dõi và xóa các “track” không xuất hiện trong một khoảng thời gian từ lần phát hiện cuối vì những đối tượng đó đã rời khỏi khung hình. Ngoài ra, để loại bỏ các “track” trùng lặp, có một ngưỡng phát hiện tối thiểu cho vài khung hình đầu tiên. Sau khi đã có bouding box từ bộ lọc Kalman, chúng ta cần liên kết các phát hiện mới với các dự đoán mới. Để thực hiện điều này, chúng ta cần một thước đo khoảng cách để định lượng liên kết và một thuật toán hiệu quả để liên kết dữ liệu. Với thước đo khoảng cách, SORT sử dụng khoảng cách Mahalanobis bình phương để kết hợp với các yếu tố không chắc chắn từ bộ lọc Kalman. Với thuật toán hiệu quả, SORT sử dụng thuật toán Hungarian tiêu chuẩn, là một thuật toán liên kết dữ liệu đơn giản.

Mặc dù đạt được hiệu suất tổng thể tốt về việc theo dõi chính xác đối tượng, SORT vẫn thất bại trong việc theo dõi nhiều trường hợp thực tế do nhiều nguyên nhân như do các góc nhìn khác nhau, sự thiếu ăn khớp giữa các khung

hình. Để khắc phục vấn đề này, một mạng CNN đã được đào tạo để phân biệt người đi bộ trên một bộ dữ liệu nhận dạng lại người quy mô lớn. Thông qua việc tích hợp mạng này, SORT đã được tăng cường mạnh mẽ chống lại các lỗi trong khi giữ cho hệ thống thực hiện dễ dàng, hiệu quả và có thể áp dụng cho các tình huống thời gian thực. Mô hình mới này được gọi là Deep SORT [10].

### 4.3 Luồng ngữ nghĩa (Semantic Stream)

Chúng tôi giới thiệu luồng ngữ nghĩa, một luồng dự đoán hành động của con người trong hình ảnh RGB tùy thuộc vào các phát hiện đối tượng dự đoán từ YOLO hay SSD. Tương tự như luồng không gian, đầu vào của luồng ngữ nghĩa là một ảnh RGB.



Hình 4.4 Luồng ngữ nghĩa.

Luồng ngữ nghĩa được chia làm 2 phần: nhận dạng đối tượng và nhận dạng hành động con người. Nhận dạng hành động con người được dựa trên các tính năng phát hiện đối tượng được dự đoán từ YOLO/SSD.

Một đặc tính tiện lợi với cả YOLO và SSD là kích thước của đầu ra được cố định, bất kể có bao nhiêu đối tượng được phát hiện. Điều này làm cho việc mở rộng các mô hình để nhận dạng hành động con người chỉ bằng cách thêm các lớp mới.

#### 4.3.1 Vị trí kết nối với YOLO

Có nhiều cách kết nối các lớp bổ sung của mạng YOLO cho luồng ngữ nghĩa. Điểm kết nối đơn giản nhất là lớp khu vực cuối cùng. Sau đó, đầu vào của luồng ngữ nghĩa sẽ được object detection dự đoán.

Một cách khác là kết nối luồng ngữ nghĩa ở các lớp trước đó. Đầu ra của các lớp trước đó có thể chứa nhiều thông tin hơn, ở mức cũ của một biểu diễn phức tạp hơn.

Đối với YOLOv2, chúng tôi xem xét 3 điểm kết nối khác nhau cho luồng ngữ nghĩa, lớp ‘region’, lớp ‘conv7-1’, và lớp ghép của lớp ‘conv5-5’ và ‘conv6-7’. Lý do tại sao chúng tôi xem xét việc kết hợp giữa ‘conv5-5’ và ‘conv6-7’ là vì đầu vào của lớp ‘conv7-1’ sử dụng đầu ra từ cả 2 lớp này.

### 4.3.2 Vị trí kết nối với SSD

Tương tự như YOLO, chúng tôi nghiên cứu các điểm kết nối khác nhau cho luồng ngữ nghĩa. Tuy nhiên, một điểm khác biệt lớp với SSD là việc nhận dạng đối tượng của SSD được dựa trên đầu ra của nhiều lớp ở nhiều cấp độ khác nhau.

Đầu vào của lớp SSDs ‘region’ được dựa trên các lớp ‘conv4-3’, ‘fc7’, ‘conv6-2’, ‘conv7-2’, ‘conv8-2’ và ‘conv9-2’. Vì vậy, bên cạnh lớp ‘region’, chúng tôi sẽ kết hợp tất cả các lớp được đề cập đến luồng ngữ nghĩa.

### 4.3.3 Kiến trúc của luồng ngữ nghĩa

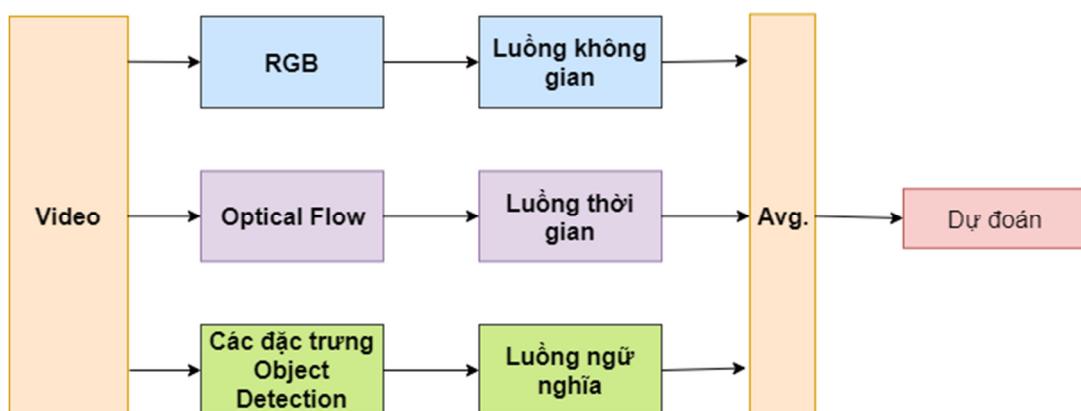
Chúng tôi xem xét một vài kiến trúc cho luồng ngữ nghĩa. Hình thức dễ nhất của luồng ngữ nghĩa chỉ là một lớp softmax. Một kết quả tốt trên lớp softmax chỉ ra rằng các đặt trưng của nhận dạng đối tượng có thể phân tách tuyến tính trên nhận dạng hành động con người.

Để tránh tình trạng overfitting, chúng tôi nhận ra tỷ lệ dropout khác nhau ảnh hưởng đến hiệu suất của luồng ngữ nghĩa. Chúng tôi cũng xem xét cách luồng ngữ nghĩa được hưởng lợi từ việc thêm các lớp ẩn bổ sung.

Các thông số của mạng YOLO/SSD bị đóng băng trong quá trình đào tạo. Điều này là để thực thi rằng luồng ngữ nghĩa nằm trong một không gian đặc trưng khác với luồng không gian và luồng thời gian.

#### 4.3.4 Sự kết hợp các luồng

Trong một số chức năng để kết hợp luồng không gian và luồng thời gian đã được nghiên cứu ở nhiều lớp liên hợp (fusion layers) khác nhau. Đưa đến kết luận rằng tổng của các lớp softmax là cách tiếp cận hiệu quả nhất. Chúng tôi theo dõi kết luận này và hợp nhất ba luồng (three stream) bằng cách tổng hợp các lớp softmax của chúng [11].



Hình 4.5 Kiến trúc ‘three stream’

#### 4.4 Tăng cường phân kỳ (Boosting Divergence)

Mục tiêu cuối cùng với luồng ngữ nghĩa là nó sẽ cung cấp thông tin tổng hợp cho các luồng không gian và thời gian. Luồng ngữ nghĩa không nhất thiết phải tự thực hiện tốt. Điều quan trọng hơn là nó tạo ra các loại “error” khác nhau đối với luồng không gian và thời gian.

Do đó, chúng tôi sẽ nghiên cứu một số cách để thực thi các dự đoán phân kỳ (diverge) lẫn nhau.

Tạo ra các mô hình mới để bổ sung cho các mô hình khác không phải là mới. Nổi tiếng nhất là AdaBoost, một kỹ thuật tăng cường cho các phân loại yếu.

AdaBoost lặp đi lặp lại tạo các mô hình mới tập trung để bổ sung cho các lần lặp trước của mô hình.

Để nhận dạng giọng nói, nghiên cứu Minimum Bayes Risk Level-aging để huấn luyện các mô hình bổ sung. Kỹ thuật này đã tạo thành công với các mô hình bổ sung và giảm tỷ lệ lỗi so với các mô hình được tạo riêng lẻ.

Chúng tôi sẽ nghiên cứu một số kỹ thuật thay thế để tăng cường sự khác biệt giữa các luồng. Những kỹ thuật này chỉ thêm một thuật ngữ mới cho hàm mất mát.

#### 4.4.1 Average Layer

Cách tiếp cận khả thi nhất là huấn luyện luồng ngữ nghĩa cùng với luồng thời gian và luồng không gian.

$$\mathcal{L}_{average}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) = \mathcal{L}_{cross}\left(\frac{w_o\tilde{\mathbf{y}}_{sem} + w_1\tilde{\mathbf{y}}_{sem} + w_2\tilde{\mathbf{y}}_{sem}}{w_o + w_1 + w_2}, \mathbf{y}\right) \quad (4.4.1)$$

Trọng số  $w_o, w_1, w_2$  được sử dụng trọng số trung bình của các luồng. Cả luồng không gian và luồng thời gian đều hoạt động quá tốt trên tập huấn luyện và do đó làm lu mờ dự đoán của luồng ngữ nghĩa.

Do đó, chúng tôi đặt  $w_o = 1$  và  $w_1 = w_2 = 0,25$ . Điều này mang lại cho dòng ngữ nghĩa ảnh hưởng nhiều hơn, ngay cả khi các dự đoán của luồng không gian và luồng thời gian quá tự tin vào true value.

Ý tưởng với kỹ thuật này là mô phỏng xác nhận mô hình. Luồng ngữ nghĩa có thể tập trung ít chính xác hơn trong trường hợp cả luồng không gian và luồng thời gian đều tự tin vào giá trị thực và tập trung chính xác hơn trong trường hợp luồng không gian và luồng thời gian không tự tin.

#### 4.4.2 Kullback-Leibler Divergence

Một cách tiếp cận khác là sử dụng phép đo phân kỳ để thực thi phân kỳ giữa các luồng dự đoán. Một phép đo phân kỳ phổ biến cho các phân phối xác suất

là Kullback-Leibler divergence  $KL$  (khoảng cách  $KL$ ), được xác định cho các phân phối xác suất rời rạc như:

$$KL(\mathbf{p} \parallel \mathbf{q}) = \sum_i p_i \log \frac{p_i}{q_i} \quad (4.4.2)$$

Trong đó,  $\mathbf{p}$  và  $\mathbf{q}$  là hai phân phối xác suất rời rạc (có cùng kích thước).  $KL(\mathbf{p} \parallel \mathbf{q})$  được phát triển khi  $p$  và  $q$  phân kỳ và  $KL(\mathbf{p} \parallel \mathbf{q}) = 0$  khi  $\mathbf{p} = \mathbf{q}$ . Lưu ý rằng  $KL(\mathbf{p} \parallel \mathbf{q}) \neq KL(\mathbf{q} \parallel \mathbf{p})$ .

Chúng ta xác định hàm mất mát  $\mathcal{L}_{KL}$  là:

$$\mathcal{L}_{KL}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) = \mathcal{L}_{cross}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) - \lambda_{KL} KL\left(\tilde{\mathbf{y}}_{sem}, \frac{\tilde{\mathbf{y}}_{spa} + \tilde{\mathbf{y}}_{tem}}{2}\right) \quad (4.4.3)$$

Trong đó,  $\lambda_{KL}$  là một yếu tố mở rộng kiểm soát Kullback-Leiber bao nhiêu phân kỳ ảnh hưởng đến sự mất mát (loss).

#### 4.4.3 Dot Product Divergence

Một cách tiếp cận hình học hơn là giảm thiểu tích vô hướng giữa các phân kỳ dự đoán. Nếu các dự đoán được xem trong một không gian vector, thì tích vô hướng được tối giảm khi các vector là trực giao. Một cách giải thích hình học là hai xác suất khác nhau khi góc của chúng lớn bằng không gian vector phân phối xác suất. Chúng ta xác định hàm mất mát  $\mathcal{L}_{dot}$ :

$$\mathcal{L}_{dot}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) = \mathcal{L}_{cross}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) - \lambda_{dot} \tilde{\mathbf{y}}_{sem} \bullet (\tilde{\mathbf{y}}_{spa} + \tilde{\mathbf{y}}_{tem}) \quad (4.4.4)$$

Trong đó  $\lambda_{dot}$  là một yếu tố tỷ lệ kiểm soát bao nhiêu tích vô hướng ảnh hưởng đến sự mất mát.

#### 4.4.4 Combining Average with Divergence

Chúng tôi cũng xem xét hai hỗn hợp của các kỹ thuật trên – cho  $\mathcal{L}_{dot}$  và  $\mathcal{L}_{KL}$ , chúng tôi thay thế  $\mathcal{L}_{cross}$  bằng  $\mathcal{L}_{arverage}$ . Chúng tôi xác định  $\mathcal{L}_{arverage+dot}$  và  $\mathcal{L}_{arverage+KL}$  như sau:

$$\mathcal{L}_{arverage+dot}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) = \mathcal{L}_{arverage}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) - \lambda_{dot} \tilde{\mathbf{y}}_{sem} \bullet (\tilde{\mathbf{y}}_{spa} + \tilde{\mathbf{y}}_{tem}) \quad (4.4.5)$$

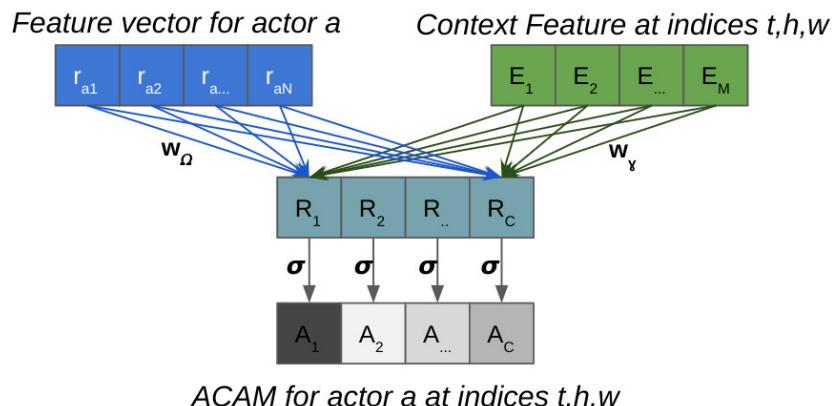
Và

$$\mathcal{L}_{arverage+KL}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) = \mathcal{L}_{arverage}(\tilde{\mathbf{y}}_{sem}, \mathbf{y}) - \lambda_{KL} KL\left(\tilde{\mathbf{y}}_{sem}, \frac{\tilde{\mathbf{y}}_{spa} + \tilde{\mathbf{y}}_{tem}}{2}\right) \quad (4.4.6)$$

## 4.5 Actor conditioned actor maps (ACAM)

### 4.5.1 Giới thiệu về phương pháp ACAM

Tương tự như các model attention trong xử lý ngôn ngữ tự nhiên và thị giác máy tính, phương pháp giới thiệu tạo ra một tập các trọng số (ACAMs) thể hiện sự chú ý của các thành phần khác nhau của media. Không giống các mô hình khác, việc phát hiện hành động của ACAM chứa nhiều actor thực hiện các hành động đồng thời có thể liên quan hoặc khác nhau. Điều này tạo ra một vấn đề “chú ý”, trong đó các actors khác nhau có sự liên quan khác nhau đến các vị trí trong không gian của bối cảnh.



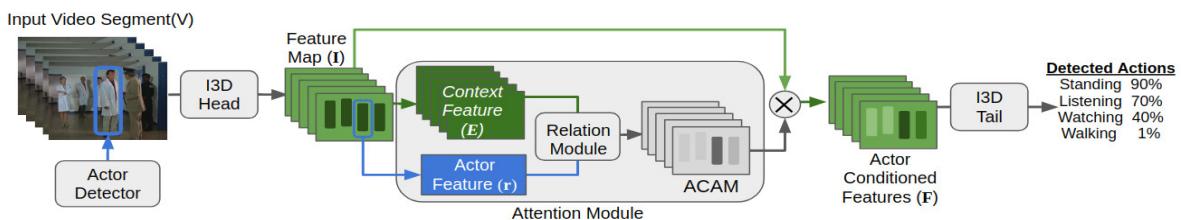
Hình 4.6 Attention module cho actor  $a$  tại một chỉ số không-thời gian duy nhất  $t, h, w$ . Trọng số chú ý duy nhất trong ACAM tại  $t, h, w$  được tạo từ đặc trưng actor  $\mathbf{r}_a$  và đặc trưng bối cảnh tại cùng chỉ số  $\mathbf{E}_{t,h,w}$  [12]

Phương pháp ACAM giải quyết các vấn đề “chú ý” bằng cách tạo ACAMs, trong đó nắm bắt mối quan hệ giữa các actor và bối cảnh. Thay vì mở rộng RoIPooling để phát hiện hành động, bản chất của ACAM là thực hiện điều

kiện các đặc trưng được trích xuất từ toàn bộ bối cảnh trên mỗi actor và động lực hành động.

#### 4.5.2 Kiến trúc ACAM

Các phân đoạn video đầu vào được xử lý bởi I3D black-bone. Các vector đặc trưng cho mỗi actor được phát hiện được tạo từ các vị trí của chúng trên bản đồ đặc trưng. Một tập hợp các trọng số được tạo cho mọi khu vực thời gian trong cảnh bằng cách kết hợp các tính năng actors và các tính năng theo ngữ cảnh được trích xuất từ toàn bộ cảnh. Các trọng số này, tức là attention map, được nhân với features map và kết quả đại diện cho các tính năng có điều kiện của actors. Bốn actors được phát hiện được đại diện bởi bốn thanh dọc trong I. Một actors tập trung (đóng hộp) đang lắng nghe một actors thân thiết. Hành động này được ghi lại bởi các trọng số lớn hơn trong attention map được hiển thị dưới dạng thanh dọc tối hơn.



Hình 4.7 Kiến trúc của ACAM. [12]

Phương pháp ACAM được viết chi tiết hơn tại [12].

#### 4.6 Mạng tích chập 3 chiều (3D CNN)

CNN là một loại mô hình sâu có thể hoạt động trực tiếp trên các đầu vào thô, do đó tự động hóa quá trình xây dựng các đặc trưng. Tuy nhiên, các mô hình như vậy hiện bị giới hạn để xử lí các đầu vào 2 chiều (2D). Trong phần này, chúng tôi sẽ nói đến mô hình CNN 3D mới để nhận dạng hành động. Mô hình này trích xuất các tính năng từ cả hai chiều không gian và thời gian bằng cách thực hiện các cấu trúc 3D, do đó thu được thông tin chuyển động được mã hóa trong nhiều frame liền kề. Mô hình được phát triển tạo ra nhiều

kênh thông tin từ các frame đầu vào và biểu diễn tính năng đầu ra có được bằng cách kết hợp thông tin từ tất cả các kênh.

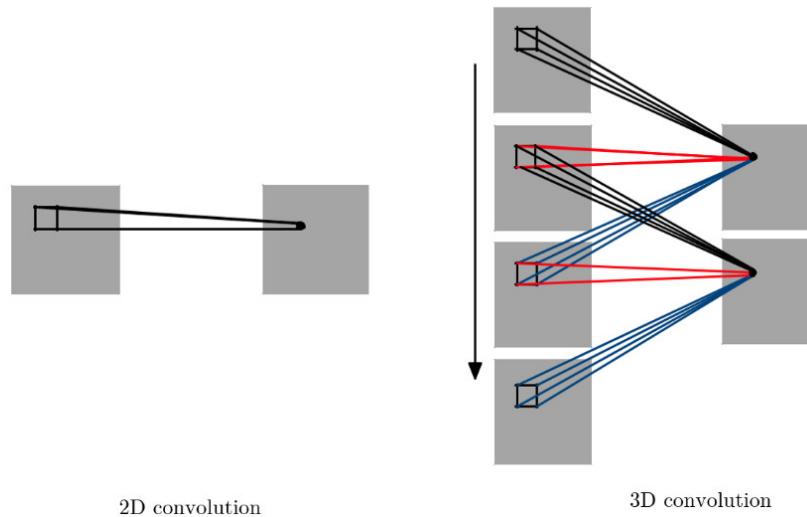
Phần 3.2 đã nêu ra các định nghĩa cơ bản về tích chập, ở đây chúng tôi sẽ nêu cụ thể về phần tích chập 3 chiều (3D CNN).

Trong 2D CNN (mạng tích chập 2 chiều), các cấu trúc được áp dụng trên feature map 2D để tính toán các đặc trưng chỉ từ các chiều không gian. Khi áp dụng cho các vấn đề phân tích video, đó là mong muốn nắm bắt được thông tin chuyển động mã hóa trong nhiều frame liền kề. Để làm được điều đó, cấu trúc 3D CNN được đề xuất thực hiện trong các giai đoạn tính tích chập của CNN để tính toán các đặc trưng từ cả hai chiều không gian và thời gian. Tích chập 3D (3D CNN) đạt được bằng cách kết hợp 3D kernel với khối (cube) được hình thành bằng các xếp chồng nhiều frame liền kề nhau. Bằng cách xây dựng này, các feature map trong lớp chập được kết nối với nhiều frame liền kề trong lớp trước, do đó nắm bắt được thông tin chuyển động. Để rõ hơn, giá trị tại vị trí  $(x,y,z)$  trên feature map thứ  $j$  trong lớp thứ  $I$  được cho như sau:

$$v_{ij}^{xyz} = \tanh(b_{ij} + \sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}) \quad (4.6.1)$$

Với  $R_i$  là kích thước của 3D kernel theo chiều thời gian,  $w$  là giá trị thứ  $(p,q,r)$  của kernel được kết nối với feature map thứ  $m$  trong lớp trước đó.

Sự so sánh của tích chập 2D và 3D được cho ở hình 4.8.



*Hình 4.8 So sánh giữa 2D convolution và 3D convolution [13]*

Lưu ý rằng một kernel 3D tích chập chỉ có thể trích xuất một loại đặc trưng từ khối khung (frame cube), vì trong số kernel được sao chép trên toàn bộ khối (cube). Nguyên tắc thiết kế chung của CNN là số lượng feature map nên được tăng lên ở các lớp sau bằng cách tạo nhiều loại đặc trưng từ cùng một bộ feature map cấp thấp hơn (lower-level). Tương tự như trường hợp tích chập 2D, điều này có thể đạt được bằng cách áp dụng nhiều tích chập 3D với các kernel riêng biệt cho cùng một vị trí ở lớp trước đó.

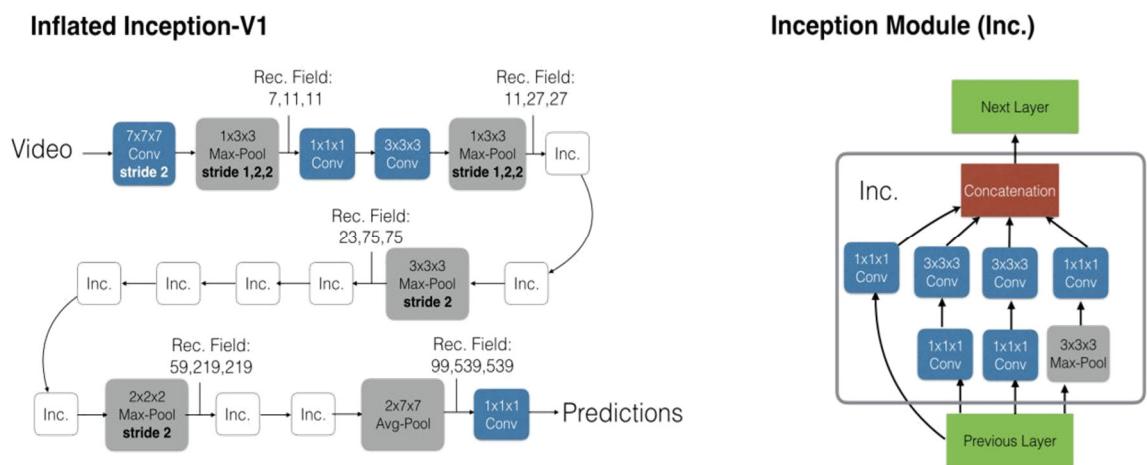
### Thỗi phồng từ 2D thành 3D

Một số kiến trúc mạng phân loại ảnh rất thành công đã được phát triển trong những năm qua. Thay vì lặp lại quy trình cho các mô hình không gian-thời gian (2 luồng), những đề xuất chuyển đổi các mô hình phân loại ảnh (2D) thành các mô hình tích chập 3D đã thành công. Điều này được thực hiện bằng cách bắt đầu với kiến trúc 2D, và thỗi phồng tất cả các filter và pooling kernel – kết hợp chúng với một chiều thời gian bổ sung. Các filter thường là hình vuông và chúng ta biến chúng thành các khối lập phương (cubic) –  $N \times N$  trở thành  $N \times N \times N$ .

### Inception3D

Với các mạng thông thường, khi thiết kế ta bắt buộc phải xác định trước các tham số của 1 lớp chập như kích thước kernel, padding, strides,... Và thường thì rất khó để nói trước được tham số nào là phù hợp, kích thước kernel là  $1 \times 1$ ,  $3 \times 3$ ,  $5 \times 5$ , ... sẽ tốt hơn. Mạng Inception ra đời để giải quyết vấn đề đó, yếu tố quan trọng nhất trong mạng Inception là Inception module, một mạng Inception hoàn chỉnh bao gồm nhiều module Inception nhỏ ghép lại với nhau. Ý tưởng của Inception module rất đơn giản, thay vì sử dụng 1 lớp chập với tham số kích thước kernel cố định, ta hoàn toàn có thể sử dụng cùng lúc nhiều lớp chập với kích thước kernel khác nhau ( $1, 3, 5, 7, \dots$ ) và sau đó concatenate các đầu ra lại với nhau. Để không bị lỗi về chiều của ma trận khi concatenate, tất cả các lớp chập đều có chung strides =  $(1, 1)$  và padding = ‘same’. Ở thời điểm này, có 3 phiên bản của mạng Inception, các phiên bản sau thường có vài điểm cải tiến so với phiên bản trước để cải thiện độ chính xác.

Như đã nói ở trên, một mạng 2D có thể được thổi phồng thành mạng 3D bằng cách kết hợp chúng với miền thời gian bổ sung, mạng Inception cũng vậy, các module Inception 2D được mô hình hóa thành các Inception3D như trong hình 4.9.



Hình 4.9 Mô hình cấu trúc mạng Inflated Inception-V1 (bên trái) và chi tiết của module Inception (bên phải) [14]

Trong hình 4.9, stride của tích tích chập và pooling là 1, các lớp batch normalizationm, ReLU và softmax ở cuối cùng không được hiển thị.

Chi tiết của mạng tích chập 3 chiều có thể được tìm thấy tại [13] [14].

## 4.7 Bộ dữ liệu AVA

Bộ dữ liệu chúng tôi sử dụng trong luận văn này để huấn luyện cho mô hình của chúng tôi là bộ dữ liệu Atomic Visual Actions (AVA) được phát triển bởi Google. Bộ dữ liệu AVA phân loại 80 atomic actions trong 430 video clip (AVAv2.1), trong đó các hành động được định vị hóa trong không gian và thời gian trong khoảng thời gian 15p. Cho kết quả 1.58 triệu nhãn hành động với nhiều hành động được chú thích cho từng người trong mỗi video clip.



Hình 4.10 Bộ dữ liệu AVA của Google

Các đặc điểm chính của bộ dữ liệu AVA là:

- Định nghĩa về atomic visual actions, thay vì hành động hỗn hợp.
- Chú thích chính xác về không-thời gian với khả năng dán nhãn nhiều hành động cho từng người.
- Chú thích đầy đủ của các atomic actions trên video clips độ dài 15p.
- Liên kết thời gian trên các phân đoạn liên tiếp cho các actors.
- Sử dụng phim trên Youtube để thu thập dữ liệu hành động.

Các đặc điểm trên cho phép bộ dữ liệu có thể nhận dạng hành động theo thời gian, cung cấp các nhãn cho các hành động hỗn hợp trong các video clip ngắn. Bộ dữ liệu AVA được công khai tại: [15].

### 4.7.1 Định nghĩa về Atomic Action

Baker & Wright đã lưu ý đến tính chất phân cấp của hành động trong nghiên cứu kinh điển “tập hành vi” (behavior episodes), nghiên cứu của họ về hành vi con người trong cuộc sống thường ngày của người dân ở một thị trấn nhỏ ở Kansas. Ở cấp độ tốt nhất, các hành động phù hợp với các chuyển động của cơ thể hoặc điều khiển đối tượng, nhưng ở các cấp độ “thô”, các mô tả tự nhiên nhất là về mặt chủ ý và hành vi hướng đến mục tiêu của các actors.

Hệ thống phân cấp này làm cho việc xác định từ vựng các “nhãn” cho hành động ít được đặc ra. Điều này dẫn đến sự tiến bộ chậm hơn ở lĩnh vực nhận dạng hành động so với nhận dạng đối tượng. Liệt kê đầy đủ các hành vi cấp cao là không thực tế, tuy nhiên, nếu chúng ta giới hạn hành động trong một phạm vi thời gian tốt, thì các hành động sẽ được thể hiện một cách tự nhiên hơn và có dấu hiệu trực quan rõ ràng. Bộ dữ liệu AVA được chú thích các khung hình chính ở tần số 1 Hz, vì ở tần số này, các khung hình đủ dày đặc (dense) để nắm bắt nội dung ngữ nghĩa hoàn chỉnh của hành động. Trong khi đó cho phép chúng ta tránh được việc yêu cầu chú thích thời gian chính xác một cách phi thực tế các ranh giới giữa hành động.

Thứ thánh THUMOS đã quan sát và thấy rằng các ranh giới giữa các hành động (không giống như các đối tượng) là rất mờ nhạt, dẫn đến sự bất đồng giữa những người chú thích. Ngược lại, các chú thích có thể dễ dàng xác định (sử dụng  $\pm 1.5$ s ngữ cảnh) cho dù một khung có chứa một hành động nhất định hay không. Trên thực tế, AVA định vị hóa các điểm bắt đầu và kết thúc hành động với độ chính xác chấp nhận được là  $\pm 0,5$ s .

### 4.7.2 Chuỗi thời gian cho từng người hành động

Trong khi các sự kiện như cây đổ không liên quan đến con người, thì chúng ta tập trung vào các hoạt động của mọi người, được coi là tác nhân duy nhất. Ví dụ: có nhiều người như trong bóng đá hoặc chỉ có hai người ôm nhau, nhưng mỗi người là một tác nhân với những lực chọn riêng biệt, vì vậy AVA làm việc với từng người một cách riêng biệt. Các nhãn hành động được gán cho một người theo thời gian là một nguồn dữ liệu phong phú cho việc mô hình hóa thời gian.

### 4.7.3 Chú thích cho bộ dữ liệu

Lý tưởng nhất là AVA có được dữ liệu cho các hành động một cách tự nhiên, nhưng chúng ta không có được điều này. Mặc dù vậy, phim ảnh được xem là gần đúng với những gì bộ dữ liệu AVA mong muốn. Đặc biệt khi chúng ta xem xét sự đa dạng của thể loại và quốc gia và ngành công nghiệp điện ảnh rất phát triển. Chúng ta mong đợi một số bias trong quá trình này. Các cảnh phim cần phải “hấp dẫn” và có một ngữ pháp của “ngôn ngữ phim” được truyền đạt thông qua các cảnh quay. Điều này nói lên rằng, ở mỗi cảnh quay, chúng ta có thể mong đợi một chuỗi hành động của con người diễn ra. Các hành động mà phần nào thể hiện được thực tế, được truyền đạt bởi các diễn viên tài năng. AVA bổ sung cho các bộ dữ liệu hiện tại nguồn từ video do người dùng tạo để chứa nhiều hoạt động hơn, phù hợp với nhiều ngữ cảnh đa dạng.

### 4.7.4 Ghi nhãn hành động toàn diện

AVA dán nhãn tất cả các hành động của tất cả mọi người trong các khung hình chính. Điều này tự nhiên sẽ dẫn đến một loại mất căng bằng Zipf's law<sup>4</sup> giữa các loại hành động. Sẽ có nhiều ví dụ hơn về các hành động thường xuyên như đứng hoặc ngồi hơn là các hành động ít xảy ra như nhảy. Các mô hình nhận dạng cần phải hoạt động trên các bản phân phối hành động thực tế thay vì sử dụng các bộ dữ liệu cân bằng giả tạo. Một hậu quả khác của

---

<sup>4</sup> [https://en.wikipedia.org/wiki/Zipf%27s\\_law](https://en.wikipedia.org/wiki/Zipf%27s_law)

giao thức là không truy xuất các ví dụ về các loại hành động bằng cách truy vấn rõ ràng cái tài nguyên video trên internet, chúng tôi tránh một loại sai lệch nhất định: mở cửa là một hành động phổ biến xảy ra thường xuyên trong các đoạn phim, tuy nhiên, một hành động mở cửa đã được gắn thẻ như vậy trên Youtube có khả năng gây “chú ý” theo cách không tự nhiên.

Chi tiết về bộ dữ liệu AVA có thể tìm thấy tại [16].

### 4.8 Kết luận chương 4

Chương 4 đã xây dựng hoàn chỉnh lý thuyết cho một hệ thống nhận dạng hành động con người. Chương trình này bày chi tiết về:

- Mạng tích chập 2 luồng và luồng ngữ nghĩa. Mạng 2 luồng kết hợp thông tin thời gian và không gian để tăng cường độ chính xác cho việc nhận dạng hành động. Luồng ngữ nghĩa có 2 phần, 1 phần nhận dạng đối tượng và 1 phần nhận dạng hành động.
- Các hệ thống nhận dạng đối tượng YOLO và SSD và Deep Sort. Chúng tôi sử dụng SSD cho việc phát hiện đối tượng (ở đây là con người) và sử dụng Deep Sort để theo dõi các đối tượng đó.
- Sự tăng cường phân kỳ. Đây là phương pháp được chúng tôi sử dụng để giảm tỷ lệ lỗi khi kết hợp giữa luồng ngữ nghĩa và luồng không-thời gian.
- Phương pháp ACAM là phương pháp nhận dạng hành động được sử dụng trong luận văn lần này. Bản chất của phương pháp là trích xuất đặc trưng từ toàn bộ bối cảnh để tạo điều kiện cho việc nhận dạng hành động
- Mạng 3D CNN, là một mạng neuron tích chập được sử dụng để trích xuất đặc trưng từ luồng không gian và luồng thời gian bằng cách thực hiện các cấu trúc 3D

## CHƯƠNG 4. NHẬN DẠNG HÀNH ĐỘNG CON NGƯỜI

---

- Giới thiệu về bộ dữ liệu AVA được chúng tôi sử dụng, đây là một bộ dữ liệu mới được Google phát triển trong vài năm gần đây cho mục đích nhận dạng hành động.

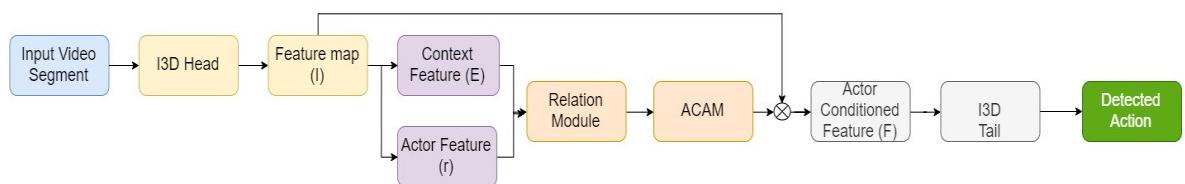
# CHƯƠNG 5

## XÂY DỰNG MÔ HÌNH NHẬN DẠNG HÀNH ĐỘNG

Trong chương 5, sử dụng các kiến thức của chương 3 và chương 4, chúng tôi sẽ trình bày công việc thực hiện việc xây dựng một mô hình nhận dạng hành động của con người. mục 5.1 sẽ trình bày nguyên lý hoạt động, các mục sau sẽ nêu chi tiết thực hiện để huấn luyện mô hình nhận dạng hành động của chúng tôi.

### 5.1 Kiến trúc hệ thống

Mô hình huấn luyện có đầu vào là các phân đoạn video 3s được xử lý bởi I3D back-bone. Các vector đặc trưng cho mỗi actor phát hiện được tạo ra từ các vị trí của chúng trên bản đồ đặc trưng. Một tập hợp các trọng số được tạo cho mọi khu vực thời gian trong ngữ cảnh bằng cách kết hợp các đặc trưng actor và các đặc trưng theo ngữ cảnh được trích xuất từ toàn bộ cảnh. Các trọng số này, tức là các attention map, được nhân với bản đồ đặc trưng và kết quả này đại diện cho các đặc trưng có điều kiện của các actors.



Hình 5.1 Kiến trúc model nhận dạng hành động

Những hành động được nhận dạng trong luận văn

Bộ dữ liệu AVA được sử dụng có khả năng huấn luyện để nhận dạng 80 hành động khác nhau. Tuy nhiên, chúng tôi chỉ quan tâm đến 10 hành động trong số 80 hành động đó, các hành động được chúng tôi tập trung trong luận văn là:

- |                      |                                      |
|----------------------|--------------------------------------|
| 1. Đứng – Stand      | 6. Nằm – Lie                         |
| 2. Ngồi – Sit        | 7. Cầm/nắm – Carry/Hold              |
| 3. Nói chuyện – Talk | 8. Trả lời điện thoại – Answer Phone |
| 4. Quỳ – Knee        | 9. Đánh nhau - Fight                 |
| 5. Đi bộ – Walk      | 10. Té ngã – Fall Down               |

## 5.2 Huấn luyện mô hình

Từ những kiến thức đã tìm hiểu được, chúng tôi tiến hành việc huấn luyện mô hình cho việc nhận dạng đối tượng, quá trình huấn luyện thực hiện theo các bước được trình bày dưới đây.

### 5.2.1 Chuẩn bị dữ liệu huấn luyện

Để huấn luyện một mô hình DL thì đầu tiên chúng ta cần chuẩn bị dữ liệu huấn luyện. Bộ dữ liệu được chúng tôi sử dụng là bộ dữ liệu AVA đã được chúng tôi giới thiệu ở mục 4.7.

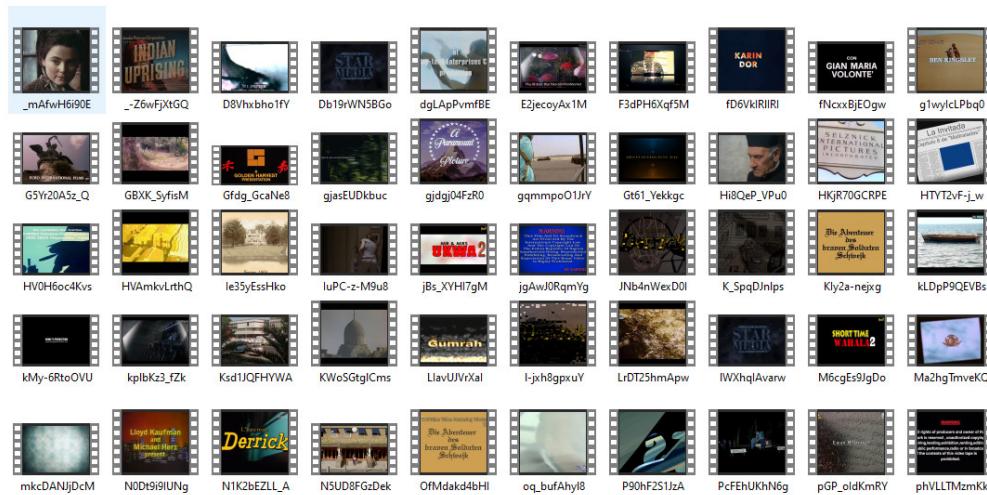
Bộ dữ liệu AVA sử dụng các video clip có sẵn trên Youtube. Chúng ta cần tải xuống các video clip từ Youtube để thực hiện việc huấn luyện. Bên cạnh đó, một video có thời lượng từ 40 – 120p, và có khoảng 200 video. Nên quá trình tải xuống bộ dữ liệu này tốn khá nhiều thời gian. Google không cung cấp các công cụ để thực hiện việc tải xuống bộ dữ liệu này mà chỉ cho người dùng youtubeid của các video clip. Vì vậy, để tải xuống bộ dữ liệu, chúng tôi sử dụng module “youtube-dl” và dựa vào danh sách youtubeid được cung cấp để tiến hành tải xuống bộ dữ liệu. Rất tiếc là một số video không còn có sẵn trên Youtube và một số video bị chặn ở Việt Nam. Dù vậy, số lượng dữ liệu vẫn là rất lớn cho việc huấn luyện.

## CHƯƠNG 5. XÂY DỰNG MÔ HÌNH NHẬN DẠNG HÀNH ĐỘNG

```
[ 'GBXK_SyfisM;1457;0.339;0.257;0.683;0.777;17;493' ]
[youtube] GBXK_SyfisM: Downloading webpage
[youtube] GBXK_SyfisM: Downloading video info webpage
[youtube] GBXK_SyfisM: Downloading js player vfl7Ksmll
[youtube] GBXK_SyfisM: Downloading js player vfl7Ksmll
[WARNING: Requested formats are incompatible for merge and will be merged into mkv.
[download] Destination: outdir/GBXK_SyfisM;1457;0.339;0.257;0.683;0.777;17;493.f135.mp4
[download] 5.2% of 202.30MiB at 1.32MiB/s ETA 02:24 ]
```

Hình 5.2 Tải xuống bộ dữ liệu A VA

Video sau khi tải xuống sẽ có định dạng “mkv” và “mp4”. Chúng tôi sử dụng tổng cộng 190 video, chia ra cho tập huấn luyện và validation.



Hình 5.3 Bộ dữ liệu A VA sau khi tải về.

Vì một video có dung lượng và thời lượng rất lớn, trong khi bộ dữ liệu A VA chỉ được chú thích trong khoảng thời gian 15p của video. Nên để chuẩn bị cho việc huấn luyện, chúng tôi cắt nhỏ các videos ra thành những video ngắn có thời lượng là 3s. Một video được cắt từ giây 899 – 1802 tính từ khi bắt đầu video, vì bộ dữ liệu A VA bắt đầu chú thích hành động trong khoảng thời gian này. Để thực hiện việc cắt video, chúng tôi sử dụng module “ffmpeg”. Sau đó chuyển tất cả các video về định dạng “mp4”, vì để 2 định dạng khác nhau có thể gây ra lỗi cho quá trình huấn luyện.

## CHƯƠNG 5. XÂY DỰNG MÔ HÌNH NHẬN DẠNG HÀNH ĐỘNG

```
Input #0, matroska,webm, from 'train/rJKeqfTlAeY.mkv':
  Metadata:
    COMPATIBLE_BRANDS: iso6avc1mp41
    MAJOR_BRAND      : dash
    MINOR_VERSION   : 0
    ENCODER         : Lavf58.20.100
  Duration: 01:27:59.98, start: -0.007000, bitrate: 1659 kb/s
  Stream #0:0: Video: h264 (High), yuv420p(tv, bt709, progressive), 1920x1080 [SAR 1:1 DAR
  Metadata:
    HANDLER_NAME    : ISO Media file produced by Google Inc. Created on: 02/16/2019.
    DURATION        : 01:27:59.960000000
  Stream #0:1(eng): Audio: opus, 48000 Hz, stereo, fltp (default)
  Metadata:
    DURATION        : 01:27:59.981000000
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> h264 (libx264))
  Stream #0:1 -> #0:1 (copy)
Press [q] to stop, [?] for help
[libx264 @ 0x56531fdcd740] using SAR=1/1
[libx264 @ 0x56531fdcd740] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 A
[libx264 @ 0x56531fdcd740] profile High, level 4.0
[libx264 @ 0x56531fdcd740] 264 - core 152 - H.264/MPEG-4 AVC codec - Copyleft 2003-2017 - ht
x113 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x8c
eads=2 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bfra
=250 keyint_min=25 scenecut=40 intra_refresh=0 rc_lookahead=40 rc=crf mbtree=1 crf=23.0 qcomp
[mp4 @ 0x56531fdcae40] track 1: codec frame size is not set
Output #0, mp4, to 'train_ver2/rJKeqfTlAeY_1586_1589.mp4':
```

Hình 5.4 Quá trình cắt nhỏ video.

Video sau khi được cắt ra sẽ có tên như sau: “youtubeid\_timestart\_timend”.



Hình 5.5 Video sau khi được xử lý

Sau khi quá trình tiền xử lý hoàn tất, video sẽ được tải lên Google Drive, vì chúng tôi sẽ tiến hành việc huấn luyện bằng dịch vụ đám mây Google Colab, chứ không sử dụng máy tính cá nhân.

### 5.2.2 Chú thích cho dữ liệu

Các chú thích cho bộ dữ liệu AVA được Google cung cấp trong file csv. File csv có định dạng gồm: mỗi hàng chứa “một chú thích” cho “một người” thực

hiện “một hành động” trong “một khoảng thời gian”. Trong đó, chú thích được liên kết với middle frame. Những người khác nhau và nhiều nhãn hành động khác nhau được mô tả trong các hàng riêng biệt. Định dạng của một hàng như sau: video\_id, middle\_frame\_timestamps, person\_box, action\_id, person\_id. Ý nghĩa của từng cột như sau:

- Video\_id: Mã định danh của Youtube.
- Middle\_frame\_timestamps: số giây tính từ khi bắt đầu video.
- Person\_box: trên cùng bên trái (x1,y1) và dưới cùng bên phải (x2,y2) được chuẩn hóa theo kích thước khung hình. Trong đó, (0.0, 0.0) tương ứng với trên cùng bên trái và (1.0, 1.0) tương ứng với dưới cùng bên phải.
- Action\_id: định danh của một lớp hành động. Được đánh số từ 1-80.
- Person\_id: một số nguyên duy nhất cho phép hộp này được liên kết với các hộp khác để miêu tả cùng một người trong các khung liền kề của video.

File csv của AVA được chuẩn bị cho việc nhận dạng 80 hành động nên có số lượng chú thích rất lớn (~837.000 dòng cho file train). Trong khi đó, mô hình của chúng tôi chỉ tập trung nhận dạng 10 hành động, vì vậy chúng tôi tiến hành chỉnh sửa lại file csv việc huấn luyện mô hình của chúng tôi . Các chú thích cho các hành động không được nhận dạng trong luận văn sẽ bị bỏ đi, và các action\_id sẽ được đánh số lại theo thứ tự từ 1 – 10.

	A	B	C	D	E	F	G	H
1	_Z6wFjXtGQ	1475	0.494	0.134	0.734	0.62	1	303
2	_Z6wFjXtGQ	1479	0.209	0.413	0.431	0.99	1	304
3	_Z6wFjXtGQ	1126	0.138	0	0.663	0.978	2	65
4	_Z6wFjXtGQ	1136	0.015	0.541	0.631	0.991	2	68
5	_Z6wFjXtGQ	1140	0.302	0.454	0.444	0.723	2	74
6	_Z6wFjXtGQ	1390	0.453	0.802	0.915	0.996	2	217
7	_145Aa_xkuE	1666	0.266	0.153	0.836	0.942	2	212
8	_145Aa_xkuE	1225	0.191	0.077	0.508	0.975	4	85
9	_145Aa_xkuE	1226	0.182	0.066	0.498	0.952	4	85
10	_145Aa_xkuE	1227	0.16	0.067	0.468	0.962	4	85
11	_145Aa_xkuE	1228	0.207	0.122	0.477	0.979	4	85
12	_145Aa_xkuE	1247	0.164	0.103	0.657	0.98	4	88
13	_145Aa_xkuE	1248	0.159	0.112	0.627	0.975	4	88
14	_145Aa_xkuE	1249	0.171	0.088	0.616	0.975	4	88
15	_145Aa_xkuE	1269	0.007	0.014	0.724	0.985	4	93

Hình 5.6 File csv cho tập train sau khi được chúng tôi chỉnh sửa lại

### 5.2.3 Cài đặt các thông số huấn luyện

Có nhiều thông số huấn luyện, chúng tôi sẽ trình bày một số thông số chính trong quá trình huấn luyện mô hình:

#### Batch size

Chúng tôi sử dụng batch size là 5 cho việc huấn luyện. Khi huấn luyện trên mô hình 3D CNN với GPU trên Google Colab. Các trọng số cho mỗi vòng lặp được cập nhật với average gradient của các batch trên GPU.

#### Epochs

Việc đào tạo được chia thành các epochs. Chúng tôi sử dụng epoch bằng 20, một epoch tương ứng với 1215 lần cập nhật tham số SGD. Sau mỗi epoch, the average validation loss được tính toán trên 145 mẫu của validation set.

#### Learning Rate Decay

Learning rate sẽ được giảm với tỉ lệ 0.1 nếu validation loss không giảm trong 3 chu kỳ. Việc huấn luyện sẽ bị ngưng khi validation loss không giảm trong hơn 10 chu kỳ. Đánh giá cuối cùng dựa trên cơ sở clip(?) được thực hiện với mô hình có validation loss thấp nhất. Trừ khi đề cập, learning rate khởi tạo ban đầu là 0.1.

#### Weight Decay

Weight Decay được set là  $10 \times 10^6$  cho tham số trọng số, và 0.0 cho tham số bias trong tất cả quá trình huấn luyện.

### 5.2.4 Google Colab

Để huấn luyện dữ liệu thì chúng ta cần một GPU mạnh mẽ, vì GPU thực hiện việc huấn luyện nhanh hơn rất nhiều lần so với CPU và bộ dữ liệu AVA dù đã được tiền xử lý vẫn rất lớn cho quá trình huấn luyện. Chúng tôi không có được một GPU mạnh mẽ như vậy cho quá trình huấn luyện. Vì vậy, thay

vì huấn luyện trên máy tính cá nhân, chúng tôi quyết định sử dụng một dịch vụ đám mây miễn phí của Google là Google Colab.

Google Colab cung cấp một dịch vụ dựa trên Jupyter Notebook và hỗ trợ GPU miễn phí. Điều này cho phép người dùng phát triển các ứng dụng Deep Learning trên các thư viện phổ biến như Pytorch, Tensorflow, Keras hay OpenCV.



*Hình 5.7 Tesla K80 GPU của Google Colab. Một GPU mạnh mẽ được Google cho sử dụng miễn phí.*

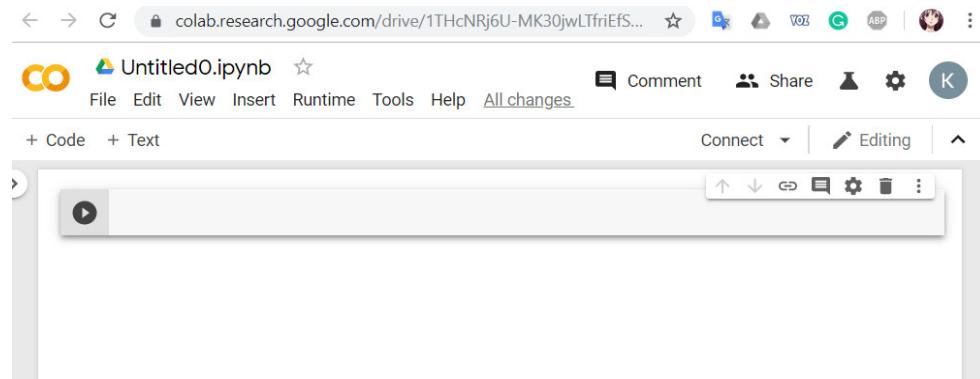
Tuy nhiên, ngoài các tiện ích trên thì dịch vụ này cũng có một số hạn chế. Google Colab chỉ hỗ trợ trên Python 2.7 và 3.6. Ngoài ra, Google Colab còn hạn chế về thời gian khi chỉ cho phép người dùng huấn luyện trong 12 tiếng liên tục.

### 5.2.5 Huấn luyện mô hình

Sau các bước chuẩn bị dữ liệu và thông số huấn luyện, chúng tôi tiến hành huấn luyện trên Google Colab.

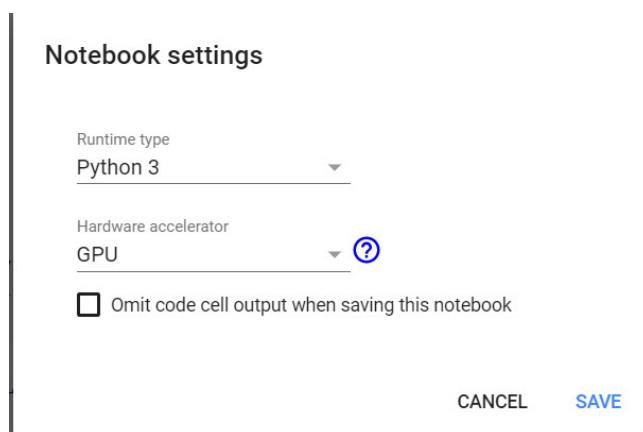
Để sử dụng Google Colab, đầu tiên chúng ta truy cập vào url: "<https://colab.research.google.com>". Đăng nhập tài khoản Google, sau đó chọn New Python 3 notebook. Một notebook sẽ được tạo ra có giao diện như hình 5.9.

## CHƯƠNG 5. XÂY DỰNG MÔ HÌNH NHẬN DẠNG HÀNH ĐỘNG



Hình 5.8 Giao diện notebook của Google Colab.

Để sử dụng GPU của Google Colab, chúng ta cần thực hiện việc thay đổi runtime type. Chọn Runtime → Change Runtime Type → Hardware accelerator → GPU → Save.



Hình 5.9 Cài đặt để được sử dụng GPU của Google Colab

Tiếp theo, cần kết nối Google Colab với Google Drive để truy cập vào dữ liệu huấn luyện đã được tải lên Google Drive. Sử dụng đoạn code dưới đây truy cập vào Google Drive từ Google Colab.

```
[1] from google.colab import drive  
drive.mount('/content/drive')  
  
↳ Go to this URL in a browser: https://  
Enter your authorization code:  
.....  
Mounted at /content/drive
```

Hình 5.10 Kết nối Google Drive và Google Colab

Truy cập vào thư mục được tạo chứa dữ liệu, sau đó chạy chương trình huấn luyện. Quá trình huấn luyện sẽ được bắt đầu, chúng tôi sử dụng phương pháp transfer learning, nên chương trình sẽ kiểm tra các checkpoint trước khi bắt đầu huấn luyện.

```
cd /content/drive/My Drive/Google_Colab/PyVideoResearch  
/content/drive/My Drive/Google_Colab/PyVideoResearch  
  
!python i3d_ava.py  
  
parsing arguments  
Logging to file ./nfs.yoda/gsigurds/caches/i3d_ava//log.txt  
{'name': 'i3d_ava', 'resume': './nfs.yoda/gsigurds/caches/i3d_ava/model.pth.tar;./nfs.'}  
experiment folder: /content/drive/My Drive/Google_Colab/PyVideoResearch  
fatal: not a git repository (or any parent up to mount point /content)  
Stopping at filesystem boundary (GIT_DISCOVERY_ACROSS_FILESYSTEM not set).  
Command '['git', 'describe', '--always']' returned non-zero exit status 128.  
git hash:  
setting Dropout p to 0.5  
=> loading checkpoint './nfs.yoda/gsigurds/caches/i3d_ava/model.pth.tar'  
=> loaded checkpoint './nfs.yoda/gsigurds/caches/i3d_ava/model.pth.tar' (epoch 18)  
setting start epoch to model epoch 18
```

*Hình 5.11 Chạy chương trình huấn luyện*

Một lưu ý nhỏ là Google Colab sẽ tự động ngắt kết nối nếu người dùng không có hoạt động trong 30 phút. Nên sau khi bắt đầu huấn luyện mô hình, chúng ta cần thêm đoạn code sau vào Console của Google Colab để tự động kết nối sau mỗi 30 phút.

```
function ClickConnect () {  
  console.log("Working");  
  document.querySelector("colab-toolbar-button#connect").click()  
}  
setInterval(ClickConnect, 60000)
```

Mô hình của chúng tôi sẽ được lưu sau mỗi epoch, với thời gian để huấn luyện cho một epoch là gần 8 tiếng. Vì Google Colab chỉ cho phép huấn luyện mô hình trong 12 tiếng, nên sau mỗi epoch, chúng ta cần phải reset Runtime rồi mới tiếp tục huấn luyện epoch tiếp theo.

## CHƯƠNG 5. XÂY DỰNG MÔ HÌNH NHẬN DẠNG HÀNH ĐỘNG

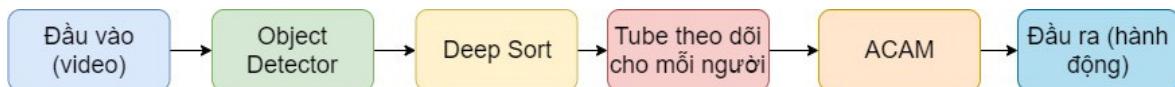
```
[i3d_ava] Train Epoch: [2][63/1215(1215)]      Time 16.297 (18.317)    Data 15.676 (17.690)    Loss 0.0242 (0.0436)
converting NxTxC a+target to N2xC
/usr/local/lib/python3.6/dist-packages/torch/nn/functional.py:2404: UserWarning: nn.functional.upsample is deprecated.
  warnings.warn("nn.functional.upsample is deprecated. Use nn.functional.interpolate instead.")
losses class: 0.004248006269335747
computing accuracy for multi-label case
computing accuracy for multi-label case
gradnorms: tensor(0.0003, device='cuda:0')
updating parameters
```

Hình 5.12 Quá trình huấn luyện bằng Google Colab

Sau 20 epochs, mô hình đã hội tụ và có thể được sử dụng để nhận dạng hành động.

### 5.3 Chương trình nhận dạng hành động

Ở phần nhận dạng hành động, chương trình có đầu vào là video (có sẵn hoặc streaming). Đối tượng được phát hiện bằng model nhận dạng đối tượng (ở đây là model SSD), sau đó được theo dõi bằng model Deep SORT. Mỗi đối tượng phát hiện được phân ra thành một tube riêng biệt để theo dõi, và được phát hiện hành động bằng model ACAM. Cuối cùng, trả kết quả là hành động được thực hiện bởi đối tượng.



Hình 5.13 Nguyên lý hoạt động chương trình chính

#### 5.3.1 Đầu vào

Đầu vào chương trình là hình ảnh từ camera. Chúng tôi sử dụng hình ảnh được truyền từ camera ip, với mong muốn có thể áp dụng vào các camera giám sát. Chương trình thiết lập ip để đọc hình ảnh từ camera:

```
url = "192.168.8.100:8080"
```

Sử dụng OpenCV để đọc hình ảnh lấy từ url

```
reader = cv2.VideoCapture("http://"+url+"/video")
```

Sau khi có hình ảnh, chúng tôi thực hiện việc đọc frames:

```
ret, frame = reader.read()
cur_img = frame[:, :, ::-1]
frame_q.put(cur_img)
```

```
if frame_q.qsize() > 100:  
    time.sleep(1)  
else:  
    time.sleep(DELAY/1000.)
```

### 5.3.2 Nhận dạng đối tượng

Tiếp theo, hình ảnh được xử lý bởi module object detection, mô hình object detection được chúng tôi sử dụng là mô hình SSD\_Mobilenet để giảm tài nguyên cho máy tính:

```
obj_detection_graph =  
"./object_detection/weights/ssd_mobilenet_v2_coco_2018_03_29/frozen_infer  
ence_graph.pb"
```

### 5.3.3 Deep SORT

Chúng tôi sử dụng deep SORT để theo dõi đối tượng, sau khi phát hiện đối tượng (ở luận văn này là person), Deep SORT sẽ theo dõi đối tượng trong 1 khoảng thời gian là T frames.

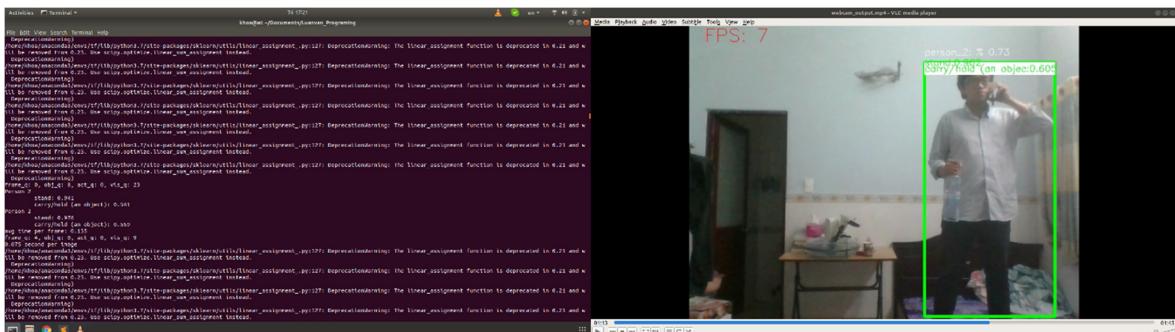
```
MODEL_CKPT = "./object_detection/deep_sort/weights/mars-small128.pb"
```

### 5.3.4 Đầu ra

Sau khi đã phát hiện được đối tượng là “person” dựa vào mô hình SSD\_Mobilenet và theo dõi đối tượng dựa vào mô hình deep SORT thì chúng ta có được một chuỗi T frames thể hiện được hành động của từng đối tượng (person). Bước phát hiện đối tượng xử lí trên tất cả các video và lưu vị trí các actors được phát hiện.

Kết quả là, sau khi đưa chuỗi T frames đi qua mô hình nhận diện hành động con người, ta thu được kết quả hành động của từng actors với vị trí tương ứng trên video đầu ra. Chúng tôi cho hiển thị 5 hành động có xác suất cao nhất, đồng thời dùng Opencv xuất hình ảnh ra màn hình để kiểm tra kết quả và lưu video nhận dạng.

Chúng tôi muốn kết quả được hiển thị dưới dạng tiếng Việt, tuy nhiên, font chữ Hershey của OpenCv chỉ hiển thị được một số ký tự hữu hạn trong bảng ASCII, vì vậy, kết quả sẽ được trả về bằng tiếng Anh.



Hình 5.14 Đầu ra của chương trình, kết quả nhận dạng hành động được hiển thi đồng thời trên Terminal và màn hình máy tính.

## 5.4 Kết luận chương 5

Tóm lại, chương 5 đã giải thích chi tiết cách thực hiện việc xây dựng và huấn luyện một mô hình nhận dạng hành động áp dụng các kiến thức từ các chương trước. Đồng thời trình bày kiến trúc của mô hình huấn luyện, nguyên lý hoạt động của chương trình. Các kết quả thực hiện và việc đánh giá mô hình sẽ được chúng tôi tiếp tục trình bày ở 2 chương sau.

# CHƯƠNG 6

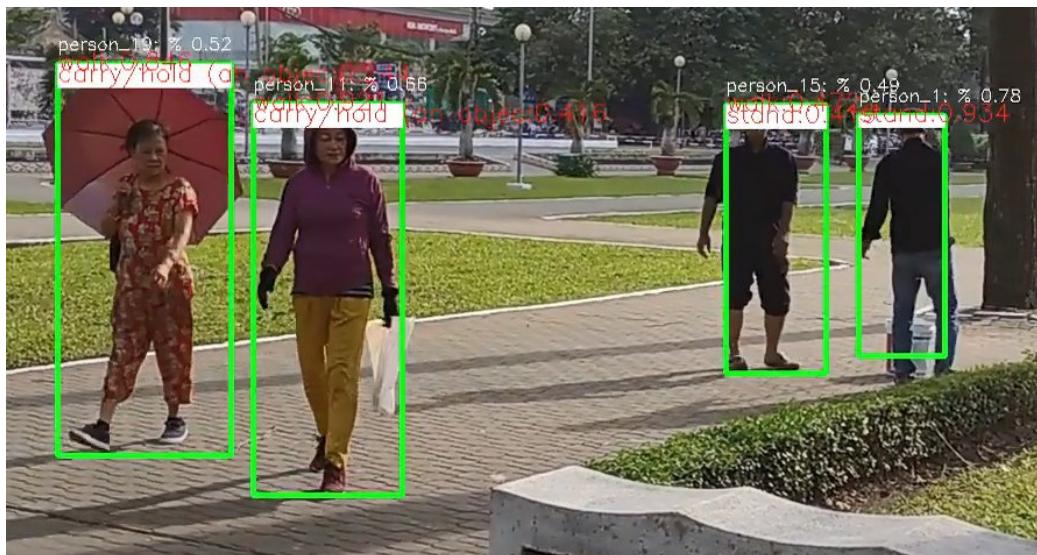
## KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ MÔ HÌNH

Chương 6 sẽ trình bày kết quả thực hiện của chương 5, đồng thời đánh giá kết quả thực hiện.

### 6.1 Kết quả thực hiện

Chương trình nhận dạng của chúng tôi được chạy trên hệ điều hành Ubuntu, sử dụng framework Keras [17] và backend Tensorflow, xử lý trên CPU core I5-8300H và GPU GeForce GTX 1050 Ti. Video được quay từ camera điện thoại với độ phân giải  $1920 \times 1080$ .

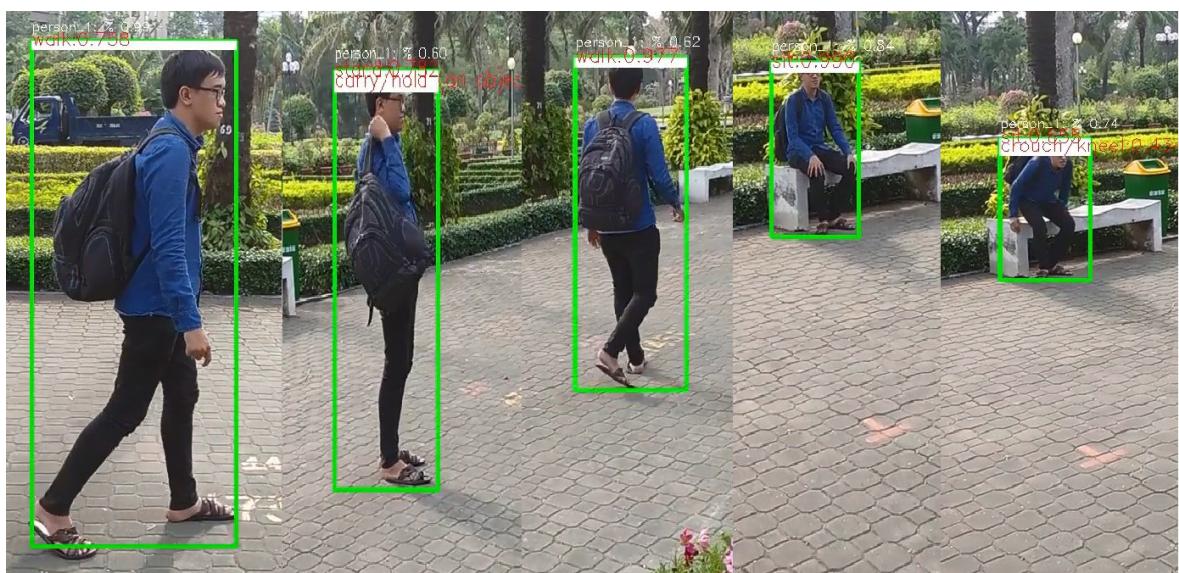
Kết quả nhận dạng hành động trong điều kiện không gian rộng rãi, ánh sáng đầy đủ. Ngữ cảnh là các hành động tại công viên.



Hình 6.1 Kết quả nhận dạng hành động nhiều đối tượng cùng lúc

Ta có thể thấy mô hình đã hoạt động chính xác trên các đối tượng. Cụ thể mô hình đã nhận dạng được sự khác nhau giữa người đứng và người đi bộ. Giữa người cầm nắm vật và không cầm, nắm vật.

Kết quả nhận dạng hành động ở các góc máy khác nhau và các hành động khác nhau. Ở hình 6.2, chúng ta có thể thấy mô hình vẫn đang làm tốt khi nhận dạng chính xác các hành động.



Hình 6.2 Kết quả nhận dạng hành động nhiều hành động khác nhau ở các góc máy khác nhau

## CHƯƠNG 6. KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ MÔ HÌNH

Kết quả nhận dạng cùng một hành động ở các điều kiện ánh sáng khác nhau, actor nhìn thẳng vào camera.



Hình 6.3 Kết quả nhận dạng hành động ở điều kiện ánh sáng khác nhau

Tiếp tục nhận dạng hành động ở điều kiện ánh sáng khác nhau, actor quay lưng lại camera.



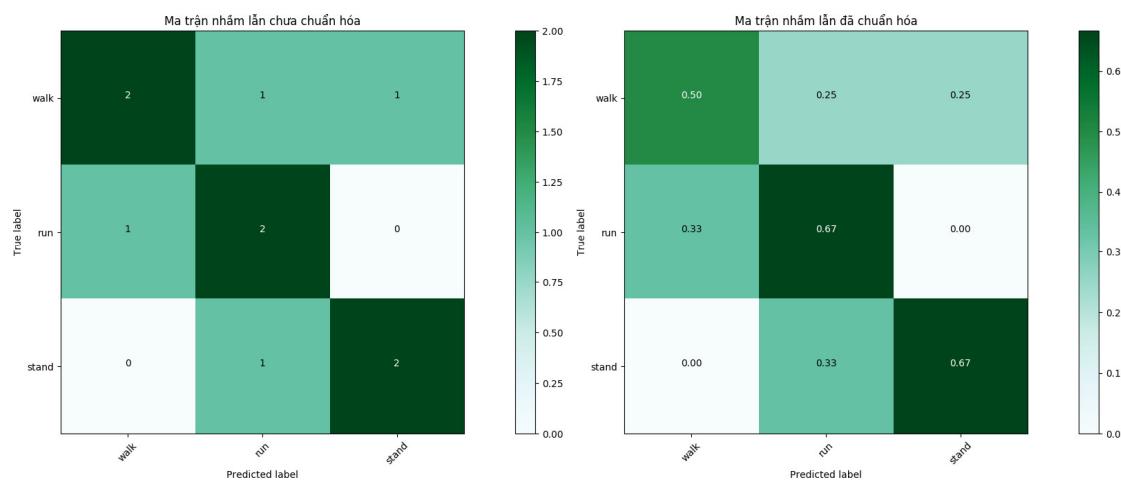
Hình 6.4 Kết quả nhận dạng hành động ở điều kiện ánh sáng khác nhau

## 6.2 Đánh giá

### 6.2.1 Ma trận nhầm lẫn (confusion matrix)

Với việc đánh giá mô hình, chúng tôi xem xét các giá trị như lớp nào được đánh giá đúng nhiều nhất, dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác, mỗi loại dữ liệu được phân loại như thế nào. Để có thể đánh giá được các giá trị này, chúng tôi sử dụng một ma trận được gọi là ma trận confusion. Về cơ bản, ma trận nhầm lẫn thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc về một lớp, và được dự đoán là rơi vào một lớp. Ma trận nhầm lẫn có 2 dạng là non-normalized confusion matrix nghĩa là ma trận nhầm lẫn chưa chuẩn hóa và normalized confusion matrix nghĩa là ma trận nhầm lẫn đã chuẩn hóa.

Ma trận nhầm lẫn thường được minh họa bằng màu sắc để có cái nhìn rõ ràng hơn. Với các bài toán với nhiều lớp dữ liệu, các biểu diễn bằng màu này sẽ rất hữu ích. Các ô màu đậm thể hiện các giá trị cao. Một mô hình tốt cho một ma trận nhầm lẫn có các phần tử trên các đường chéo chính có giá trị lớn, các phần tử còn lại sẽ có giá trị nhỏ. Nói cách khác, khi biểu diễn bằng màu sắc, đường chéo có màu càng đậm so với phần còn lại sẽ còn tốt. [2]

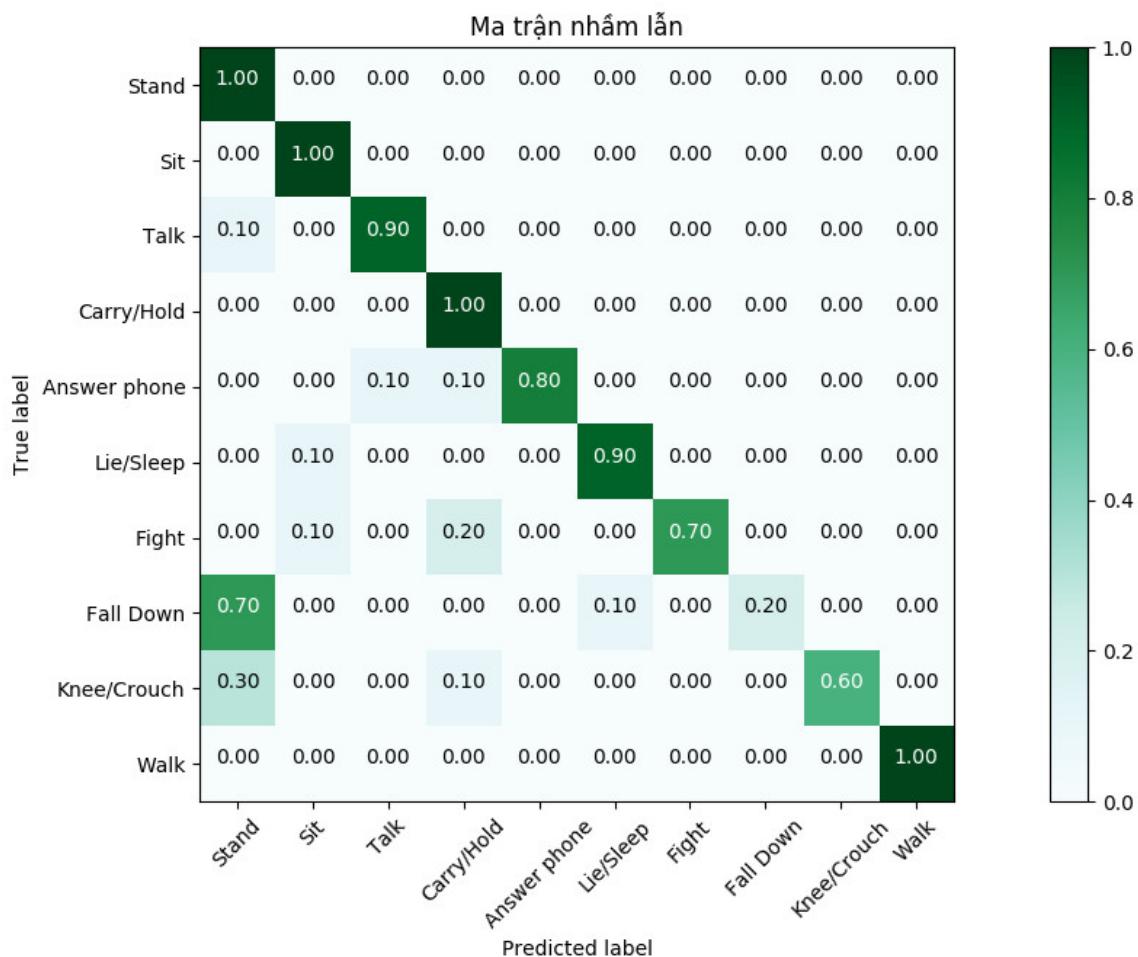


*Hình 6.5 Minh họa ma trận nhầm lẫn chưa chuẩn hóa (bên trái) và đã chuẩn hóa (bên phải)*

Từ hình 6.2, ta thấy rằng ma trận nhầm lẩn đã chuẩn hóa mang nhiều thông tin hơn. Sự khác nhau được thấy ở ô trên cùng bên trái. Lớp dữ liệu “walk” được phân loại không thực sự tốt nhưng trong ma trận nhầm lẩn chưa chuẩn hóa, nó vẫn có màu đậm như hai ô còn lại trên đường chéo chính.

### 6.2.2 Đánh giá mô hình

Để đánh giá mô hình của mình, chúng tôi sử dụng ma trận nhầm lẩn. Các video dự đoán được lấy ngẫu nhiên ở validation set của bộ dữ liệu AVA, mỗi video có thời lượng 3s và được chú thích trong file csv đi kèm theo bộ dữ liệu. Chúng tôi sử dụng kết quả “top 5” để đánh giá độ chính xác của mô hình này. Kết quả được trình bày trong hình 6.3



Hình 6.6 Ma trận nhầm lẩn của hệ thống

## CHƯƠNG 6. KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ MÔ HÌNH

---

Từ ma trận nhầm lẫn trên, chúng ta có thể thấy mô hình giới thiệu có khả năng nhận dạng rất tốt các hành động có tính chất “liên tục” (xảy ra trong nhiều frame liên tiếp nhau của video) như: Đứng, Ngồi, Nói chuyện, Nằm, Đi bộ. Nhưng nhận dạng kém chính xác với các hành động có tính chất “nhất thời” (chỉ xảy ra trong vài frames của video) như: Đánh nhau, quỳ gối. Đặc biệt nhận dạng rất kém với hành động té ngã, điều này là do hành động này là một hành động “nhất thời” và có lượng dữ liệu ít nhất (chỉ có 290 chú thích trong tập train và 88 chú thích trong tập validation) trong số 10 hành động .

# CHƯƠNG 7

## KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chương 7 là chương cuối cùng của luận văn, ở chương này sẽ trình bày kết luận về việc hoàn thành nhiệm vụ đặt ra ở đầu luận văn, đồng thời nêu ra hướng phát triển cho luận văn.

### 7.1 Kết luận

Từ mục tiêu ban đầu đề ra, tuy gấp phải khó khăn, thử thách, nhưng qua quá trình tìm tòi, học hỏi những kiến thức liên quan, và quyết tâm kiên trì thực hiện. Chúng tôi cơ bản cũng đã hoàn thành mục tiêu đề ra như sau:

- Luận văn đã trình bày được rõ ràng, mạch lạc các kiến thức cần thiết cho việc xây dựng một hệ thống nhận dạng đối tượng. Giới thiệu về Deep Learning và các thuật toán của nó, cũng như giải thích về Deep CNN được dùng trong Luận văn lần này. Chúng tôi cũng giới thiệu chi tiết về việc nhận dạng hành động con người, mạng tích chập 2 luồng, phương pháp ACAM và bộ dữ liệu AVA được chúng tôi sử dụng
- Luận văn đã xây dựng thành công một mô hình để huấn luyện từ bộ dữ liệu AVA để thực hiện nhiệm vụ nhận dạng hành động.
- Chúng tôi cũng xây dựng một chương trình xử dụng mô hình đã được huấn luyện cho việc nhận dạng một số hành động thực tế. Chương trình có khả năng xử lý việc nhận dạng thời gian thực, có thể xử lý video được tải sẵn, hình ảnh từ webcam, hay hình ảnh được truyền từ camera điện thoại.

Tuy nhiên, vẫn có vài mặc hạn chế chưa được khắc phục:

- Việc chạy nhiều model song song cần rất nhiều tài nguyên máy tính. Nên không thể thực hiện trên các máy tính nhúng có lượng Ram và Cpu hạn chế như Raspberry.
- FPS còn thấp, hình ảnh bị chậm 15 – 30s so với thực tế.

### 7.2 Hướng phát triển

Mặc dù phương pháp nhận diện hành động con người của chúng tôi trình bày đã đạt được những kết quả khả quan, nhưng việc áp dụng phương pháp nhận diện hành động con người hiện tại trong các hệ thống và ứng dụng thời gian thực và thực tế vẫn chưa thực sự tốt. Với sự hạn chế về mặt kiến thức và thời gian trong việc thực hiện luận văn này, chúng tôi vẫn tồn tại nhiều khuyết điểm cho luận văn này.

Hầu hết các ứng dụng bây giờ đều áp dụng cho thời gian thực và các thiết bị di động, nhưng trong phạm vi đề tài rất khó để thực hiện như vậy, vì các hệ thống thời gian thực và di động có sức tính toán hạn chế. Để đạt được hiệu suất tương đương, chúng tôi có các đề xuất như sau:

- Sử dụng những cảm biến bổ sung để hỗ trợ nhận dạng hoặc phát triển vi mạch cho các thiết bị nhúng.
- Bên cạnh giải pháp định hướng phần cứng, từ góc độ thị giác máy tính, chúng ta có các phương pháp trích xuất đặc trưng hiệu quả hơn, phương pháp dự đoán sự phân loại sẽ huấn luyện các mô hình nhận dạng nhanh.
- Một cách khác, chúng ta có thể làm giảm chất lượng hình ảnh đầu vào và cân bằng giữa thông tin đầu vào, hiệu quả thuật toán và tỷ lệ nhận dạng.

Trên đây, chúng tôi đã đưa ra những giải pháp cho định hướng phát triển tiếp tục phương pháp nhận diện hành động con người trong luận văn này.

## THAM KHẢO

- [1] M. Vrigkas, C. Nikou and A. I. Kakadiaris, "A Review of Human Activity Recognition Methods," *Front. RoBot. AI*, vol. 2, no. 28, 16 November 2015.
- [2] V. H. Tiệp, Machine Learning Cơ Bản, Nhà Xuất Bản Khoa Học và Kỹ Thuật, 2018.
- [3] Convolutional Neural Network, [Online]. Available: <http://cs231n.github.io/convolutional-networks/>. [Accessed 19 05 2017].
- [4] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.
- [5] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos," *arXiv preprint arXiv:1406.2199v2*, 12 Nov 2014.
- [6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [7] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788, 2016.
- [8] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv:1612.08242v1*, 25 Dec 2016.
- [9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu and A. C. Berg, "SSD: Single shot multiple box," *European Conference on Computer Vision*, pp. 21-37, 2016.

- [10] A. Bewley, N. Wojke and D. Paulus, "SIMPLE ONLINE AND REALTIME TRACKING WITH A DEEP ASSOCIATION METRIC," *arXiv preprint arXiv:1703.07402v1*, 21 Mar 2017.
- [11] C. Feichtenhofer, A. Pinz and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1933-1941, 2016.
- [12] O. Ulutan, Torres Carlos and B. S. Manjunath, "Actor Conditioned Attention Maps for Video Action Detection," *arXiv preprint arXiv:1812.11631v2*, 29 Mar 2019.
- [13] S. Ji, W. Xu, M. Yang and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221-231, Jan 2013.
- [14] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *arXiv preprint arXiv:1705.07750v3*, 12 Feb 2018.
- [15] Google, 2018. [Online]. Available: <https://research.google.com/ava/>.
- [16] C. Gu, C. Sun, D. A. Ross, C. Vondrick, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid and M. Jitendra, "AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions," *arXiv preprint arXiv:1705.08421v4*, 30 Apr 2018.
- [17] F. Chollet, Keras, 2015. [Online]. Available: <https://github.com/fchollet/keras>.