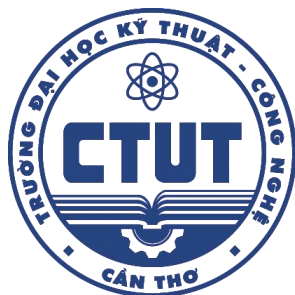


TRƯỜNG ĐẠI HỌC KỸ THUẬT - CÔNG NGHỆ CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN KHOA HỌC MÁY TÍNH 3

Đề tài

NHẬN DẠNG HÀNH ĐỘNG CON NGƯỜI
DỰA TRÊN CỬ CHỈ

GIẢNG VIÊN HƯỚNG DẪN
ThS Trầm Vũ Kiệt

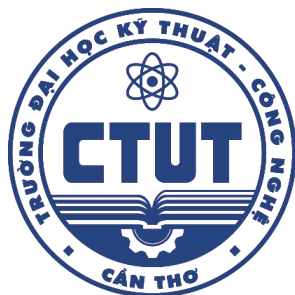
SINH VIÊN THỰC HIỆN
Phạm Minh Sang (MSSV: 2001185)

Lê Thanh Xuân (MSSV: 2001188)

NGÀNH : KHOA HỌC MÁY TÍNH 2020

Cần Thơ 2023

TRƯỜNG ĐẠI HỌC KỸ THUẬT - CÔNG NGHỆ CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN 3
KHOA HỌC MÁY TÍNH
Đề tài

**NHẬN DẠNG HÀNH ĐỘNG CON NGƯỜI DỰA
TRÊN CỬ CHỈ**

GIẢNG VIÊN HƯỚNG DẪN

ThS Trầm Vũ Kiệt

SINH VIÊN THỰC HIỆN

Phạm Minh Sang (MSSV: 2001185)

Lê Thanh Xuân (MSSV: 2001188)

NGÀNH : KHOA HỌC MÁY TÍNH 2020

Cần Thơ 2023

LỜI CẢM ƠN

Lời đầu tiên em xin cảm ơn quý thầy cô Khoa Công Nghệ Thông Tin Trường Đại Học Kỹ Thuật - Công Nghệ Cần Thơ đã truyền dạy kiến thức cho em trong thời gian qua để em có thể hoàn thành nghiên cứu và thực hiện đề án. Trong quá trình hoàn thành đề án khoa học, ngoài những cố gắng của bản thân, em sẽ không thể nào hoàn thành tốt được công việc của mình nếu không có sự chỉ bảo và hướng dẫn tận tình của Giảng viên Trầm Vũ Kiệt .

Em xin được gửi lời cảm ơn chân thành nhất tới thầy vì đã trang bị cho em những kiến thức, kỹ năng cơ bản cần có để hoàn thành đề tài khoa học này.

Trong quá trình làm đề án, khó tránh khỏi sai sót, rất mong Thầy (cô) bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm của nhóm em còn hạn chế nên đề án không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của Thầy (cô) để nhóm em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn những bài đề án sắp tới.

Xin chân thành cảm ơn!

LỜI CAM ĐOAN

Đây là đề tài nghiên cứu được thực hiện dưới sự hướng dẫn của ThS. Trầm Vũ Kiệt. Đề tài này đã được hoàn thành sau một thời gian nghiên cứu, tìm hiểu các nguồn tài liệu và thông tin trên mạng đáng tin cậy. Nội dung của bài được tôi tập hợp lại từ các nguồn tài liệu tham khảo (cuối đề tài), không sao chép toàn bộ các đề tài và quá trình nghiên cứu của các tác giả khác.

Tôi xin chịu hoàn toàn trách nhiệm về nội dung trong đồ án của mình đã thực hiện.

Cần Thơ, ngày.....tháng.....năm 2023

Sinh viên thực hiện

Phạm Minh Sang

Lê Thanh Xuân

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Giảng viên hướng dẫn: ThS Trần Vũ Kiệt

Nhận xét của giảng viên hướng dẫn:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Cần Thơ, ngày tháng năm 2023

GIẢNG VIÊN HƯỚNG DẪN

TRẦN VŨ KIẾT

MỤC LỤC

LỜI CẢM ƠN	iii
PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN.....	v
DANH MỤC HÌNH.....	3
DANH MỤC TỪ VIẾT TẮT.....	5
KẾ HOẠCH THỰC HIỆN.....	6
PHẦN GIỚI THIỆU TỔNG QUAN.....	8
1. Đặc vấn đề	8
2. Mục đích nghiên cứu của đề tài	8
3. Đối tượng và phạm vi nghiên cứu	9
4. Phương pháp nghiên cứu	9
5. Kết quả đạt được	9
6. Bố cục đồ án	9
PHẦN CƠ SỞ LÝ THUYẾT.....	11
CHƯƠNG 1. TỔNG QUAN VỀ MẠNG NEURAL TÍCH CHẬP VÀ BỘ NHỚ	
NGẮN HẠN DÀI HẠN TRONG NHẬN DẠNG HÀNH ĐỘNG	11
1. Giới thiệu mạng Nơ-ron(ANN)	11
1.1. Kiến trúc mạng nơ ron.....	11
1.2. Các tham số chính	12
2. Tìm hiểu mạng nơ ron tích chập (CNN).....	12
2.1 Giới thiệu.....	12
2.2 Kiến trúc mạng nơ ron tích chập	12
2.3 Các phân lớp chính	13
2.4 Tham số tối ưu	16
2.5 Nguyên lý hoạt động của một mạng nơ ron tích chập điển hình	20
2.6 Ứng dụng	20
3. Bộ nhớ ngắn hạn dài hạn (LSTM)	21
3.1 Mạng lưới thần kinh tái phát bộ nhớ ngắn hạn dài hạn.....	21
3.2 Cấu trúc của mạng LSTM.....	24
CHƯƠNG 2: CƠ CHẾ NHẬN DẠNG HÀNH ĐỘNG TRONG VIDEO	28
1. Cơ chế nhận dạng	28
2. Nghiên cứu liên quan	28

3. Thách thức trong nhận dạng hành động	30
4. Quy trình thực hiện	31
5. Công cụ hỗ trợ	33
PHẦN QUÁ TRÌNH THỰC HIỆN	35
1. Thư viện sử dụng	35
1.1 Giới thiệu thư viện TensorFlow	35
1.2 Chuẩn bị tập dữ liệu	35
2. Bài toán nhận dạng hành động trong video	36
3. Chương trình ứng dụng cho bài toán nhận dạng hành động.....	37
4. Tạo một giao diện chương trình ứng dụng	45
PHẦN KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	48
1. Kết luận.....	48
2. Những hạn chế và hướng phát triển của đề tài	48
2.1 Hạn chế đề tài.....	48
2.2 Hướng phát triển đề tài	48
TÀI LIỆU THAM KHẢO	50

DANH MỤC HÌNH

<i>Hình 1. 1 Kiến trúc mạng nơ ron</i>	11
<i>Hình 2 1 Kiến trúc hoạt động của mạng nơ ron tích chập</i>	13
<i>Hình 2 3 Ma trận pixel 5x5 và ma trận bộ lọc 3x3</i>	13
<i>Hình 2 4 Feature Map: lớp tích chập của ma trận ảnh 5x5 nhân với ma trận bộ lọc 3x3</i>	14
<i>Hình 2 5 Một số ma trận bộ lọc phổ biến</i>	14
<i>Hình 2 6 Feature Map: lớp tích chập của ma trận ảnh 5x5 nhân với ma trận bộ lọc 3x3</i>	15
<i>Hình 2 7 Nguyên lý hoạt động của một mạng nơ ron</i>	20
<i>Hình 2 8 Cách mà một RNN sử dụng bộ nhớ của nó.</i>	21
<i>Hình 2 9 Mô hình Many-to-One RNN</i>	22
<i>Hình 2 10 Thông tin lưu giữ của một RNN</i>	22
<i>Hình 2 11 Cấu trúc của một ô LSTM (Long Short-Term Memory)</i>	24
<i>Hình 3. 1 Sự giảm thiểu của sự phụ thuộc lẫn nhau giữa các neuron trước và sau khi thực hiện dropout</i>	26
<i>Hình 3. 2 Phân tích hoạt động của con người.</i>	30
<i>Hình 3. 3 A 2 x 2 sample Confusion Matrix</i>	33
<i>Hình 3. 4 Công cụ phát triển Visual studio code.</i>	34
<i>Hình 3. 5 Tập dữ liệu ucf101</i>	36
<i>Hình 3. 6 Ảnh kết tải tập dữ liệu</i>	39
<i>Hình 3. 7 Ảnh giao diện chương trình</i>	46
<i>Hình 3. 8 Kết quả nhận dạng trong video</i>	47
<i>Hình 3. 9 Kết quả nhận dạng trong ảnh</i>	47

DANH MỤC BẢNG

<i>Bảng 1 1 Bảng thông tin liên lạc.....</i>	<i>6</i>
<i>Bảng 1 2 Bảng phân công thực hiện công việc nhóm.</i>	<i>6</i>
<i>Bảng 1 3 Lớp gộp Pooling</i>	<i>16</i>

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

KẾ HOẠCH THỰC HIỆN

1. Nội dung chung

- **Tên đề tài:**
 - Nhận dạng hành động con người dựa trên cử chỉ
- **Giảng viên hướng dẫn:**
 - Trần Vũ Kiệt
- **Thời gian giao nhiệm vụ**
 - Từ ngày 01/9/2023 đến 01/12/2023
- **Sinh viên thực hiện:**
 - Lê Thanh Xuân
 - Phạm Minh Sang
- **Thông tin liên lạc của sinh viên**

STT	Tên	MSSV	Email
1	Lê Thanh Xuân	2001188	ltxuan2001188@student.ctuet.edu.vn
2	Phạm Minh Sang	2001185	pmsang2001185@student.ctuet.edu.vn

Bảng 1 1 Bảng thông tin liên lạc

- **Chương trình, ứng dụng sử dụng:**
 - Chương trình sử dụng: Sử dụng mạng CNN + LSTM kết hợp với thư viện OpenCV và TensorFlow để nhận dạng hành động con người từ dữ liệu hình ảnh, video.
- 2. Ứng dụng: Áp dụng mô hình cho việc giải quyết bài toán nhận dạng hành động con người trong lĩnh vực như giáo dục, y tế, công nghiệp...
- 3. Kế hoạch thực hiện

Nội dung công việc	Người thực hiện	Tiến độ
Lập kế hoạch và nghiên cứu chủ đề	Phạm Minh Sang	100%
Thiết kế nội dung báo cáo		
Kiểm thử		
Tinh chỉnh mô hình		
Viết giao diện		
Nội dung thực hiện	Người thực hiện	Tiến độ
Tìm kiếm, tổng hợp tài liệu	Lê Thanh Xuân	100%
Viết mã xây dựng mô hình		
Tinh chỉnh mô hình		
Chỉnh sửa, điều chỉnh bài báo cáo		
Trình bày bài báo cáo		

Bảng 1 2 Bảng phân công thực hiện công việc nhóm.

TÓM TẮT

Nhận dạng hành động trong video là một lĩnh vực quan trọng trong thị giác máy tính với nhiều ứng dụng rộng rãi như gán chỉ mục đa phương tiện, khôi phục thông tin, theo dõi và kiểm soát bệnh nhân, giám sát không gian công cộng và nhiều ứng dụng khác. Nghiên cứu này tập trung vào việc triển khai nhận dạng hành động của con người trên video bằng việc kết hợp Mạng nơ-ron tích chập và Mạng bộ nhớ dài-ngắn hạn. Chúng tôi sẽ thực hiện điều này thông qua hai kiến trúc và phương pháp tiếp cận khác nhau trong TensorFlow. Chúng tôi đề xuất một kiến trúc CNN-LSTM, trong đó mạng nơ-ron tích chập được huấn luyện trước để trích xuất các đặc trưng từ video đầu vào. Sau đó, mạng LSTM sẽ phân loại video thành các lớp hành động cụ thể.

Để đánh giá hiệu suất của hệ thống, chúng tôi sử dụng độ chính xác kết hợp giữa precision và recall. Kết quả thể hiện rằng, với việc sử dụng bộ dữ liệu UCF-101, hệ thống của chúng tôi đạt được độ chính xác khoảng 80% trong quá trình huấn luyện và 85% trong quá trình thử nghiệm.

Từ khóa: Nhận dạng hành động · Mạng nơ-ron tích chập · Bộ nhớ ngắn hạn dài · UCF-101.

PHẦN GIỚI THIỆU TỔNG QUAN

1. Đặc vấn đề

Sự phát triển nhanh chóng của Internet đã dẫn đến việc tăng lượng video được chia sẻ hàng phút, tạo ra một nguồn thông tin đa dạng và phong phú. Bản chất của video là một thông tin phương thức truyền thông đặc biệt với nhiều loại tin tức và có can nhiễu phức tạp, ví dụ như chuyển động máy ảnh, hình nền hỗn độn, hoặc điều kiện chiếu sáng khác nhau v.v.. Với sự bùng nổ thông tin, việc hiểu và phân tích những đoạn video này cho các mục đích khác nhau như tìm kiếm, giới thiệu, xếp hạng v.v. trở nên ngày càng cần thiết. Bài toán nhận dạng hành động trong video là một trong những bài toán cơ bản của thị giác máy tính với nhiều ứng dụng khác nhau như giám sát, lập mối quan hệ, phục hồi, đến tương tác giữa con người và máy tính.

Những năm gần đây, chúng ta đã chứng kiến được nhiều thành tựu vượt bậc trong ngành Thị giác máy tính (Computer Vision). Các hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động. Bài toán nhận dạng hoạt động video là bài toán khó đã được nghiên cứu từ lâu, nhưng gần đây mới có nhiều kết quả khả quan do sự phát triển mạnh mẽ của công nghệ. Đặc biệt, các thao tác video quy mô lớn cũng là vấn đề mang tính cấp thiết hiện nay.

Có nhiều phương pháp được sử dụng để nhận dạng hành động trong video, trong đó Convolutional Neural Network (CNNs - Mạng nơ-ron tích chập) và Long Short-Term Memory (LSTM) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Được sự hướng dẫn tận tình của thầy Trần Vũ Kiệt và với mong muốn học một công nghệ nhận dạng hành động mới, đồng thời có ứng dụng thực tiễn trong ngành công tác của bản thân. Chúng em đã mạnh dạn chọn đề tài " Nhận dạng hành động con người dựa trên cử chỉ" để nghiên cứu. Với phương pháp này, kỳ vọng sẽ đạt kết quả tốt hơn các nghiên cứu trước đây.

2. Mục đích nghiên cứu của đề tài

Nghiên cứu các cơ sở lý thuyết và cũng như cách thức hoạt động của mạng nơ-ron tích chập (CNN) và mạng nơ-ron dài ngắn hạn (LSTM). Từ đó tạo ra một hệ thống thông minh có khả năng nhận dạng hành động con người dựa trên video. Để đạt được mục tiêu này, dự án tập trung vào xây dựng và huấn luyện một tập dữ liệu đa dạng về hành động con người. Chúng tôi sẽ sử dụng mô hình CNN để trích xuất đặc trưng từ các khung hình video và LSTM để học cấu trúc thời gian trong dữ liệu.

Sự kết hợp giữa CNN và LSTM giúp cải thiện khả năng dự đoán và phân loại hành động con người từ dữ liệu trở nên chính xác và tin cậy hơn. Kết quả cuối cùng sẽ là một hệ thống có khả năng nhận dạng và phân loại hành động con người từ video, mang lại giá trị ứng dụng cho các lĩnh vực như chăm sóc sức khỏe, an ninh, và công nghiệp, giáo dục,...

3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Các chuỗi video hoặc hình ảnh về hành động của con người từ nhiều nguồn khác nhau, bao gồm video clip và các nguồn dữ liệu hình ảnh khác.

Phạm vi nghiên cứu của mô hình: Tập trung vào việc áp dụng kiến thức về mạng nơ-ron tích chập và mạng nơ-ron dài ngắn. Đánh giá trên một số cơ sở dữ liệu chuẩn như UCF101. Công cụ lập trình Visual studio code , Python 3.6, các thư viện khác...

4. Phương pháp nghiên cứu

Phương pháp nghiên cứu lý thuyết: Tổng hợp, nghiên cứu tài liệu các tài liệu, nghiên cứu gần đây về CNN, LSTM và ứng dụng của chúng trong lĩnh vực nhận dạng hành động để phân loại, nhận dạng.

Phương pháp trao đổi khoa học: Tham khảo các tài liệu, bài báo, bài luận văn trên internet...

5. Kết quả đạt được

Đề tài giúp sinh viên nắm vững các kiến thức cần thiết và quan trọng về mạng tích chập , mạng nơ ron ngắn hạn dài hạn , cũng như biết cách thức hoạt động của nó trong việc nhận dạng hành động, giúp cho việc vận dụng các kiến thức ấy vào các đồ án khác trong tương lai trở nên dễ dàng và đạt hiệu quả cao.

6. Bố cục đồ án

Phần giới thiệu

Giới thiệu tổng quan về đề tài

Phần nội dung lý thuyết

Chương 1: Tổng quan về mạng neural tích chập và bộ nhớ dài ngắn hạn trong nhận dạng hành động.

Chương 2: Cơ chế nhận dạng hành động qua video

Chương 3: Quá trình thực hiện

Chương 4: Kết luận và hướng phát triển

Phần kết luận

Kết quả đạt được và hướng phát triển mô hình

Phụ lục: Tài liệu tham khảo

PHẦN CƠ SỞ LÝ THUYẾT

CHƯƠNG 1. TỔNG QUAN VỀ MẠNG NEURAL TÍCH CHẬP VÀ BỘ NHỚ NGẮN HẠN DÀI HẠN TRONG NHẬN DẠNG HÀNH ĐỘNG

1. Giới thiệu mạng Nơ-ron(ANN)

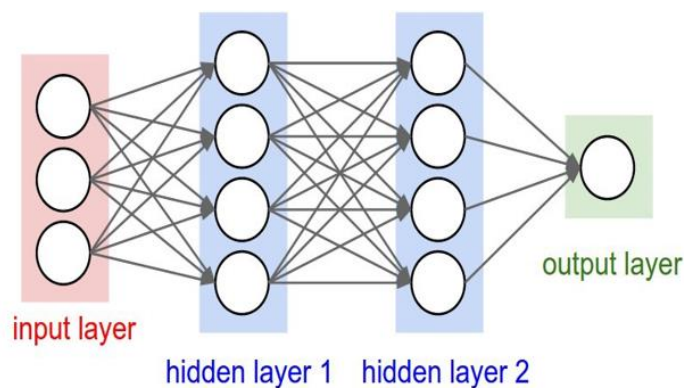
Mạng neural nhân tạo (Artificial Neural Network - ANN), hay thường gọi ngắn gọn là mạng neural, là mô hình xử lý thông tin được mô phỏng theo cách thức xử lý thông tin của các hệ neural sinh học. Nó bao gồm có một nhóm các neural nhân tạo (mỗi neural là một nút) kết nối với nhau qua các liên kết có trọng số. Thông tin được truyền qua các liên kết này để xử lý thông tin và giải quyết các vấn đề trong máy học và deep learning.

Mạng neuron được sử dụng rộng rãi trong machine learning và deep learning do khả năng học được từ dữ liệu. Các deep learning models như CNN (Convolutional Neural Network) và LSTM (Long Short-Term Memory) là các kiểu mô hình mạng neuron phổ biến được áp dụng trong việc xử lý dữ liệu hình ảnh, video, và dữ liệu tuần tự.

1.1. Kiến trúc mạng nơ ron

Mạng nơ-ron là tập hợp các nơ-ron được kết nối trong một đồ thị không tuần hoàn. Các đầu ra của một số nơ-ron có thể trở thành đầu vào của các nơ-ron khác. Mô hình mạng nơ-ron thường được tổ chức thành các lớp riêng biệt của nơ-ron.

Kiến trúc chung của một ANN gồm 3 thành phần đó là lớp đầu vào (Input Layer), lớp ẩn (Hidden Layer) và lớp đầu ra (Output Layer) (xem hình 1-1). Trong đó, lớp ẩn (Hidden Layer) gồm các nơ-ron, nhận dữ liệu input từ các nơ-ron ở lớp trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo, trong một ANN có thể có nhiều Hidden Layer.



Hình 1. 1 Kiến trúc mạng nơ ron

1.2. Các tham số chính

Inputs: Thông tin từ thế giới bên ngoài đi vào mạng nơ-ron nhân tạo qua lớp đầu vào. Các nút đầu vào xử lý dữ liệu, phân tích hoặc phân loại và sau đó chuyển dữ liệu sang lớp tiếp theo.

Hidden : Dữ liệu đi vào lớp ẩn đến từ lớp đầu vào hoặc các lớp ẩn khác. Mạng nơ-ron nhân tạo có thể có một số lượng lớn lớp ẩn. Mỗi lớp ẩn phân tích dữ liệu đầu ra từ lớp trước, xử lý dữ liệu đó sâu hơn và rồi chuyển dữ liệu sang lớp tiếp theo.

Output: Lớp đầu ra cho ra kết quả cuối cùng của tất cả dữ liệu được xử lý bởi mạng nơ-ron nhân tạo. Lớp này có thể có một hoặc nhiều nút. Ví dụ: giả sử chúng ta gặp phải một vấn đề phân loại nhị phân (có/không), lớp đầu ra sẽ có một nút đầu ra, nút này sẽ cho kết quả 1 hoặc 0. Tuy nhiên, nếu chúng ta gặp phải vấn đề phân loại nhiều lớp, lớp đầu ra sẽ có thể bao gồm nhiều hơn một nút đầu ra.

2. Tìm hiểu mạng nơ ron tích chập (CNN)

2.1 Giới thiệu

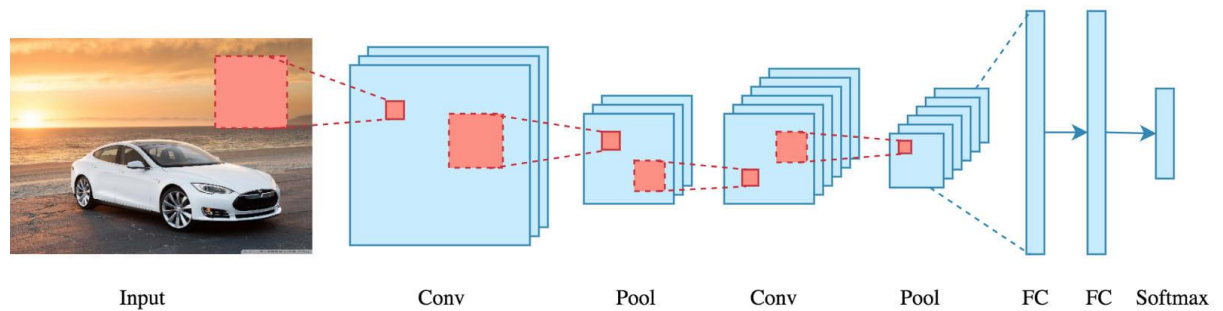
Convolutional Neural Network (CNNs - Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

Mạng nơ-ron tích chập được tạo thành từ các nơ-ron có trọng số và sai số. Mỗi một nơ-ron nhận một số đầu vào, thực hiện nhân chập và tùy chọn sau đó với một hàm phi tuyến tính. Mạng nơ-ron tích chập được áp dụng khá nhiều trong các bài toán nhận dạng như nhận dạng vật thể trong ảnh, nhận dạng chữ viết tay (chuyển đổi chữ viết trong hình ảnh thành văn bản thô trong máy tính), nhận dạng vật thể 3D, xử lý tiếng nói, xử lý ngôn ngữ tự nhiên, với độ chính xác cao.

2.2 Kiến trúc mạng nơ ron tích chập

Mô hình kiến trúc mạng nơ-ron tích chập ra đời đã được áp dụng nhiều trong các bài toán nhận dạng khác nhau, với một kiến trúc khác so với mạng truyền thống. Thay vì toàn bộ ảnh nối với một node thì chỉ có một phần cục bộ trong ảnh nối đến một node trong lớp tiếp theo. Dữ liệu hình ảnh thông qua các lớp của mô hình này sẽ được “học” ra các đặc trưng để tiến hành phân lớp một cách hiệu quả.

Về cơ bản, mô hình kiến trúc mạng nơ-ron tích chập bao gồm các lớp sau: lớp Convolutional, lớp chuyển đổi (RELU- Rectified Linear Unit), lớp Pooling, lớp Fully Connected. Sự sắp xếp về số lượng và thứ tự giữa các lớp này sẽ tạo ra những mô hình khác nhau phù hợp cho các bài toán khác nhau.



Hình 2 1 Kiến trúc hoạt động của mạng nơ ron tích chập

2.3 Các phân lớp chính

2.3.1. Lớp tích chập – Convolution Layer

Lớp Convolution trong đóng vai trò quan trọng trong việc trích xuất các đặc trưng từ ảnh. Đầu vào của lớp convolution là các ma trận ảnh và các ma trận trọng số được gọi là bộ lọc (filter). Quá trình tích chập được thực hiện bằng cách trượt bộ lọc qua ảnh và tính tổng các tích chập tại từng vị trí. Các đặc trưng cần tìm sẽ được biểu diễn dưới dạng các bản đồ đặc trưng (feature maps).

Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

Hình 2 2 Ma trận pixel 5x5 và ma trận bộ lọc 3x3

Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.



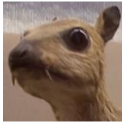

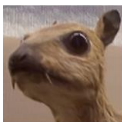

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Hình 2 3 Feature Map: lớp tích chập của ma trận ảnh 5x5 nhân với ma trận bộ lọc 3x3

Feature là đặc điểm, các CNN sẽ so sánh hình ảnh dựa theo từng mảnh và những mảnh này được gọi là Feature. Thay vì phải khớp các bức ảnh lại với nhau thì CNN sẽ nhìn ra sự tương đồng khi tìm kiếm thô các Feature khớp với nhau bằng 2 hình ảnh tốt hơn. Mỗi Feature được xem là một hình ảnh mini có nghĩa chúng là những mảng 2 chiều nhỏ. Các Feature này đều tương ứng với các khía cạnh nào đó của hình ảnh và chúng có thể khớp lại với nhau

Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau

Input	Convolution filter	Feature
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	 Edge
	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	 Blurred
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	 Sharpen

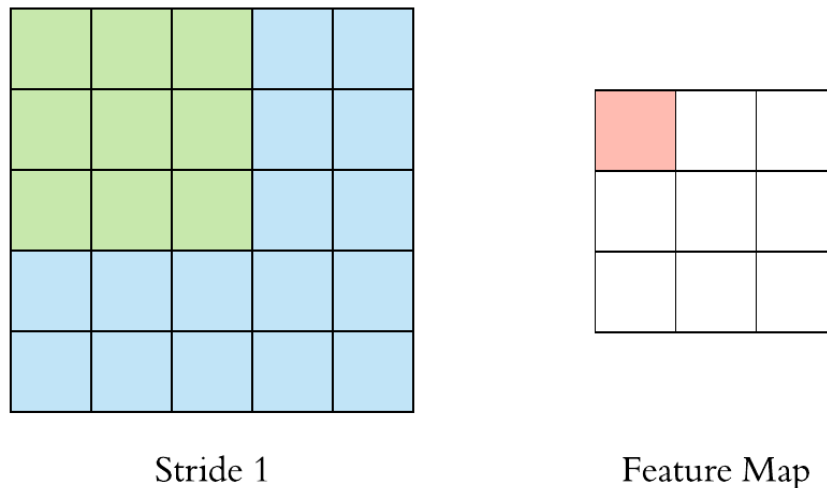
Hình 2 4 Một số ma trận bộ lọc phổ biến

2.3.2. Bước nhảy – Stride

Stride là thông số quy định bước di chuyển của các bộ lọc (filters) hoặc cửa sổ trượt (sliding window) khi thực hiện phép tích chập trên dữ liệu đầu vào. Stride xác định khoảng cách giữa hai lần áp dụng bộ lọc lên dữ liệu. Khi bộ lọc di chuyển trên dữ liệu đầu vào để tạo ra các feature map, stride xác định bao nhiêu bước bộ lọc sẽ di chuyển qua mỗi lần áp dụng.

Ví dụ, nếu bạn có một ma trận đầu vào kích thước 5x5 và áp dụng một bộ lọc kích thước 3x3 với stride là 1, sau mỗi lần tích chập, bộ lọc sẽ di chuyển 1 pixel sang phải hoặc xuống dưới (hoặc theo chiều xác định bởi stride) để tiếp tục tích chập.

Nếu stride là 2, bộ lọc sẽ di chuyển 2 pixel sang phải hoặc xuống dưới sau mỗi lần tích chập. Điều này sẽ làm giảm kích thước của feature map được tạo ra sau mỗi lần tích chập.



Hình 2 5 Feature Map: lớp tích chập của ma trận ảnh 5x5 nhân với ma trận bộ lọc 3x3

Sử dụng stride trong các lớp convolutional có thể ảnh hưởng đến kích thước của feature map sau mỗi lần áp dụng bộ lọc. Stride lớn hơn có thể dẫn đến giảm kích thước của feature map đầu ra, trong khi stride nhỏ hơn giữ nguyên hoặc tăng kích thước của feature map đầu ra.

2.3.3 Lớp gộp – Pooling Layer

Lớp Pooling được sử dụng để giảm kích thước của bản đồ đặc trưng, từ đó giảm số lượng tham số cần tính toán và làm giảm overfitting. Phương pháp phổ biến trong lớp Pooling là Max Pooling, trong đó giá trị lớn nhất trong mỗi vùng pooling sẽ được giữ lại và các giá trị khác sẽ bị loại bỏ. Các phương thức lấy mẫu phổ biến trong lớp Pooling:

Kiểu	Max Pooling	Average Pooling
Chức năng	Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng	Từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng
Minh Họa		
Nhận xét	Bảo toàn các đặc trưng đã phát hiện Được sử dụng thường xuyên	Giảm kích thước feature map Được sử dụng trong mạng LeNet

Bảng 1 3 Lớp gộp Pooling

2.3.4 Lớp FC – Fully connected

Lớp kết nối đầy đủ (hay còn gọi là lớp ẩn) lớp cuối cùng trong mạng tích chập. Tại lớp mạng này, mỗi một nơ-ron của layer này sẽ liên kết tới mọi nơ-ron của lớp khác. Để đưa ảnh từ các layer trước vào mạng này, buộc phải dãn phẳng bức ảnh ra thành 1 vector thay vì là mảng nhiều chiều như trước. Tại layer cuối cùng sẽ sử dụng 1 hàm kinh điển trong học máy mà bất kì ai cũng từng sử dụng đó là softmax để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các lớp trước đó.

2.4 Tham số tối ưu

2.4.1 Activation Functionx (Hàm kích hoạt)

Một tính năng quan trọng khác trong mạng neural là hàm kích hoạt (Activation Function). Chúng có thể được đặt trong hoặc ở cuối của một mạng neural. Hàm kích hoạt cho phép một mạng neural thực hiện biến đổi phi tuyến tính trên ánh xạ tuyến tính giữa đầu vào và đầu ra của lớp mà nó được kết nối . Nói một cách đơn giản, khi áp dụng vào một đầu ra của 5 loại khác nhau, hàm kích hoạt giúp dự đoán loại nào có thể là đúng nhất và do đó trả về 1 cho loại đó và 0 cho phần còn lại. Hàm kích hoạt có một ngưỡng, nếu đạt được, sẽ khiến nó trả về 1, nếu không sẽ trả về 0. Ba hàm kích hoạt phổ biến nhất bao gồm: sigmoid, Tanh - tangent hyperbolique, và ReLU - Rectified Linear Units.

2.4.2 Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) đã được sử dụng rất phổ biến trong những ngày gần đây. Lý do chính là toán học đơn giản phía sau nó làm cho việc áp dụng nó vào các bộ phân loại với số lớp cao trở nên dễ dàng hơn. Logic đơn giản của nó được hiển thị trong hai phương trình sau:

$$\text{Đầu vào} < 0: f(x) = 0 \quad (1)$$

$$\text{Đầu vào} \geq 0: f(x) = x \quad (2)$$

Điều này cho thấy độ dốc của đồ thị của hàm tại bất kỳ điểm nào sẽ là hoặc 0 hoặc 1. Tại bất kỳ điểm nào độ dốc cũng không bao giờ bão hòa, do đó hàm kích hoạt này tránh được vấn đề gradient biến mất (vanishing gradient problem). Tuy nhiên, một vấn đề tiêu cực là nếu một nơ-ron cụ thể không bao giờ được kích hoạt với bất kỳ đầu vào nào, độ dốc luôn là không, và có thể xảy ra vấn đề gradient chết (dead gradient problem).

2.4.3 Softmax

Một hạn chế của hàm kích hoạt ReLU là nó chỉ nên được sử dụng trong các lớp ẩn của mạng neural. Do đó, cho lớp đầu ra cuối cùng của mạng neural tích chập, một lớp Softmax được sử dụng. Hàm softmax có khả năng chuyển đổi đầu ra của lớp tuyến tính cuối cùng của một mạng neural đa lớp thành xác suất của lớp nào có khả năng cao nhất là được dự đoán đúng. Nó tạo ra một phân phối xác suất của các lớp được dự đoán - tổng của đó nên bằng một.

2.4.4 Forward Propagation (Lan truyền thuận)

Khi ba lớp chính - lớp tích chập, lớp gom nhóm và lớp kết nối đầy đủ - được kết hợp với một hàm kích hoạt và một lớp softmax, quá trình lấy đầu vào từ lớp tích chập đến sinh ra đầu ra ở lớp đầu ra được gọi là lan truyền thuận (forward propagation). Mỗi lớp riêng lẻ nhận đầu vào từ lớp trước đó và xử lý nó theo hàm kích hoạt, sau đó chuyển mỗi đầu ra đến lớp tiếp theo. Mạng truyền ngược là quan trọng cho lan truyền thuận; để tạo ra một đầu ra, việc không cho phép dữ liệu được đưa ngược lại trong mạng trong quá trình tạo ra đầu ra là điều cần thiết. Điều này diễn ra tại mỗi nơ-ron trong một lớp ẩn hoặc lớp đầu ra trong hai bước: giai đoạn tiền kích hoạt và giai đoạn kích hoạt.

Trong giai đoạn tiền kích hoạt, tổng trọng số của các đầu vào - phép biến đổi tuyến tính của trọng số đối với các đầu vào có sẵn. Tổng hợp này sau đó được lấy và dựa trên hàm kích hoạt, nơ-ron quyết định liệu thông tin này có được truyền đi hay

không. Tổng trọng số tính toán này sau đó được chuyển đến giai đoạn kích hoạt, nơi phi tuyến tính được thêm vào mạng dựa trên hàm toán học của hàm kích hoạt.

2.4.5 Backward Propagation (Lan truyền ngược)

Các mạng neural sử dụng một giá trị ngẫu nhiên của trọng số khi chúng được khởi tạo ban đầu. Điều này có vẻ ngược đi với trực giác, nhưng trong học có giám sát, không quan trọng điểm xuất phát ban đầu của trọng số của các nơ-ron trong mỗi lớp. Nếu mạng được huấn luyện đủ lần và bộ phân loại đủ mạnh mẽ và lan truyền đều đặn, cuối cùng sẽ đạt được trọng số chính xác. Các trọng số của mỗi nơ-ron trong mỗi lớp phải được cập nhật liên tục và điều này được thực hiện thông qua lan truyền ngược (backward propagation).

2.4.6 Loss Function (Hàm mất mát) và Optimization (Tối ưu hóa)

Khi có một đầu ra từ mạng trong một vòng lặp, đầu ra đó được so sánh với giá trị thực tế. Việc này được thực hiện thông qua hàm mất mát (loss function) - một phép đo hiệu suất có thể tổng quát hóa cho phép người dùng hiểu được mức độ gần giá trị dự đoán của Mạng Neural đến đầu ra Thực tế. Mục tiêu của chương trình học máy là luôn luôn làm giảm thiểu hàm mất mát này về không. Độ dốc của hàm mất mát được sử dụng để tối ưu hóa các giá trị trọng số nội tại của mạng để giảm giá trị của hàm mất mát. Độ dốc này giúp người dùng hiểu được mức độ thay đổi của tổng lỗi nếu chúng ta thay đổi trọng số của nơ-ron bằng bất kỳ giá trị tùy ý nhỏ nào. Một độ dốc dương sẽ chỉ vào việc giảm kích thước của trọng số vì tại trọng số được khởi tạo ngẫu nhiên đó, nếu tăng kích thước trọng số, tổng lỗi sẽ tăng. Một độ dốc âm sẽ chỉ vào việc tăng kích thước của trọng số vì tại trọng số được khởi tạo ngẫu nhiên đó, tổng lỗi đang giảm. Mục tiêu là đạt được một độ dốc bằng 0 sau quá trình tăng hoặc giảm kích thước của các trọng số. Quá trình này của việc tìm kiếm một giá trị nhỏ nhất được gọi là gradient descent. Cần phải chọn kích thước bước phù hợp để đảm bảo bộ phân loại có thể tìm được một giá trị nhỏ nhất toàn cầu và không phải là một giá trị nhỏ nhất cục bộ. Việc huấn luyện cẩn thận và các kích thước epoch (vòng lặp huấn luyện) phù hợp có thể đảm bảo điều này.

Khi mỗi lỗi được tính toán và độ dốc của mỗi lỗi được tính toán, sau đó thông tin này được gửi lại đến đầu của các lớp. Quá trình lan truyền ngược này cho phép các trọng số được cập nhật với một giá trị phù hợp để đảm bảo hiệu suất của bộ phân loại có thể được cải thiện. Khi mỗi trọng số được cập nhật, quan trọng là đảm bảo rằng mỗi thay đổi của trọng số là rất nhỏ, vì một sự thay đổi lớn trong các trọng số có thể làm cho hành vi của mạng neural trở nên không ổn định. Những thay đổi nhỏ có thể giúp xác định giá trị chính xác nhanh hơn vì quỹ đạo của mạng neural có thể được

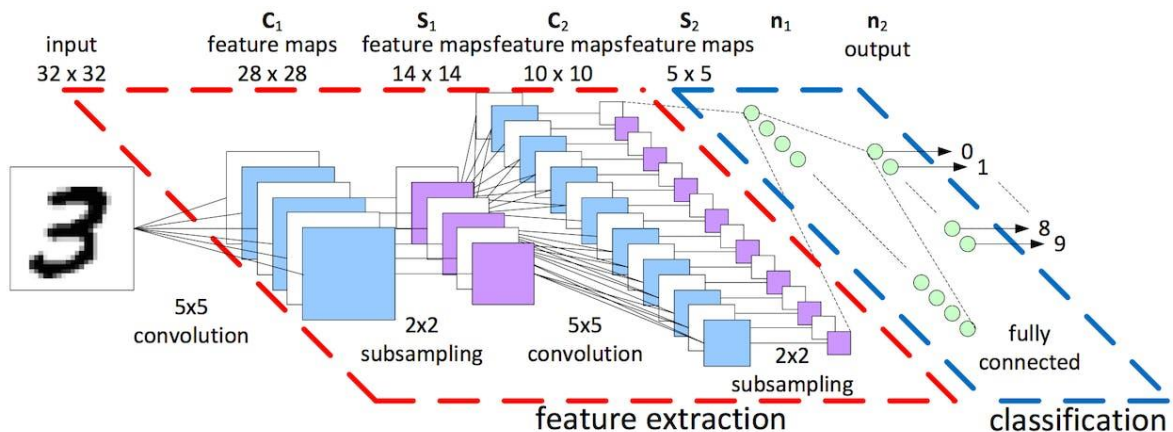
dự đoán và cục bộ hóa dễ dàng hơn. Đặc biệt khi các mạng neural này thường có nhiều phi tuyến tính và độ dốc hiện tại mà mạng sẽ có được được cục bộ hóa ở điểm cụ thể mà nó đang được tính toán, những thay đổi lớn đối với các trọng số cập nhật sẽ làm cho việc quan sát và phân tích kết quả của những cập nhật đó rất khó khăn. Các phương pháp sử dụng để cập nhật các trọng số này được gọi là các trình tối ưu hóa (optimizers). Một trình tối ưu hóa thông dụng là trình tối ưu hóa Adam - một biến thể của gradient descent ngẫu nhiên (stochastic gradient descent).

Adam là một thuật toán tối ưu hóa tốc độ học thích nghi được thiết kế đặc biệt cho việc huấn luyện các mạng neural sâu. Thuật toán hoạt động đặc biệt tốt vì nó tìm ra các tốc độ học cá nhân cho mỗi tham số thông qua sức mạnh của các phương pháp tốc độ học thích nghi. Adam có thể được coi là sự kết hợp của RMSprop và Stochastic Gradient Descent với momentum. Ý tưởng chính của RMSprop là giữ giá trị trung bình di chuyển của độ dốc bình phương cho mỗi trọng số. Sau đó, chúng ta chia độ dốc cho căn bậc hai của trung bình bình phương. Trong Gradient Descent Ngẫu nhiên, sử dụng xác suất ngẫu nhiên, thuật toán cố gắng làm giảm thiểu một hàm chi phí nhất định bằng cách di chuyển theo hướng hạ lưu dốc nhất định được xác định bởi đối của độ dốc âm.

2.4.5 Epochs (Vòng lặp huấn luyện)

Toàn bộ quá trình lan truyền thuận và lan truyền ngược sau đó được lặp lại một số lần để đạt được một điểm mà mạng neural được huấn luyện đủ trên dữ liệu huấn luyện để có thể đưa ra dự đoán với thông tin tương tự. Mạng neural mất vài vòng lặp để đạt được các tham số phù hợp để học. Mỗi vòng lặp được gọi là một epoch. Số lượng epoch phụ thuộc vào một số yếu tố: chất lượng của dữ liệu, tốc độ học của thuật toán, độ phức tạp của các lớp, phương pháp tối ưu hóa được sử dụng, và thậm chí cả việc khởi tạo ngẫu nhiên của các trọng số.

2.5 Nguyên lý hoạt động của một mạng nơ ron tích chập điển hình



Hình 2.6 Nguyên lý hoạt động của một mạng nơ ron

Hình ảnh sẽ được đưa vào lớp Conv1 (Convolutional kết hợp RELU) với bộ lọc (filter) có kích thước 5x5, áp dụng bước trượt 2, mỗi filter sẽ được dùng để tính tích chập với ảnh và cho ra một ảnh kết quả tương ứng là C_1 có kích thước 28x28.

Mỗi ảnh trên đều có kích thước tương ứng là 28x28. Sau đó, cả 28 ảnh này đều được cho qua lớp Pooling, sử dụng bộ lọc kích thước 2x2, bước trượt 2 ta được kết quả đầu ra sẽ là S_2 ảnh có kích thước 14x14.

Tiếp tục dữ liệu sẽ đi vào lớp Conv2. Tương tự như Conv1, ảnh sẽ được tính tích chập với filter và trả ra kết quả ảnh đầu ra C_2 có kích thước 10x10. Lớp Pooling tiếp theo sẽ tiếp tục giảm kích thước của ảnh xuống còn 5x5. Với kích thước đủ nhỏ như vậy, lớp Fully-connected tiếp theo sẽ xử lý và đưa ra kết quả phân lớp hay kết quả nhận dạng.

2.6 Ứng dụng

- **Nhận dạng hình ảnh:** CNN được sử dụng để nhận dạng đối tượng trong hình ảnh, như nhận dạng khuôn mặt, phân loại đối tượng, nhận dạng biển số xe, và nhận diện vật thể trong ảnh y khoa.
- **Xử lý ngôn ngữ tự nhiên:** CNN có thể được sử dụng để xử lý văn bản và ngôn ngữ tự nhiên, như phân loại văn bản, phân loại cảm xúc trong văn bản, hoặc tạo ra mô hình dịch máy.
- **Nhận dạng âm thanh:** CNN có thể giúp trong việc nhận dạng âm thanh, ví dụ như nhận dạng tiếng nói hoặc nhận dạng âm nhạc.
- **Xử lý video và chuỗi thời gian:** CNN được sử dụng rộng rãi trong việc phân loại và nhận dạng hành động trong video, cũng như trong các ứng dụng xử lý chuỗi thời gian khác như dự đoán chuỗi thời tiết, chuỗi tài chính, hoặc dữ liệu chuỗi thời gian khác.

- *Tự động lái xe và nhận dạng vật cản*: Trong lĩnh vực tự động lái xe, CNN được sử dụng để nhận dạng và phân loại vật cản, định hình môi trường xung quanh.
- *Y học và chẩn đoán y khoa*: Trong lĩnh vực y học, CNN có thể được sử dụng để phân loại và dự đoán bệnh lý từ ảnh chụp X-quang, MRI, hoặc các hình ảnh y khoa khác.

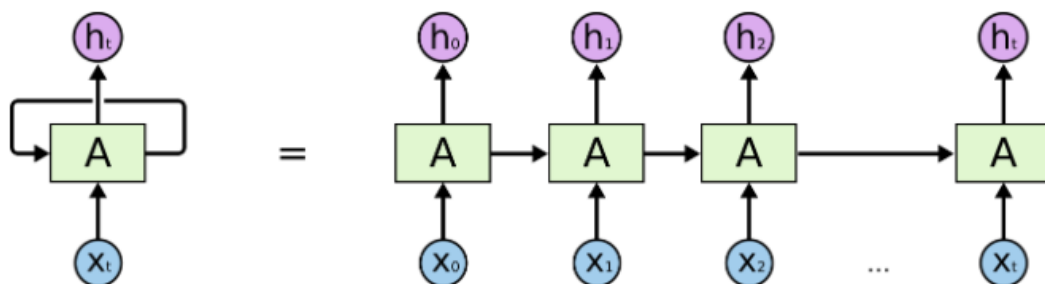
3. Bộ nhớ ngắn hạn dài hạn (LSTM)

3.1 Mạng lưới thần kinh tái phát bộ nhớ ngắn hạn dài hạn

LSTM RNN là một tập hợp các Mạng Nơ-ron có chức năng và cách hoạt động được lấy cảm hứng từ não bộ người. Mạng Nơ-ron Tích Lũy (Recurrent Neural Network - RNN) khác biệt so với Mạng Nơ-ron Nhân Tạo thông thường ở cách nó cụ thể sử dụng thông tin tuần tự. Nếu hai hình ảnh của mèo và chó được đưa vào một mạng nơ-ron nhân tạo, mạng sẽ nhận ra một số mẫu và dự đoán một hình ảnh là của chó hoặc mèo. Tại bất kỳ thời điểm nào, dự đoán của mạng nhân tạo không phụ thuộc vào hình ảnh trước đó. Mỗi dự đoán về chó hoặc mèo sẽ dựa trên hình ảnh đó và không phụ thuộc vào dữ liệu trước đó.

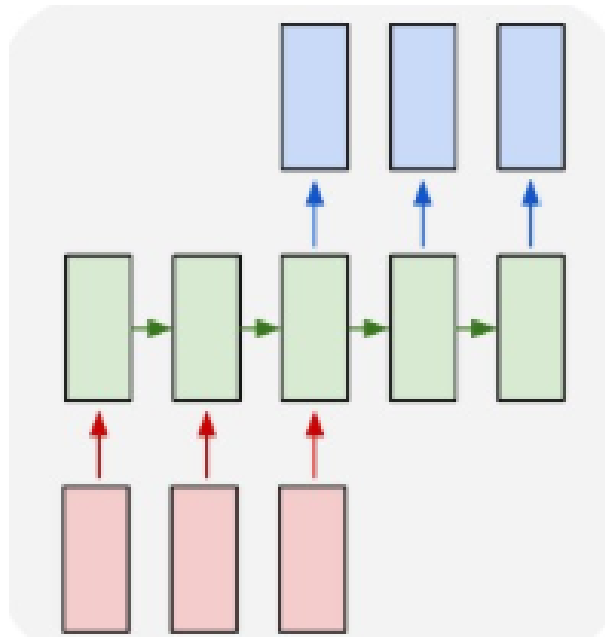
Tuy nhiên, có thể có nhiều tình huống trong đó dữ liệu trước đó rất quan trọng để dự đoán dữ liệu hiện tại. Ví dụ, dự đoán giá trị thị trường tài chính của cổ phiếu của một công ty, dự đoán từ tiếp theo trong một câu v.v. Trong những tình huống như vậy, một mạng nơ-ron - như mạng nơ-ron tái phát - là cần thiết, nơi mà nó có thể có "bộ nhớ" để lưu trữ thông tin tuần tự và sử dụng nó khi thực hiện một dự đoán cụ thể.

Trong mạng nơ-ron truyền thống, thường được cho rằng đầu vào và đầu ra là độc lập với nhau. Tuy nhiên, điều này có thể trở nên bất lợi khi ví dụ như trong việc dự đoán một từ trong một câu như đã đề cập trước đó. Từ 'recurrent' được sử dụng cho mạng nơ-ron này bởi vì nó thực hiện cùng một nhiệm vụ trên mỗi phần tử của một chuỗi dữ liệu, với mỗi đầu ra phụ thuộc vào dữ liệu trước đó. Như có thể thấy từ hình ảnh dưới đây, trong đó x_t đại diện cho đầu vào, A đại diện cho hàm kích hoạt của lớp ẩn, và h đại diện cho giá trị của trạng thái hiện tại.



Hình 2 7 Cách mà một RNN sử dụng bộ nhớ của nó.

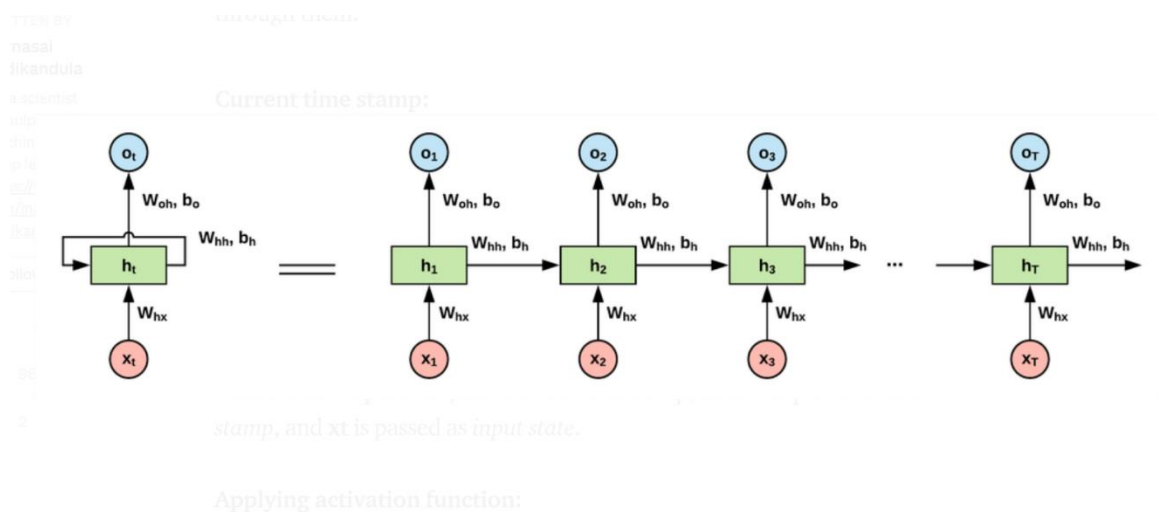
Có nhiều loại RNN khác nhau: một đến một, một đến nhiều, nhiều đến một, nhiều đến nhiều và hai chiều nhiều đến nhiều. Nhiều đến một là loại RNN mà nó nhận một chuỗi của nhiều dữ liệu đầu vào và cung cấp một đầu ra có kích thước cố định. Hình ảnh sau mô tả kiến trúc của một RNN nhiều đến một trong đó mũi tên màu đỏ đại diện cho đầu vào và mũi tên màu xanh lá cây và xanh dương đại diện cho đầu ra của các lớp ẩn:



Hình 2 8 Mô hình Many-to-One RNN

Các trọng số và sai số áp dụng vào các lớp ẩn của một RNN luôn luôn giống nhau để cho phép nó ghi nhớ thông tin được xử lý qua chúng .

$$h_t = f(h_{t-1}, x_t)$$



Hình 2 9 Thông tin lưu giữ của một RNN

Ở đây O_t biểu thị trạng thái đầu ra, h_t là dấu thời gian hiện tại, h_{t-1} là dấu thời gian trước đó và x_t được chuyển tiếp làm đầu vào cho giai đoạn tiếp theo. Hàm kích hoạt tanh được áp dụng trong mỗi ô ứng viên để xác định giá trị trạng thái nội tại và sau đó được sử dụng để cập nhật giá trị của trạng thái ẩn. Điều này được mô tả trong phương trình sau:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t)(1)$$

Trong phương trình 4W, đại diện cho trọng số, với W_{hh} biểu thị các trọng số từ lớp ẩn trước đó, và W_{hx} biểu thị các trọng số tại trạng thái đầu vào hiện tại. Hàm kích hoạt tanh được sử dụng vì nó thực hiện một phi tuyến tính trong đó giá trị đầu ra nằm trong phạm vi từ -1 đến 1. Tính phi tuyến của nó cho phép nhiều lớp LSTM được xếp chồng lên nhau. Đầu ra ở giai đoạn cuối được mô tả tốt nhất qua phương trình sau:

$$y_t = (W_{hy}h_t)(2)$$

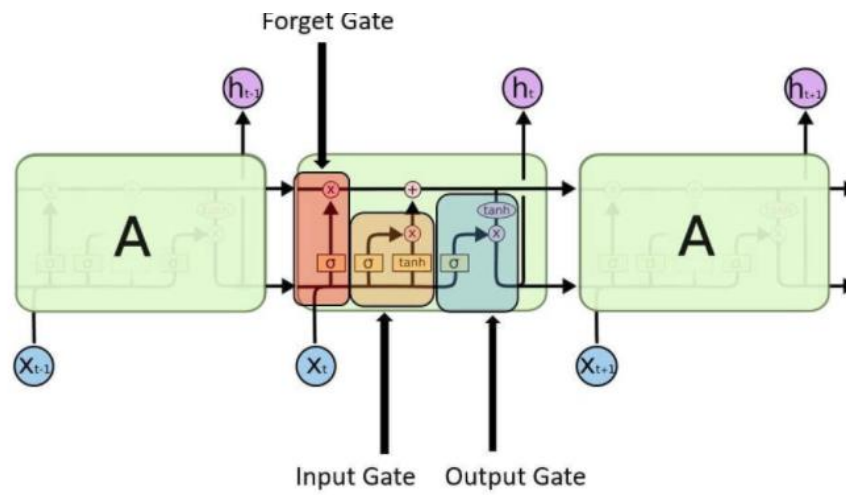
Trong phương trình (2), y_t biểu thị trạng thái đầu ra và W_{hy} biểu thị trọng số tại trạng thái đầu ra.

Mạng nơ-ron tái phát LSTM sử dụng lan truyền ngược để cải thiện giá trị của trọng số và sai số được gán tại mỗi tầng trong quá trình huấn luyện để đạt được trạng thái đầu ra chính xác hơn. Hàm mất mát giúp tính toán sự khác biệt giữa trạng thái đầu ra hiện tại dự đoán và trạng thái đầu ra thực tế. Hàm mất mát thông thường được sử dụng là mean squared error (sai số bình phương trung bình). Như đã đề cập trong phần 3.1, độ dốc của hàm mất mát được sử dụng để tối ưu hóa giá trị của các trọng số nội tại để cố gắng giảm giá trị của hàm mất mát về gần zero. Vì trọng số là giống nhau cho tất cả các tầng ẩn trong mạng nơ-ron tái phát, độ dốc được tính toán cho mỗi bước thời gian có thể được kết hợp với nhau. Kết quả, trọng số của nơ-ron tái phát và tầng dày đặc đầu ra có thể được cập nhật cùng nhau.

Có hai loại vấn đề mà người ta có thể gặp trong quá trình lan truyền ngược: độ dốc biến mất và độ dốc bùng nổ. Nói một cách đơn giản, độ dốc biến mất là một vấn đề xảy ra khi đóng góp từ các bước trước trở nên không đáng kể trong bước gradient descent. Điều này có thể xảy ra khi sai số giữa đầu ra dự đoán và đầu ra thực tế có đạo hàm riêng với trọng số nhỏ hơn 1. Giá trị này sau đó được nhân với tỷ lệ học tập - đã là một giá trị rất không đáng kể - dẫn đến giá trị cuối cùng trở nên còn nhỏ hơn. Sau đó, khi trọng số được cập nhật bằng giá trị này, không có sự thay đổi đáng kể trong trọng số và do đó gặp phải vấn đề độ dốc biến mất. Nó khiến cho RNN quên đi các phụ thuộc dài hạn vì các giá trị đó không được chuyển tiếp sang các vòng lặp tiếp

theo. Tương tự, độ dốc bùng nổ là vấn đề xảy ra khi đạo hàm riêng của sai số theo trọng số là cực kỳ cao, dẫn đến gán một giá trị cực kỳ lớn cho trọng số và do đó độ dốc bùng nổ.

Các vấn đề này được giải quyết bằng cách thêm LSTM vào RNN: Bộ nhớ ngắn hạn dài. Trong các trạng thái ô LSTM, các phụ thuộc và mối quan hệ dài hạn được mã hóa trong các vector trạng thái và chính là đạo hàm của trạng thái ô này có thể ngăn gradient của LSTM biến mất. LSTM bao gồm ba bước: Cổng quên, Cổng đầu vào, Cổng đầu ra. Đây được mô tả trong hình ảnh dưới đây:



Hình 2 10 Cấu trúc của một ô LSTM (Long Short-Term Memory)

3.2 Cấu trúc của mạng LSTM

- **Cổng Quên:** Cơ bản, cổng này cho phép ô LSTM biết được phải nhớ bao nhiêu và phải quên bao nhiêu thông tin từ dữ liệu được cung cấp. Tại mỗi bước thời gian, cổng này xác định thông tin nào sẽ được bỏ đi từ ô LSTM. Điều này được thực hiện bằng cách sử dụng một hàm sigmoid. Nó xem xét dữ liệu đầu vào trước đó và trạng thái trước đó, sau đó đưa ra một giá trị giữa 0 và 1. Giá trị 0 có nghĩa là loại bỏ thông tin này và giá trị 1 có nghĩa là giữ thông tin này. Phương trình đầu ra của cổng quên được biểu diễn như sau:

$$f_t = \sigma[W_t(h_{t-1}, x_t)] \quad (3)$$

- **Cổng Đầu Vào:** Cổng đầu vào còn được gọi là cổng cập nhật. Nó quyết định một phần nào đó của đơn vị này sẽ được thêm vào trạng thái hiện tại. Hàm sigmoid quyết định các giá trị nào được thông qua bằng cách đưa ra đầu ra là 0 hoặc 1. Hàm tanh gán trọng số cho các giá trị này trên một thang điểm từ -1 đến 1 dựa trên mức độ quan trọng của chúng. Công thức của cổng đầu vào được phản ánh trong phương trình sau:

$$\tanh = (W_c[h_{t-1}, x_t])\phi\sigma[W_i h_{t-1}, x_t] \quad (4)$$

- **Cổng Đầu Ra:** Cổng này chịu trách nhiệm quyết định phần nào của ô hiện tại sẽ được đưa ra đầu ra. Hàm sigmoid được sử dụng để quyết định các giá trị nào sẽ đi qua, tanh đánh trọng số cho các giá trị này trên thang điểm từ -1 đến 1 và sau đó giá trị từ hàm sigmoid và tanh được nhân với nhau. Công thức cho đầu ra của cổng đầu ra có thể được biểu diễn trong phương trình sau:

$$o_t = \sigma([W_0.[h_{t-1}, x_t]]) \quad (5)$$

Đầu ra từ ô LSTM sau đó có thể được biểu diễn bằng phương trình sau:

$$h_t = o_t \odot \tanh(c_t) \quad [25] \quad (6)$$

Như đã đề cập trước đó, thuật ngữ lỗi trong RNN là tổng của tất cả các gradient được tính trong các lớp vì tất cả các trọng số và sai số đều giống nhau cho tất cả các lớp ẩn. Nếu đạo hàm của lỗi theo trọng số được biểu diễn như tổng của các đạo hàm riêng lẻ thông qua phương trình sau:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W} \quad (7)$$

Để xảy ra vấn đề gradient biến mất, tổng của các đạo hàm riêng lẻ này sẽ cần phải tiến gần đến giá trị 0. Để ngăn chặn điều này, cách đơn giản nhất là ngăn một số đạo hàm riêng lẻ này hội tụ về giá trị 0. Chúng phải hội tụ về một giá trị không bằng không. Khi một ô LSTM được thêm vào một RNN, vector trạng thái của ô LSTM được thêm vào thuật ngữ lỗi. Nếu lấy đạo hàm riêng của thuật ngữ gradient được đề cập trong phương trình (7), đạo hàm riêng của vector trạng thái của ô LSTM chứa vector của cổng quên trong việc kiểm soát giá trị gradient tốt hơn ở mỗi vòng lặp thông qua việc cập nhật tham số thích hợp của cổng quên. Sự hiện diện của cổng quên này cho phép hệ thống quyết định tại mỗi bước thời gian thông tin nào không nên bị loại bỏ và sau đó có thể cập nhật các thông số của mô hình tương ứng.

Ví dụ, nếu tại một thời điểm t , gradient của lỗi, $\frac{\partial E}{\partial W}$, tiến gần đến 0, vector các hoạt động của cổng quên trong thuật ngữ gradient cùng với cấu trúc cộng của nó cho phép LSTM tìm ra cập nhật tham số sao cho tại thời điểm $t+1$, gradient của lỗi, $\frac{\partial E}{\partial W}$, không nên bằng không. Do đó, gradient không biến mất.

Ngoài hàm kích hoạt của cổng quên, hình dạng và cách thức nó xuất hiện trong gradient của lỗi cũng quan trọng. Sau khi vector trạng thái của ô LSTM được vi phân, nó bao gồm bốn điều kiện cộng khác nhau. Khi bốn điều kiện khác nhau này hiện có trong gradient của lỗi, tại mỗi vòng lặp khi bốn điều kiện này được cập nhật, không có khả năng tất cả bốn điều kiện đều hội tụ về 0. Khác với hàm gradient lỗi của một

RNN thông thường chỉ bao gồm một điều kiện, với ô LSTM được thêm vào RNN, các điều kiện cộng này làm cho việc hội tụ về 0 trở nên ít có khả năng hơn.

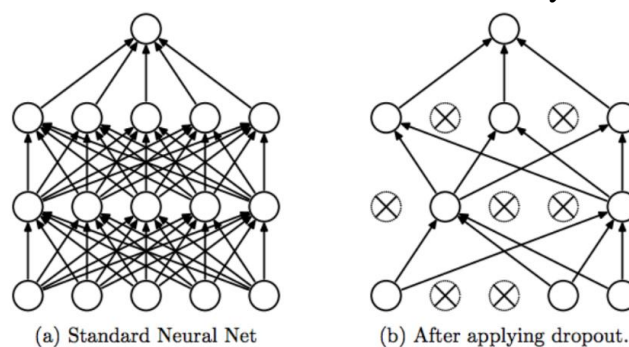
Điều này giúp tránh vấn đề gradient biến mất nhờ sự hiện diện của hàm kích hoạt của cổng quên và tính chất cộng của hàm. Vấn đề gradient phồng lên xảy ra theo cách tương tự như vấn đề gradient biến mất. Khi các thuật ngữ con gradient được kết hợp, nếu giá trị của chúng lớn hơn một, vấn đề gradient phồng lên sẽ xảy ra. Tuy nhiên, ô LSTM giải quyết vấn đề này cũng như giải quyết vấn đề gradient biến mất.

Còn một số tính năng khác cần thiết để triển khai mạng nơ-ron:

- **Dropout:** Tại mỗi tầng của LSTM, dropout được thêm vào từng tầng. Đơn giản, dropout là việc loại bỏ một số đơn vị trong tập các đơn vị trong quá trình huấn luyện. Việc loại bỏ một số đơn vị hoặc neuron có nghĩa là những đơn vị hoặc neuron cụ thể này không được xem xét trong quá trình lan truyền thuận hoặc lan truyền ngược. Những nút này bị loại bỏ với xác suất $1-p$ hoặc giữ lại với xác suất p để giảm tổng thể của mạng. Các cạnh đi vào và đi ra của một nút bị loại bỏ cũng được loại bỏ.

Dropout là cần thiết để ngăn chặn hiện tượng quá khớp của mạng. Quá khớp xảy ra khi một mô hình huấn luyện 'quá tốt' trên dữ liệu huấn luyện. Vì hầu hết các thông số được chiếm bởi tầng kết nối đầy đủ, các neuron bên trong nó phát triển một sự phụ thuộc lẫn nhau trong quá trình huấn luyện. Điều này làm hại cho sức mạnh của từng neuron dẫn đến quá khớp với dữ liệu huấn luyện. Quá khớp có hại cho mô hình vì nó học các mẫu và nhiễu trong dữ liệu huấn luyện đến mức hiệu suất của mô hình trên một bộ dữ liệu mới sẽ bị ảnh hưởng tiêu cực.

Trong giai đoạn huấn luyện, tại mỗi lớp ẩn, một phân tử ngẫu nhiên p của các nút và chức năng kích hoạt của chúng sẽ bị bỏ qua. Trong giai đoạn kiểm thử, tất cả các chức năng kích hoạt sẽ được sử dụng nhưng sẽ được giảm xuống một hệ số p để tính đến các chức năng kích hoạt bị thiếu trong quá trình huấn luyện. Cách các sự phụ thuộc lẫn nhau của các nơ-ron được loại bỏ có thể được thấy trong hình sau:



Hình 3. 1 Sự giảm thiểu của sự phụ thuộc lẫn nhau giữa các neuron trước và sau khi thực hiện dropout

Các ưu điểm của việc sử dụng dropout như đã được đề cập trước đó bao gồm việc tránh quá khớp, nó cho phép mạng neural học được các đặc trưng mạnh mẽ hơn từ dữ liệu huấn luyện có thể được áp dụng cho các bộ dữ liệu ngẫu nhiên khác, và nó cũng giảm tổng thời gian cần thiết để chạy một epoch.

- **Batch Normalization (Chuẩn hóa Batch):**

Huấn luyện LSTMs đôi khi trở thành một công việc khó khăn. Tại mỗi vòng lặp, các tham số của mỗi tầng sẽ thay đổi, dẫn đến sự thay đổi phân phối của đầu vào của các tầng khác. Điều này làm chậm quá trình huấn luyện. Batch normalization cải thiện sự ổn định của mạng bằng cách giảm sự thay đổi covariance của các giá trị đơn vị ẩn. Đầu ra của tầng kích hoạt trước đó được chuẩn hóa bằng cách trừ trung bình của batch và sau đó chia cho độ lệch chuẩn của batch.

Tuy nhiên, việc chuẩn hóa này có thể ảnh hưởng tiêu cực đến tối ưu hóa trọng số của tầng kế tiếp. Nếu việc tối thiểu hóa hàm mất mát ở giai đoạn đó liên quan đến loại bỏ chuẩn hóa này, thì gradient descent ngẫu nhiên sẽ thực hiện thao tác đó. Trong trường hợp như vậy, batch normalization có thể trở nên vô dụng.

Để khắc phục vấn đề này, batch normalization thêm hai tham số có thể điều chỉnh vào mỗi tầng: gamma và beta. Những tham số này hoạt động như các hệ số tỉ lệ và chuyển đổi đầu ra đã được chuẩn hóa, cho phép gradient descent ngẫu nhiên tối ưu hóa chỉ hai tham số này. Phương pháp này duy trì tính ổn định của mạng mà không cần sửa đổi quá nhiều trọng số.

- **Dense Layer (Tầng Dày Đặc):**

Tầng dày đặc là nơi mà mạng neural có một neuron cho mỗi kết quả dự kiến. Vì vậy, nếu có 5 kết quả, sẽ có 5 neuron trong tầng dày đặc. Mỗi neuron sẽ kết nối với tất cả các neuron trong các tầng trước đó, do đó tầng kết nối này mật độ cao được gọi là tầng dày đặc. Tầng này có ma trận trọng số riêng W , một vector sai số b , và kết quả tính toán từ các tầng trước đó.

CHƯƠNG 2: CƠ CHẾ NHẬN DẠNG HÀNH ĐỘNG TRONG VIDEO .

1. Cơ chế nhận dạng

Cơ chế nhận dạng hành động qua video sử dụng mạng tích chập (CNN) và LSTM là một phương pháp kết hợp hai kiến trúc mạng neural network để nhận dạng và hiểu các hành động trong dữ liệu video. Quá trình này bắt đầu bằng việc thu thập và chuẩn bị dữ liệu video, trong đó mỗi video được chia thành các khung hình (frames) để xử lý. Mục tiêu của mạng CNN là trích xuất các đặc trưng không gian từ các khung hình, như là các đặc điểm và cấu trúc của các hành động trong video. Các khung hình sẽ được đưa qua các lớp Convolutional và Pooling trong CNN để học và trích xuất thông tin. Sau khi đã trích xuất các đặc trưng không gian từ CNN, đầu ra sẽ được chuyển qua mạng LSTM. Mạng LSTM chủ yếu là để hiểu thông tin về thời gian từ dữ liệu video. LSTM có khả năng học cấu trúc chuỗi thời gian, giúp mô hình nhận biết mối quan hệ giữa các khung hình liên tiếp, từ đó dự đoán hành động tiếp theo dựa trên thông tin thời gian đã học.

Quá trình huấn luyện và điều chỉnh mô hình bao gồm việc sử dụng dữ liệu video huấn luyện để mô hình hóa và điều chỉnh các tham số. Việc kết hợp CNN và LSTM cung cấp khả năng học và nhận biết cả thông tin không gian và thời gian từ dữ liệu video, từ đó giúp mô hình nhận dạng hành động một cách hiệu quả và chính xác.

2. Nghiên cứu liên quan

Nhiệm vụ của bài toán nhận dạng hành động con người là xác định và phân loại các hành động mà con người thực hiện từ dữ liệu video hoặc hình ảnh. Nhận diện hoạt động của con người đóng vai trò quan trọng trong tương tác con người-với-con người và các mối quan hệ giữa cá nhân. Bởi vì nó cung cấp thông tin về danh tính của một người, tính cách và trạng thái tâm lý của họ, việc trích xuất thông tin này là khó khăn. Khả năng của con người trong việc nhận diện hoạt động của người khác là một trong những chủ đề chính của các lĩnh vực khoa học như thị giác máy tính và học máy. Kết quả của nghiên cứu này đã tạo ra nhiều ứng dụng, bao gồm hệ thống giám sát video, tương tác giữa con người và máy tính, và robot hóa cho việc đặc trưng hành vi của con người, yêu cầu một hệ thống nhận diện nhiều hoạt động.

Trong nhiều kỹ thuật phân loại khác nhau, xuất hiện hai câu hỏi chính: "Hành động là gì?" (tức là vấn đề nhận diện) và "Ở đâu trong video?" (tức là vấn đề xác định vị trí). Khi cố gắng nhận diện hoạt động của con người, người ta phải xác định các trạng thái động lực của một người, để máy tính có thể nhận diện hoạt động này một cách hiệu quả. Các hoạt động của con người, như "đi bộ" và "chạy", xuất hiện tự nhiên trong cuộc sống hàng ngày và tương đối dễ nhận diện. Tuy nhiên, các hoạt động

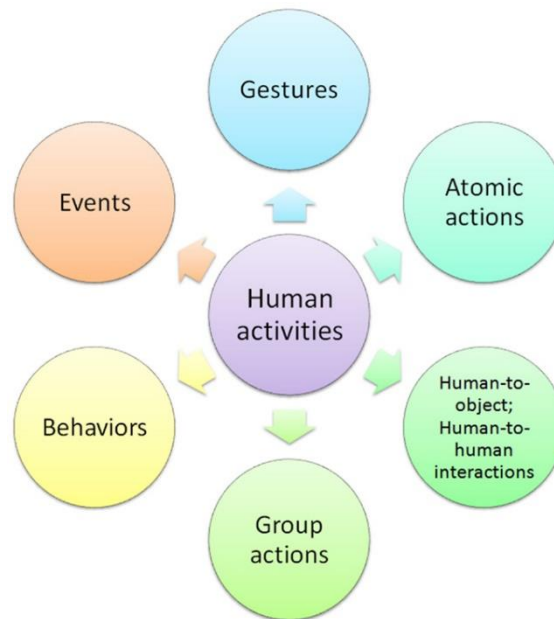
phức tạp hơn, như "bóc quả táo", khó nhận diện hơn. Các hoạt động phức tạp có thể được phân rã thành các hoạt động đơn giản hơn, mà thường dễ nhận diện hơn. Thông thường, việc phát hiện các đối tượng trong một cảnh có thể giúp hiểu rõ hơn về các hoạt động của con người, bởi nó có thể cung cấp thông tin hữu ích về sự kiện đang diễn ra (Gupta và Davis, 2007).

Hầu hết các công việc trong việc nhận diện hoạt động của con người giả định một cảnh tượng trung tâm của hình nền không quá phức tạp, nơi diễn viên tự do thực hiện một hoạt động. Việc phát triển một hệ thống nhận diện hoạt động của con người hoàn toàn tự động, có khả năng phân loại hoạt động của một người với sai số thấp, là một nhiệm vụ khó khăn do các vấn đề như nhiễu nền, che khuất một phần, thay đổi tỷ lệ, góc nhìn, ánh sáng và diện mạo, và độ phân giải khung hình. Ngoài ra, việc ghi chú vai trò hành vi mất thời gian và yêu cầu kiến thức về sự kiện cụ thể. Hơn nữa, sự tương đồng trong và giữa các lớp làm cho vấn đề trở nên đầy thách thức. Điều này có nghĩa là các hành động thuộc cùng một lớp có thể được thể hiện bởi các người khác nhau với các chuyển động cơ thể khác nhau, và các hành động giữa các lớp khác nhau có thể khó phân biệt vì chúng có thể được biểu diễn bằng thông tin tương tự. Cách mà con người thực hiện một hoạt động phụ thuộc vào thói quen của họ, điều này khiến việc xác định hoạt động cơ bản khá khó khăn. Ngoài ra, việc xây dựng một mô hình hình ảnh để học và phân tích các chuyển động của con người theo thời gian thực với các bộ dữ liệu thử nghiệm không đầy đủ là nhiệm vụ khó khăn.

Để vượt qua những vấn đề này, cần có một nhiệm vụ bao gồm ba thành phần, bao gồm: (i) loại bỏ nền (Elgammal et al., 2002; Mumtaz et al., 2014), trong đó hệ thống cố gắng phân tách các phần của hình ảnh không thay đổi theo thời gian (nền) khỏi các đối tượng đang di chuyển hoặc thay đổi (foreground); (ii) theo dõi con người, trong đó hệ thống xác định chuyển động của con người theo thời gian (Liu et al., 2010; Wang et al., 2013; Yan et al., 2014); và (iii) phát hiện hành động và đối tượng của con người (Pirsiavash and Ramanan, 2012; Gan et al., 2015; Jainy et al., 2015), trong đó hệ thống có khả năng xác định vị trí một hoạt động của con người trong một hình ảnh.

Mục tiêu của việc nhận dạng hoạt động của con người là phân tích các hoạt động từ chuỗi video hoặc hình ảnh cố định. Được truyền cảm hứng từ thực tế này, các hệ thống nhận diện hoạt động con người hướng đến việc phân loại chính xác dữ liệu đầu vào vào các danh mục hoạt động cơ bản tương ứng. Tùy thuộc vào mức độ phức tạp, các hoạt động của con người được phân loại thành: (i) các cử chỉ; (ii) các hành động nguyên tử; (iii) tương tác con người-với-vật hoặc con người-với-con người; (iv) các

hành động nhóm; (v) hành vi; và (vi) sự kiện. Hình 3.2 minh họa sự phân tách của các hoạt động của con người dựa trên độ phức tạp của chúng.



Hình 3. 2 Phân tích hoạt động của con người.

3. Thách thức trong nhận dạng hành động

Việc xây dựng một hệ thống nhận dạng hành động hoàn toàn tự động với khả năng nhận dạng chính xác cao là một thách thức đối với các nghiên cứu. Điều này là do các yếu tố (chủ quan và khách quan) ảnh hưởng quá trình thu nhận ảnh và tạo ra các bức ảnh có độ khác biệt rất lớn của cùng một hành động. Có thể liệt kê ra đây các yếu tố ảnh hưởng chủ yếu tới độ chính xác của một hệ thống nhận dạng hành động con người:

- **Sự đa dạng của hành động:** Con người có thể thực hiện rất nhiều loại hành động khác nhau, từ đơn giản đến phức tạp. Điều này đòi hỏi các hệ thống nhận dạng phải có khả năng phân biệt được các hành động này.
- **Sự thay đổi của hành động:** Cùng một hành động có thể được thực hiện bởi nhiều người khác nhau với các cách thức khác nhau. Ví dụ, cách một người đi bộ có thể khác nhau tùy thuộc vào giới tính, chiều cao, trọng lượng, tuổi, v.v.
- **Sự che giấu của hành động:** Một số hành động có thể được thực hiện một cách che giấu, ví dụ như khi một người đang giấu tay trong túi hoặc đang đeo kính mặt dày.
- **Sự nhiễu sóng:** Các hệ thống nhận dạng hành động có thể gặp khó khăn trong việc phân biệt các hành động khi có nhiễu sóng, ánh sáng yếu hoặc các điều kiện môi trường khác.

4. Qui trình thực hiện

• Bước 1: Chuẩn bị dữ liệu

Để huấn luyện mô hình nhận dạng hành động con người, chúng ta cần có một tập dữ liệu gồm các video về các hành động con người với nhãn là tên hành động tương ứng. Một trong những tập dữ liệu phổ biến được sử dụng trong lĩnh vực này là tập dữ liệu UCF101¹. Tập dữ liệu này bao gồm 13.320 đoạn video được phân loại thành 101 loại hành động khác nhau. Tất cả các video được thu thập từ YouTube và có tốc độ khung hình cố định là 25 FPS với độ phân giải 320×240 .

Sau khi thu thập dữ liệu, chúng ta cần tiền xử lý để loại bỏ nhiễu, cân bằng phân phối lớp, chuẩn hóa kích thước và định dạng của video. Một số kỹ thuật tiền xử lý phổ biến là:

- Chuyển video sang định dạng có chất lượng thấp hơn để giảm số chiều của dữ liệu.
- Thay đổi kích thước của video về cùng một giá trị nhất định, ví dụ 64×64 .
- Phân chia dữ liệu thành tập huấn luyện, tập kiểm tra theo tỉ lệ phù hợp

• Bước 2: Tiền xử lý dữ liệu

Quá trình tập trung vào việc tiền xử lý dữ liệu để chuẩn bị cho việc huấn luyện mô hình như sau:

- **Tách khung hình từ video:** Bước đầu tiên là xác định và trích xuất các khung hình từ video. Các khung hình này sẽ được sử dụng như các điểm dữ liệu để huấn luyện và kiểm thử mô hình nhận dạng hành động.

- **Chuẩn hóa dữ liệu:** Sau khi có được các khung hình, quá trình chuẩn hóa có thể được thực hiện để đảm bảo rằng các khung hình đều có cùng định dạng và các giá trị tương đối như nhau, nhằm loại bỏ sự biến đổi không cần thiết và tăng tính đồng nhất của dữ liệu.

- **Chuyển đổi nhãn:** Nếu ban đầu dữ liệu nhãn được cung cấp dưới dạng nguyên thì quá trình này bao gồm chuyển đổi nhãn từ định dạng nguyên thành dạng mã hóa (ví dụ: từ chuỗi ký tự thành các mã số hoặc các vector one-hot encoding) để phục vụ cho việc huấn luyện mô hình.

- **Chia tập dữ liệu:** Dữ liệu sau khi được chuẩn hóa và nhãn được chuyển đổi sẽ được chia thành hai phần: tập huấn luyện và tập kiểm thử. Tập huấn luyện sẽ được

sử dụng để huấn luyện mô hình, trong khi tập kiểm thử sẽ được sử dụng để đánh giá hiệu suất của mô hình sau quá trình huấn luyện.

- **Bước 3: Huấn luyện mô hình**

- Quá trình huấn luyện mô hình nhận dạng hành động con người từ dữ liệu video bao gồm các bước quan trọng. Đầu tiên, dữ liệu video được chuẩn bị và chia thành các khung hình, sau đó gán nhãn các hành động trong video. Nhãn này thường được chuyển đổi thành định dạng số hoặc mã hóa one-hot để mô hình có thể hiểu và dự đoán chúng dưới dạng số liệu. Tiếp theo, tập dữ liệu video được chia thành hai phần: tập huấn luyện và tập kiểm định. Tập huấn luyện được dùng để huấn luyện mô hình, trong khi tập kiểm định sẽ đánh giá hiệu suất của mô hình sau quá trình huấn luyện.

- Quá trình huấn luyện mô hình sử dụng các thuật toán máy học như mạng nơ-ron, CNN, RNN/LSTM để xây dựng một mô hình nhận dạng hành động. Các tham số của mô hình được điều chỉnh để đảm bảo độ chính xác và hiệu suất cao trên tập dữ liệu huấn luyện. Sau đó, mô hình được đánh giá bằng cách sử dụng các phương pháp như cross-validation hoặc validation set để đánh giá hiệu suất trên tập kiểm định và tránh overfitting.

- Việc kiểm thử và đánh giá mô hình là bước cuối cùng, trong đó mô hình được đánh giá trên tập dữ liệu kiểm định hoặc tập dữ liệu mới mà nó chưa từng thấy trước đó. Kết quả đánh giá sẽ đưa ra thông tin cần thiết để cải thiện hoặc điều chỉnh cuối cùng cho mô hình, đảm bảo tính khả dụng và độ chính xác của nó trên dữ liệu thực tế.

- **Bước 4: Đánh giá mô hình**

- Phương pháp triển khai phân loại các hoạt động này và đánh giá phân loại đó được thực hiện sau khi tập dữ liệu được chia thành tập huấn luyện và tập kiểm tra. Việc phân chia tập dữ liệu được thực hiện sao cho 75% dữ liệu được sử dụng để huấn luyện bộ phân loại trong khi 25% còn lại được sử dụng để kiểm tra phân loại. Để tránh việc mô hình quá khớp, sẽ có một bộ dữ liệu xác thực riêng để kiểm tra bộ phân loại và hiểu về khả năng tổng quát của nó.

Sẽ có hai tập hợp chỉ số chính sẽ được sử dụng để kiểm tra khả năng của bộ phân loại và đánh giá kết quả của nó:

- Độ chính xác – tỷ lệ phần trăm của số dự đoán chính xác so với tổng số dự đoán. Công thức được định nghĩa như sau:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Trong đó:

- TP (True Positive) đại diện cho số bản ghi dương tính được dự đoán chính xác là dương tính.
 - TN (True Negative) đại diện cho số bản ghi âm tính được dự đoán chính xác là âm tính.
 - FP (False Positive) đại diện cho số bản ghi âm tính bị dự đoán sai là dương tính.
 - FN (False Negative) đại diện cho số bản ghi dương tính bị dự đoán sai là âm tính.
- Ma trận nhầm lẫn: đây là một kỹ thuật giúp tóm tắt kết quả và hiệu suất của một bộ phân loại học máy. Nó sử dụng cùng 4 giá trị mà hàm độ chính xác sử dụng TP, TN, FP và FN. Một bảng phân loại 2 x 2 đơn giản được mô tả trong hình dưới đây:

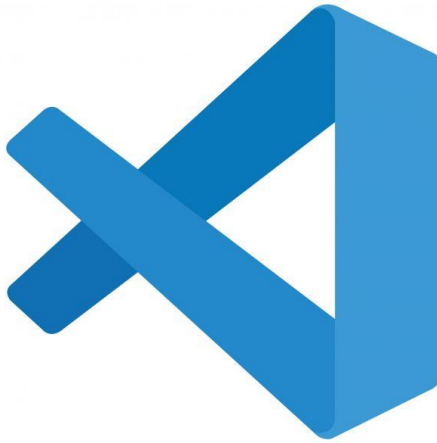
		Actual values	
		P	N
Predicted values	P'	True Positive	False Negative
	N'	False Positive	True Negative

Hình 3. 3 A 2 x 2 sample Confusion Matrix

Trong một ma trận nhầm lẫn lớn hơn, kết quả tốt nhất là khi các ô từ trái lên phải đến đường chéo chính xuống dưới (đường chéo chính từ trái trên xuống phải) có giá trị lớn nhất, vì chúng biểu thị các dự đoán chính xác.

5. Công cụ hỗ trợ

Visual Studio Code (VS Code) là một ứng dụng mã nguồn mở được phát triển bởi Microsoft, chuyên dùng để lập trình và phát triển phần mềm. Với sự hỗ trợ từ nền tảng Electron, VS Code cung cấp cho người dùng một trải nghiệm lập trình đơn giản, linh hoạt và hiệu quả với nhiều tính năng hữu ích để tăng năng suất cho quá trình lập trình. Với VS Code, người lập trình có thể dễ dàng quản lý các tệp mã nguồn cho dự án của mình, tối ưu hóa quá trình lập trình bằng tính năng đa nhiệm, gỡ lỗi và chạy chương trình một cách dễ dàng. Điều này giúp cho quá trình phát triển và triển khai mô hình máy học trở nên thuận tiện hơn. Ngoài ra, VS Code còn cung cấp cho người dùng nhiều tiện ích mở rộng và plugin để tùy chỉnh và mở rộng tính năng của ứng dụng.



Hình 3. 4 Công cụ phát triển Visual studio code.

PHẦN QUÁ TRÌNH THỰC HIỆN

1. Thư viện sử dụng

1.1 Giới thiệu thư viện TensorFlow

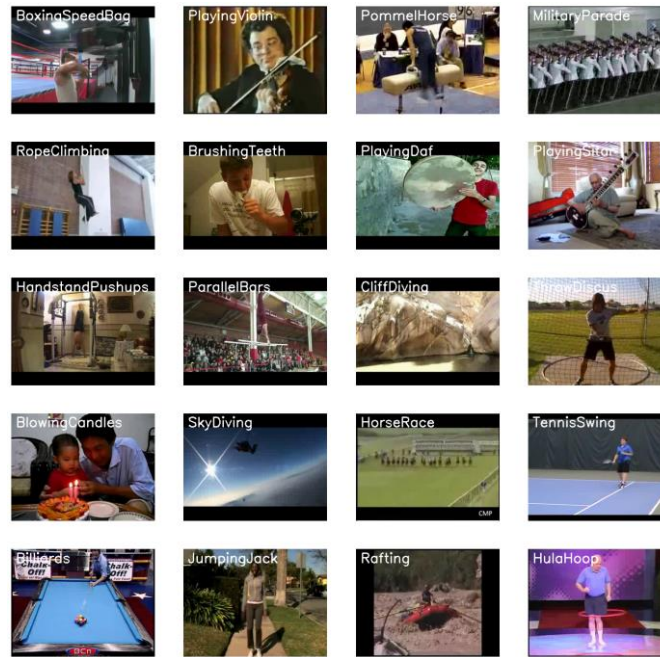
TensorFlow là một thư viện phần mềm mở cho tính toán số, sử dụng biểu đồ luồng dữ liệu. Các nút trong đồ thị biểu diễn cho hoạt động toán học, trong khi các cạnh đồ thị biểu diễn cho các mảng dữ liệu đa chiều (tensors) trao đổi giữa chúng. Kiến trúc linh hoạt cho phép chúng ta triển khai tính toán trên một hoặc nhiều CPU hoặc GPU trong một máy tính để bàn, máy chủ, hoặc thiết bị di động với một API đơn. TensorFlow ban đầu được phát triển bởi các nhà nghiên cứu và kỹ sư làm việc trong nhóm Google Brain cho các nghiên cứu máy học và deep neural network.

TensorFlow có các API với một số ngôn ngữ lập trình cho cả xây dựng và thực thi một đồ thị TensorFlow. Python API là hiện tại hoàn thiện nhất và dễ sử dụng nhất, nhưng API C++ có một vài ưu điểm về hiệu năng trong việc thực thi đồ thị, và hỗ trợ triển khai các thiết bị nhỏ như Android.

1.2 Chuẩn bị tập dữ liệu

Thông tin về tập dữ liệu: Nghiên cứu sử dụng tập dữ liệu UFC101 với các thông tin sau:

- Gồm có 13320 video được lấy từ YouTube, được chia thành 101 hành động khác nhau như chạy, nhảy, make up, bơi, đánh đàn...
- Độ phân giải 256x256
- Kích thước tải xuống 6.48 GB
- Link: <https://www.crcv.ucf.edu/data-sets/ucf101/>



Hình 3. 5 Tập dữ liệu ucf101

2. Bài toán nhận dạng hành động trong video

Nhận diện hành động của con người được phát triển bắt đầu vào đầu năm 1980. Hiện nay, các nghiên cứu chủ yếu tập trung vào việc học và nhận biết các hành động từ chuỗi video. Nhận dạng là một ngành thuộc lĩnh vực trí tuệ nhân tạo. Nhận dạng mẫu là khả năng phát hiện sự sắp xếp các đặc tính hoặc dữ liệu mang lại thông tin về một hệ thống hoặc tập dữ liệu nhất định. Nhận dạng mẫu chia thành nhiều lĩnh vực trong công nghệ thông tin, bao gồm phân tích dữ liệu lớn, nhận dạng sinh trắc học, bảo mật và trí tuệ nhân tạo. Nhận dạng đối tượng trong hình ảnh là một nhánh của nhận dạng mẫu. Nhận dạng đối tượng trong hình ảnh thể hiện qua các công nghệ máy tính có thể nhận ra người, động vật, vật thể hoặc các đối tượng mục tiêu khác thông qua việc sử dụng các thuật toán và khái niệm học máy.

Một hành động là một chuỗi các chuyển động cơ thể con người, và có thể bao gồm nhiều bộ phận cơ thể đồng thời. Từ quan điểm của thị giác máy tính, việc nhận dạng hành động này là để phù hợp với các quan sát (ví dụ: video) với các mẫu được xác định trước đó và sau đó gán cho nó một nhãn là loại hành động. Tùy thuộc vào độ phức tạp, hoạt động của con người có thể được phân loại thành bốn cấp độ: cử chỉ, hành động, tương tác và hoạt động của nhóm, và nhiều nghiên cứu theo hướng một cấu trúc từ dưới lên về nhận dạng hoạt động của con người. Các phần chính của hệ thống như vậy bao gồm trích xuất đặc trưng, học tập hành động, phân loại, nhận dạng hành động và phân đoạn. Một quy trình đơn giản gồm ba bước, cụ thể là phát hiện của con người hoặc các bộ phận cơ thể, theo dõi, và sau đó nhận bằng cách sử dụng

kết quả theo dõi. Ví dụ, để nhận ra hành động “bắt tay”, cánh tay và bàn tay của hai người được phát hiện trước tiên và theo dõi để tạo ra một mô tả không gian-thời gian của chuyển động của họ. Mô tả này được so sánh với các mẫu hiện có trong dữ liệu huấn luyện để xác định loại hành động. Mô hình này dựa rất nhiều vào tính chính xác của việc theo dõi, điều này không đáng tin cậy trong những cảnh lộn xộn.

Nhiều phương pháp đã được đề xuất, và có thể được phân loại theo nhiều tiêu chí khác nhau. Poppe thảo luận nhận dạng hành động của con người từ biểu diễn hình ảnh và phân loại hành động riêng rẽ. Weinland khảo sát các phương pháp cho biểu diễn hành động, phân đoạn và nhận dạng. Turaga chia vấn đề nhận dạng thành hành động và hoạt động theo độ phức tạp, và các hướng tiếp cận phân loại theo khả năng của mình để xử lý các mức độ phức tạp khác nhau. Có nhiều tiêu chí phân loại khác nhau. Trong số đó, Aggarwal và Ryoo là một trong những tổng kết toàn diện mới nhất và so sánh của sự tiến bộ quan trọng nhất trong lĩnh vực này. Dựa vào hành động được nhận dạng từ hình ảnh đầu vào trực tiếp, Aggarwal và Ryoo phân chia các phương pháp nhận dạng thành hai loại chính: phương pháp tiếp cận đơn lớp và phương pháp tiếp cận phân cấp. Cả hai đều là thêm loại con phụ thuộc vào các phương pháp biểu diễn đặc trưng và học.

3. Chương trình ứng dụng cho bài toán nhận dạng hành động

- Bước 1: Tải và trực quan hóa dữ liệu bằng nhãn.
- Bước 2: Xử lý trước tập dữ liệu
- Bước 3: Chia dữ liệu thành Train và Test
- Bước 4: Thực hiện Phương pháp tiếp cận LRCN
- Bước 4.1: Xây dựng mô hình
- Bước 4.2: Biên dịch và huấn luyện đánh giá

Tập dữ liệu UCF-101 là một nguồn tài nguyên quan trọng trong lĩnh vực nhận dạng hành động con người. Nó bao gồm một bộ sưu tập lớn các video khác nhau từ các hoạt động thể thao đến các hoạt động hàng ngày, cung cấp một bộ dữ liệu đa dạng và phong phú để nghiên cứu và phát triển các mô hình nhận dạng hành động.

Tuy nhiên trong phạm vi nghiên cứu của đề tài này chỉ thực hiện lựa chọn một vài hành động cụ thể như: Archery: (Bắn cung) . Biking: (Đạp xe) . HorseRiding: (Cưỡi ngựa). WalkingWithDog: (Đi bộ cùng chó). BodyWeightSquats: Chuỗi động tác Squats (Động tác uốn cong cơ bắp mông và đùi), Bowling: (Bowling). BoxingPunchingBag: (Đấm bóng quyền). Diving: (Nhảy từ trampoline hoặc nước). Drumming: (Đánh trống). GolfSwing: (Đánh gậy golf).

Mục tiêu của việc lựa chọn chỉ một số hành động từ tập dữ liệu lớn như UCF-101 là tập trung phát triển và cải thiện khả năng nhận dạng cho các hành động cụ thể này. Việc giảm phạm vi của dữ liệu giúp tăng hiệu quả trong việc huấn luyện mô hình cho các tác vụ nhận dạng cụ thể, thay vì phải xử lý tất cả các loại hành động có trong tập dữ liệu gốc.

Ưu điểm: Lựa chọn các hành động cụ thể giúp tối ưu hóa việc sử dụng tài nguyên tính toán và thời gian huấn luyện mô hình. Bạn có thể tập trung vào việc tối ưu hóa mô hình để phát triển một hệ thống nhận dạng chính xác và hiệu quả cho các hành động quan trọng mà bạn chọn.

Quá trình xây dựng mô hình nhận dạng hành động con người được mô tả như sau:

Nhập thư viện :

```
# Import the required libraries.
import os
import cv2
import random
import numpy as np
import datetime as dt
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from moviepy.editor import *
%matplotlib inline
```

– Import các thư viện được sử dụng cho cả mô hình: như OpenCV (cv2) để xử lý video và hình ảnh, NumPy (numpy) để thực hiện các phép toán số học, TensorFlow (tensorflow) để xây dựng và huấn luyện mô hình học sâu, Matplotlib (matplotlib) để tạo biểu đồ và hiển thị hình ảnh.

– Import các lớp và công cụ từ TensorFlow/Keras: sử dụng các lớp và công cụ từ thư viện TensorFlow/Keras như Sequential để tạo mô hình tuần tự, các lớp như Conv2D, MaxPooling2D, Dense để xây dựng kiến trúc mạng neural, cùng với các công cụ khác như to_categorical để mã hóa dữ liệu về dạng ma trận nhị phân.

– Import một số công cụ hỗ trợ khác: train_test_split từ thư viện scikit-learn để phân chia dữ liệu thành tập huấn luyện và tập kiểm tra, cùng với moviepy.editor để thực hiện các tác vụ chỉnh sửa video.

Bước 1. Chuẩn bị dữ liệu và trực quan hóa dữ liệu với nhãn tương ứng

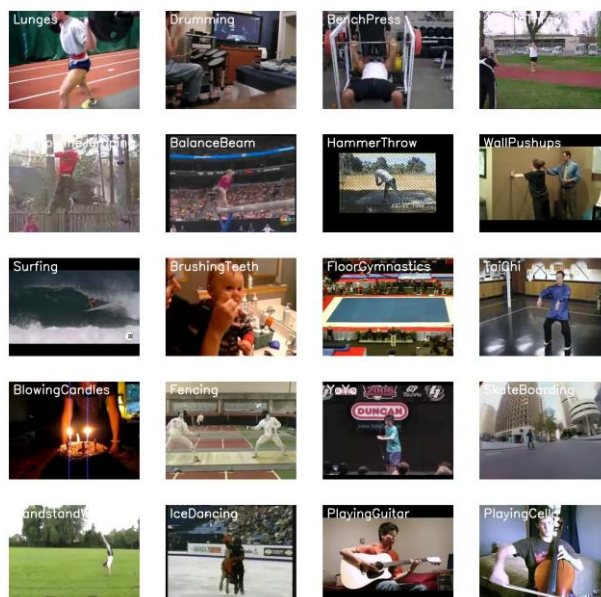
Để trực quan hóa, chúng ta sẽ chọn ngẫu nhiên các danh mục từ tập dữ liệu và một video ngẫu nhiên từ mỗi danh mục được chọn và sẽ trực quan hóa khung hình đầu tiên của các video được chọn với nhãn đi kèm. Như vậy, chúng ta sẽ có thể trực quan hóa một phần nhỏ (các video ngẫu nhiên) của tập dữ liệu.

```
plt.figure(figsize=(20, 20))
all_classes_names = os.listdir("UCF-101")
random_range = random.sample(range(len(all_classes_names)), 20)

for counter, random_index in enumerate(random_range, 1):
    selected_class_Name = all_classes_names[random_index]
    video_files_names_list = os.listdir(f'UCF-101/{selected_class_Name}')
    selected_video_file_name = random.choice(video_files_names_list)
    video_reader = cv2.VideoCapture(f'UCF-101/{selected_class_Name}/{selected_video_file_name}')
    _, bgr_frame = video_reader.read()
    video_reader.release()
    rgb_frame = cv2.cvtColor(bgr_frame, cv2.COLOR_BGR2RGB)
    cv2.putText(rgb_frame, selected_class_Name, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
    plt.subplot(5, 4, counter)
    plt.imshow(rgb_frame)
    plt.axis('off')
```

– Đoạn này tạo một biểu đồ Matplotlib để hiển thị các hành động từ tập dữ liệu. Nó chọn ngẫu nhiên 20 hành động từ tập dữ liệu UCF-101 và hiển thị một frame đầu tiên của mỗi video tương ứng với từng hành động. Khi hiển thị frame, tên của hành động cũng được ghi lên đó để xác định từng video thuộc hành động nào.

– Kết quả hiển thị :



Hình 3. 6 Ảnh kết tải tập dữ liệu

Bước 2. Tiền xử lý tập dữ liệu

- Tiếp theo, chúng ta sẽ thực hiện một số bước tiền xử lý trên tập dữ liệu. Đầu tiên, chúng ta sẽ đọc các tệp video từ tập dữ liệu và thay đổi kích thước các khung hình của video về một độ rộng và chiều cao cố định, nhằm giảm thiểu tính toán và chuẩn hóa dữ liệu về một phạm vi [0-1] bằng cách chia giá trị pixel cho 255, điều này giúp tăng tốc quá trình hội tụ trong quá trình huấn luyện mạng.
- Nhưng trước hết, hãy khởi tạo một số hằng số.

```

IMAGE_HEIGHT, IMAGE_WIDTH = 64, 64
SEQUENCE_LENGTH = 20

DATASET_DIR = "UCF-101"
CLASSES_LIST = ["Archery", "Biking", "HorseRiding", "WalkingWithDog", "BodyWeightSquats", "Bowling", "BoxingPunchingBag", "Diving", "Drumming", "GolfSwing"]
    
```

- Lưu ý: Các hằng số IMAGE_HEIGHT, IMAGE_WIDTH và SEQUENCE_LENGTH có thể được tăng lên để đạt được kết quả tốt hơn, tuy nhiên việc tăng độ dài chuỗi (SEQUENCE_LENGTH) chỉ hiệu quả đến một mức nhất định, và việc tăng giá trị này sẽ làm tăng độ phức tạp tính toán trong quá trình xử lý.

Tạo một Hàm để Trích Xuất, Thay Đổi Kích Thước & Chuẩn Hóa Khung Hình

- Chúng ta sẽ tạo một hàm frames_extraction() sẽ tạo ra một danh sách chứa các khung hình đã được thay đổi kích thước và chuẩn hóa của một video, đường dẫn đến video sẽ được truyền vào hàm như một đối số. Hàm này sẽ đọc tệp video từng khung hình một, mặc dù không tất cả các khung hình được thêm vào danh sách vì chúng ta chỉ cần một chuỗi các khung hình phân bố đều.

```

def frames_extraction(video_path):
    frames_list = []
    video_reader = cv2.VideoCapture(video_path)
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

    for frame_counter in range(SEQUENCE_LENGTH):
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)
        success, frame = video_reader.read()
        if not success:
            break

        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        normalized_frame = resized_frame / 255
        frames_list.append(normalized_frame)

    video_reader.release()
    return frames_list
    
```

Tạo một hàm để tạo tập dữ liệu.

– Bây giờ chúng ta sẽ tạo một hàm `create_dataset()` sẽ lặp qua tất cả các lớp được chỉ định trong hằng số `CLASSES_LIST` và gọi hàm `frame_extraction()` cho mỗi tập video trong các lớp được chọn, sau đó trả về các khung hình (features), chỉ số lớp (labels), và đường dẫn tập video (`video_files_paths`).

```
def create_dataset():
    features = []
    labels = []
    video_files_paths = []

    for class_index, class_name in enumerate(CLASSES_LIST):
        print(f'Extracting Data of Class: {class_name}')
        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

        for file_name in files_list:
            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)
            frames = frame_extraction(video_file_path)

            if len(frames) == SEQUENCE_LENGTH:
                features.append(frames)
                labels.append(class_index)
                video_files_paths.append(video_file_path)

    features = np.asarray(features)
    labels = np.array(labels)

    return features, labels, video_files_paths
```

– Bây giờ chúng ta sẽ sử dụng hàm `create_dataset()` đã tạo ở trên để trích xuất dữ liệu từ các lớp được chọn và tạo tập dữ liệu cần thiết.

```
# Create the dataset.
features, labels, video_files_paths = create_dataset()

Extracting Data of Class: Archery
Extracting Data of Class: Biking
Extracting Data of Class: HorseRiding
Extracting Data of Class: WalkingWithDog
Extracting Data of Class: BodyWeightSquats
Extracting Data of Class: Bowling
Extracting Data of Class: BoxingPunchingBag
Extracting Data of Class: Diving
Extracting Data of Class: Drumming
Extracting Data of Class: GolfSwing
```

– Tạo ra tập dữ liệu bằng cách sử dụng hàm `create_dataset()` đã được định nghĩa trước đó. Kết quả sẽ trả về các features (đặc trưng), labels (nhãn) và `video_files_paths` (đường dẫn tập video) từ quá trình trích xuất dữ liệu từ tập dữ liệu video theo danh sách các lớp đã chọn.

Bây giờ chúng ta sẽ chuyển đổi (chỉ số lớp) thành các vector mã hóa one-hot.

```
#Chuyển đổi nhãn dạng nguyên thành dạng one shot
one_hot_encoded_labels = to_categorical(labels)
```

Bước 3: Phân chia dữ liệu thành tập huấn luyện và tập kiểm tra

– Hiện tại, chúng ta đã có các đặc trưng cần thiết (một mảng NumPy chứa tất cả các khung hình được trích xuất từ các video) và `one_hot_encoded_labels` (cũng là

một mảng NumPy chứa tất cả nhãn lớp ở định dạng mã hóa one-hot). Bây giờ, chúng ta sẽ chia dữ liệu để tạo ra các tập huấn luyện và kiểm tra. Chúng ta cũng sẽ trộn dữ liệu trước khi chia để tránh bất kỳ thiên lệch nào và có được các tập chia phản ánh phân phối tổng quát của dữ liệu.

```
#Chia tập dữ liệu
# Split the Data into Train ( 75% ) and Test Set ( 25% ).
features_train, features_test, labels_train, labels_test = train_test_split(features, one_hot_encoded_labels,
                                                                            test_size=0.25, shuffle=True,
                                                                            random_state=42)
```

Bước 4: Thực hiện phương pháp tiếp cận LRCN

– Trong bước này, chúng ta sẽ thực hiện Phương pháp LRCN bằng cách kết hợp các lớp Convolution và LSTM trong một mô hình duy nhất. Một phương pháp tương tự khác có thể là sử dụng một mô hình CNN và một mô hình LSTM được huấn luyện một cách riêng biệt. Mô hình CNN có thể được sử dụng để trích xuất các đặc trưng không gian từ các khung hình trong video, và với mục đích này, một mô hình được huấn luyện trước có thể được sử dụng, có thể được điều chỉnh lại để phù hợp với vấn đề cụ thể. Và mô hình LSTM sau đó có thể sử dụng các đặc trưng được trích xuất bởi CNN để dự đoán hành động đang được thực hiện trong video.

– Nhưng ở đây, chúng ta sẽ thực hiện một phương pháp khác được biết đến là Mạng Tổ hợp Convolutional Dài hạn (LRCN), kết hợp các lớp CNN và LSTM trong một mô hình duy nhất. Các lớp Convolution được sử dụng để trích xuất đặc trưng không gian từ các khung hình, và các đặc trưng không gian được trích xuất được đưa vào các lớp LSTM tại mỗi bước thời gian để mô hình chuỗi thời gian. Như vậy, mạng học các đặc trưng không gian-thời gian trực tiếp trong quá trình huấn luyện end-to-end, tạo ra một mô hình mạnh mẽ và ổn định.

Bước 4.1: Xây dựng mô hình

– Mô hình này bao gồm các lớp Conv2D, MaxPooling2D, Dropout, LSTM và Dense. Nó xây dựng một kiến trúc mạng nơ-ron sử dụng mạng học sâu kết hợp với mạng nơ-ron hồi quy dài hạn để nhận dạng hành động trong video. Lớp Conv2D và MaxPooling2D giúp trích xuất đặc trưng từ từng khung hình trong chuỗi, trong khi LSTM hỗ trợ mô hình hóa các mối quan hệ thời gian giữa các khung hình. Lớp Dense với softmax activation được sử dụng để dự đoán lớp hành động mong muốn từ video. Hàm kích hoạt softmax cho phép xác định xác suất của từng lớp dự đoán cho mỗi video.

```
def create_LRCN_model():
    model = Sequential()

    model.add(TimeDistributed(Conv2D(64, (4, 4), padding='same', activation='relu'),
                                input_shape=(SEQUENCE_LENGTH, IMAGE_WIDTH, IMAGE_HEIGHT, 3)))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.3)))

    model.add(TimeDistributed(Conv2D(128, (4, 4), padding='same', activation='relu')))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.3)))

    model.add(TimeDistributed(Conv2D(256, (2, 2), padding='same', activation='relu')))
    model.add(TimeDistributed(GlobalMaxPooling2D()))

    model.add(LSTM(256, return_sequences=True, kernel_regularizer=tf.keras.regularizers.l2(0.1)))
    model.add(Flatten())

    model.add(Dense(len(CLASSES_LIST), activation='softmax'))

    model.summary()

    return model
```

– Bây giờ chúng ta sẽ sử dụng hàm `create_LRCN_model()` đã tạo ở trên để xây dựng mô hình LRCN cần thiết.

```
# Construct the required LRCN model.
LRCN_model = create_LRCN_model()

# Display the success message.
print("Model Created Successfully!")
```

– Kết quả hiển thị:

```
Model: "sequential"
-----
Layer (type)                Output Shape              Param #
-----
time_distributed (TimeDist  (None, 20, 64, 64, 64)    3136
ributed)

time_distributed_1 (TimeDi  (None, 20, 16, 16, 64)    0
stributed)

time_distributed_2 (TimeDi  (None, 20, 16, 16, 64)    0
stributed)

time_distributed_3 (TimeDi  (None, 20, 16, 16, 128)   131200
stributed)

time_distributed_4 (TimeDi  (None, 20, 4, 4, 128)     0
stributed)

time_distributed_5 (TimeDi  (None, 20, 4, 4, 128)     0
stributed)
...
Trainable params: 842186 (3.21 MB)
Non-trainable params: 0 (0.00 Byte)

Model Created Successfully!
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Bước 5.2: Biên dịch & Huấn luyện mô hình


```
#Compile the model and specify loss function, optimizer and metrics to the model.
LRCN_model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metrics = ["accuracy"])

# Start training the model.
LRCN_model_training_history = LRCN_model.fit(x = features_train, y = labels_train, epochs = 150, batch_size = 32 ,
                                              shuffle = True, validation_split = 0.2)
```

Đánh giá Mô hình đã Huấn luyện

- Chúng ta sẽ đánh giá mô hình trên tập dữ liệu kiểm tra (test set) (LRCN).

```
# Evaluate the trained model.
model_evaluation_history = LRCN_model.evaluate(features_test, labels_test)

12/12 [=====] - 3s 209ms/step - loss: 0.7523 - accuracy: 0.8536
```

Lưu Mô hình

- Sau đó, chúng ta sẽ lưu mô hình để sử dụng trong tương lai bằng cùng một kỹ thuật đã được sử dụng cho mô hình trước đó.

```
model_evaluation_loss, model_evaluation_accuracy = model_evaluation_history

date_time_format = 'Y_%m_%d_%H_%M_%S'
current_date_time_dt = dt.datetime.now()
current_date_time_string = dt.datetime.strftime(current_date_time_dt, date_time_format)

model_file_name = f'LRCN_model__Date_Time_{current_date_time_string}__Loss_{model_evaluation_loss}__Accuracy_{model_evaluation_accuracy}.h5'

# Save the Model.
LRCN_model.save(model_file_name)
```

```
...      precision    recall  f1-score   support

0         0.82        0.82        0.82         34
1         0.73        0.79        0.76         34
2         0.85        0.91        0.88         44
3         0.50        0.45        0.48         22
4         0.78        0.94        0.85         34
5         0.97        0.94        0.96         34
6         0.97        0.86        0.92         44
7         0.98        1.00        0.99         40
8         0.84        0.95        0.89         39
9         0.96        0.68        0.79         37

 accuracy          0.85        362
 macro avg         0.84        0.84        0.83        362
 weighted avg      0.86        0.85        0.85        362
```

- Nhận xét về kết quả mô hình đạt được

Chính xác tổng thể (Accuracy): Đạt khoảng 85%, tức là mô hình dự đoán chính xác khoảng 85% trên tập dữ liệu kiểm tra.

Precision, Recall và F1-score:

Precision cao cho thấy mô hình có khả năng dự đoán chính xác các hành động.

Recall cao cho biết mô hình có khả năng bao phủ tốt các trường hợp thực tế của các hành động đó.

F1-score là một sự kết hợp của Precision và Recall, điểm số này cũng cao đồng nghĩa với việc cả Precision và Recall đều tốt.

Có sự chênh lệch giữa các lớp:

Một số lớp (như lớp 3) có precision và recall thấp hơn, tức là mô hình không thể dự đoán và bao phủ tốt cho lớp đó.

Có những lớp có precision và recall cao (như lớp 7 và 8), cho thấy mô hình hoạt động tốt trong việc dự đoán và bao phủ.

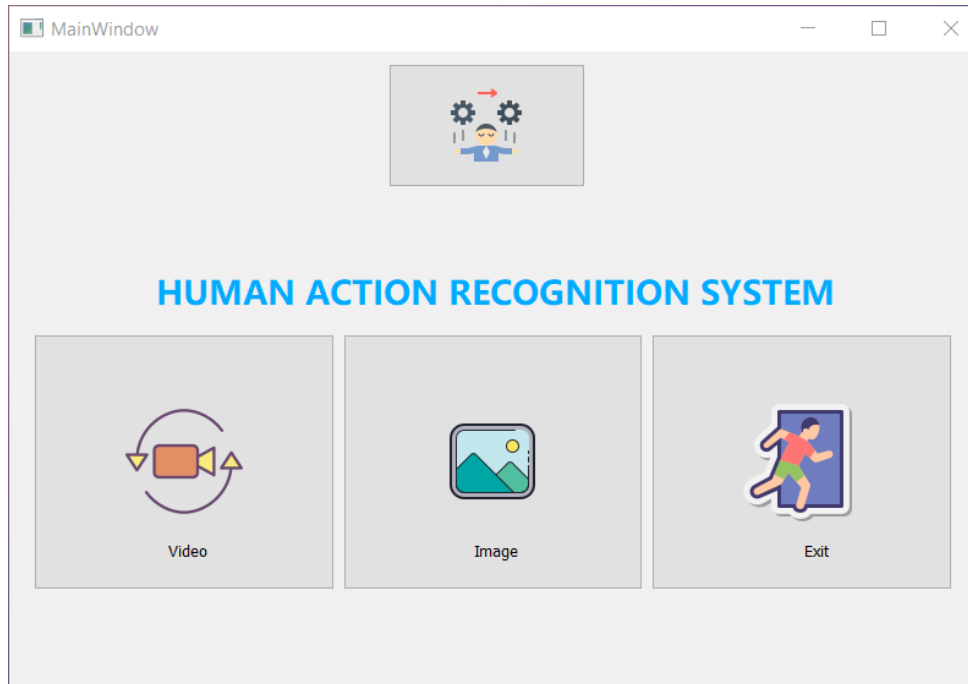
4. Tạo một giao diện chương trình ứng dụng

PyQt5 là một thư viện Python rất mạnh mẽ và phổ biến để tạo giao diện người dùng đồ họa (GUI). Nó cung cấp khả năng kết hợp giữa Python và các công cụ thiết kế giao diện người dùng từ Qt, một bộ công cụ phát triển ứng dụng đa nền tảng rất mạnh mẽ.

Một số tính năng của PyQt5 như:

- Tính đa nền tảng: PyQt5 hỗ trợ phát triển trên nhiều hệ điều hành như Windows, macOS và Linux. Điều này cho phép bạn xây dựng ứng dụng đa nền tảng một cách linh hoạt.
- Giao diện người dùng đồ họa (GUI) tương tác: PyQt5 cung cấp các thành phần giao diện người dùng như cửa sổ, nút, hộp văn bản, và các widget khác để tạo ra các ứng dụng tương tác và hấp dẫn.
- Hỗ trợ Qt Designer: Qt Designer là một công cụ thiết kế giao diện người dùng GUI tương tác, giúp bạn dễ dàng tạo và thiết kế giao diện một cách trực quan.
- Hỗ trợ tốt cho các tập tin và ứng dụng đa phương tiện: PyQt5 không chỉ hỗ trợ các loại widget cơ bản mà còn cho phép bạn làm việc với các tập tin, đa phương tiện, đồ họa và nhiều tính năng nâng cao khác.
- Tích hợp tốt với Python: PyQt5 có cấu trúc chặt chẽ với Python, giúp các lập trình viên Python dễ dàng học và sử dụng.
- Cộng đồng lớn và tài liệu phong phú: PyQt5 có một cộng đồng sáng tạo và phát triển đông đảo cùng với tài liệu chi tiết và ví dụ minh họa rất hữu ích.

Thành phần giao diện chương trình:



Hình 3. 7 Ảnh giao diện chương trình

Tính năng chính:

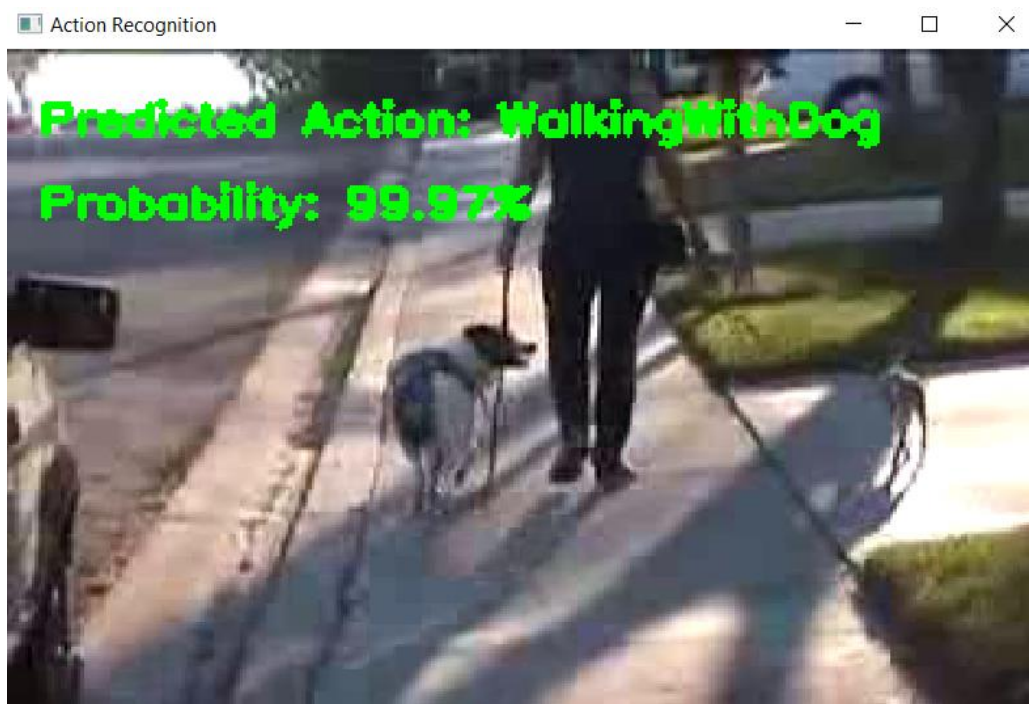
- Tải video hoặc ảnh : Cho phép người dùng tải video từ máy tính của họ để phân loại hành động.
- Hiển thị kết quả: Hiển thị kết quả dự đoán về hành động con người trong video, bao gồm cả xác suất của mỗi hành động.
- Thoát chương trình

Giao diện đồ họa:

- Cửa sổ chính: Bao gồm các tùy chọn để tải lên video hoặc ảnh, cùng với các nút để bắt đầu quá trình nhận dạng.
- Kết quả dự đoán: Hiển thị kết quả của việc nhận dạng hành động, hiển thị tên hành động và xác suất tương ứng.

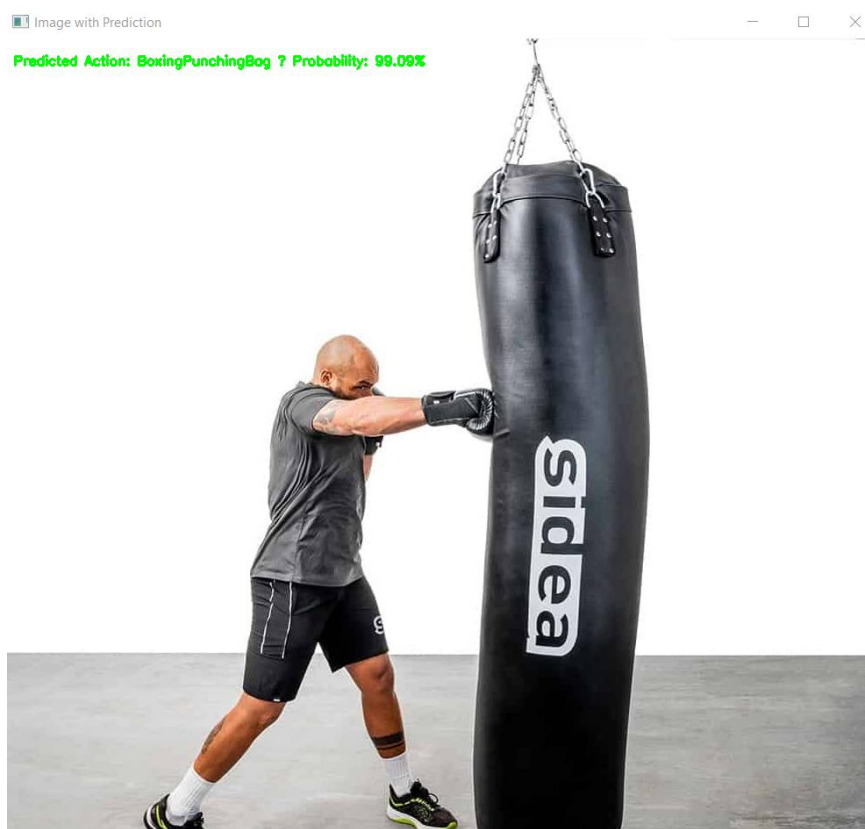
Mục tiêu:

- Mục tiêu của chúng tôi là cung cấp cho người dùng một giao diện thân thiện và dễ sử dụng để hiểu rõ về việc nhận dạng hành động con người thông qua mô hình học sâu.
- Nhận dạng trong video



Hình 3. 8 Kết quả nhận dạng trong video

– Nhận dạng trong ảnh



Hình 3. 9 Kết quả nhận dạng trong ảnh

PHẦN KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Đồ án đã nghiên cứu và thực nghiệm giải thuật về một số cách tiếp cận để thực hiện phân loại video và tìm hiểu về tầm quan trọng của khía cạnh thời gian của dữ liệu để đạt được độ chính xác cao hơn trong phân loại video và triển khai hai kiến trúc CNN + LSTM trong TensorFlow để thực hiện Nhận dạng hành động của con người trên video bằng cách sử dụng thông tin thời gian cũng như không gian của dữ liệu.

Tuy nhiên, mặc dù mô hình đã đạt được kết quả khả quan, nhưng vẫn còn một số hạn chế. Một số hành động có thể khó nhận dạng hơn do sự tương tự về mặt hình ảnh hoặc do sự khác biệt về góc nhìn, độ phân giải, v.v... Ngoài ra, việc huấn luyện mô hình yêu cầu nhiều tài nguyên tính toán và thời gian.

Dựa trên việc nghiên cứu cơ sở lý thuyết và các công nghệ liên quan, đề tài xây dựng mô hình mạng CNN với LSTM, cho ra kết quả đáp ứng những mục tiêu đã đề ra.

2. Những hạn chế và hướng phát triển của đề tài

2.1 Hạn chế đề tài

Một số vấn đề ảnh hưởng đến quá trình huấn luyện và dự đoán kết quả:

- Bộ nhận dạng hành động không thể xử lý nhiều người thực hiện các hoạt động khác nhau cùng một lúc. Chỉ có thể nhận diện chính xác hành động của một người duy nhất trong khung hình.
- Xử lý video và sử dụng mô hình để huấn luyện sâu, cần có thiết bị có phần cứng có khả năng xử lý tính toán cao để có thể áp dụng cho một lượng dữ liệu lớn.
- Tạo giao diện đồ họa sử dụng PyQt5 để xử lý khung hình có thể gặp hạn chế về hiệu suất khi hiển thị và xử lý dữ liệu hình ảnh trong thời gian thực.

2.2 Hướng phát triển đề tài

Cải tiến mô hình: Thử nghiệm với các kiến trúc mạng khác nhau, tinh chỉnh các tham số của mô hình hiện tại để cải thiện hiệu suất.

Sử dụng các bộ dữ liệu khác nhau đã được ghi chú cho việc thực hiện hoạt động của nhiều người. Đồng thời, cung cấp thông tin về tọa độ của hộp giới hạn của từng người cùng với hoạt động mà họ đang thực hiện. Điều này có thể giúp khắc phục hạn

ché của mô hình chỉ nhận diện hành động của một người duy nhất trong khung hình, bằng cách cho phép mô hình xử lý thông tin từ nhiều người cùng một lúc.

Một cách “hack” khác có thể là cắt ảnh để tách riêng từng người và thực hiện việc nhận dạng hành động độc lập trên từng người. Tuy nhiên, cách tiếp cận này sẽ tốn kém về mặt tính toán, vì đòi hỏi phải xử lý một lượng lớn dữ liệu hình ảnh và thực hiện quá trình nhận dạng trên từng phần ảnh cắt riêng lẻ. Vấn đề gia tăng độ chính xác có thể giải quyết bằng việc kết hợp thuật toán trong lĩnh vực xử lý ảnh như HOG, HOF... để trích xuất các đặc trưng từ video để đưa vào huấn luyện mô hình mạng đã trình bày trong đề tài.

TÀI LIỆU THAM KHẢO

- [1] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series", In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks, MIT Press, 1995.
- [2] F. D, "Batch normalization in Neural Networks,," 20 October 2017. [Online]. Available: <https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c>. [Accessed 7 11 2023].
- [3] A. S. Walia, "AcEvaEon funcEons and it's types-Which is be_er?," [Online]. Available: https://towardsdatascience.com/acEvaEon-funcEons-and-its-types-which-is-be_era9a5310cc8f. [Accessed 8 11 2023].
- [4] Uniqtech, "Understand the Soimax FuncEon in Minutes," 30 2 2018. [Online]. Available: <https://medium.com/data-science-bootcamp/understand-the-soimax-funcEon-in-minutes-f3a59641e86d>. [Accessed 5 11 2023].
- [5] V. Luhaniwal, "Forward propagation in neural networks — Simplified math and code version," 7 5 2019. [Online]. Available: <https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250>. [Accessed 5 December 2023].
- [6] Michael A. Nielsen, ""Neural Networks and Deep Learning",," Thu November 2015. [Online]. Available: <http://neuralnetworksanddeeplearning.com/chap6.html>. [Accessed 22 10 2023].
- [7] A. MOAWAD, "Neural networks and back-propagation explained in a simple way," 1 February 2018. [Online]. Available: <https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e>. [Accessed 4 December 2023].
- [8] V. Bushaev, "Adam – latest trends in deep learning optimization," 22 October 2018. [Online]. Available: [\[https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c\]](https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c). [Accessed 12 December 2023].
- [9] V. Bushaev, "Understanding RMSprop – faster neural network learning," 2 September 2018. [Online]. Available:

[<https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a>]. [Accessed 16 11 2023].

- [10] N. e. a. Srivastava, "Dropout: a simple way to prevent neural networks from overfitting," 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>.. [Accessed 11 11 2023].