|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU, XÂY DỰNG CAMERA AN NINH THÔNG MINH DỰA TRÊN HỌC MÁY**  Ngành: Công nghệ thông tin  Mã số: 7.48.02.01 |
| *Sinh viên thực hiện:*  **Hoàng Thanh Bình**  Lớp: CT1CN  **Hà Nội – 2021**   |  | | --- | | BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM | | ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU, XÂY DỰNG CAMERA AN NINH THÔNG MINH DỰA TRÊN HỌC MÁY**  Ngành: Công nghệ thông tin  Mã số: 7.48.02.01 | | *Sinh viên thực hiện:*  **Hoàng Thanh Bình**  Lớp: CT1CN  *Cán bộ hướng dẫn:*  **TS. Phạm Văn Hưởng**  Khoa Công nghệ thông tin – Học viện Kỹ thuật mật mã  **Hà Nội - 2021** | |

LỜI CẢM ƠN

Trong quá trình thực hiện khoá luận tốt nghiệp, tôi đã nhận được sự giúp đỡ, hướng dẫn, hỗ trợ và động viên từ gia đình, quý thầy cô cùng các bạn. Nhờ đó mà tôi đã hoàn thành được luận văn như mong muốn. Trong những dòng đầu tiên của luận văn tốt nghiệp này, tôi muốn gửi lời cảm ơn và biết ơn chân thành nhất của mình tới tất cả những người đã hỗ trợ, giúp đỡ tôi về kiến thức và tinh thần trong quá trình thực hiện khoá luận.

Đầu tiên, tôi xin gửi lời cảm ơn chân thành tới các thầy cô trong Học viện Kỹ thuật mật mã nói chung và khoa Công nghệ thông tin nói riêng đã tận tình giảng dạy, truyền đạt cho tôi những kiến thức và kinh nghiệm quý báu trong suốt thời gian là sinh viên học viện; cũng như sự quan tâm và tạo mọi điều kiện thuận lợi cho tôi trong quá trình thực hiện luận văn tốt nghiệp này.

Và để hoàn thành đồ án tốt nghiệp này, tôi xin được gửi lời cảm ơn chân thành và sự biết ơn sâu sắc tới giảng viên hướng dẫn TS. Phạm Văn Hưởng đã tận tình giúp đỡ, trực tiếp chỉ dạy, hướng dẫn tôi trong suốt quá trình làm khóa luận.

Sau cùng xin được gửi lời cảm ơn chân thành tới gia đình, anh chị, bạn bè đã động viên, đóng góp ý kiến và giúp đỡ chúng tôi trong quá trình học tập, nghiên cứu và hoàn thành khóa luận.

Tôi xin chân thành cảm ơn!

|  |  |
| --- | --- |
|  | **Sinh viên thực hiện**  Hoàng Thanh Bình |
|  |  |

**MỤC LỤC**

LỜI CẢM ƠN i

DANH MỤC KÝ HIỆU, TỪ VIẾT TẮT v

DANH MỤC HÌNH VẼ vi

MỞ ĐẦU 1

CHƯƠNG 1. TỔNG QUAN VỀ CAMERA AN NINH THÔNG MINH 3

1.1 Khái niệm và cấu tạo của camera an ninh 3

1.1.1 Khái niệm camera an ninh 3

1.1.2 Cấu tạo của camera an ninh 3

1.2 Các loại camera 4

1.3 Camera an ninh thông minh 7

1.3.1 Khái niệm 7

1.3.2 Mạng Nơ-ron nhân tạo 8

1.3.2.1 Khái niệm Nơ-ron 8

1.3.2.2 Cấu trục mạng nơ-ron nhân tạo 8

1.3.2.3 Mạng Deep Learning Neural Network 12

1.3.2.4 Cách mô hình hoạt động 14

1.3.2.5 Lan truyền tiến 15

1.3.2.6 Hàm mất mát 16

1.3.2.7 Lan truyền ngược 17

1.3.3 Ứng dụng của camera an ninh thông minh 18

CHƯƠNG 2. CƠ SỞ LÍ THUYẾT 21

2.1 Máy tính nhúng Raspberry 21

2.1.1 Giới thiệu chung về máy tính nhúng Raspberry 21

2.1.2 Hệ điều hành cho Raspberry 24

2.2 Mạng nơ-ron tích chập (Convolutional Neural Network) 25

2.2.1 CNN cho bài toán phân loại ảnh 25

2.2.1.1 Convolution layer 27

2.2.1.2 Pooling Layer 34

2.2.1.3 Fully Connected Layer 34

2.2.2 Mô hình YOLO 35

2.2.2.1 YOLO cho bài toán phát hiện đối tượng 35

2.2.2.2 Kiến trúc YOLO 36

2.3 Thư viện OpenCV 44

2.3.1 OpenCV 44

2.3.2 Các module phổ biến trong OpenCV 44

2.3.3 Ứng dụng của OpenCV 45

2.4 Tổng quan về mã hóa công khai RSA 45

2.4.1 Giải thuật mã hóa công khai RSA 45

2.4.1.1 Tạo khóa 46

2.4.1.2 Mã hóa 46

2.4.1.3 Giải mã 47

2.4.1.4 Ví dụ 47

2.4.2 Các vấn đề của RSA 48

2.5 Mã hóa AES 48

2.5.1 Khái niệm 48

2.5.2 Hoạt động của AES 48

2.5.3 Thuật toán AES 48

2.5.3.1 Bước SubBytes 50

2.5.3.2 Bước ShiftRows 51

2.5.3.3 Bước MixColumns 51

2.5.3.4 Bước AddRoundKey 52

2.5.4 Bảo mật AES 52

2.6 Flask Framework 52

2.7 Hàng đợi tin nhắn (Message Queue) 53

2.7.1 Hàng đợi tin nhắn 53

2.7.1.1 Các loại Message Queue 54

2.7.1.2 Ưu nhược điểm của Message queue: 55

2.7.2 ZMQ 56

2.7.2.1 Khái niệm 56

2.7.2.2 Các mô hình ZMQ 56

2.7.3.1 Khái niệm 58

2.7.3.2 Đặc trưng Image ZMQ 58

CHƯƠNG 3. XÂY DỰNG CAMERA AN NINH THÔNG MINH SỬ DỤNG RASPBERRY 60

3.1 Mô tả hoạt động của hệ thống 60

3.1.1 Sơ đồ khối của hệ thống 60

3.1.2 Hoạt động của hệ thống 61

3.1.3 Chức năng của hệ thống 62

3.2 Thiết kế phần cứng 62

3.2.1 Cấu hình Server 62

3.2.2 Hệ thống camera an ninh 63

3.2.2.1 Raspberry Pi 3 Model B+ 63

3.2.2.2 Camera Pi 64

3.3 Lập trình và tích hợp phần mềm 64

3.3.1 Môi trường lập trình 64

3.3.2 Cài đặt phần mềm cho Raspberry Pi 65

3.3.2.1 Chuẩn bị các thiết bị dụng cụ cần thiết để thiết lập 65

3.3.2.2 Cài đặt Raspbian cho Raspberry 66

3.3.3 Lập trình phần mềm 66

3.3.3.1 Trao đổi khóa mã hóa AES bằng mã hóa công khai RSA 66

3.3.3.2 Mã hóa dữ liệu bằng AES 68

3.3.3.3 Bảo mật lưu trữ video 68

3.3.3.4 Phát hiện người bằng Yolo 69

3.3.3.5 Nhận ảnh và đưa lên stream 71

3.3.3.6 Lưu video và gửi cảnh báo khi phát hiện bất thường 72

3.3.3.7 Xem lại video đã lưu 73

3.3.3.8 Chức năng đăng nhập 74

3.4 Kết quả và thực nghiệm 75

3.4.1 Kết quả đạt được 75

3.4.2 Thực nghiệm 76

3.4.2.1 Web server 76

3.4.2.2 Xem video trực tiếp 76

3.4.2.3 Gửi mail cảnh báo khi phát hiện bất thường 77

3.4.2.4 Xem lại các video đã lưu khi phát hiện bất thường 77

3.4.2.5 Đăng nhập 77

KẾT LUẬN 80

TÀI LIỆU THAM KHẢO 81

PHỤ LỤC 83

DANH MỤC KÝ HIỆU, TỪ VIẾT TẮT

|  |  |
| --- | --- |
| Viết tắt | Tiếng anh |
| ARM | Acorn RISC Machine |
| API | Application Programming Interface |
| AES | Advanced Encryption Standard |
| CCTV | Closed-circuit television |
| CPU | Central Processing Unit |
| CNN | Convolutional Neural Networks |
| GPU | Graphics Processing Unit |
| HTTP | HyperText Transfer Protocol |
| IoT | Internet of Things |
| I/O | Input/Output |
| IP | Internet Protocol |
| IEEE | Institute of Electrical and Electronics Engineers |
| LAN | Local Area Network |
| LED | Light Emitting Diode |
| OpenCV | Open Source Computer Vision Library |
| RAM | Random Access Memory |
| ReLU | Rectified Linear Units |
| RSA | Rivest-Shamir-Adleman |
| ZMQ | Zero Message Queue |
| VGA | Video Graphics Array |

# **DANH MỤC HÌNH VẼ**

[Hình 1.1 Cấu tạo camera an ninh 3](#_Toc76753399)

[Hình 1.2 Camera hồng ngoại 5](#_Toc76753400)

[Hình 1.3 Camera ip 6](#_Toc76753401)

[Hình 1.4 Camera không dây 6](#_Toc76753402)

[Hình 1.5 Hình ảnh camera phát hiện người 7](#_Toc76753403)

[Hình 1.6 Nơ-ron trong não bộ người 8](#_Toc76753404)

[Hình 1.7 Mô hình nơ-ron nhân tạo 9](#_Toc76753405)

[Hình 1.8 Cấu trúc nơ-ron nhân tạo 9](#_Toc76753406)

[Hình 1.9 Đồ thị hàm Sigmoid 11](#_Toc76753407)

[Hình 1.10 Đồ thị hàm Tanh 11](#_Toc76753408)

[Hình 1.11 Đồ thị hàm RELU 12](#_Toc76753409)

[Hình 1.12 Mô hình Deep Learning Neural Network 13](#_Toc76753410)

[Hình 1.13 Mô hình Deep Learning tổng quát 13](#_Toc76753411)

[Hình 1.14 Cách mô hình được đào tạo và dự đoán 14](#_Toc76753412)

[Hình 1.15 Các tham số trong mạng neural networks 15](#_Toc76753413)

[Hình 1.16 Camera an ninh trong lĩnh vực bảo vệ 18](#_Toc76753414)

[Hình 1.17 Camera an ninh trong trường học 19](#_Toc76753415)

[Hình 2.1 Máy tính nhúng Raspberry Pi 21](#_Toc76753416)

[Hình 2.2 Raspberry Pi Zero 21](#_Toc76753417)

[Hình 2.3 Raspberry Pi 22](#_Toc76753418)

[Hình 2.4 Raspberry Pi 2 Model B 22](#_Toc76753419)

[Hình 2.5 Raspberry Pi 3 Model B 23](#_Toc76753420)

[Hình 2.6 Raspberry Pi 4 23](#_Toc76753421)

[Hình 2.7 Mô hình học sâu CNN 26](#_Toc76753422)

[Hình 2.8 Filter 5x5 27](#_Toc76753423)

[Hình 2.9 Phép tính tích chập trên ma trận 28](#_Toc76753424)

[Hình 2.10 Kích thước của ma trận đầu ra của phép tích chập 29](#_Toc76753425)

[Hình 2.11 Ma trận được thêm các phần tử 0 làm padding 29](#_Toc76753426)

[Hình 2.12 Các phần tử được thực hiện tích chập với bước nhảy k = 2 30](#_Toc76753427)

[Hình 2.13 Phép tích chập trên ảnh màu 31](#_Toc76753428)

[Hình 2.14 Phép tính tích chập trên ma trận 3 chiều 32](#_Toc76753429)

[Hình 2.15 Thực hiện tính tích chập trên ảnh màu 32](#_Toc76753430)

[Hình 2.16 Đầu ra của lớp tích chập trên ảnh màu 33](#_Toc76753431)

[Hình 2.17 Mô hình tổng quát cho lớp tích chập 33](#_Toc76753432)

[Hình 2.18 Max pooling 34](#_Toc76753433)

[Hình 2.19 Phẳng hoá ma trận 35](#_Toc76753434)

[Hình 2.20 Lớp Fully Connected 35](file:///D:\KMA\Do%20An%20Tot%20Nghiep\baocaoDATNvFinal.docx#_Toc76753435)

[Hình 2.21 Sơ đồ kiến trúc mạng yolo 37](#_Toc76753436)

[Hình 2.22 Các layer trong mạng darknet 53 37](#_Toc76753437)

[Hình 2.23 Outut của Yolo 39](#_Toc76753438)

[Hình 2.24 Anchor box cho việc xác định vật thể 40](#_Toc76753439)

[Hình 2.25 Cách tính chỉ số IoU 41](#_Toc76753440)

[Hình 2.26 Dự đoán các thông số bounding box 42](#_Toc76753441)

[Hình 2.27 Non-max suppression 44](#_Toc76753442)

[Hình 2.28 Sơ đồ biểu diễn thuật toán RSA 46](#_Toc76753443)

[Hình 2.29 Mã hóa AES 49](#_Toc76753444)

[Hình 2.30 SubBytes 50](#_Toc76753445)

[Hình 2.31 Bảng S-box 50](#_Toc76753446)

[Hình 2.32 ShiftRows 51](#_Toc76753447)

[Hình 2.33 MixColumns 51](#_Toc76753448)

[Hình 2.34 Phân tích đa thức 52](#_Toc76753449)

[Hình 2.35 Hàng đợi tin nhắn (Message queue) 54](#_Toc76753450)

[Hình 2.36 Message queue Point-to-point 54](#_Toc76753451)

[Hình 2.37 Message queue Publisher-Subscriber 55](#_Toc76753452)

[Hình 2.38 Mô hình Pair trong ZeroMQ 56](#_Toc76753453)

[Hình 2.39 Mô hình REQ/REP trong ZeroMQ 57](#_Toc76753454)

[Hình 2.40 Mô hình Pub/ Sub trong ZeroMQ 57](#_Toc76753455)

[Hình 2.41 Mô hình Push/Pull trong ZeroMQ 58](#_Toc76753456)

[Hình 3.1 Sơ đồ hoạt động hệ thống 60](#_Toc76753457)

[Hình 3.2 Cấu hình của hệ thống máy chủ 62](#_Toc76753458)

[Hình 3.3 Raspberry Pi 3 Model B+ 63](#_Toc76753459)

[Hình 3.4 Camera pi 64](#_Toc76753460)

[Hình 3.5 Giao diện của sublime text 65](#_Toc76753461)

[Hình 3.6 Giao diện của Raspberry khi cài đặt hệ điều hành Raspbian 66](#_Toc76753462)

[Hình 3.7 Giao diện Web Server 76](#_Toc76753463)

[Hình 3.8 Video stream hệ thống khi phát hiện người 76](#_Toc76753464)

[Hình 3.9 Mail cảnh báo khi phát hiện người 77](#_Toc76753465)

[Hình 3.10 Các video đã lưu khi phát hiện người 77](#_Toc76753466)

[Hình 3.11 Thông báo khi nhập sai thông tin đăng nhập 78](#_Toc76753467)

[Hình 3.12 Giao diện khi nhập đúng thông tin đăng nhập 78](#_Toc76753468)

# 

# **MỞ ĐẦU**

Việc ứng dụng camera để giám sát an ninh khu vực đã được đưa vào thực tế từ rất lâu. Theo dòng phát triển khoa học công nghệ, các mạng camera giám sát phát triển không ngừng và trở thành thiết bị không còn xa lạ gì trong cuộc sống của chúng ta.

Nó xuất hiện ở trường học, đường phố, văn phòng, bãi gửi xe….Tuy nhiên, còn có nhiều vấn đề hơn là việc chỉ giám sát thông thường vì camera hiện nay còn thực hiện chức năng như một “vệ sĩ thực thụ”. Đó chính là những chiếc camera thông minh thế hệ mới đầy tính năng hiện đại.

Xu hướng phát triển công nghệ giám sát đang là cuộc cách mạng công nghệ cao hiện nay trên thế giới. Một loạt giải pháp được kết hợp trong giám sát nhằm đưa ra các biện pháp an toàn cho xã hội và an ninh quốc gia. Chính vì vậy việc sử dụng hệ thống giám sát là sự lựa chọn tối ưu và được ứng dụng rộng rãi trên thế giới nói chung và Việt Nam nói riêng. Trong tương lai gần, camera giám sát được ứng dụng phổ biến trong tất cả các căn nhà và các công ty trong việc giám sát, quản lý và bảo vệ tài sản.

Với các lý do trên, tôi chọn đề tài “**Nghiên cứu, xây dựng camera an ninh thông minh dựa trên học máy**” làm đề tài đồ án tốt nghiệp.

Nội dung báo cáo được chia làm bốn chương cụ thể như sau:

**Chương 1:** Tổng quan về camera an ninh thông minh.

**Chương 2:** Cơ sở lý thuyết để xây dựng hệ thống camera.

**Chương 3:** Triển khai xây dựng camera thông minh sử dụng Raspberry Pi.

Mục tiêu của đề tài là xây dựng hệ thống camera thông minh dựa trên học máy có những chức năng:

* Giám sát trực tiếp thông qua camera Raspberry Pi.
* Cảnh báo khi phát hiện bất thường.
* Lưu video khi phát hiện bất thường.
* Xem lại video đã lưu khi phát hiện bất thường.

Đối tượng nghiên cứu của đề tài:

* Phần cứng: Raspberry Pi 3, Camera Pi.
* Ngôn ngữ lập trình:
* Web ( front-end: Html, css, js, bootstrap ; backend: python) .
* Raspberry: hệ điều hành raspbian, Flask Framework.
* Công cụ phát triển: Sublime text.

Phạm vi nghiên cứu của đề tài:

* Nghiên cứu về Raspberry Pi và thiết kế xây dựng camera an ninh dựa trên Raspberry Pi.
* Nghiên cứu một số kĩ thuật thư viện học máy cho Raspberry Pi.
* Tích hợp kĩ thuật học máy để xây dựng camera thông minh dựa trên Raspberry Pi.

Kết quả dự kiến của đề tài:

* Hệ thống camera an ninh hoạt động ổn định, có thể phát hiện bất thường và gửi cảnh báo, lưu video khi phát hiện bất thường.
* Xây dựng server để xem video trực tiếp từ camera cũng như xem lại video đã lưu khi phát hiện bất thường.
* Các tài liệu liên quan

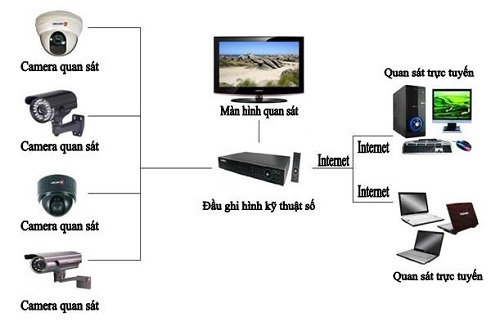
# **CHƯƠNG 1. TỔNG QUAN VỀ CAMERA AN NINH THÔNG MINH**

## **Khái niệm và cấu tạo của camera an ninh**

### **Khái niệm camera an ninh**

Camera an ninh là một thiết bị được lắp đặt để hỗ trợ việc giám sát, quản lý từ xa. Nó được coi là “con mắt điện tử” có thể nhìn, tiếp nhận hình ảnh một cách chính xác tuyệt đối sau đó truyền tín hiệu về thiết bị di động hoặc là thiết bị phát. Thiết bị camera giám sát phù hợp để lắp đặt ở mọi nơi, mọi điều kiện có thể ở nhà, xưởng máy, văn phòng, khách sạn, ngân hàng….

* + 1. **Cấu tạo của camera an ninh**



Hình 1.1 Cấu tạo camera an ninh

Hệ thống camera an ninh thường có những bộ phận sau:

* Ống kính

Dùng để quan sát hình ảnh tùy theo yêu cầu về phạm vi, mục tiêu quan sát sử dụng ống kính khác nhau.

* Màn hình

Để hiện thị hình ảnh trực tiếp từ camera cũng như các video đã lưu lại trong thời gian trước đó.

* Bộ ghi hình

Bộ ghi hình để cho người dùng có thể xem lại những video hình ảnh đã quay trước đó với mục đích khảo sát về sau.

* Bàn phím điều khiển

Với hệ thống nhiều camera giám sát để thuận lợi cho việc điều khiển thì nên có một bàn phím điều khiển. Nó giúp cho người quản trị thuận lợi hơn trong việc giám sát một hệ thống lớn và quản lý tốt hơn.

* Chân đế cố định/ quay

Chân cố định dùng để cố định camera. Chân quay để chỉnh camera theo một hướng nhất định cần quan sát.

* Hộp bảo vệ camera

Để bảo về tránh cho camera hỏng khi ngoài trời, để cho có thẩm mĩ khi đặt trong nhà hay là tránh ảnh hưởng khi quay.

## **Các loại camera**

Hiện nay sự phát triển của công nghệ và nhu cầu sử dụng camera ngày càng nhiều, camera đã trở thành một trong những thiết bị thông dụng trong các nhà máy, những nơi làm việc có tính bảo mật cao hay là ở tất cả các ngân hàng trên thế giới hiện nay. Vì vậy các nhà sản xuất nghiên cứu ra các loại camera phù hợp với các khu vực cần đến sự giám sát cao và chính xác. Sau đây tôi có đưa ra một số loại camera thông dụng được sử dụng hiện nay.

* + 1. **Camera hồng ngoại**

Camera hồng ngoại là loại camera được tích hợp thêm cảm biến ánh sáng và cảm biến hồng ngoại để có thể quan sát trong đêm tối.

Cảm biến ánh sáng là bộ phận được dùng để nhận biết sự thay đổi của ánh sáng bên ngoài chuyển từ ngày sang đêm và ngược lại.

Đèn hồng ngoại hay còn gọi là led hồng ngoại hoạt động dựa vào cảm biến ánh sáng. Nếu cảm biến phát hiện sự thiếu ánh sáng thì led hồng ngoại sẽ được kích hoạt giúp camera có thể ghi hình trong bóng tối và xuất hình ảnh dưới dạng màu trắng đen.



Hình 1.2 Camera hồng ngoại

* + 1. **Camera IP**

**Camera IP là loại camera quan sát có hình ảnh được số hóa, xử lí và mã hóa ngay bên trong camera, sau đó truyền tải tín hiệu số đó qua một kết nối mạng về máy tính, hệ thống server hoặc đầu ghi hình IP, ….**

**Ưu điểm của hệ thống camera IP là nó không phải dùng các phần cứng khác như đầu thu video, màn hình CCTV, các thiết bị chuyển mạch, ….Camera IP chụp ảnh và xử lí hình ảnh trước khi truyền nên nó cung cấp hình ảnh video sắc nét hơn so với camera analog.Một ưu điểm của camera IP nữa là khả năng xem từ xa. Ta có thể truy cập xem camera IP từ bất cứ đâu khi có internet.**

**Vì camera IP sử dụng internet nên nhược điểm lớn nhất của camera IP đó là khả năng bị tấn công qua internet nếu bảo mật không cao. Một nhược điểm khác đó là chi phí của camera IP cao hơn rất nhiều so với camera analog và nó đòi hỏi phải có một hạ tầng mạng hoạt động ổn định.**



Hình 1.3 Camera ip

* + 1. **Camera không dây**

Là camera IP nhưng nó không kết nối internet thông qua dây mạng mà nó kết nốt qua sóng wifi. Loại camera này rất tiện lợi, nó có thể đặt ở bất kì đâu mà không phải lo về độ dài dây dẫn. Tuy nhiên nhược điểm của loại camera này chi phí cao và hoạt động không ổn định khi mạng wifi yếu.



Hình 1.4 Camera không dây

## **1.3 Camera an ninh thông minh**

### **1.3.1 Khái niệm**

Camera thông minh ngoài việc là một chiếc camera giám sát, nó còn được tích hợp AI để nhận diện đối tượng và gửi cảnh báo đến người dùng.



Hình 1.5 Hình ảnh camera phát hiện người

Một vài hệ thống hiện tại được xây dựng dựa trên những ứng dụng của thị giác máy tính, các mô hình học sâu, thư viện OpenCV.Các mô hình học sâu (Deep learning) đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn thơ hay âm nhạc.

Để làm được điều đó, việc xây dựng các mô hình Deep Learning được lấy cảm hứng từ bộ não con người. Các thuật toán Deep Learning cố gắng đưa ra các kết luận tương tự như con người bằng cách liên tục phân tích dữ liệu với một cấu trúc logic nhất định. Cấu trúc này chạy qua một thuật toán gồm có nhiều lớp và được gọi là Neural Networks (Mạng Nơ-ron).

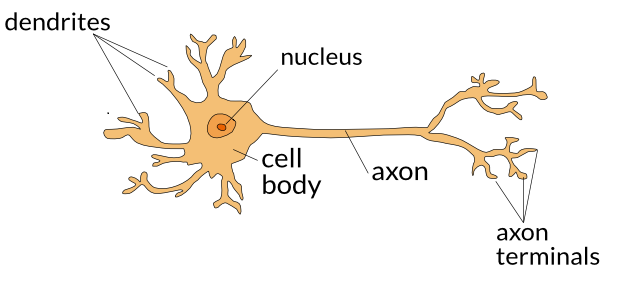
### **1.3.2 Mạng Nơ-ron nhân tạo**

#### **1.3.2.1 Khái niệm Nơ-ron**

Nơ-ron là đơn vị cơ bản cấu tạo hệ thống thần kinh và là một phần quan trọng nhất của não. Não chúng ta gồm khoảng 10 triệu nơ-ron và mỗi nơ-ron liên kết với 10.000 nơ-ron khác.

Ở mỗi nơ-ron có phần thân (soma) chứa nhân, các tín hiệu đầu vào qua sợi nhánh (dendrites) và các tín hiệu đầu ra qua sợi trục (axon) kết nối với các nơ-ron khác. Hiểu đơn giản mỗi nơ-ron nhận dữ liệu đầu vào qua sợi nhánh và truyền dữ liệu đầu ra qua sợi trục, đến các sợi nhánh của các nơ-ron khác.

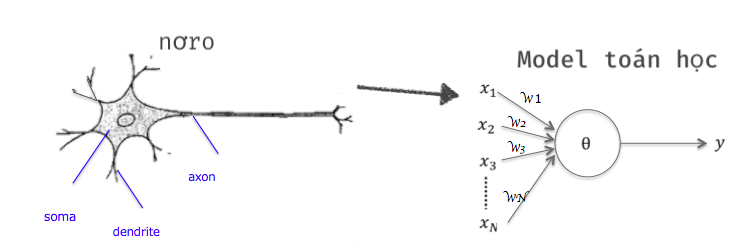
Mỗi nơ-ron nhận xung điện từ các nơ-ron khác qua sợi nhánh. Nếu các xung điện này đủ lớn để kích hoạt nơ-ron, thì tín hiệu này đi qua sợi trục đến các sợi nhánh của các nơ-ron khác.



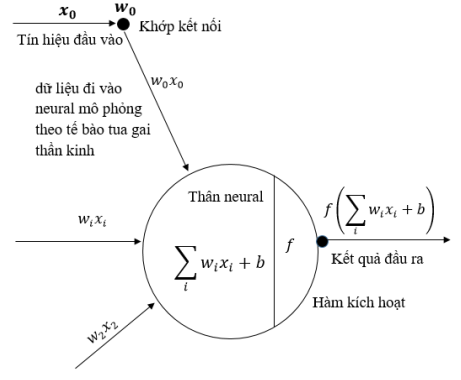
Hình 1.6 Nơ-ron trong não bộ người

#### **1.3.2.2 Cấu trục mạng nơ-ron nhân tạo**

Dựa vào nguyên tắc hoạt động của nơ-ron sinh học, các nhà khoa học và lập trình đã đưa ra cấu trúc của nơ-ron nhân tạo.



Hình 1.7 Mô hình nơ-ron nhân tạo



Hình 1.8 Cấu trúc nơ-ron nhân tạo

Trong đó:

* + Danh sách đầu vào: Là các thuộc tính đầu vào của một nơ-ron. Số lượng thuộc tính đầu vào thường nhiều hơn một, do dữ liệu thô đầu vào thường là một vector nhiều chiều, hoặc nhiều nơ-ron tầng trước kết nối tới một nơ-ron tầng sau
  + Trọng số liên kết: Các liên kết được thể hiện độ mạnh yếu qua một giá trị được gọi là trọng số liên kết. Kết hợp với các đầu truyền, tín hiệu đến các nơ-ron nhân tạo khác sẽ được tính bằng
  + Hàm tổng: Tổng các tích của các đầu vào với trọng số liên kết mô phỏng các khớp kết nối. Được tính theo công thức:

|  |
| --- |
|  |

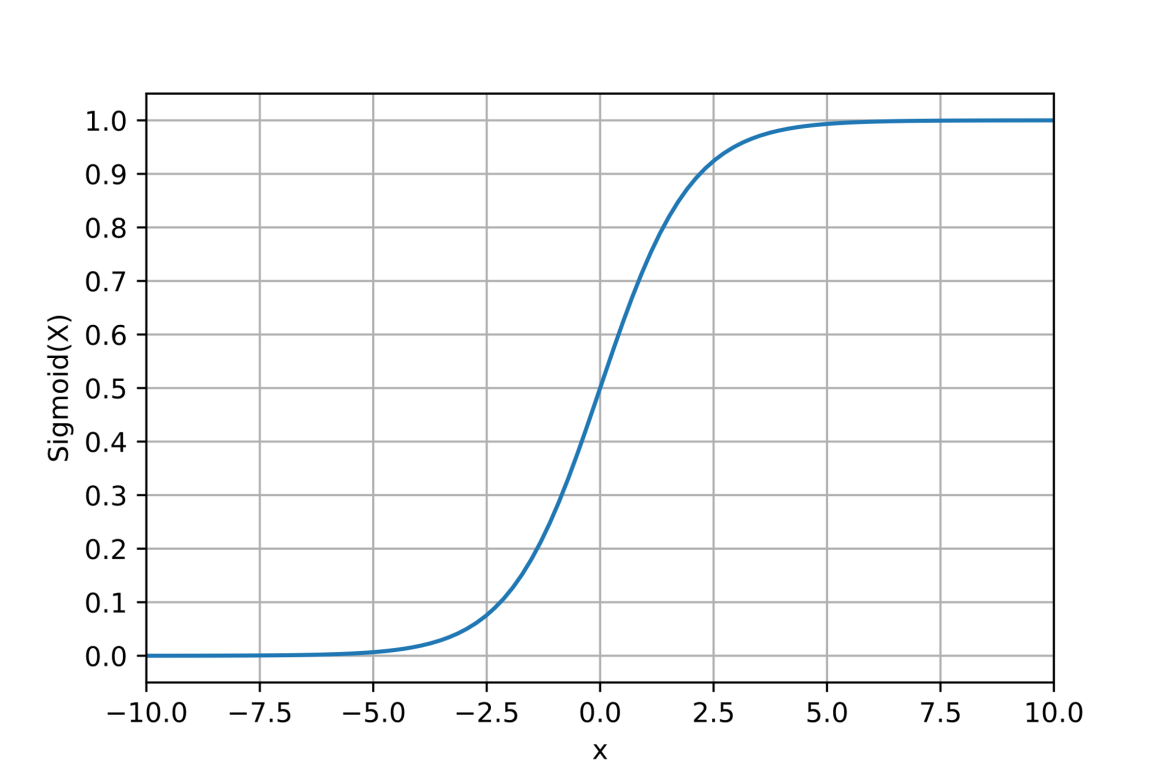
* Hàm kích hoạt: Hàm này được sử dụng để tính toán giá trị của đầu ra dựa vào giá trị của hàm tổng. Một số hàm kích hoạt được sử dụng trong mạng nơ-ron nhân tạo
* Hàm sigmoid
* Biểu diễn hàm



* Đạo hàm của hàm



Hàm Sigmoid được sử dụng vì ngưỡng của nó nằm trong khoảng (0,1). Do đó, hàm này được sử dụng nhiều cho các mô hình dự đoán xác suất đầu ra, tức kết quả chỉ tồn tại trong khoảng từ 0 đến 1: Khi đầu vào là số dương lớn, đầu ra của hàm sigmoid gần bằng 1. Khi đầu vào nhỏ hơn 0, đầu ra gần bằng 0. Tuy nhiên, việc tối ưu của hàm này tương đối khó khăn, nguyên nhân vì nếu giá trị đầu vào của hàm là 1 số rất lớn thì đầu ra của hàm càng về 2 đầu xấp xỉ 1 hoặc 0, nên tốc độ hội tụ sẽ rất chậm.



Hình 1.9 Đồ thị hàm Sigmoid

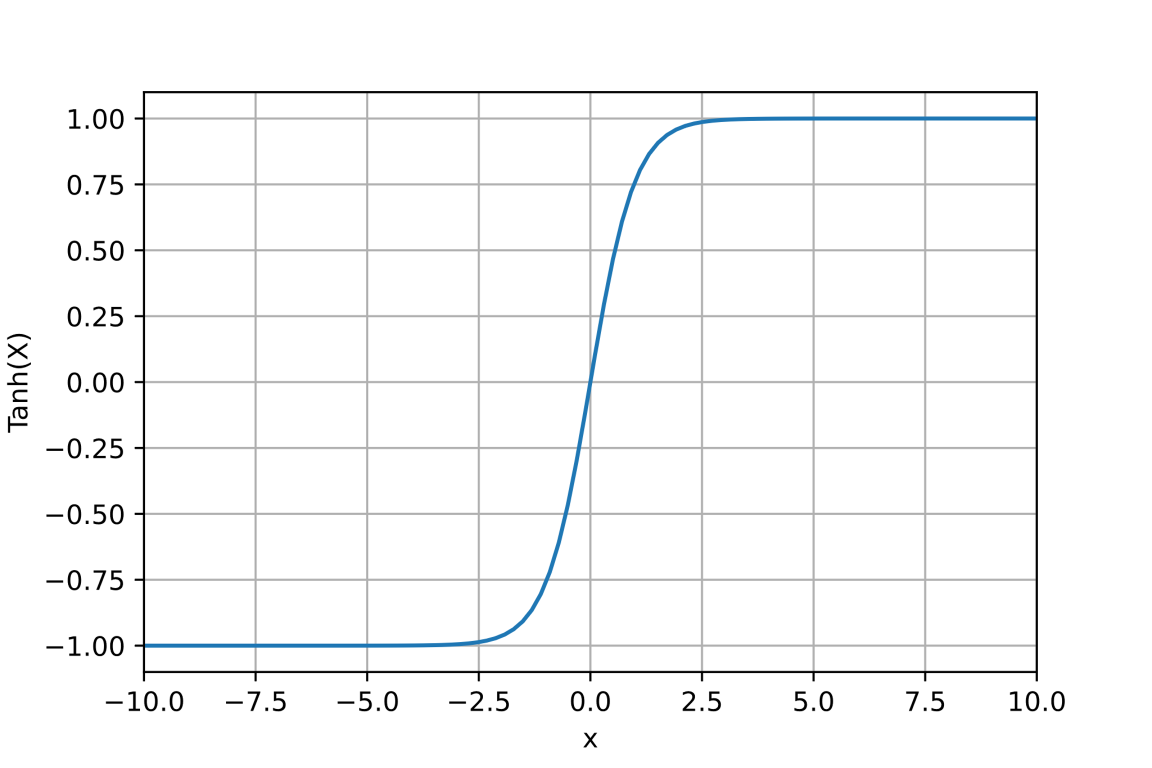
* Hàm tanh
* Biểu diễn hàm



* Đạo hàm của hàm



Hàm tanh được sử dụng vì đầu ra của hàm nằm trong khoảng (-1,1), thích hợp với các mô hình đầu ra có ba giá trị: âm, trung tính, dương.



Hình 1.10 Đồ thị hàm Tanh

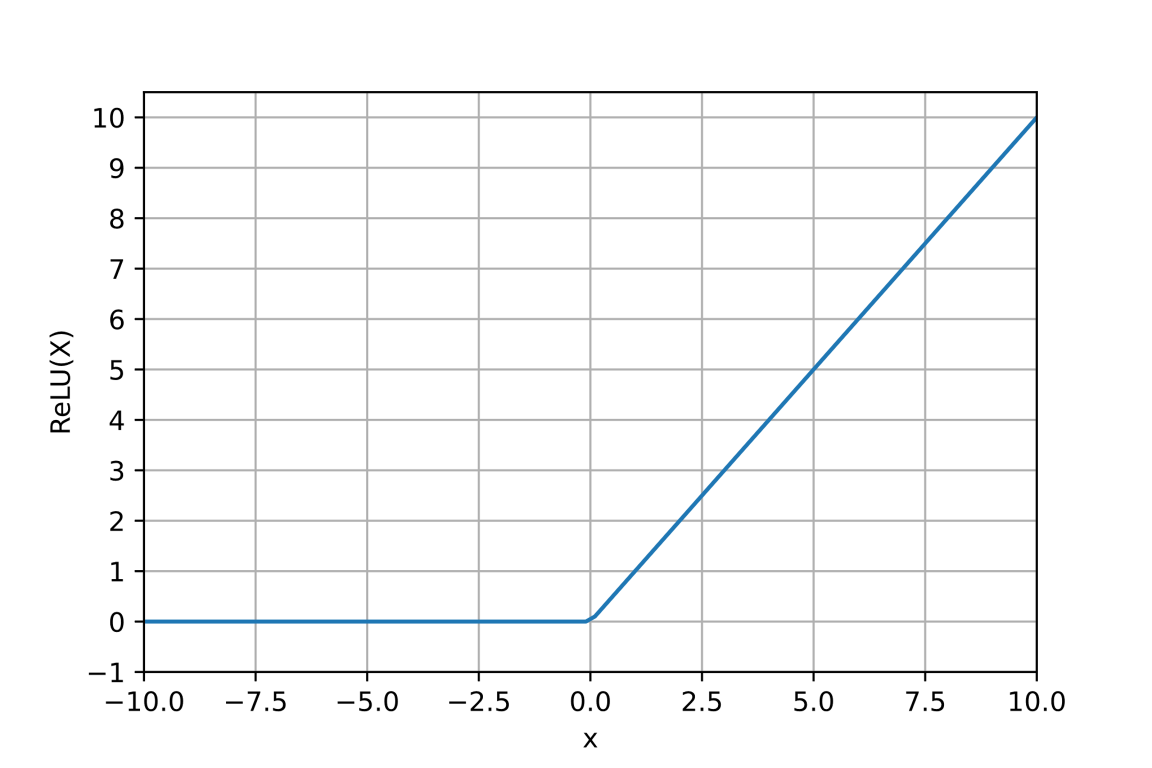
* Hàm RELU
* Biểu diễn hàm



* Đạo hàm của hàm



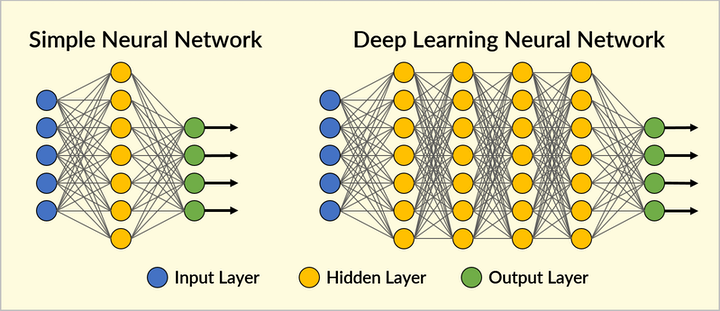
Hàm ReLU áp dụng với nhưng trường hợp cần đầu ra nằm trong khoảng (0,). Hàm ReLU có tốc độ tính toán rất nhanh, gán các giá trị âm trở thành 0 ngay lập tức, phù hợp cho việc huấn luyện từ dữ liệu chuẩn. Tuy nhiên, điều này khiến hàm ReLU không ánh xạ các giá trị âm một cách thích hợp.



Hình 1.11 Đồ thị hàm RELU

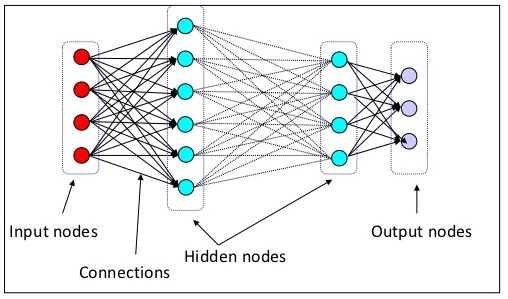
#### **1.3.2.3 Mạng Deep Learning Neural Network**

Cũng giống như việc chúng ta sử dụng bộ não của mình để xác định và phân loại các thông tin khác nhau. Bất cứ khi nào chúng ta nhận được thông tin mới, bộ não sẽ cố gắng so sánh nó với các đối tượng đã biết. Mạng nơ-ron cũng có thể được dạy để thực hiện các tác vụ giống nhau trên tập dữ liệu bằng cách thức tương tự như với bộ não của chúng ta. Để bộ não có thể thực hiện được điều đó, các thông tin cần được lan truyền qua nhiều nơ-ron, sau đó đưa ra quyết định và mô hình mạng nơ-ron muốn làm được điều tương tự thì nó cũng cần phải được xây dựng theo cấu trúc tương tự - cấu trúc nhiều lớp, nhiều nơ-ron hay còn được gọi là Deep Learning Neural Network.



Hình 1.12 Mô hình Deep Learning Neural Network

Deep Learning cũng giống như Neural Network nhưng có nhiều lớp ẩn. Việc có nhiều lớp ẩn như vậy giúp mô hình trích xuất được nhiều đặc trưng khác nhau giúp tăng độ chính xác dự đoán của output.



Hình 1.13 Mô hình Deep Learning tổng quát

Cấu trúc của mô hình deep learning được mô tả như hình 1.13, lớp đầu tiên là Input Layer, các lớp ở giữa được gọi là hidden layer, lớp cuối cùng được gọi là output layer. Còn các hình tròn được gọi là các nơ-ron hay các node. Các hình mũi tên thể hiện sự liên kết, lan truyền thông tin giữa các lớp, các node với nhau.

Mỗi mô hình luôn có 1 input layer, 1 output layer, có thể có hoặc không có các hidden layer. Tổng số lớp trong mô hình được quy ước là số lớp – 1 (không tính lớp input). Ví dụ hình ở trên có 1 input layer, 2 hidden layer và 1 output layer. Số lượng lớp của mô hình là 3.

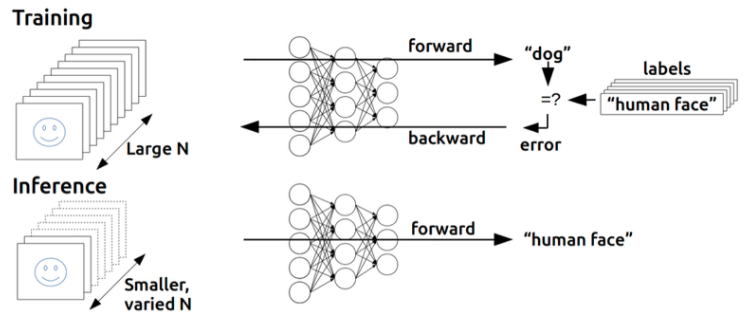
Mỗi node trong hidden layer và output layer có các đặc điểm sau:

* Liên kết với tất cả các node ở lớp trước đó với các hệ số riêng
* Mỗi node có một hệ số bias riêng
* Diễn ra 2 bước tương tự như cấu trúc nơ-ron ở hình 1.18: tính tổng và thực hiện hàm kích hoạt

Ở mỗi lớp, số lượng các node (Nơ-ron) có thể khác nhau tuỳ thuộc vào bài toán và cách giải quyết. Các nơ-ron ở các tầng thường được liên kết đôi một với nhau tạo mạng kết nối đầy đủ (fully connected network).

#### **1.3.2.4 Cách mô hình hoạt động**

Để mô hình có thể dự đoán, chúng ta không chỉ xây dựng cấu trúc mô hình mà còn phải thực hiện đào tạo mô hình trên tập dữ liệu theo mô tả của bài toán.



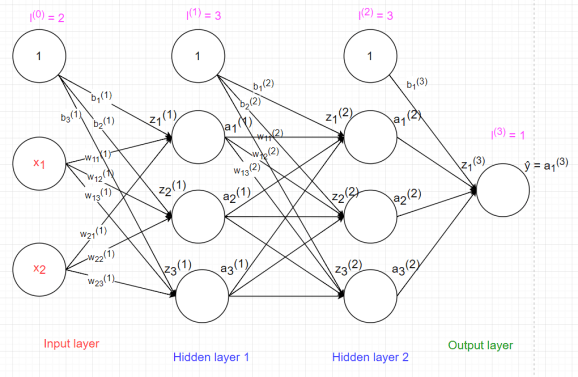
Hình 1.14 Cách mô hình được đào tạo và dự đoán

Quá trình training là quá trình mô hình học được các tham số để rút trích đặc trưng trên tập dữ liệu, từ các đặc trưng này, mô hình sẽ đưa ra dự đoán về thông tin mà nó nhận được. Các tham số (trọng số và bias) sẽ được cập nhật theo mỗi lần mô hình duyệt qua hết một lượt dữ liệu, số lần mô hình duyệt qua hết tập dữ liệu này được gọi là số epochs. Mỗi khi mô hình hoàn thành một epochs, các tham số sẽ được cập nhật dựa vào tổng lỗi của toàn bộ tập dữ liệu. Việc cập nhật này được gọi là lan truyền ngược, còn việc thực hiện tính toán, rút trích đặc trưng trên dữ liệu để đưa ra dự đoán được gọi là lan truyền tiến.

Sau khi training với một số lượng epochs nhất định, tổng lỗi của mô hình có xu hướng hội tụ thì quá trình training kết thúc. Toàn bộ tham số ở epochs có lỗi thấp nhất sẽ được mô hình sử dụng để thực hiện dự đoán những thông tin mới, những thông tin tương tự như trong tập dữ liệu huấn luyện nhưng không có nhãn (label).

#### **1.3.2.5 Lan truyền tiến**

Tất cả các nốt mạng (nơ-ron) được kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào tới tầng ra. Tức là mỗi node ở một tầng nào đó sẽ nhận đầu vào là tất cả các nốt ở tầng trước đó mà không suy luận ngược lại. Hay nói cách khác, việc suy luận trong mạng Neural Networks là suy luận tiến (feedforward).



Hình 1.15 Các tham số trong mạng neural networks

Biểu diễn toán học của quá trình lan truyền tiến:





Trong đó, là số lượng nút ở tầng l tương ứng và là nút mạng thứ j của tầng l. Còn là tham số trọng lượng của đầu vào đối với nút mạng thứ i của tầng l + 1 và là độ lệch (bias) của nút mạng thứ i của tầng l + 1. Đầu ra của nút mạng này được biểu diễn bằng ứng với hàm kích hoạt tương ứng.

Riêng với lớp đầu vào, cũng chính là các đầu vào **x** tương ứng của mạng. Để tiện tính toán, ta coi là một đầu vào và là tham số trọng lượng của đầu vào lúc này. Lúc đó ta có thể viết lại công thức trên dưới dạng vector:



Nếu nhóm các tham số của mỗi tầng thành một ma trận có các cột tương ứng với tham số mỗi nút mạng thì ta có thể tính toán cho toàn bộ các nút trong một tầng bằng vector:

#### **1.3.2.6 Hàm mất mát**

Cũng tương tự như các bài toán học máy khác thì quá trình học vẫn là tìm lấy một hàm mất mát (hàm lỗi) để đánh giá và tìm cách tối ưu hàm lỗi đó để được kết quả hợp lý nhất có thể. Chúng ta xem mỗi nút mạng của Neural Networks là một bộ phân loại logistic regression có hàm lỗi là:

Trong đó, m là số lượng dữ liệu trong tập dữ liệu huấn luyện, là đầu ra thực tế của dữ liệu thứ i trong tập huấn luyện. Còn là kết quả ước lượng được ứng với dữ liệu thứ i.

Hàm mất mát của Neural Network cũng tương tự như vậy, chỉ khác là đầu ra của mạng Neural Network có thể có nhiều node nên khi tính đầu ra ta cũng cần phải tính cho từng node ra đó. Giả sử số node ra là K và là đầu ra thực tế của node thứ k, còn là đầu ra ước lượng được cho node thứ k tương ứng. Khi đó, công thức tính hàm mất mát sẽ thành:

Chúng ta mong muốn rằng sự sai khác giữa giá trị thực và giá trị dự đoán là nhỏ nhất. Hay nói cách khác, chúng ta muốn là tổng sai số là nhỏ nhất, tương đương với việc tìm W để hàm số J(W) đạt giá trị nhỏ nhất.

Giá trị của W làm cho hàm mất mát J(W) có giá trị nhỏ nhất được gọi là điểm tối ưu. Để tìm điểm tối ưu này, phương pháp phổ biến nhất là giải phương trình đạo hàm của hàm mất mát J(W) theo W bằng 0, được ký hiệu như sau:



#### **1.3.2.7 Lan truyền ngược**

Để tính đạo hàm của hàm mất mát trong mạng Neural Network, ta sử dụng một giải thuật đặc biệt là giải thuật lan truyền ngược (backpropagation). Nhờ có giải thuật đuộc sáng tạo vào năm 1986 này mà mạng Neural Network được thực thi hiệu quả và ứng dụng ngày một nhiều cho tới tận ngày nay.

Về cơ bản, phương pháp này được dựa theo quy tắc chuỗi đạo hàm của hàm hợp và phép tính ngược đạo hàm để thu được đạo hàm theo tất cả các tham số cùng lúc với chỉ 2 lần duyệt mạng.

Giải thuật lan truyền ngược được thực hiện như sau:

1. Lan truyền tiến

Lần lượt tính các từ l = 2 L theo công thức:

Trong đó, tầng vào chính bằng giá trị vào của mạng **x**

1. Tính đạo hàm theo ở tầng ra
2. Lan truyền ngược
3. Tính đạo hàm

Tính đạo hàm theo tham số w bằng công thức:

### **1.3.3 Ứng dụng của camera an ninh thông minh**

Các ứng dụng phổ biến gồm có:

* Ứng dụng trong lĩnh vực bảo vệ



Hình 1.16 Camera an ninh trong lĩnh vực bảo vệ

Có thể giúp cho quản lý được người ra vào, báo động danh sách đen khi đã có sẵn trong dữ liệu. Theo dõi được khuôn mặt trong khi cần tìm kiếm.

* Ứng dụng nhận diện trong trường học



Hình 1.17 Camera an ninh trong trường học

* Cảnh báo người lạ trong khu vực ra vào để đảm bảo an toàn, cảnh báo ngay lập tức.
* Ghi chép lại việc đi lại của học sinh.
* Theo dõi khuôn mặt học sinh trong quá trình hoạt động. Nếu học sinh xuất hiện ở những điểm mù cũng được cảnh báo để tránh những trường hợp đáng tiếc xảy ra.

Đảm bảo an toàn trong trường học là việc vô cùng cần thiết trong cuộc sống hiện nay.

* Ứng dụng trong ngân hàng

Ngân hàng là khu vực không thể thiếu camera thông minh. Nhất là thiết bị cao cấp có tính bảo mật cao để tránh tội phạm công nghệ xâm nhập.

* Nhận diện được khách hàng hoặc danh sách đen để cảnh báo khi đối tượng xuất hiện.
* Xác định danh tính đối tượng là khách hàng hay nhân viên bình thường.
* Cảnh bảo lập tức cảnh báo khi đối tượng xấu xuất hiện.

Camera thông minh nhất hiện nay được ngân hàng đầu tư rất lớn, chuyên nghiệp.Nó cung cấp dịch vụ tốt hơn cho ngân hàng, nâng cao tính hiệu quả tính chính xác trong công việc và đảm bảo an toàn tuyệt đối cho ngân hàng.

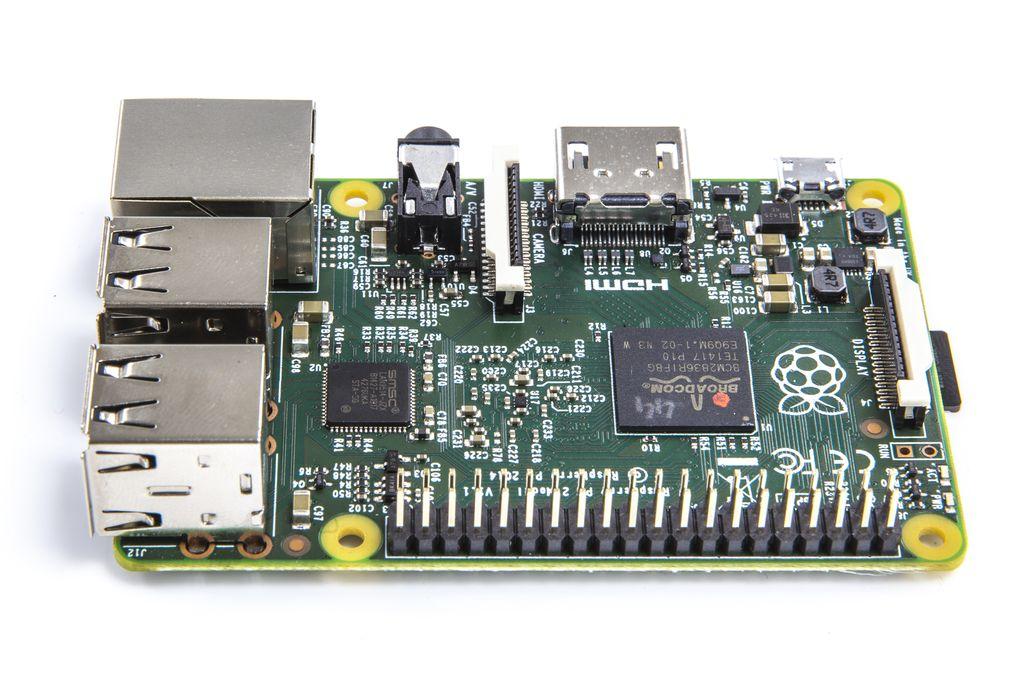
* Ứng dụng nhận diện trong cửa hàng
* Trong khu vực cửa hàng là khu vực kinh doanh. Do vậy, nhận diện khuôn mặt sẽ giúp cho việc tìm kiếm khách hàng như một thói quen mua sắm.
* Quản lý ra vào, báo động ra vào hoặc khi có người lạ xâm nhập. Cảnh bảo người lạ xuất báo cáo để phân tích thống kê cửa hàng.
* Ứng dụng trong văn phòng
* Hệ thống công ty chuyên nghiệp vẫn cần có camera để kiểm soát được người ra vào.
* Lợi ích của nó là xác định nhân viên một cách thuận tiện, chính xác và hiệu quả hơn. Qua đó, nó đảm bảo an toàn các thông tin và tài sản của công ty.

# **CHƯƠNG 2. CƠ SỞ LÍ THUYẾT**

## **2.1 Máy tính nhúng Raspberry**

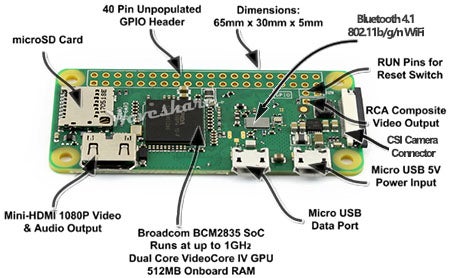
### **2.1.1 Giới thiệu chung về máy tính nhúng Raspberry**

Raspberry Pi là chiếc máy tính kích thước nhỏ được tích hợp nhiều phần cứng đủ khả năng chạy hệ điều hành và cài đặt được nhiều ứng dụng trên nó.Raspbery Pi phù hợp cho những ứng dụng cần khả năng xử lý mạnh mẽ, đa nhiệm hoặc giải trí và đặc biệt cần chi phí thấp.

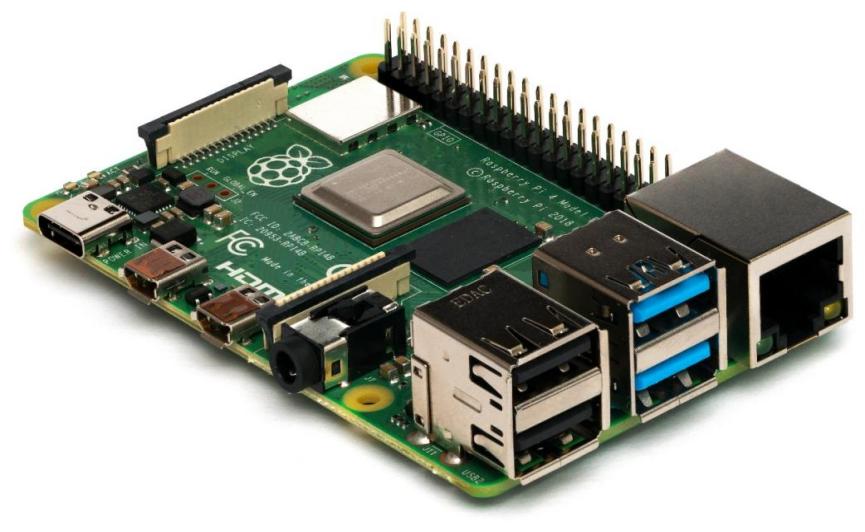


Hình 2.1 Máy tính nhúng Raspberry Pi

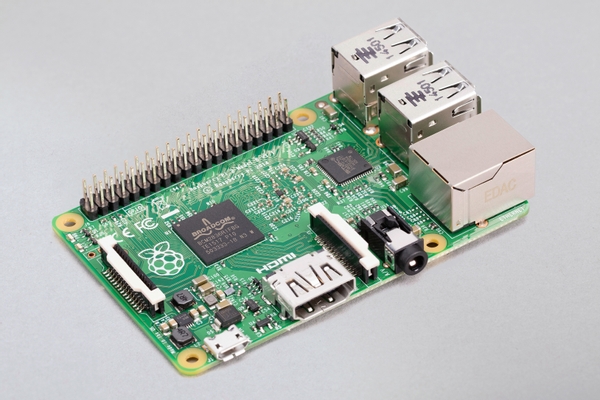
Các model của Raspberry hiện nay:



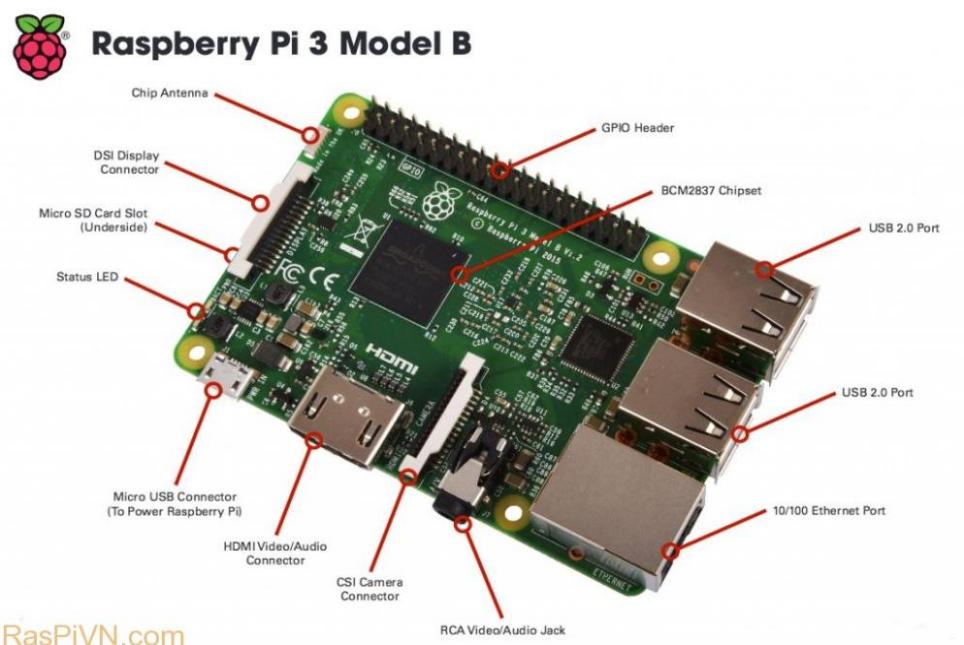
Hình 2.2 Raspberry Pi Zero



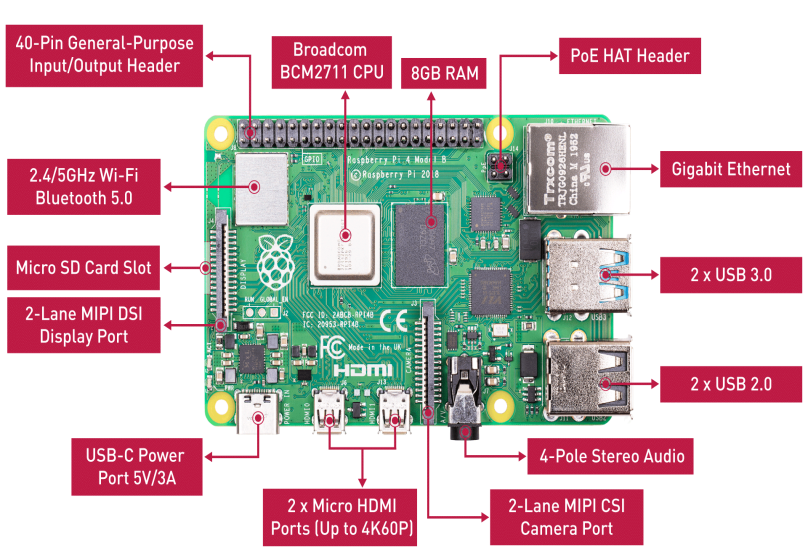
Hình 2.3 Raspberry Pi



Hình 2.4 Raspberry Pi 2 Model B



Hình 2.5 Raspberry Pi 3 Model B



Hình 2.6 Raspberry Pi 4

Bảng 1: Các model của Raspberry

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Họ | Model | Form Factor | Ethernet | Wireless | GPIO | Đã phát hành |
| Raspberry Pi | B | Standard | Có | Không | 26 chân | 2012 |
| A | Không | 2013 |
| B+ | Có | 40 chân | 2014 |
| A+ | Compact | Không | 2014 |
| Raspberry Pi 2 | B | Chuẩn | Có | Không | 2015 |
| Raspberry Pi Zero | Zero | Zero | Không | Không | 2015 |
| W/  WH | Có | 2017 |
| Raspberry Pi 3 | B | Chuẩn | Có | Có | 2016 |
| A+ | Thu nhỏ | Không | 2018 |
| B+ | Chuẩn | Có | 2018 |
| Raspberry Pi 4 | B (1 GiB) | Standard | Có (Gigabit Ethernet) | Có | 2019 |
| B (2 GiB) |
| B (4 GiB) |
| B (8 GiB) | 2020 |

### **2.1.2 Hệ điều hành cho Raspberry**

Một số hệ điều hành được sử dụng cho Raspberry Pi:

* Raspbian: Đây là hệ điều hành phổ biến và được ưa chuộng nhất.
* Sử dụng Raspberry Pi như máy tính thu nhỏ với chức năng văn phòng
* Nghiên cứu và xây dựng hệ thống điều khiển tự động.
* Cung cấp các dịch vụ như web, File server, printer server, ofice, game…
* Ubuntu Mate:

Ubuntu Mate hệ điều hành phù hợp cho nhân viên văn phòng với những tiện ích hỗ trợ. Và hệ điều hành này so với hệ điều hành Raspbian có giao diện đẹp hơn và được phát triển cùng với Ubuntu.

* Snappy Core Ubuntu:

Snappy Core Ubuntu: là một hệ điều hành thu nhỏ của Ubuntu. Và được sử dụng và chạy các ứng dụng đám mây và là một phần quan trọng trong IoT, giúp cho các thiết bị kết nối với nhau thông qua internet một cách hoàn hảo. Nó có thể chạy được nhiều ứng dụng mà không cần một hệ điều hành Ubuntu đầy đủ.

* Windows 10 IoT Core:

Được sử dụng với mục đích phát triển các ứng dụng IoT. Nó được thiết kế để có thể hoạt động với một loạt các ngôn ngữ mã nguồn mở, giúp phát triển và dễ dàng cài đặt trên các thiết bị của riêng mình.

## **2.2 Mạng nơ-ron tích chập (Convolutional Neural Network)**

### **2.2.1 CNN cho bài toán phân loại ảnh**

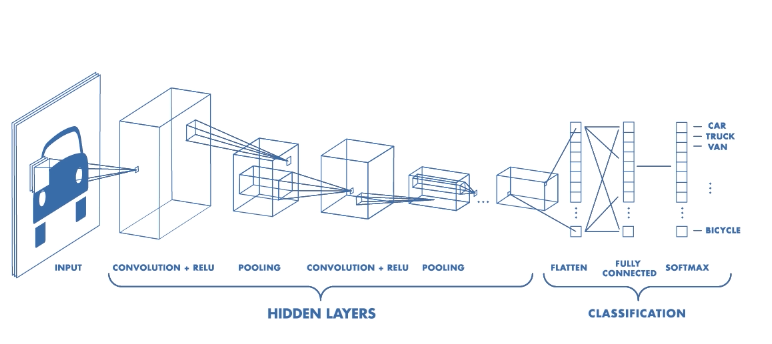
CNN là một trong những mô hình deep learning phổ biến nhất và có ảnh hưởng nhiều nhất trong cộng đồng Computer Vision. CNN được dùng trong nhiều bài toán như nhận dạng ảnh, phân tích video, ảnh MRI, hoặc cho lĩnh vực xử lý ngôn ngữ tự nhiên, và hầu hết đều giải quyết tốt các bài toán này.Với thị giác máy tính và mô hình CNN ta có thể xử lý một số tác vụ tiêu biểu như: phân loại hình ảnh (image classification), định vị vật thể (object localization), phát hiện đối tượng (object detection).

Với bài toán phân loại hình ảnh, đầu vào của bài toán là một hình ảnh, trong đó có chứa một đối tượng chẳng hạn như một chữ số, một bông hoa, một con vật, …. Và nhiệm vụ của mô hình là dự đoán đó là chữ số gì, loài hoa nào, hay con vật thuộc loài gì.

Bài toán định vị vật thể cho phép ta xác định vị trí của vật thể đó ở trong ảnh, hoặc xác định vị trí của nhiều vật thể trong ảnh, với đầu vào của bài toán là hình ảnh có chứa một hoặc nhiều đối tượng, vật thể. Mô hình sẽ trả về những toạ độ của các vật thể trong ảnh và các vật thể sẽ được đóng khung xung quanh chúng. Các khung này còn được gọi là bounding box. Ngoài việc sử dụng các bounding box để xác định toạ độ của đối tượng, ta còn có một định nghĩa khác trong thị giác máy tính để giải quyết bài toán này, đó là object segmentation (phân đoạn đối tượng). Thuật toán phân đoạn này thay vì đóng khung đối tượng như bounding box thì các đối tượng trong ảnh sẽ được làm nổi bật các điểm ảnh cụ thể của đối tượng đó.

Với bài toán phân loại hình ảnh, đầu vào của mô hình chỉ là ảnh có chứa một đối tượng. Như vậy với ảnh chứa nhiều đối tượng, chúng ta không thể phân loại được các đối tượng trong ảnh. Từ đây, bài toán phát hiện đối tượng giải quyết vấn đề trên bằng cách kết hợp hai bài toán phân loại hình ảnh và định vị đối tượng. Với đầu vào của bài toán là một bức ảnh có chứa nhiều đối tượng. Mô hình sẽ xác định vị trí của các đối tượng ở trong ảnh và tạo bounding box xung quanh chúng. Từ những bounding box này, mô hình tiếp tục gán nhãn cho những đối tượng ở trong đó. Đầu ra của bài toán sẽ là một hình ảnh với các đối tượng được xác định nằm trong bounding box và kèm theo nhãn tương ứng.

CNN bao gồm tập hợp các lớp cơ bản: convolution layer + nonlinear layer, pooling layer, fully connected layer. Thông thường, một ảnh sẽ được lan truyền qua tầng convolution layer + nonlinear layer đầu tiên, sau đó các giá trị tính toán được sẽ lan truyền qua pooling layer, bộ ba convolution layer + nonlinear layer + pooling layer có thể được lặp lại nhiều lần trong network. Và sau đó được lan truyền qua tầng fully connected layer và softmax để tính xác suất ảnh đó có đối tượng.

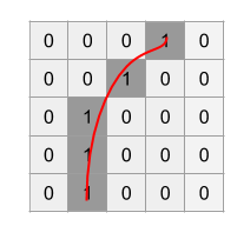


Hình 2.7 Mô hình học sâu CNN

**2.2.1.1 Convolution layer**

Convolution layer (Lớp tích chập): Là lớp đầu tiên đồng thời cũng là lớp giữ vai trò quan trọng nhất trong mô hình CNN. Một hình ảnh sẽ được truyền qua tầng Convolution layer để phát hiện được các đặc trưng cụ thể trong từng bức ảnh. Trong tầng bao gồm 4 đối tượng: ma trận vào, filters, receptive field và cuối cùng là feature map. Đầu tiên tầng nhận vào một ma trận đầu vào và một bộ filter cần học. Bộ filter này sẽ quét qua từng vị trí trên bức ảnh và tiến hành tích chập giữa phần tương ứng trên bức ảnh (receptive) với bộ filter đầu vào. Feature map là vùng mà một neruron có thể nhìn thấy để đưa ra quyết định hay còn gọi là ma trận đầu ra.

Chức năng chính của tầng Convolution là phát hiện được đặc trưng cụ thể của bức ảnh bao gồm: đặc trưng cơ bản của góc, cạnh hay màu sắc của bức ảnh.



Hình 2.8 Filter 5x5

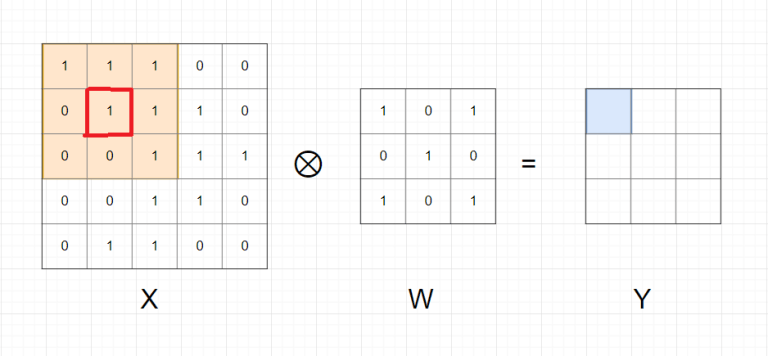
Filter là một trong những tham số không thể thiếu trong tầng thứ nhất. Kính thước của nó tỉ lệ thuận với tham số cần học tại mỗi tầng Convolution. Kích thước của filter được sử dụng phổ biến nhất là 3 x 3. Sự lựa chọn filter nhỏ để có thể rút trích được nhưng đặc trưng có độ chính xác cao và có thể phát hiện được những đặc trưng nhỏ trong từng bức ảnh thay vì sử dụng những filter lớn.

Nonlinear Layer (ReLU, (f(x) = max(0,x)) là hàm kích hoạt phổ biến nhất cho CNN thời điểm hiện tại. ReLU là hàm phi tuyến, được đặt ngay sau tầng convolution. Vì những dữ liệu bên trong thế giới mà chúng ta tìm hiểu là các giá trị không âm. Chính vì thế nên cần sử dụng đến ReLU bởi vì nó sẽ gán những giá trị âm bằng 0 và giữ nguyên giá trị của đầu vào khi lớn hơn 0.

Ví dụ với ảnh xám, là ảnh được biểu diễn dưới dạng ma trận X kích thước m\*n. Ta định nghĩa filter là một ma trận vuông kích thước k\*k trong đó k là số lẻ, k có thể bằng 1, 3, 5, 7, 9, …. Ví dụ filter kích thước 3\*3



Khi áp dụng phép tính convolution trên ma trận X với filter W ta được ma trận đầu ra Y với ký hiệu . Trong đó, phép tính convolution được ký hiệu . Để thực hiện tính toán với phép tính convolution, với mỗi phần tử trong ma trận X ta lấy ra một ma trận có kích thước bằng kích thước của filter W có phần tử  ​làm trung tâm, gọi là ma trận B. Sau đó ta tính tổng các phần tử của phép tính element-wise của ma trận B và ma trận W, rồi viết vào ma trận kết quả Y.



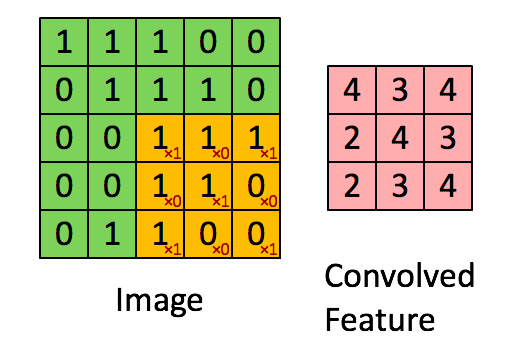
Hình 2.9 Phép tính tích chập trên ma trận

Với phần từ (ô đang được khoanh đỏ trong hình), ma trận B có cùng kích thước với W, có làm trung tâm có màu nền da cam như trong hình. Sau đó tính:

=

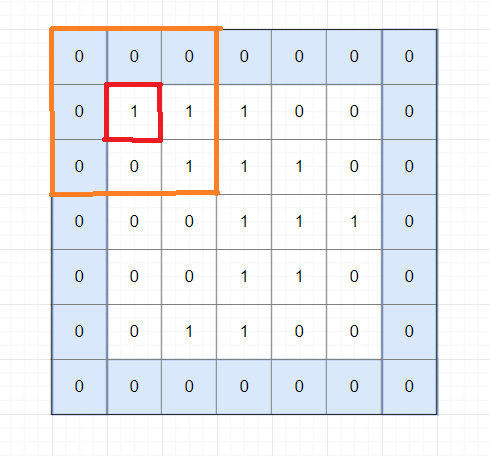
Các phần tử còn lại, chúng ta làm tương tự.

Với các phần tử ở viền ngoài như , ta không thể dùng để làm trung tâm cho ma trận B ở bên trong X. Nên ma trận Y có kích thước nhỏ hơn ma trận X. Kích thước của ma trận Y là (m-k+1) \* (n-k+1).



Hình 2.10 Kích thước của ma trận đầu ra của phép tích chập

Ta có thể quan sát thấy mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X. Tuy nhiên, ta muốn ma trận Y thu được có kích thước bằng ma trận X, ta cần tìm cách giải quyết cho các phần tử ở viền, việc đơn giản bây giờ là thêm giá trị 0 ở viền ngoài ma trận X để các phần tử ở viền có thể được thực hiện tích chập.



Hình 2.11 Ma trận được thêm các phần tử 0 làm padding

Như vậy, ta đã giải quyết được vấn đề tìm ma trận B cho phần tử , ma trận kết quả Y thu được sẽ bằng kích thước của ma trận đầu vào X. Phép tính này được gọi là phép tích chập với padding = 1. Từ đây, ta thấy với padding = k có nghĩa là ta thêm k vector 0 vào mỗi phía của ma trận.

Với phép tích chập như ở trên, ta thực hiện với từng phần tử trong ma trận đầu vào thì sẽ thu được ma trận đầu ra có cũng kích thước, việc thực hiện tuần tự như vậy ta gọi là bước nhảy (hay stride). Như vậy, ở trên, ta đang thực hiện convolution với stride = 1. Tuy nhiên với stride = 2, 3, …, k thì phép tính convolution chỉ được thực hiện trên các phần tử



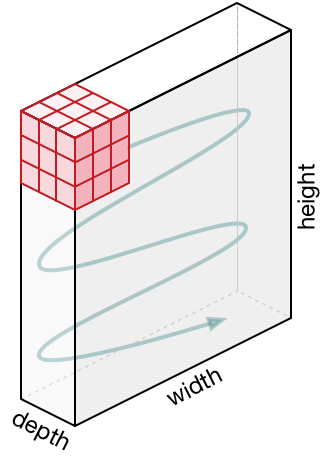
Hình 2.12 Các phần tử được thực hiện tích chập với bước nhảy k = 2

Ta có thể hiểu đơn giản việc thực hiên convolution từ phần tử đầu tiên là rồi trượt k bước theo hàng ngang, sau đó là hàng dọc cho đến hết ma trận đầu vào. Ta cũng nhận thấy ma trận đầu ra bây giờ đã giảm kích thước xuống còn 3\*3, số lượng phần tử đã được giảm đi đáng kể so với ma trận đầu vào với kích thước 5\*5. Như vậy, ta có thể thấy bước nhảy stride sẽ được sử dụng cho việc giảm kích thước của ma trận sau phép tích chập, với kích thước của ma trận đầu ra sẽ được tính theo công thức sau:

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

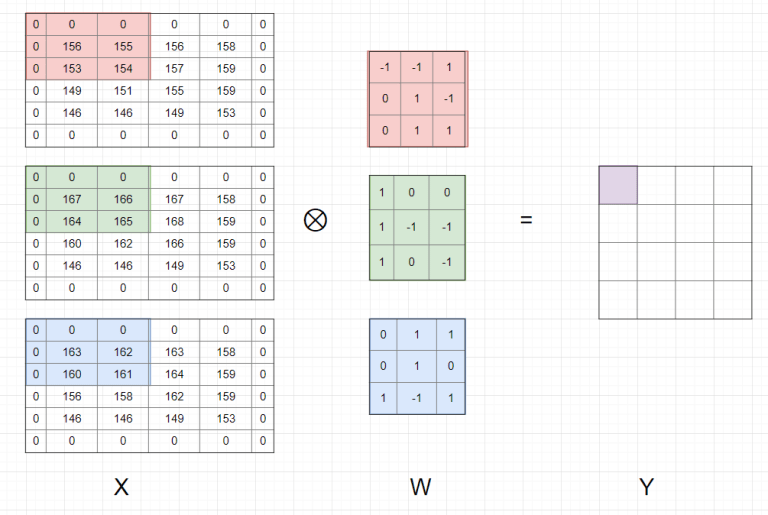
Trong đó, ma trận đầu vào có kích thước m\*n với filter có kích thước k\*k, hệ số stride = s, padding = p

Như đã trình bày ở trên, một ảnh màu được lưu trữ trong máy tính có 3 kênh màu là Red, Green, Blue nên khi biểu diễn dưới dạng ma trận, nó cần sử dụng ma trận 3 chiều. Vậy việc thực hiện tính toán phép tích chập trên ảnh màu sẽ có phần phức tạp hơn. Lúc này, phần filter cũng sẽ được định nghĩa là một ma trận 3 chiều có kích thước k\*k\*3.



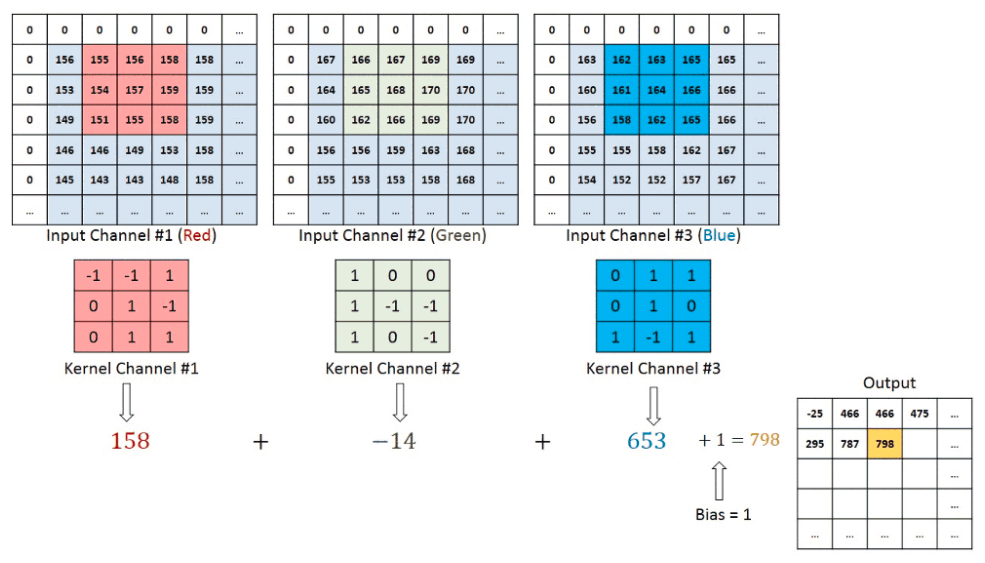
Hình 2.13 Phép tích chập trên ảnh màu

Để thực hiện phép tích chập trên ảnh màu, ta phải định nghĩa filter có cùng độ sâu(depth) với ma trận biểu diễn ảnh, sau đó việc tính toán cũng tương tự như trên ma trận 2 chiều, ta thực hiện di chuyển filter tuần tự từng phần tử trong ma trận biểu diễn ảnh



Hình 2.14 Phép tính tích chập trên ma trận 3 chiều

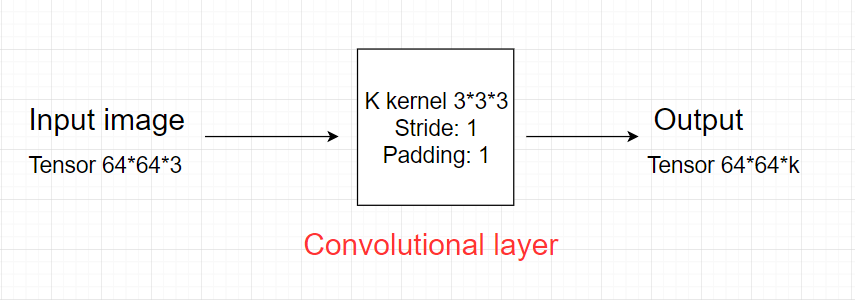
Với ma trận 2 chiều, ta cần có chỉ số hàng và cột tương ứng (i, j). Khi biểu diễn ma trận ảnh với 3 chiều ta cần thêm chỉ số về độ sâu k. Lúc này, mỗi phần tử trong ma trận biểu ảnh được biểu diễn dưới dạng .



Hình 2.15 Thực hiện tính tích chập trên ảnh màu

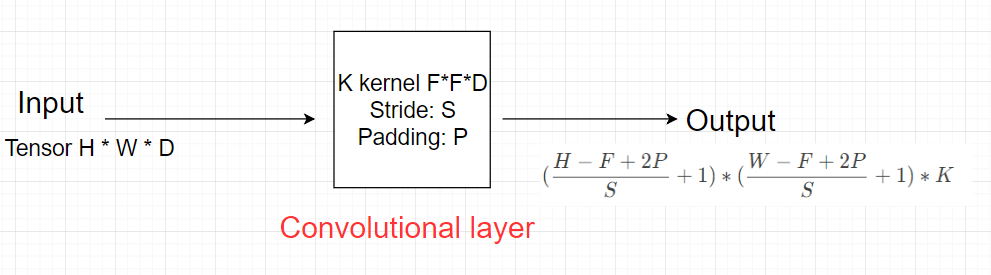
Phần tử được tính như sau:

Việc thực hiện với các phần tử khác cũng tương tự.



Hình 2.16 Đầu ra của lớp tích chập trên ảnh màu

Do với mỗi filter, mô hình sẽ học được một đặc trưng cụ thể của ảnh, để có thể rút trích được nhiều đặc trưng, ta cần sử dụng nhiều filter trong mỗi lớp convolution. Vì với mỗi một filter sẽ cho ra một ma trận nên với k filter thì đầu ra sẽ có k ma trận. Ta lại kết hợp các ma trận này lại thành một ma trận 3 chiều có chiều sâu k. Và đầu ra của lớp tích chập này sẽ thành đầu vào của lớp tích chập tiếp theo. Vậy kích thước của ma trận đầu ra có thể được tính như sau:



Hình 2.17 Mô hình tổng quát cho lớp tích chập

Mô hình trên nhận đầu vào là một ma trận 3 chiều có kích thước H\*W\*D. Filter có độ sâu depth luôn bằng với độ sâu của ma trận ảnh đầu vào, có kích thước F\*F\*D, các hệ số stride = S, padding = P. Áp dụng phép tích chập ta được ma trận đầu ra có kích thước:

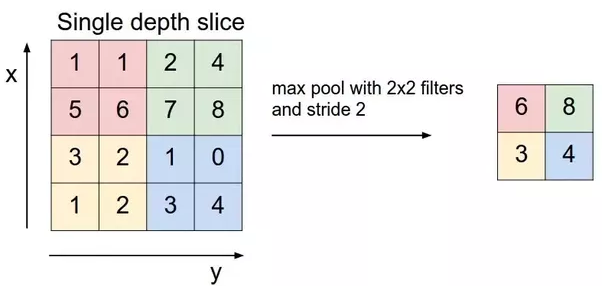
|  |  |
| --- | --- |
|  | (1.26) |

Ta thấy với mỗi filter có kích thước F\*F\*D và 1 hệ số bias thì sẽ có số parameter là F\*F\*D+1. Vậy một lớp tích chập sử dụng k filter khác nhau thì tổng số parameter trong lớp này sẽ được tính theo công thức

|  |  |
| --- | --- |
| K\*(F\*F\*D+1) |  |

#### **2.2.1.2 Pooling Layer**

Tầng này sẽ giảm số lượng tham số khi hình ảnh qua lớn. Ý tưởng đằng sau tầng Pooling là giảm kích thước của mỗi map nhưng vẫn giữ được thông tin quan trọng để có thể phân loại hình ảnh.

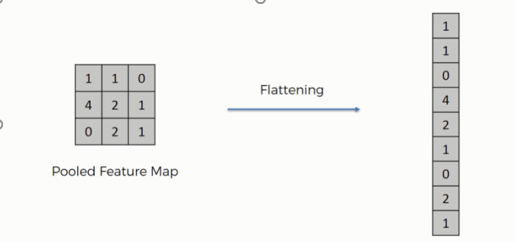


Hình 2.18 Max pooling

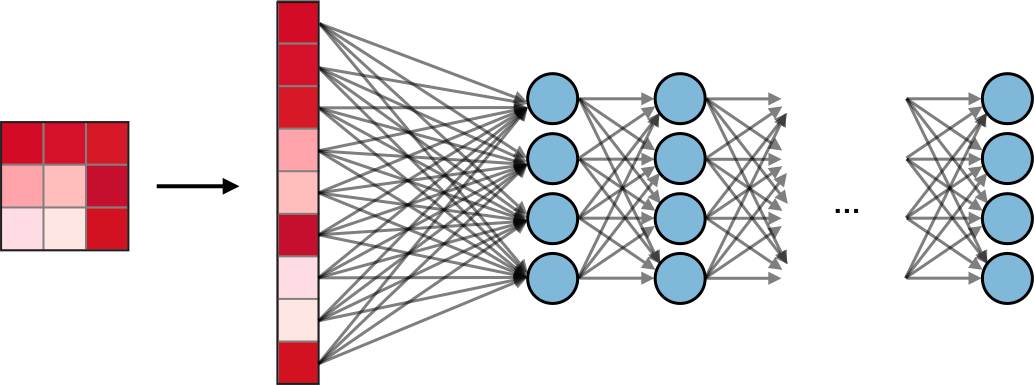
Ta thấy được với ma trận đầu vào có kích thước 4\*4 và ma trận filter pooling có kích thước 2\*2 với stride = 2 và padding = 0. Max pooling cho đầu ra là giá trị của phần tử lớn nhất ở cùng filter đi qua

#### **2.2.1.3 Fully Connected Layer**

Khi ảnh đã được truyền qua một vài lớp tích chập và lớp pooling, lúc này model đã có tương đối các đặc trưng của ảnh thì đầu ra của lớp cuối cùng có kích thước H\*W\*D sẽ được chuyển về 1 vector có kích thước (H\*W\*D) hay còn gọi là phẳng hoá ma trận.



Hình 2.19 Phẳng hoá ma trận

Để phẳng hoá ma trận đặc trưng ảnh có kích thước m\*n, ta lấy từng phần tử từ trái qua phải, từ trên xuống dưới xếp vào một cột theo chiều từ trên xuống, cột này được gọi là vector đặc trưng (m\*n) chiều.

Hình 2.20 Lớp Fully Connected

Ta có thể thấy, phần Fully Connected Layer cũng chính là một mạng neural network được gắn vào phần cuối của mạng CNN và có chức năng chuyển vector đặc trưng thành vector chứa xác suất của các đối tượng được dự đoán. Ví dụ với bài phân loại số viết tay, ta có 10 class tương đương với 10 số từ 0-9, tầng fully connected sẽ chuyển ma trận đặc trưng thành vector có 10 chiều thể hiện xác suất của 10 số tương ứng.

### **Mô hình YOLO**

#### **2.2.2.1 YOLO cho bài toán phát hiện đối tượng**

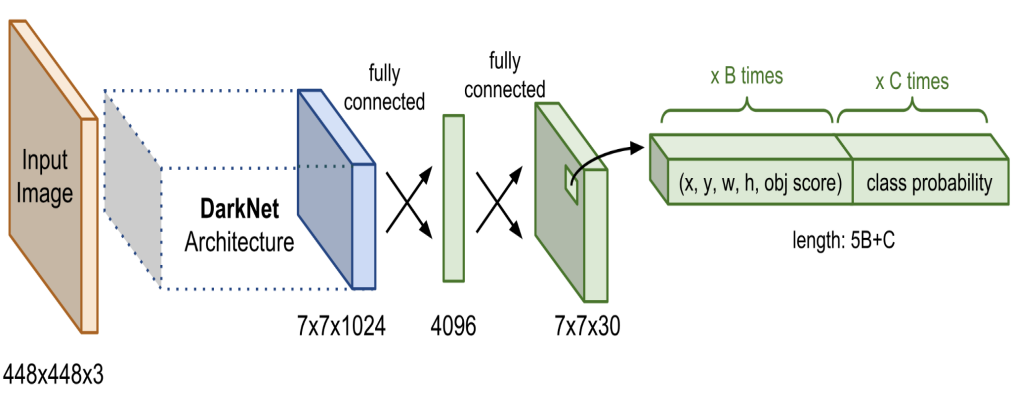
Trong các ứng dụng của mô hình CNN, phát hiện đối tượng là một đề tài nhận được rất nhiều sự quan tâm trong vài năm trở lại đây bởi tính ứng dụng cao, dữ liệu dễ dàng chuẩn bị. Việc tích hợp nhận dạng đối tượng trong ảnh có thể giúp con người đếm số lượng vật thể, chấm công tự động, phát hiện vũ khí nguy hiểm như dao, súng, …. Để phát hiện đối tượng trở nên phổ biến như vậy phải kể đến sự bùng nổ của internet, khiến cho nguồn dữ liệu hình ảnh giờ đây đã vô cùng dồi dào, đa dạng và luôn sẵn sàng trên google cho chúng ta tìm kiếm và sử dụng, nó đã phá bỏ đi những khó khăn trong quá trình huấn luyện mô hình deep learning đó là sự khan hiếm về dữ liệu. Bên cạnh đó, sự xuất hiện của các thuật toán mới và hiện đại như YOLO, SSD cùng với khả năng xử lý thông tin nhanh và độ chính xác cao của mình, các thuật toán này đã khiến cho phát hiện đối tượngcó thể được thực hiện trong thời gian thực (real-time) mà độ chính xác không giảm. Bên cạnh đó, các mô hình cũng được phát triển để có thể triển khai trên các thiết bị IoT, tạo nên các thiết bị thông minh.

Trong các thuật toán thuộc lớp mô hình phát hiện đối tượng, YOLO được đề xuất với tốc độ thực thi nhanh nhất, mặc dù nó không phải thuật toán tốt nhất về độ chính xác. Mô hình YOLO có thể đạt được tốc độ xử lý thông tin gần như real-time mặc dù độ chính xác không bằng nhưng cũng không quá khác biệt so với các model thuộc top đầu.

YOLO trong bài toán phát hiện đối tượng có nghĩa là “You only look once”. Tức là mô hình chỉ cần nhìn bức ảnh 1 lần là có thể phát hiện ra vật thể trong ảnh. Để thực hiện được, YOLO chỉ dùng một mạng CNN và áp dụng lên ảnh đầu vào một lần duy nhất để rút trích các thông tin của toàn bộ bức ảnh, sau đó dựa vào thông tin trên để dự đoán các bounding box chứa đối tượng.Kiến trúc của Yolo

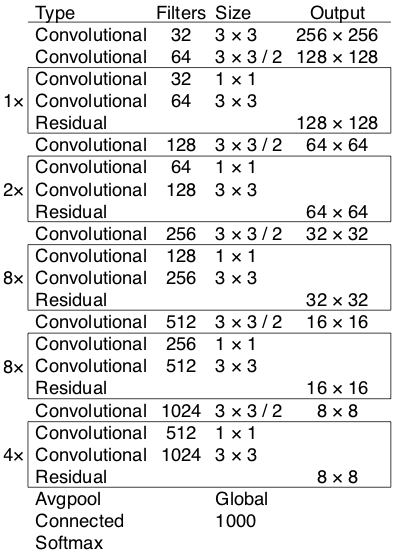
#### **2.2.2.2 Kiến trúc YOLO**

Kiến trúc của mạng YOLO bao gồm phần base network và Extra Layers. Phần Base network là các mạng CNN có nhiệm vụ rút trích đặc trưng, thông tin của bức ảnh sau đó đưa ra feature map để cho phần tiếp theo là Extra Layers sử dụng để phát hiện các vật thể. Base network của mô hình YOLO chủ yếu là các lớp tích chập và mạng fully connected.Base Network của YOLO sử dụng chủ yếu là các Convolutional Layer và các Fully Connected Layer.



Hình 2.21 Sơ đồ kiến trúc mạng yolo

Thành phần DarkNet Architecture là base network làm nhiệm vụ rút trích thông tin, đặc trưng của ảnh. Đầu ra của phần này là một feature map với kích thước 7x7x1024. Feature map này sẽ được làm đầu vào cho phần Extra Layers để dự đoán các nhãn và toạ độ bounding box của các vật thể trong ảnh. Ở phần DarkNet Architecture, tác giả YOLO đã sử dụng mạng darknet-53 để thực hiện feature extractor cho ảnh. Mạng này gồm có 53 tầng tích chập được kết nối liên tiếp, theo sau mỗi tầng là một batch normalization cùng với activation Leaky ReLU. Để giảm thiểu số lượng tham số cho mô hình, tác giả đã down sample bằng các filter với kích thước là 2 sau mỗi tầng tích chập.



Hình 2.22 Các layer trong mạng darknet 53

Mô hình YOLO đang cho phép các ảnh đầu vào có kích thước là 416x416 và 608x608. Các bức ảnh khi được đưa vào mô hình sẽ được scale về chung kích thước phù hợp với kích thước đầu vào của mô hình và sau đó được gom lại thành từng nhóm để bắt đầu huấn luyện. Các nhóm này được gọi là batch. Với mỗi kích thước ảnh đầu vào như trên, mô hình sẽ có một thiết kế các tầng, các lớp riêng phù hợp với kích thước của ảnh. Mỗi khi ảnh được đi qua các lớp tích chập thì kích thước sẽ giảm dần theo cấp số nhân là 2. Sau cùng ta thu được một feature map có kích thước nhỏ, tiếp theo trên từng ô của feature map này, mô hình sẽ tính xác suất xuất hiện của các vật thể. Kích thước của feature map trên sẽ phụ thuộc vào kích thước ảnh đầu vào. Với ảnh có kích thước 416x416, feature map có các kích thước 13x13, 26x26, 52x52. Còn với kích thước 608x608 sẽ tạo ra các feature map 19x19,38x38,72x72.

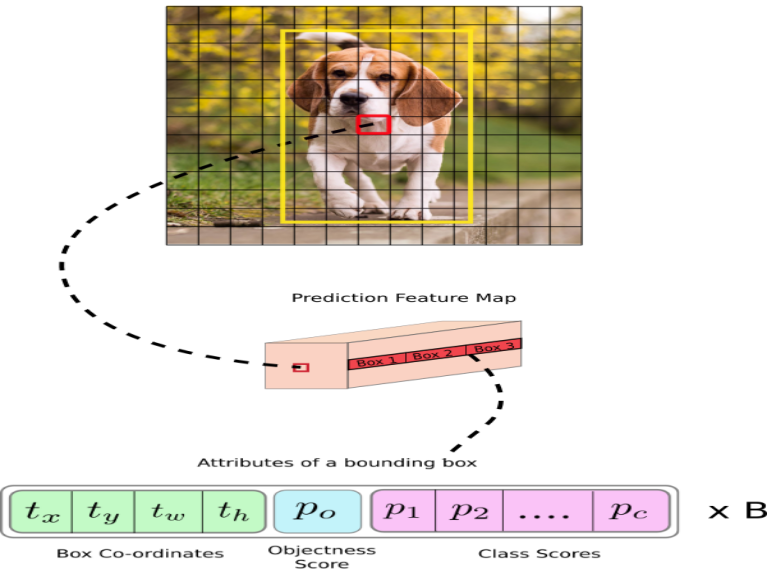
Output của YOLO:

Output của mô hình YOLO là một vectơ sẽ bao gồm các thành phần:

=[,⟨…,⟩]

Trong đó:

* là xác suất dự báo vật thể xuất hiện trong bounding box.
* () bounding box giúp xác định bounding box. Trong đó là tọa độ tâm và là kích thước rộng, dài của bounding box.
* (…,) là vectơ phân phối xác suất dự báo của các classes.



Hình 2.23 Outut của Yolo

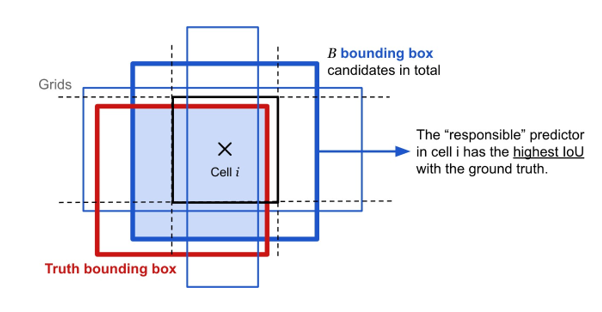
Hình trên mô tả một feature map có kích thước 13x13. Giả sử mô hình của ta lựa chọn 3 anchor boxes với kích thước khác nhau lần lượt là Box 1, Box 2, Box 3. Tại mỗi ô của feature map lúc này sẽ là một vector được ghép bởi 3 vector như đã trình bày ở trên. Như vậy, để ta có thể cấu hình tham số một cách chuẩn xác khi huấn luyện mô hình thì việc hiểu ma trận đầu ra là rất quan trọng. Ta có thể xác định số lượng tham số của ma trận đầu ra với công thức . Số 5 bao gồm các phần tử , ở trong công thức tính như trên. B là số lượng anchors box trên một ô mà mô hình lựa chọn. Ví dụ, ta huấn luyện mô hình nhận diện 5 class, áp dụng 3 anchors box cho một ô thì số lượng tham số trên ma trận đầu ra là: (5+5) x 3 = 30.

Với anchor box có kích thước cố định, nếu như đầu ra là ma trận 13x13 thì mô hình sẽ chỉ dự đoán được các đối tượng có kích thước lớn. Do vậy, YOLO dự đoán trên nhiều feature map. Như đã đề cập ở trên, mô hình cho ra 3 feature map với kích thước 13x13, 26x26, 52x52. Các feature map có kích thước lớn hơn sẽ giúp dự báo được các đối tượng có kích thước nhỏ.Trên mỗi một cell của các feature map ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 feature map x 3 anchor box).

Đồng thời trên một feature map hình vuông SxS, mô hình YOLOv3 sinh ra một số lượng anchor box là: SxSx3. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là:

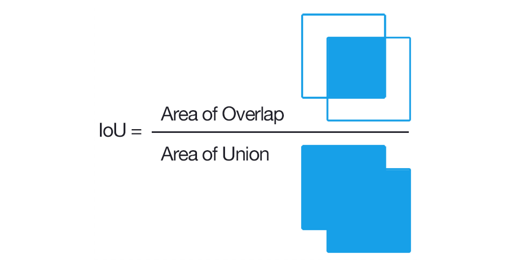
(13 x 13 + 26 x 26 + 52 x 52) x 3 = 10647 (anchor boxes)

Để tìm được bounding box cho đối tượng, mô hình YOLO sẽ cần các anchor box làm cơ sở để ước lượng. Như đã đề cập ở trên, kích thước của các anchor box sẽ được xác định trước và bao quanh vật thể một cách tương đối chính xác. Sau này, thuật toán regression bounding box sẽ được sử dụng để điều chỉnh lại các thông số kích thước của anchor box, từ đó tạo ra các bounding box dự đoán cho đối tượng.



Hình 2.24 Anchor box cho việc xác định vật thể

Với mỗi một đối tượng xuất hiện trong ảnh, nó được phân bố về một anchor box. Trong trường hợp đối tượng có nhiều hơn một anchor box bao quanh thì ta sẽ xác định anchor box có chỉ số IoU với ground truth bounding box cao nhất. Trong đó, ground truth bounding box là khung hình bao quanh vật thể được xác định trước từ bộ dữ liệu huấn luyện, có toạ độ (𝑐𝑥, 𝑐𝑦, 𝑤, ℎ).



Hình 2.25 Cách tính chỉ số IoU

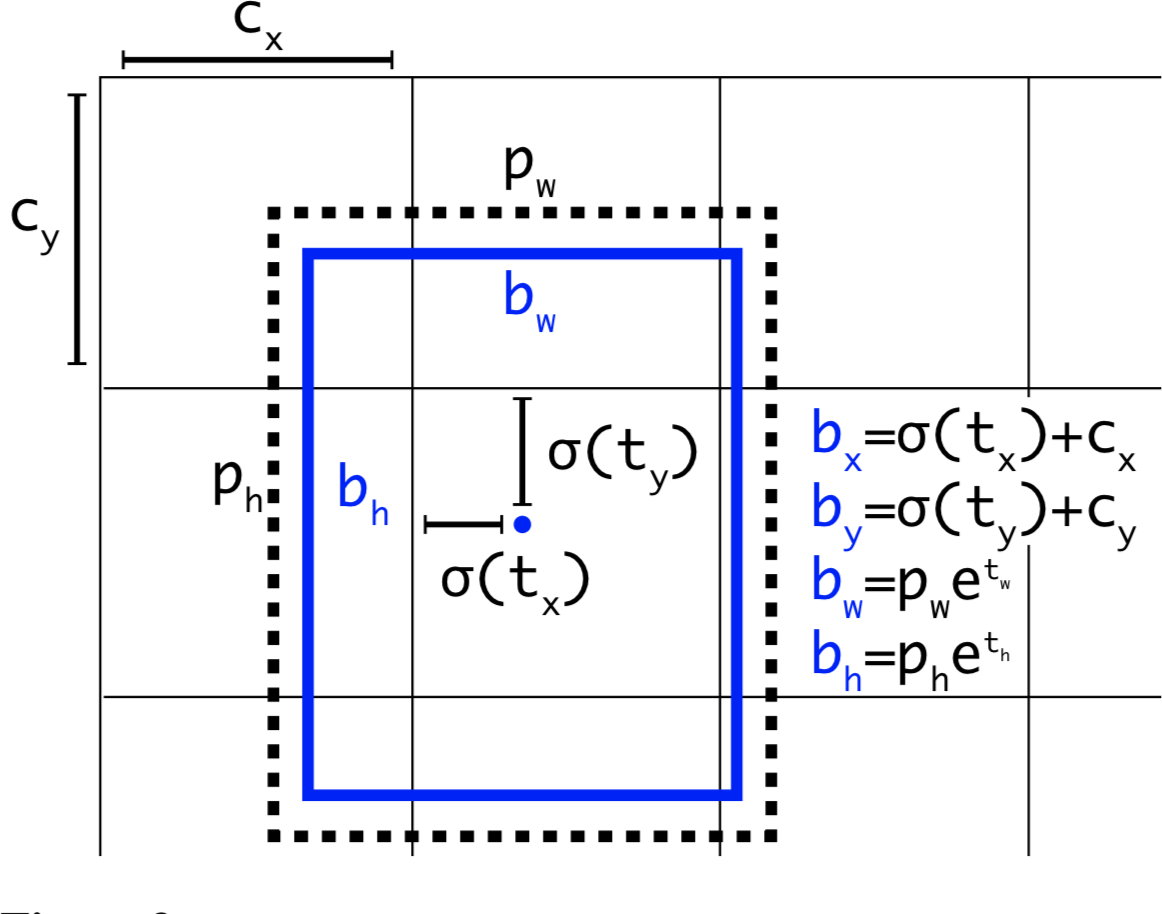
IoU (Tỷ lệ Intersection of Union) là tỷ lệ giao nhau giữa 2 bounding box, thường là bounding box dự đoán và ground truth bounding box. Tỷ lệ này được tính theo phần phần diện tích giao nhau giữa 2 bouding box và phần tổng diện tích giao nhau và không giao nhau giữa chúng.

Mỗi đối tượng trong ảnh sẽ được phân bố về một ô trên feature map mà ô đó chứa điểm trung tâm của đối tượng đó. Chẳng hạn như hình 2.23, hình ảnh của chú chó sẽ được phân vào trong ô màu đỏ vì điểm trung tâm của chú chó nằm trong ô này. Từ ô này, ta sẽ xác định các anchor boxes bao quanh hình chú chó. Như vậy, khi xác định một đối tượng, ta cần phải có 2 thành phần là cell, anchor box.

Để dự đoán bounding box cho một đối tượng, ta cần sử dụng phép biến đổi từ anchor box và cell. Với một anchor box có kích thước (𝑝𝑤, 𝑝ℎ) tại ô nằm trên feature map với góc trên cùng bên trái của ô đó có toạ độ (𝑐𝑥, 𝑐𝑦). Mô hình sẽ dự đoán các tham số (𝑡𝑥, 𝑡𝑦, 𝑡𝑤, 𝑡ℎ). Trong đó 2 tham số 𝑡𝑥, 𝑡𝑦 là độ lệch so với toạ độ (𝑐𝑥, 𝑐𝑦). Các tham số trên sẽ giúp xác định toạ độ tâm và kích thước của bounding box (𝑏𝑥, 𝑏𝑦, 𝑏𝑤, 𝑏ℎ) thông qua hàm sigmoid và hàm exponential như công thức sau:

|  |  |
| --- | --- |
| bx   (tx )  cx by   (ty )  cy  b  p etw  w w  b  p eth  h h | (2.1) |

Do các toạ độ đã được hiệu chỉnh theo width và height của bức ảnh nên giá trị của nó luôn nằm trong khoảng [0,1]. Điều này giúp ta giới hạn được toạ độ không vượt quá xa các ngưỡng này khi sử dụng hàm sigmoid.



Hình 2.26 Dự đoán các thông số bounding box

Cũng giống như mọi mô hình học máy khác, YOLO cũng cần phải có hàm mất mát để tối ưu các tham số. Hàm loss của YOLO được chia làm 2 phần: Localization loss được dùng để đo sự sai số của bounding box và Confidence loss đo sự sai số của phân phối xác suất cho các class.

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

Trong đó:

* : Có giá trị 0,1 nhằm xác định ô thứ i có chứa đối tượnghay không. Nếu có đối tượng, giá trị bằng 1 và ngược lại
* : Cho biết bounding box thứ j tại ô thứ i có phải là khung của đối tượng được dự đoán hay không?
* : Điểm số tin cậy của ô thứ i, được tính theo công thức:

P(contain object) \* IoU(predict box, ground truth bbox)

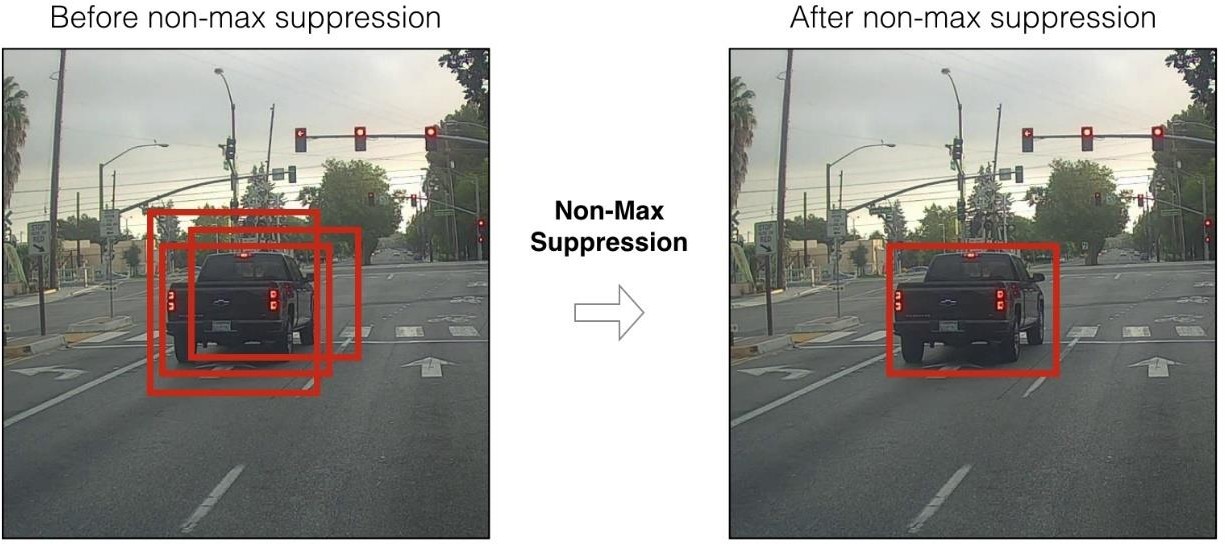
* : Điểm tự tin dự đoán
* C: Tập hợp các lớp cần dự đoán
* : Xác suất xuất hiện một đối tượng trong ô thứ i của lớp c
* : Xác suất dự đoán

Ngoài ra, hệ số được sử dụng để điều chỉnh mức độ trừng phạt của hàm mất mát đối với các tham số trong trường hợp dự đoán sai bounding box. Tương tự, nếu ta muốn giảm nhẹ hàm mất mát trong trường hợp ô đó không chứa đối tượng thì sử dụng hệ số

Ta có thể hiểu đơn giản mục đích của các hàm mất mát như sau:

* : hàm mất mát của bouding box dự báo so với ground truth bounding box
* : hàm mất mát của phân phối xác suất. Trong đó, tổng đầu tiên là mất mát cho việc có đối tượng nào trong ô đó hay không? Tổng thứ 2 là mất mát cho phân phối xác suất của đối tượng đó thuộc vào lớp nào, nếu có đối tượng ở trong ô.

DDo mỗi ô có thể dự đoán nhiều bounding box, nên ở những ô gần nhau, sẽ có nhiều bounding box nhận diện ra cùng một đối tượng. Điều này khiến cho một đối tượng bị đóng khung nhiều lần. Trong trường hợp đó, ta cần sử dụng Non- max suppression để giảm số lượng khung hình được sinh ra.



Hình 2.27 Non-max suppression

Việc sử dụng Non-max suppression đã giảm từ 3 bounding box bao quanh xe xuống còn 1 bounding box. Để chọn được bounding box này, ta cần lọc bỏ toàn bộ những bounding box có xác suất chứa đối tượng nhỏ hơn một ngưỡng nào đó, thường là 0.5, sau đó đối với các bbox giao nhau, non-max suppression sẽ lựa chọn một bounding box có xác suất chứa đối tượng lớn nhất bằng cách tính so sánh chỉ số IoU giữa các bounding box còn lại.

## **2.3 Thư viện OpenCV**

### **2.3.1 OpenCV**

OpenCV là thư viện mã nguồn mở hàng đầu cho Computer Vision, Machine Learning và xử lý ảnh. OpenCV được lập trình bằng C/C++ nên có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực (real – time).

OpenCV sở hữu cộng đồng người dùng khá đông hoạt động trên khắp thế giới và số lượng này sẽ còn tăng bởi xu hướng về AI cảu các công ty công nghệ.

### **Các module phổ biến trong OpenCV**

* Core: module được sử dụng bởi tất cả các module khác, định nghĩa các cấu trúc dữ liệu cơ bản, mảng đa chiều và nhiều chức năng cơ bản của thư viện
* Imgproc: module gồm các bộ lọc hình ảnh, phép biến đổi hình học, chuyển đổi không gian màu, biểu đồ
* Video: module có chức năng phân tích, nhận dạng chuyển động, tách nền và một vài thuật toán theo dõi đối tượng
* Calib3d: module sử dụng cho việc hiệu chỉnh, tái cấu trúc hình ảnh của camera single hoặc camera stereo và một vài thành phần giúp cho việc tái tạo ảnh 3D
* Objdetect: phát hiện các đối tượng ở trong ảnh như khuôn mặt, con người, phương tiện, ….
* Features2d: phát hiện các đặc tính nổi bật của bộ nhận diện, truy xuất thông số
* Highgui: Giao diện để dùng cho việc thực hiện giao tiếp UI đơn giản
* Videoio: module dùng cho việc đọc video mà mã hoá video
* GPU: gồm các thuật toán tăng tốc sử dụng GPU cho các module khác

### **Ứng dụng của OpenCV**

* Nhận dạng và xử lý hình ảnh qua camera
* Hình ảnh street view
* Phân tích hình ảnh y học
* Phục hồi ảnh/video
* Dựng phim - cấu trúc 3D từ chuyển động

## **2.4 Tổng quan về mã hóa công khai RSA**

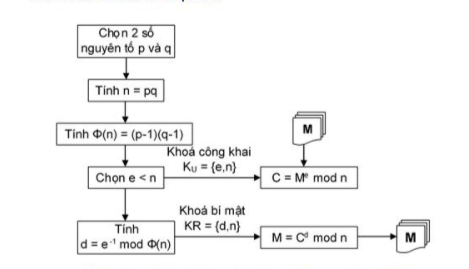
### **2.4.1 Giải thuật mã hóa công khai RSA**

Mã hóa khóa công khai (còn gọi là phương pháp mã hóa bất đối xứng, hay hệ thống hai khóa) trong đó khóa để mã hóa và khóa để giải mã là khác nhau. Các khóa này tạo thành một cặp chuyển đổi ngược nhau và không khóa nào có thể suy ra được từ khóa kia.

Quy trình mã hóa công khai:

* Tạo một cặp khóa gồm khóa công khai và khóa bí mật.
* Bên nhận sẽ gửi mã công khai và bên nhận nhận mã công khai dùng để mã hóa dữ liệu gửi cho bên nhận.
* Bên nhận nhận bản mã và giải mã bằng khóa bí mật.

Sơ đồ biểu diễn thuật toán



Hình 2.28 Sơ đồ biểu diễn thuật toán RSA

#### **2.4.1.1 Tạo khóa**

Giả sử bên nhận và bên gửi cần trao đổi thông tin bí mật. Với thuật toán RSA, bên gửi sẽ tạo ra một cặp khóa gồm khóa công khai và khóa bí mật theo các bước sau:

1. Chọn 2 số nguyên tố lớn và với, lựa chọn ngẫu nhiên và độc lập.
2. Tính:
3. Tính giá trị hàm số Ơle của :
4. Chọn một số tự nhiên e sao cho và là số nguyên tố cùng nhau với .
5. Tính: sao cho .

#### **2.4.1.2 Mã hóa**

Giả sử bên nhận muốn gửi thông điệp M cho bên nhận. Thông điệp M là một số nguyên thỏa mãn . Lúc này bên nhận có và biết cũng như do bên gửi gửi. Bên nhận sẽ tính là bản mã hóa của theo công thức:

|  |  |
| --- | --- |
|  | (1.1) |

Hàm trên có thể tính dễ dàng sử dụng phương pháp tính hàm mũ bằng thuật toán bình phương và nhân. Cuối cùng bên nhận gửi cho bên gửi.

#### **2.4.1.3 Giải mã**

Bên gửi nhận từ bên nhận và biết khóa bí mật . Bên gửi có thể tìm được từ theo công thức sau:

|  |  |
| --- | --- |
|  | (1.2) |

Tính đúng đắn của công thức giải mã (1.2) được giải thích như sau:

|  |  |
| --- | --- |
|  | (1.3) |

* + - Do và theo định lý Ferma nhỏ, nên:

|  |  |
| --- | --- |
| và | (1.4) |

* + - Do và là hai số nguyên tố cùng nhau nên áp dụng định lý số dư Trung Quốc ta được:

|  |  |
| --- | --- |
|  | (1.5) |

Tức là

|  |  |
| --- | --- |
|  | (1.6) |

#### **2.4.1.4 Ví dụ**

Để làm rõ hơn hoạt động của hệ mật RSA, sau đây ta lấy một ví dụ với những số cụ thể. Tất nhiên, ở đây ta sử dụng những số nhỏ để tiện tính toán còn trong thực tế phải dùng các số có giá trị đủ lớn.

Lấy:

* + - * + – số nguyên tố thứ nhất (giữ bí mật hoặc hủy sau khi tạo khóa)
        + – số nguyên tố thứ hai (giữ bí mật hoặc hủy sau khi tạo khóa)
        + – mô-đun (công bố công khai)
        + – số mũ công khai
        + – số mũ bí mật
        + – thông điệp cần mã hóa

Khi đó, ta có khóa công khai là cặp , còn khóa bí mật là cặp . Việc mã hóa được thực hiện như sau:

Và công thức để giải mã là:

* + 1. **Các vấn đề của RSA**

Vét cạn khóa: Thử tất cả các khóa d để giải mã. Nhưng khi n đủ lớn thì không thể dùng phương pháp này để giải mã.

Phân tích n thành các thừa số nguyên tố n=ab: Vì tốc độ xử lý của máy tính ngày càng nhanh nên phân tích n rất đơn giản.Để đảm bảo an toàn tuyệt đối ta nên sử dụng khóa có độ dài tối thiểu 2048 bit.

## **Mã hóa AES**

### **2.5.1 Khái niệm**

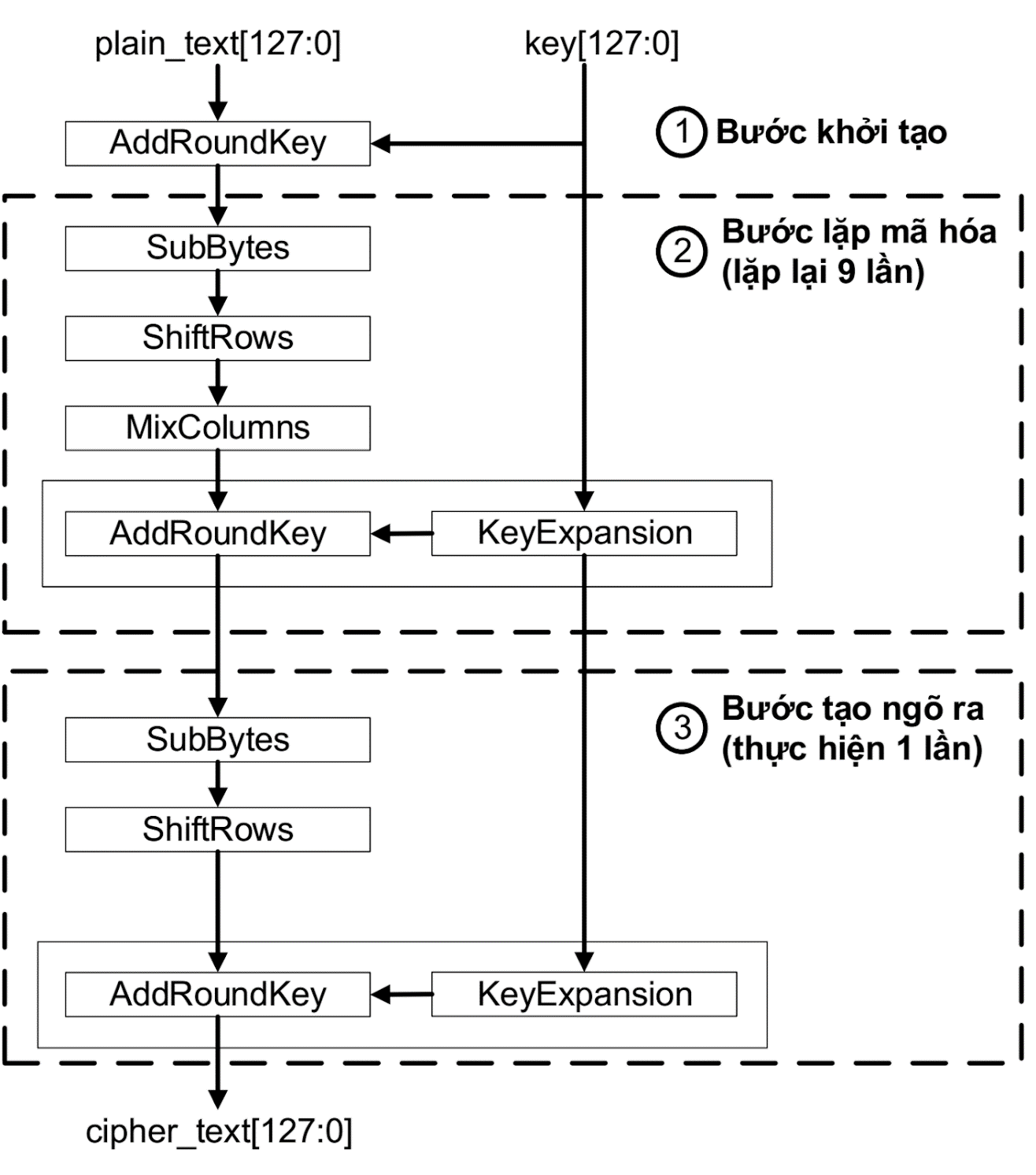
AES là viết tắt của Advanced Encryption Standard, chuẩn mã hóa dữ liệu rất phổ biến, dùng cho nhiều mục đích và được cả chính phủ Mỹ sử dụng để bảo vệ các dữ liệu tuyệt mật.AES là kiểu mã hóa đối xứng dạng khối, nghĩa là mỗi khối văn bản có một kích thước nhất định được mã hóa.

### **Hoạt động của AES**

AES là kiểu mã hóa khối, mỗi khối kích thước 128 bit. Khóa đối xứng với 3 kích thước là 128, 192 và 256 bit, trong đó 2 kích thước sau được chính phủ Mỹ dùng cho các tài liệu mật cấp cao, được gọi là “Top Secret”.

AES dùng thuật toán mã hóa khối mạng thay thế hoán đổi (SPN - Substitution Permutation Network). Dữ liệu được chuyển thành dạng an toàn trong vài bước, bắt đầu là khối plain text kích thước chuẩn, sau đó chèn vào hàng và sau đó là mã hóa. Mỗi lần đều có các bước thay thế, chuyển đổi, hòa trộn.

### **2.5.3 Thuật toán AES**



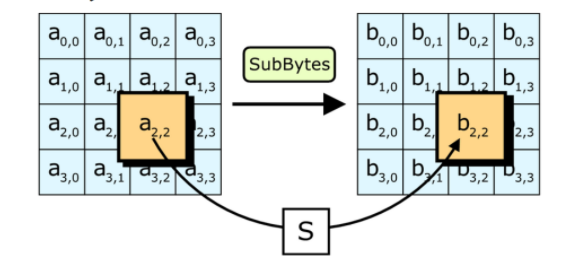
Hình 2.29 Mã hóa AES

Thuật toán AES được mô tả khái quát gồm 3 bước:

* 1 vòng khởi tạo AddRoundKey.
* Nr-1 vòng lặp gồm 4 phép biến đổi: SubBytes, ShiftRows, MixColumns, AddRoundKey.
* 1 vòng cuối gồm các phép biến đổi giống vòng lặp và không có MixColumns.

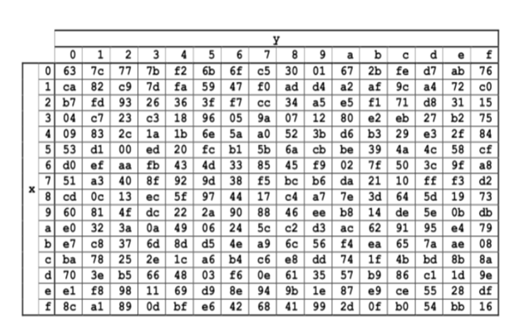
Phương pháp mã hóa AES đơn giản, có thể thực hiện hiệu quả trên các vi xử lý 8 bit (dùng trong smartcard) cũng như trên các vi xử lý 32 bit, chỉ dùng phép XOR và phép Shift bit.

#### **2.5.3.1 Bước SubBytes**



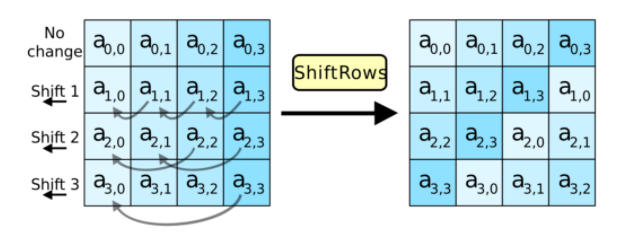
Hình 2.30 SubBytes

Phép biến đổi SubBytes: Là phép thay thế byte phi tuyến tính được thực hiện bằng cách tra cứu bảng thay thế (S-box thuận hoặc nghịc) với tham số đầu vào là các byte trong bảng trạng thái.



Hình 2.31 Bảng S-box

#### **2.5.3.2 Bước ShiftRows**

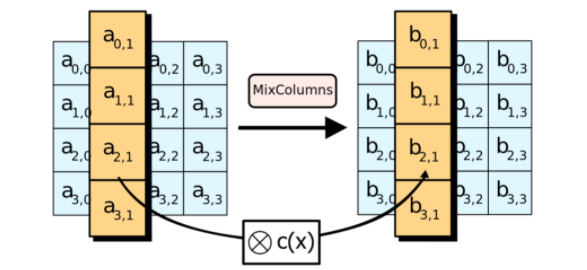


Hình 2.32 ShiftRows

Các bước trong ShiftRows thực hiện hoán vị các byte trong ma trận state theo cách thức:

* Dòng thứ nhất giữ nguyên
* Dòng thứ 2 dịch vòng trái 1 byte
* Dòng thứ 3 dịch vòng trái 2 byte
* Dòng thứ 4 dịch vòng trái 3 byte

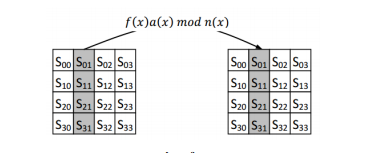
#### **2.5.3.3 Bước MixColumns**

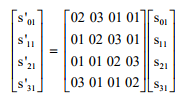


Hình 2.33 MixColumns

Phép biến đổi MixColumns thực hiện biến đổi đọc lập từng cột trong ma trận state bằng phép nhân đa thức. Mỗi cột của state đươc coi là biểu diễn của một đa thức f(x) trong GF() như vậy phép biến đổi MixColumns chính là phép nhân theo modulo với +1 với một đa thức cố định: 3++.







Hình 2.34 Phân tích đa thức

#### **2.5.3.4 Bước AddRoundKey**

Trong bước AddRoundKey, 128 bit của ma trận state được XOR với 128 bit của khóa con của từng vòng.

### **Bảo mật AES**

AES tuyệt đối an toàn khi nó được triển khai đúng cách. Tuy nhiên, các khóa mã hóa AES phải được bảo mật. Các hệ thống mã hóa bảo mật nhất cũng dễ dàng bị tấn công khi tin tặc có quyền truy cập vào khóa AES. Việc sử dụng mật khẩu mạnh, bảo mật mật khẩu, xác thực đa yếu tố (MFA), tường lửa và phần mềm chống vi-rút là rất quan trọng.

* 1. **Flask Framework**
     1. **Giới thiệu chung về Flask Framework**

Flask là một framework ứng dụng web được viết bằng Python. Nó được phát triển bởi Armin Ronacher, người dẫn đầu một nhóm những người đam mê Python quốc tế có tên là Poocco. Flask dựa trên bộ công cụ Werkzeg WSGI và template engine Jinja2. Cả hai đều là các dự án của Pocco. Flask được coi là một microframework. Nó được thiết kế để giữ cho cốt lõi của ứng dụng đơn giản và có thể mở rộng (scalable).

* + 1. **Tính năng của Flask Framework**

Flask cung cấp khả năng tùy biến khi phát triển ứng dụng web, nó cung cấp các công cụ, thư viện và cơ chế cho phép xây dựng một ứng dụng web nhưng nó sẽ không thực thi bất kỳ sự phụ thuộc nào.

* Khả năng tương thích công cụ dựa trên ứng dụng Google.
* Có nhiều tiện ích cho các chức năng khác nhau.
* Tính modular và thiết kế gọn nhẹ.
* Có độ linh hoạt cao.
* Cung cấp xử lý HTTP request.
* API có độc đáo và mạch lạc.
* Dễ dàng triển khai.
  + 1. **Ưu điểm của Flask Framework**

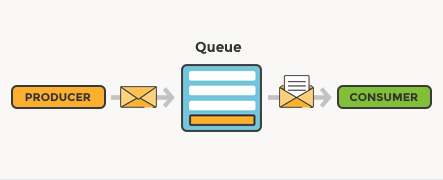
Flask được coi là một Web Framework nhỏ nhẹ. Và là một framework tối giản hoặc không phụ thuộc vào các thư viện ở bên ngoài.

* Tốc độ nhanh chóng.
* Hỗ trợ cho NoQuery.
* Có độ phúc tạp thấp.
* Dễ dàng kết nối với tiện ích mở rộng.
* Trình gỡ lỗi được tích hợp trong trình duyệt.
* Code ngắn và đơn giản so với các thư viện hỗ trợ Python khác.
* Điểm nổi bật khi sử dụng Flask để lập trình web là sẽ rất ít phụ thuộc bên thứ 3, do đó những vấn đề rủi ro về các lỗi bảo mật xảy ra rất ít.
  1. **Hàng đợi tin nhắn (Message Queue)**
     1. **Hàng đợi tin nhắn**

Hàng đợi tin nhắn cho phép các ứng dụng giao tiếp bằng cách gửi tin nhắn cho nhau. Hàng đợi tin nhắn cung cấp lưu trữ tin nhắn tạm thời khi chương trình đích bận hoặc không được kết nối. Hàng đợi tin nhắn cung cấp một giao thức truyền thông không đồng bộ, không yêu cầu phản hồi ngay lập tức để tiếp tục xử lý. Ý nghĩa của queue ở đây chính là một hàng đợi chứa message chờ để được xử lý tuần tự theo cơ chế vào trước thì ra trước (FIFO - First In First Out). Một message là các dữ liệu cần vận chuyển giữa người gửi và người nhận. Một hệ thống Message Queue thường có những thành phần sau:

* Message: Thông tin được gửi (có thể là text, binary hoặc JSON).
* Message Queue: Nơi chứa những message này, cho phép producer và consumer có thể trao đổi với nhau.
* Producer: Service tạo ra thông tin, đưa thông tin vào message queue.
* Consumer: Service nhận message từ message queue và xử lý.
* Một service có thể vừa làm producer, vừa làm consumer.

Một số Message Queue được dùng hiện nay: Kafka, Pulsar,…

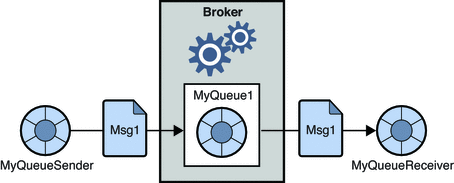


Hình 2.35 Hàng đợi tin nhắn (Message queue)

#### **2.7.1.1 Các loại Message Queue**

* Point-to-point

Message queue có thể là kiểu point-to-point, tức là khi đó ta chỉ có một hàng đợi và một consumer duy nhất để xử lý các tin nhắn trong hàng đợi:



Hình 2.36 Message queue Point-to-point

* Publisher-Subscriber

Message queue có thể sử dụng định dạng Publisher-Subscriber, trong đó publisher (nhà sản xuất) gửi tin nhắn đến hàng đợi (trong trường hợp này được gọi là Topic) và tất cả subscriber (người đăng ký) vào cùng một Topic đều sẽ nhận được tin nhắn trong Topic đó:



Hình 2.37 Message queue Publisher-Subscriber

#### **2.7.1.2 Ưu nhược điểm của Message queue:**

* Ưu điểm về Message queue:
* Dễ scaling hệ thống: Vào giờ cao điểm, nhiều truy vấn, ta có thể tăng số lượng consumer lên để xử lý được nhiều message hơn. Khi không cần ta có thể giảm lại.
* Phân tán hệ thống : Giúp phân tách hệ thống thành nhiều service nhỏ hơn, mỗi service chỉ xử lý 1 chức năng nhất định theo cấu trúc microservice.
* Đảm bảo duration/recovery: Do message đã được lưu trong queue, khi một service đang xử lý nhưng bị crash, ta không lo bị mất dữ liệu vì có thể lấy message từ trong queue ra và thử lại.
* Nhược điểm về Message queue
* Làm phức tạp hệ thống.
* Phải có message format: Từ hai phía producer và consumer cần phải thống nhất định dạng message để có thể gửi và nhận message.
* Khó xử lý đồng bộ: Không phải lúc nào message queue cũng là lựa chọn hàng đầu khi chúng ta xây dựng hệ thống. Sẽ có nhiều trường hợp hệ thống bắt buộc cần phải xử lý đồng bộ giữa các service, khi đó ta sẽ cần lựa chọn những cơ chế phù hợp hơn như RPI.
  + 1. **ZMQ**

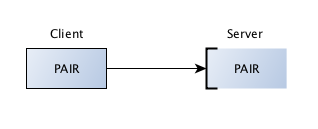
#### **2.7.2.1 Khái niệm**

ZeroMQ (còn được gọi là ØMQ , 0MQ hoặc ZMQ) là một thư viện nhắn tin không đồng bộ, nhằm mục đích sử dụng trong các ứng dụng phân tán hoặc đồng thời. Nó cung cấp một hàng đợi tin nhắn, nhưng không giống như phần mềm trung gian hướng thông điệp, hệ thống ØMQ có thể chạy mà không cần một trình môi giới thông điệp chuyên dụng. Thư viện được thiết kế để có một API kiểu socket quen thuộc.

#### **2.7.2.2 Các mô hình ZMQ**

* Đôi (Pair)

Cung cấp các socket gần giống với các socket thông thường: Giao tiếp là hai chiều, chỉ có thể có một đồng đằng được kết nối, máy chủ lắng nghe trên một cổng nhất định và một máy khách kết nối với nó.

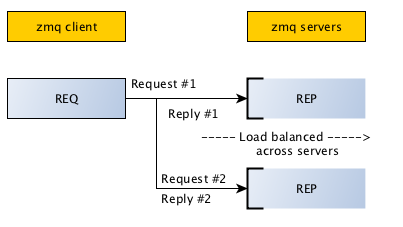


Hình 2.38 Mô hình Pair trong ZeroMQ

* Máy khách/ máy chủ (Client/ Server)

Mô hình cơ bản nhất là mô hình máy khách/ máy chủ, trong đó máy khách gửi một yêu cầu và máy chủ trả lời yêu cầu. Có một điểm so với Pair là ZMQ REQ socket có thể kết nối với nhiều máy chủ và các yêu cầu sẽ được xen kẽ hoặc phân phối đến cả hai máy chủ.

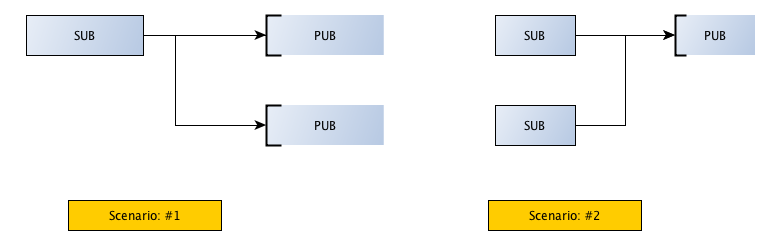
Socket zmq.PAIR có thể gửi bất kì số lượng tin nhắn nào giữa các đồng nghiệp được kết nối nhưng socket zmq.REQ sẽ chặn gửi cho đến khi nó đã nhận được phản hồi thành công.



Hình 2.39 Mô hình REQ/REP trong ZeroMQ

* Theo dõi công khai (Publish/ Subscribe)

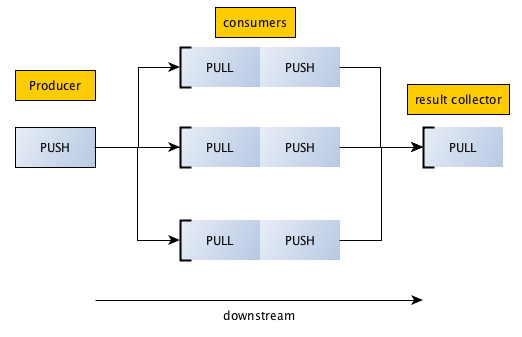
Publish/ Subscribe là một mô hình cổ điển khác trong đó người gửi tin nhắn, được gọi là publisher, không lập trình tin nhắn để gửi trực tiếp đến người nhận cụ thể, được gọi là subscriber. Tin nhắn được gửi mà không cần có subscriber kết nối đến publisher. Khi publisher gửi tin nhắn mà không có subscriber kết nối, nó sẽ xóa bỏ tất cả tin nhắn. Nếu subscriber nhận tin nhắn chậm, các tin nhắn sẽ được đưa vào hàng đợi tin nhắn trên publisher.



Hình 2.40 Mô hình Pub/ Sub trong ZeroMQ

* Đẩy/ Kéo (Push/ Pull)

Các socket đẩy và kéo cho phép phân phối tin nhắn cho nhiều thiết bị nhận, các tin nhắn được sắp xếp trong một đường ống. Một socket Push sẽ phân phối đồng đều các tin nhắn đã gửi đến các máy khách Pull của nó. Điều này tương đương với mô hình nhà Publish/ Subscribe nhưng các thiết bị Pull sẽ không gửi ngược tin nhắn lại cho thiết bị Push được mà sẽ phải gửi vào một đường ống khác. Dữ liệu luôn chảy xuống đường ống và mỗi giai đoạn của đường ống được kết nối với ít nhất một nút. Khi một giai đoạn đường ống được kết nối với nhiều nút, dữ liệu được cân bằng tải giữa tất cả các nút được kết nối.



Hình 2.41 Mô hình Push/Pull trong ZeroMQ

* + 1. **Image ZMQ**

#### **2.7.3.1 Khái niệm**

Image ZMQ là một cơ chế vận chuyển hình ảnh dễ sử dụng cho mạng xử lý hình ảnh phân tán. Ví dụ, một mạng hàng chục Raspberry Pi có camera có thể gửi hình ảnh đến một máy tính trung tâm mạnh hơn. Raspberry Pi thực hiện chụp ảnh và xử lý ảnh đơn giản như lật, làm mờ và phát hiện chuyển động. Sau đó, hình ảnh được chuyển qua Image ZMQ đến máy tính trung tâm để xử lý hình ảnh phức tạp hơn như gắn thẻ hình ảnh, trích xuất văn bản, nhận dạng tính năng, v.v.

#### **2.7.3.2 Đặc trưng Image ZMQ**

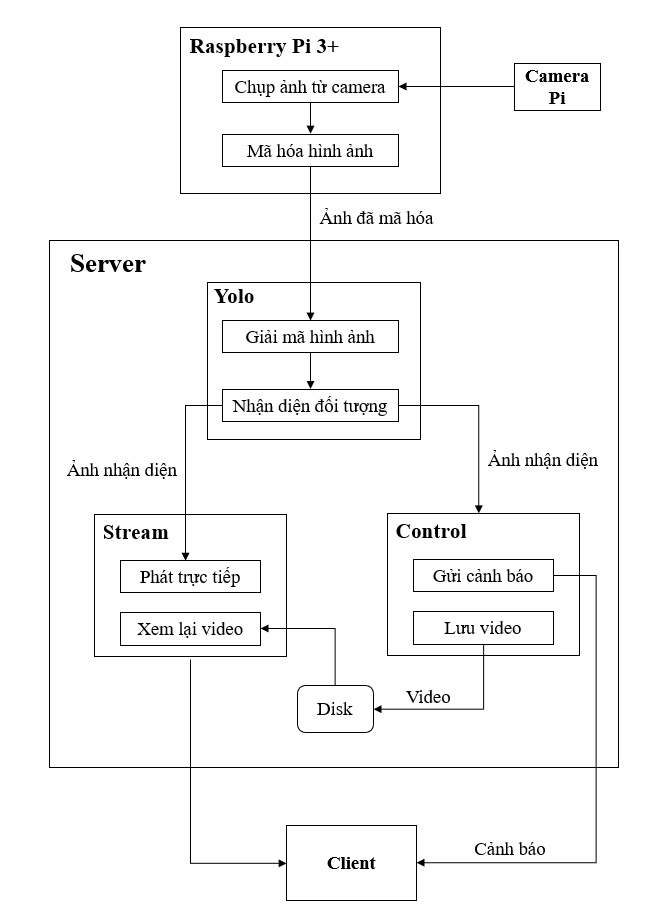
Gửi hình ảnh bằng ZMQ.

* Sử dụng thư viện nhắn tin ZMQ thông qua các liên kết PyZMQ.
* Cho phép lựa chọn 2 mẫu nhắn tin ZMQ khác nhau (REQ/ REP hoặc PUB/ SUB).
* Cho phép trung tâm hình ảnh nhận và xử lý hình ảnh từ nhiều người gửi hình ảnh đồng thời.

# **CHƯƠNG 3. XÂY DỰNG CAMERA AN NINH THÔNG MINH SỬ DỤNG RASPBERRY**

## **3.1 Mô tả hoạt động của hệ thống**

### **3.1.1 Sơ đồ khối của hệ thống**



Hình 3.1 Sơ đồ hoạt động hệ thống

* **Raspberry Pi 3+**: Có chức năng chụp ảnh từ Camera Pi sau đó mã hóa và gửi lên server để xử lí.
* **Yolo**: Giải mã hình ảnh nhận được sau đó sử dụng Yolo V4 tiny để nhận diện và phát hiện người trong ảnh. Sau đó gửi ảnh đã xử lí đến Stream và Control.
* **Control**: Nhận ảnh đã xử lí từ Yolo để lưu video và gửi cảnh báo khi phát hiện người.
* **Stream**: Có 2 chức năng: Thứ nhất là nhận ảnh từ YOLO và phát trực tiếp đến Client. Thứ hai là đọc video từ ổ đĩa để phát những video đã lưu.
* **Client**: Truy cập website để xem video trực tiếp hoặc video đã lưu.

### **3.1.2 Hoạt động của hệ thống**

* Khi khởi động server và raspberry sẽ trao đổi khóa mã hóa AES để mã hóa hình ảnh truyền giữa server và raspberry

1. Raspberry sẽ tạo cặp khóa RSA và gửi cho server khóa công khai.

2. Server nhận khóa công khai sẽ tạo, mã hóa khóa AES bằng khóa công khai và gửi lại bản mã cho raspberry.

3. Sau khi nhận bản mã của khóa AES raspberry sẽ dùng khóa bí mật để giải mã.

4. Khi có khóa AES, raspberry sẽ mã hóa hình ảnh và gửi cho server qua ZeroMQ.

* Nhận ảnh và phát hiện người

1. Worker Yolo trên server sẽ nhận ảnh từ ZeroMQ sau đó giải mã và dùng mô hình Yolo để phát hiện người.

2. Sau khi phát hiện người, Worker Yolo sẽ gửi hình ảnh đã nhận diện, tin nhắn đến worker Stream để phát trực tiếp và worker Control để gửi cảnh báo và lưu video khi phát hiện người.

* Lưu video và gửi cảnh báo

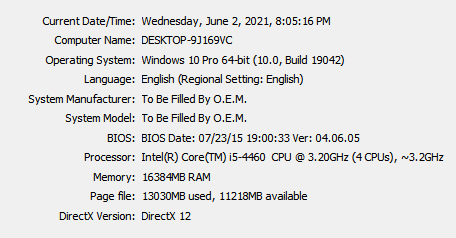
Khi nhận hình ảnh và tin nhắn từ worker Yolo, worker Control sẽ kiểm tra tin nhắn. Nếu là tin nhắn phát hiện người, worker sẽ gửi cảnh báo và lưu video phát hiện người.

### **3.1.3 Chức năng của hệ thống**

* Chức năng giám sát: Nhận ảnh từ Raspberry Pi và phát video trực tiếp lên hệ thống server.
* Chức năng cảnh báo: Khi phát hiện người trong hình ảnh camera ghi lại, hệ thống sẽ gửi mail cảnh báo về cho người dùng biết.
* Chức năng lưu trữ video: Khi phát hiện người trong camera ghi lại, hệ thống sẽ lưu video và người dùng có thể xem lại video mà hệ thống đã lưu trên server.
* Chức năng mã hóa: Khi khởi động hệ thống, Server và raspberry pi sẽ trao đổi khóa để mã hóa hình ảnh.Hệ thống sẽ sử dụng mã hóa này để mã hóa hình ảnh được gửi từ camera đến server.

## **3.2 Thiết kế phần cứng**

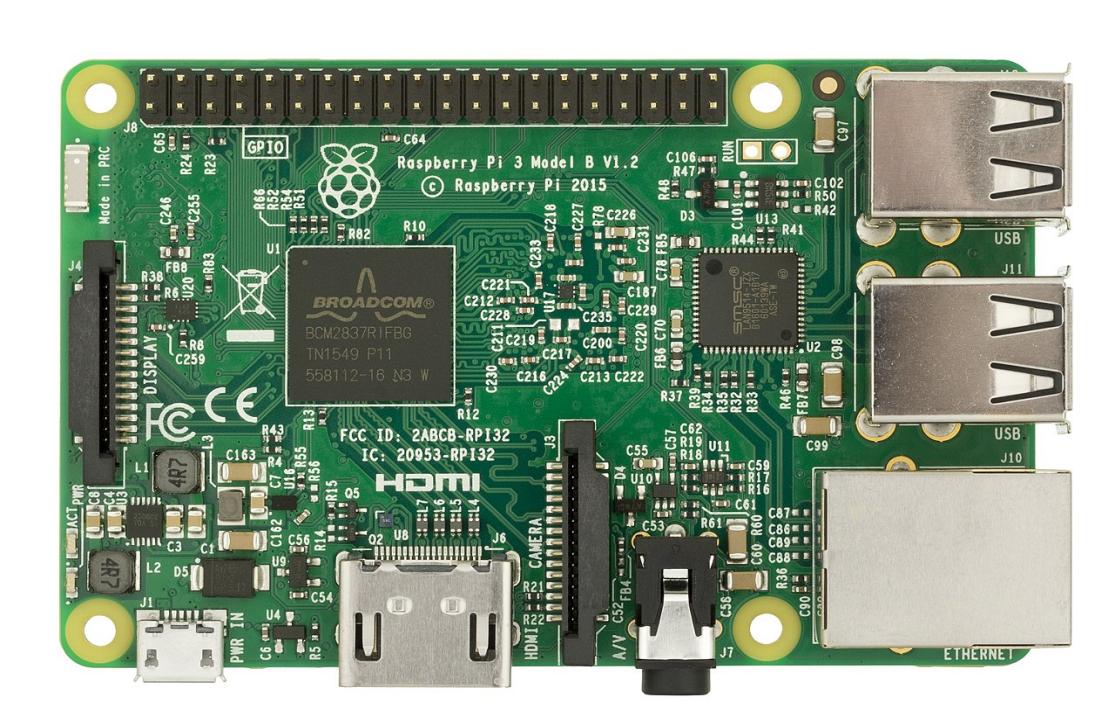
### **3.2.1 Cấu hình Server**



Hình 3.2 Cấu hình của hệ thống máy chủ

### **Hệ thống camera an ninh**

#### **3.2.2.1 Raspberry Pi 3 Model B+**



Hình 3.3 Raspberry Pi 3 Model B+

Raspberry Pi 3 Model B+ thế hệ thứ 3 của Raspberry Pi, nổi bật với chip 4 nhân 64-bit có tốc độ 1.4GHz. Broadcom là nhà sản xuất chip (SoC) cho Raspberry Pi 3, với tên mã BCM2837.

Cấu tạo phần cứng :

* Vi xử lý: Broadcom BCM2837B0, quad-core A53 (ARMv8) 64-bit SoC 1.4GHz.
* RAM: 1GB DDR2.
* Kết nối: 2.4GHz and 5GHz IEEE 802.11 b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, Gigabit Ethernet over USB 2.0.
* Cổng USB: 4 x 2.0.
* Mở rộng: 40-pin GPIO.
* Video và âm thanh: 1 cổng full-sized HDMI, Cổng MIPI DSI Display, cổng MIPI CSI Camera, cổng stereo output và composite video 4 chân.
* Lưu trữ: MicroSD.
* Nguồn điện sử dụng: 5V/2.5A DC cổng microUSB, 5V DC trên chân GPIO, Power over Ethernet (PoE).

#### **3.2.2.2 Camera Pi**



Hình 3.4 Camera pi

Raspberry Pi Camera Module V2 có một cảm biến 8-megapixel của Sony IMX219

Camera Module có thể được sử dụng để quay video độ nét cao, cũng như chụp hình ảnh tĩnh. Raspberry Pi Camera Module V2 là một bước nhảy vọt về chất lượng hình ảnh, màu sắc trung thực và hiệu suất ánh sáng thấp. Đặc biệt nó hỗ trợ video lên tới 1080P30, 720P60 và video mode VGA90, cũng như chế độ chụp hình. Và nó sử dụng đoạn cáp 15cm qua cổng CSI trên Raspberry Pi.

Thông số kỹ thuật:

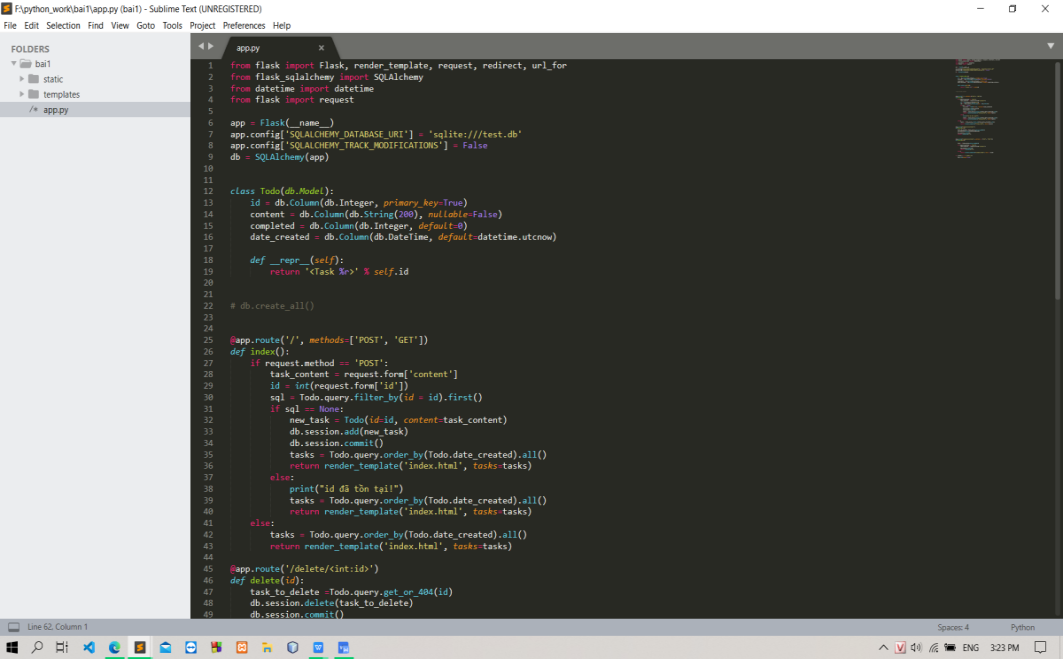
* Ống kính tiêu cự cố định.
* Cảm biến độ phân giải 8 megapixel cho khả năng chụp ảnh kích thước 3280 x 2464.
* Hỗ trợ video 1080p30, 720p60 và 640x480p90.
* Kích thước 25mm x 23mm x 9mm.
* Trọng lượng chỉ hơn 3g.
* Kết nối với Raspberry Pi thông qua cáp ribbon đi kèm dài 15 cm.
* Camera Module được hỗ trợ với phiên bản mới nhất của Raspbian.

## **3.3 Lập trình và tích hợp phần mềm**

### **Môi trường lập trình**

Sublime text là một phần mềm lập trình cho các nhà lập trình viên hiện nay sử dụng với đầy đủ các tính năng chỉnh sửa các tệp cục bộ hoặc các đoạn code trong lập trình phần mềm hiện nay.

Sublime text nhẹ, hỗ trợ nhiều ngôn ngữ lập trình khác nhau, có thể chỉnh sửa đồng thời và nhiều tiện ích khác.



Hình 3.5 Giao diện của sublime text

Ngôn ngữ lập trình: Python, HTML, CSS, Javascript

### **3.3.2 Cài đặt phần mềm cho Raspberry Pi**

Hệ điều hành Raspbian là một hệ điều hành dựa trên Debian được tối ưu hóa cho phần cứng Raspberry Pi. Hệ điều hành là tập hợp các chương trình và tiện ích cơ bản giúp Raspberry Pi chạy. Và được tối ưu hóa cho máy tính raspberry với CPU và RAM. Đồng thời Raspbian cung cấp một giao diện đồ họa như windows, có tích hợp sẵn rất nhiều ứng dụng văn phòng như LireOffice (Tương tự Microsoft office), trình nghe nhạc VLC Player, trình email client, trình duyệt web Chrome. Hơn thế nữa nó cong cung cấp cho người dùng những tiện ích khác như là camera, có thể chơi trò chơi, lập trình trên máy tính Raspberry Pi mà đã cài sẵn hệ điều hành Raspberry.

#### **3.3.2.1 Chuẩn bị các thiết bị dụng cụ cần thiết để thiết lập**

* + - * Raspberry Pi 3.
      * Nguồn 5V tối thiểu 700mA.
    - Thẻ nhớ SD: 32 GB.
    - Bàn phím, chuột sử dụng cổng USB.
    - Màn hình kết nối với cổng HDMI.
    - Dây mạng để Raspberry Pi có thể kết nối mạng.
    - Đầu đọc thẻ nhớ.

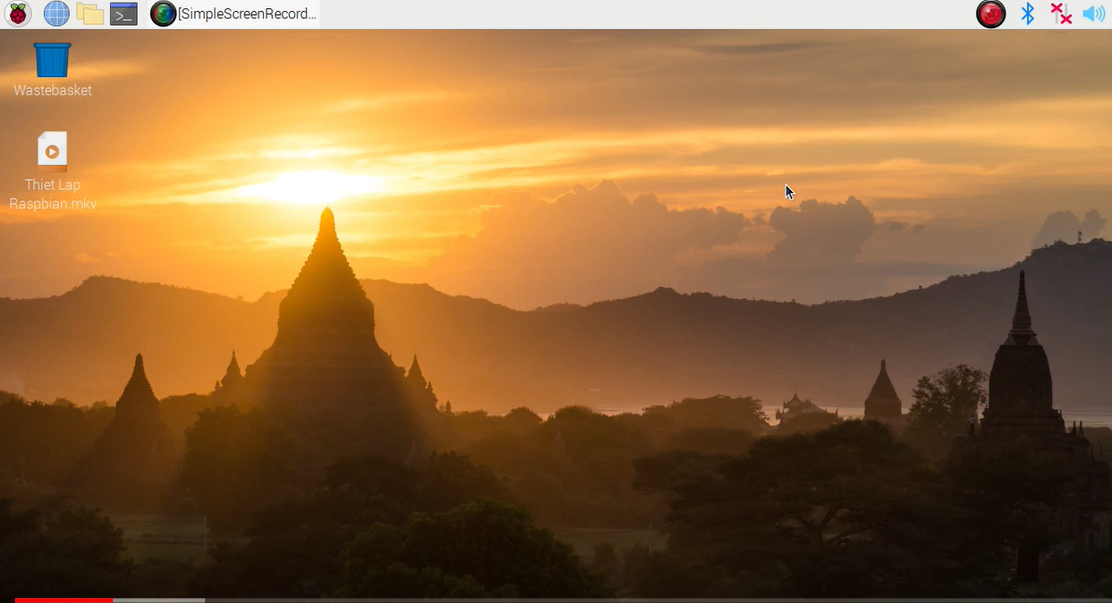
#### **3.3.2.2 Cài đặt Raspbian cho Raspberry**

Bước 1: Download hệ điều hành raspbian vào lap và ghi vào thẻ SD đã chuẩn bị.

Bước 2: Cắm thẻ nhớ đã được ghi hệ điều hành vào Pi.

Bước 3: Trước hết kết nối chuột, bàn phím và màn hình. Sau đó cắm nguồn cho Raspberry Pi.

Bước 4: Setup cơ bản ban đầu cho Raspberry Pi.



Hình 3.6 Giao diện của Raspberry khi cài đặt hệ điều hành Raspbian

Như vậy đã cài đặt được hệ điều hành Raspbian cho Raspberry.

### **3.3.3 Lập trình phần mềm**

#### **3.3.3.1 Trao đổi khóa mã hóa AES bằng mã hóa công khai RSA**

* Tạo và gửi khóa công khai RSA trên raspberry

addr = 'http://169.254.5.211:5000'

url = addr + '/api/rsa'

key = RSA.generate(1024)

pubKey = key.publickey().exportKey("PEM")

response = requests.post(url, data=pubKey)

data = response.json().get('key')

rsa\_private\_key = PKCS1\_OAEP.new(key)

* + Sử dụng phương thức RSA.generate() trong Crypto.PublicKey để tạo khóa công khai và khóa bí mật RSA.
  + Sử dụng phương thức publickey() để lấy mã khóa công khai.
  + Gửi khóa công khai bằng phương thức post(). Trong đó url là địa chỉ gửi và data là dữ liệu gửi.
* Nhận khóa công khai RSA và gửi bản mã của khóa mã hóa AES trên server

@app.route('/api/rsa', methods=['POST'])

def rsa():

    global key

    data = request.data

    pubKey = RSA.importKey(data)

    rsa\_public\_key = PKCS1\_OAEP.new(pubKey)

    aesKey = rsa\_public\_key.encrypt(key)

    response = {'key': "{}".format(aesKey)}

    response\_pickled = jsonpickle.encode(response)

    return Response(response=response\_pickled, status=200, mimetype="application/json")

* + Nhận khóa công khai RSA bằng cách nhận data từ đối tượng request.
  + Mã khóa khóa AES bằng phương thức rsa\_public\_key.encrypt().
  + Gửi lại bản mã của khóa AES bằng Response với khóa là “key” và giá trị là bản mã khóa AES.
* Nhận bản mã của khóa mã hóa AES và giải mã bằng khóa bí mật RSA

data = response.json().get('key')

rsa\_private\_key = PKCS1\_OAEP.new(key)

key = rsa\_private\_key.decrypt(ast.literal\_eval(str(data)))

sha = hashlib.sha1(key).hexdigest()

keyAES = sha[0:16].encode("utf-8")

* + Nhận bản mã khóa bằng phương thức json().get() của đối tượng response.
  + Sử dụng hàm băm SHA để băm khóa và lấy 16 byte đầu làm khóa mã hóa AES

#### **3.3.3.2 Mã hóa dữ liệu bằng AES**

* Mã hóa và gửi hình ảnh đã mã hóa trên raspberry

for frame in camera.capture\_continuous(rawCapture, format="bgr", use\_video\_port=True):

    image = frame.array

    \_,image = cv2.imencode('.jpg',image)

    imgBytes = image.tobytes()

    imgEncrypto = cipher.encrypt(pad(imgBytes, AES.block\_size))

    socket.send(imgEncrypto)

    rawCapture.truncate(0)

* + Tạo đối tượng cap dọc dữ liệu từ camera pi.
  + Chụp ảnh từ camera pi bằng phương thức read() của đối tượng cap.
  + Định dạng ảnh sang “JPG” bằng phương thức cv2.imencode().
  + Chuyển ảnh sang bytes bằng phương thức img.encode.tobytes().
  + Tạo đối tượng mã hóa AES bằng phương thức AES.new().
  + Mã hóa ảnh bằng phương thức cipher.encrypt().
  + Gửi ảnh lên queue bằng phương thức socket.send().
* Nhận ảnh và giải mã trên server

key = b'keygenerateaes'

sha = hashlib.sha1(key).hexdigest()

keyAESI = sha[0:16].encode("utf-8")

iv = b'0123456789abcdef'

cipherDecrypto = AES.new(keyAESI, AES.MODE\_CBC, iv)

imgBytes = socket.recv()

encrypto = unpad(cipherDecrypto.decrypt(imgBytes), AES.block\_size)

nparr = np.frombuffer(encrypto, np.uint8)

imgYolo = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

* + Nhận ảnh từ queue bằng socket.recv().
  + Tạo đối tượng giải mã bằng phương thức AES.new()
  + Giải mã ảnh bằng phương thức decrypt().
  + Chuyển byte sang ảnh bằng phương thức cv2.imdecode().

#### **3.3.3.3 Bảo mật lưu trữ video**

* Mã hóa video

def encrypt(name):

    videoDir = "static/video/"+name

    videoBinary = open(videoDir, "rb").read()

    iv = Random.new().read(AES.block\_size)

    cipher = AES.new(keyAES, AES.MODE\_CBC,iv)

    videoEncrypt = cipher.encrypt(pad(videoBinary, AES.block\_size))

    data = iv + videoEncrypt

    os.remove(videoDir)

    with open(videoDir,'wb') as file:

        file.write(data)

    video = Video.query.filter\_by(ten= name).first()

    video.mahoa = '1'

    db.session.commit()

* + Đọc video bằng phương thức open(videoDir, ”rb”).read().
  + Tạo và mã hóa video bằng phương thức AES.new() và encrypt().
  + Xóa video chưa mã hóa bằng phương thức os.remove().
  + Tạo video mã hóa bằng phương thức file.write().
  + Truy vấn video trên database bằng phương thức

Video.query.filter\_by(ten= name).first().

* + Sửa cột mahoa = ‘1’ và commit database bằng phương thức db.session.commit().
* Giải mã video

@app.route('/api/getVideo/<string:name>',methods=['GET','POST'])

def getVideo(name):

    global cipherDecrypto

    videoDir = "static/video/{}".format(name)

    try:

        with open(videoDir,'rb') as f:

            data = f.read()

            iv = data[:AES.block\_size]

            cipherDecrypto = AES.new(keyAES, AES.MODE\_CBC, iv)

            videoDecrypt = unpad(cipherDecrypto.decrypt(data[AES.block\_size:]), AES.block\_size)

            return base64.b64encode(videoDecrypt).decode('ascii')

    except IOError:

        print("File not accessible")

* + Đọc video bằng phương thức open(videoDir, ‘rb’).
  + Giải mã video bằng phương thức decrypt().
  + Chuyển video sang sạng base64 bằng phương thức

base64.b64encode()

#### **3.3.3.4 Phát hiện người bằng Yolo**

#load model yolo đã trên vào mạng dnn của opencv

net = cv2.dnn.readNetFromDarknet("yolov4-tiny-custom.cfg", "yolov4-tiny-custom\_final.weights")

#Sử dụng GPU cho opencv

net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

#Load các layer trong mạng Yolo

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

#Random màu khung đối tượng

colors = np.random.uniform(0, 255, size=(len(classes), 3))

while True:

#Nhận và giải mã hình ảnh

        imgBytes = socket.recv()

        encrypto = unpad(cipherDecrypto.decrypt(imgBytes), AES.block\_size)

#Định dạng và lấy kích thước của ảnh

        nparr = np.frombuffer(encrypto, np.uint8)

        imgYolo = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

        height, width, channels = imgYolo.shape

        class\_ids = []

        confidences = []

        boxes = []

#Resize ảnh về kích thước 416x416

        blob = cv2.dnn.blobFromImage(imgYolo, 1/255, (416, 416), (0, 0, 0), True, crop=False)

#Đưa ảnh vào Yolo

        net.setInput(blob)

#Output của Yolo

        outs = net.forward(output\_layers)\

#Kiểm tra và lấy các bounding box của đối tượng trong ảnh

        for out in outs:

            for detection in out:

                scores = detection[5:]

                class\_id = np.argmax(scores)

                confidence = scores[class\_id]

                if confidence > 0:

                    center\_x = int(detection[0] \* width)

                    center\_y = int(detection[1] \* height)

                    w = int(detection[2] \* width)

                    h = int(detection[3] \* height)

                    x = int(center\_x - w / 2)

                    y = int(center\_y - h / 2)

                    boxes.append([x, y, w, h])

                    confidences.append(float(confidence))

                    class\_ids.append(class\_id)

#Sử dụng non-max supression để chọn khung chính xác nhất của đối tượng

        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

        font = cv2.FONT\_HERSHEY\_PLAIN

        havePerson = 0

#Đóng khung đối tượng trong ảnh

        for i in range(len(boxes)):

            if i in indexes:

                havePerson = 1

                x, y, w, h = boxes[i]

                color = colors[class\_ids[i]]

                cv2.rectangle(imgYolo, (x, y), (x + w, y + h), color, 2)

#Gửi ảnh đã nhận diện đến Stream và Control

        imageSenderStream.send\_image("yolo",imgYolo)

        if havePerson == 1:

            imageSenderControl.send\_image(1,imgYolo)

        else:

            imageSenderControl.send\_image(0,imgYolo)

#### **3.3.3.5 Nhận ảnh và đưa lên stream**

* HTML

<div>

      <h4 class="text-center">From KMA with nove (\*\_^)'</h4>

      <br>

      <div class="text-center">

        <img id="imgDiv" src="{{ url\_for('static', filename='img/default.jpg') }}" height="480" width="640">

      </div>

    </div>

* + Tạo thẻ ảnh có id = ‘imgDiv’ có kích thước 640x480.
* Javascript

const socket = io();

var img = document.getElementById('imgDiv');

socket.on('imgChannel',(data)=>{

    img.src = "data:image/jpg;base64, " + data.data;

})

* + Tạo hàm lắng nghe sự kiện. Khi nhận được commit ảnh từ server sẽ thay đổi ảnh trong thẻ ảnh có id ‘imgDiv’. Thay đổi ảnh liên tục sẽ tạo thành video.

#### **3.3.3.6 Lưu video và gửi cảnh báo khi phát hiện bất thường**

* Gửi cảnh báo khi phát hiện bất thường

def firstBlood(ten,now):

    thoigian = time.strftime("%H:%M:%S--%d/%m/%Y", now)

    message = "Có người xuất hiện lúc {} ".format(thoigian)

    msg = MIMEText(message, 'plain')

    msg['Subject']= "Cảnh báo!!!"

    server = smtplib.SMTP('smtp.gmail.com:587')

    server.starttls()

    server.login('kmafhnib@gmail.com','Lucifer98@')

    server.sendmail('kmafhnib@gmail.com', 'fhnibkma@gmail.com', msg.as\_string())

    server.quit()

* Lưu video khi phát hiện bất thường

while True:

        msg,img = imageRecv.recv\_image()

        if msg == 1:

            if sendMail == 1:

                saveVideo.write(img)

            else:

                now = time.localtime()

                timeString = time.strftime("%d%m%Y%H%M%S", now)

                nameVideo =  timeString + ".mp4"

                saveVideo = cv2.VideoWriter('static/video/{}'.format(nameVideo), fourcc, 12, (640, 480))

                saveVideo.write(img)

                sendMail = 1

                countOut = 120

                threading.Thread(target=firstBlood, args=(nameVideo, now)).run()

        else:

            countOut -= 1

            """

            if countOut == 0:

                saveVideo.write(img)

                socket.send\_string(nameVideo)

            """

            if countOut > 0:

                saveVideo.write(img)

            else:

                if saveVideo is not None:

                    saveVideo.release()

                    saveVideo = None

                    sendMail = 0

                    socket.send\_string(nameVideo)

* + Khi phát hiện bất thường, server sẽ ghi lại video bằng cv2.VideoWriter() qua phương thức write().

**3.3.3.7 Xem lại video đã lưu**

* Python

@app.route('/api/videoplayback')

def videoplayback():

    global user

    videos = Video.query.filter\_by(mahoa='1')

    if user != None:

        return render\_template('videoplayback.html',role=user.role,videos=videos)

    else:

        return render\_template('videoplayback.html',role='normal',videos=videos)

* HTML

<h4 class="text-center">From KMA with nove (\*\_^)'</h4>

      <div class="row align-items-center">

        {% for video in videos %}

        <div class="col-md-4 p-4">

          <div class="card" style="width: 323px;">

            <video value="{{video.ten}}" class="srcVideo" width="320" height="240" controls>

            </video>

            <div class="card-body text-center">

              <b class="card-text">{{video.thoigian}}</b>

            </div>

          </div>

        </div>

        {% endfor %}

      </div>

**3.3.3.8 Chức năng đăng nhập**

* Python

@login.user\_loader

def user\_load(user\_id):

    return User.query.get(user\_id)

@app.route('/login', methods = ["POST"])

def login():

      global user

      username = request.form.get("username")

      password = str(request.form.get("password"))

      password = str(hashlib.md5(password.strip().encode("utf-8")).hexdigest())

      user = User.query.filter(User.username == username, User.password == password).first()

      if user:

          login\_user(user=user)

          return redirect("/api/streaming")

      else:

          flash('Sai tài khoản hoặc mật khẩu. Vui lòng đăng nhập lại', 'danger')

          return redirect("/")

* HTML

<div class="card"

        style="width: 500px;height: 450px;margin-left: 50%;margin-top: 20%; transform: translate(-50%,-50%);">

        <form action="{{url\_for('login')}}" method="POST">

          <br>

          <br>

          <h3 class="text-center" style="color:#f50057">ĐĂNG NHẬP</h3>

          <br>

          <br>

          {% with messages = get\_flashed\_messages(with\_categories=true) %}

            {% if messages %}

              {% for category, message in messages %}

                <div class="text-center alert alert-{{ category }}">

                  {{ message }}

                </div>

              {% endfor %}

            {% endif %}

          {% endwith %}

          <div class="form-group text-center">

            <input style="max-width: 68%; margin-left: 15%; margin-right: 15%;" type="text" name="username"

              class="form-control text-center" placeholder="Tài khoản" />

          </div>

          <br>

          <div class="form-group text-center">

            <input style="max-width: 68%; margin-left: 15%; margin-right: 15%;" type="password" name="password"

              class="form-control text-center" placeholder="Mật khẩu" />

          </div>

          <br>

          <div class="form-group text-center">

            <input type="submit" value="Đăng nhập" class="btn btn-info text-center" />

          </div>

        </form>

      </div>

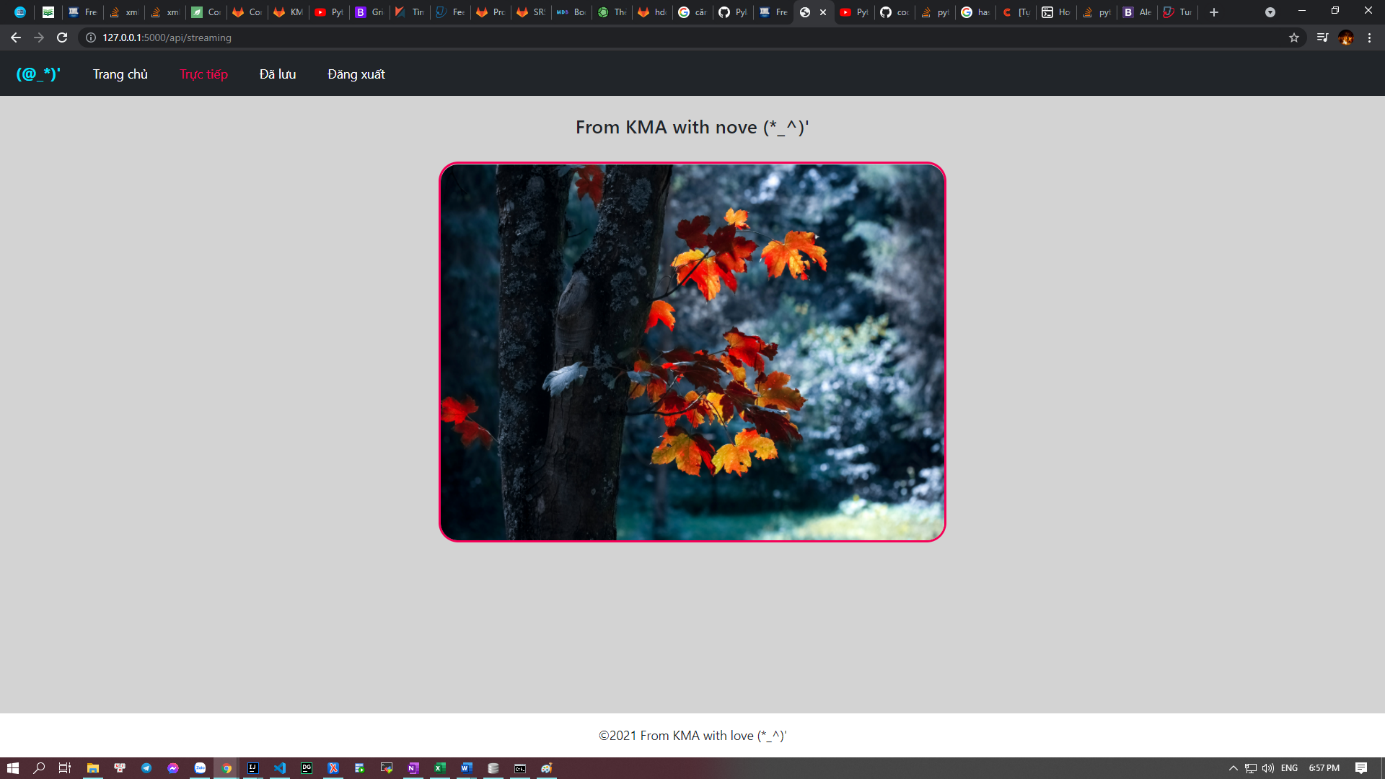
## **3.4 Kết quả và thực nghiệm**

* + 1. **Kết quả đạt được**
* Train xong model Yolo để phát hiện bất thường và áp dụng để xử lí ảnh trong hệ thống camera an ninh.
* Xây dựng hệ thống lưu video và gửi cảnh báo đến người dùng khi phát hiện người.
* Xây dựng web server để người dùng truy cập giám sát trực tiếp camera an ninh cũng như xem lại các video đã lưu.
* Mã hóa hình ảnh và video để bảo mật camera an ninh.

### **Thực nghiệm**

#### **3.4.2.1 Web server**

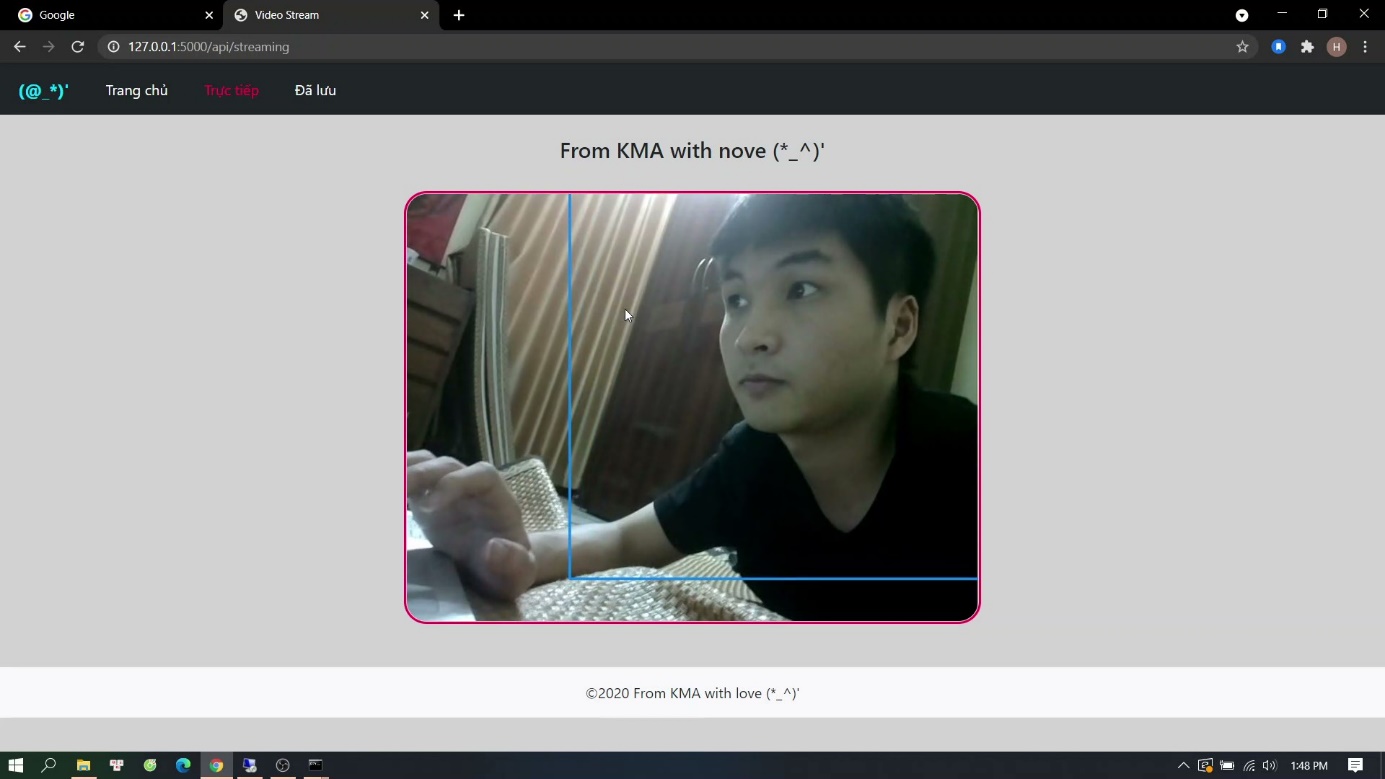
Web server hoạt động ổn định và nhanh.Giao diện hiển thị đúng như thiết kế.



Hình 3.7 Giao diện Web Server

#### **3.4.2.2 Xem video trực tiếp**

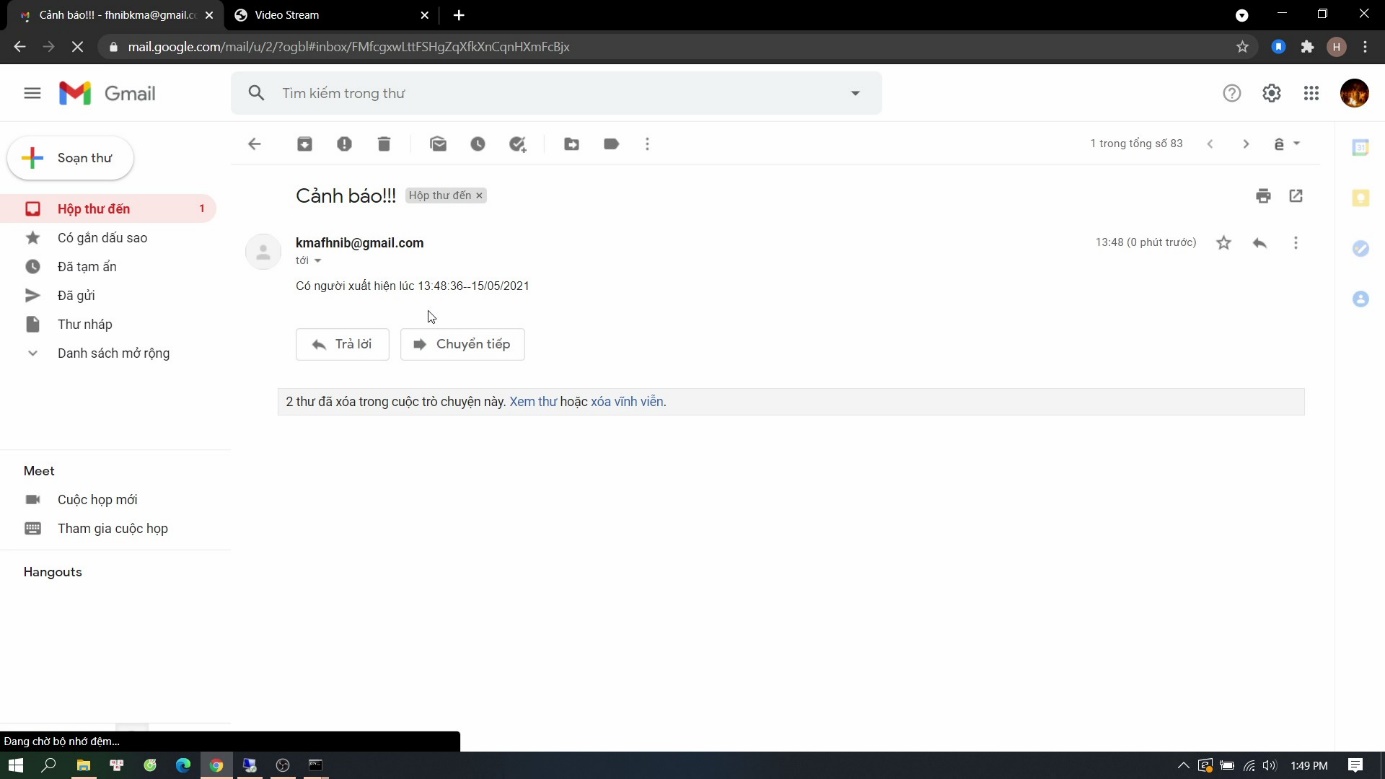
Hệ thống phát video trực tiếp ổn định, có phát hiện bất thường chính xác.



Hình 3.8 Video stream hệ thống khi phát hiện người

#### **3.4.2.3 Gửi mail cảnh báo khi phát hiện bất thường**

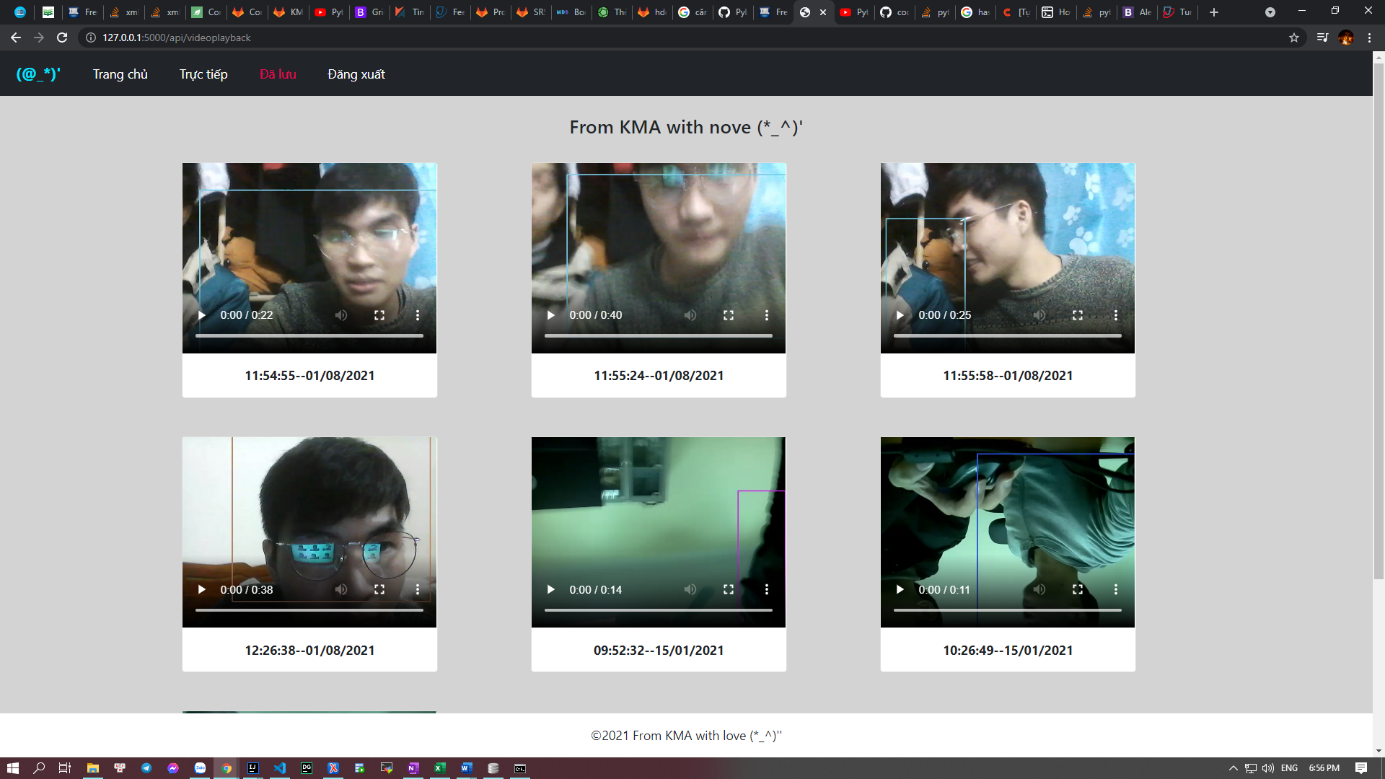
Khi phát hiện bất thường, có mail cảnh báo ngay lập tức.



Hình 3.9 Mail cảnh báo khi phát hiện người

#### **3.4.2.4 Xem lại các video đã lưu khi phát hiện bất thường**

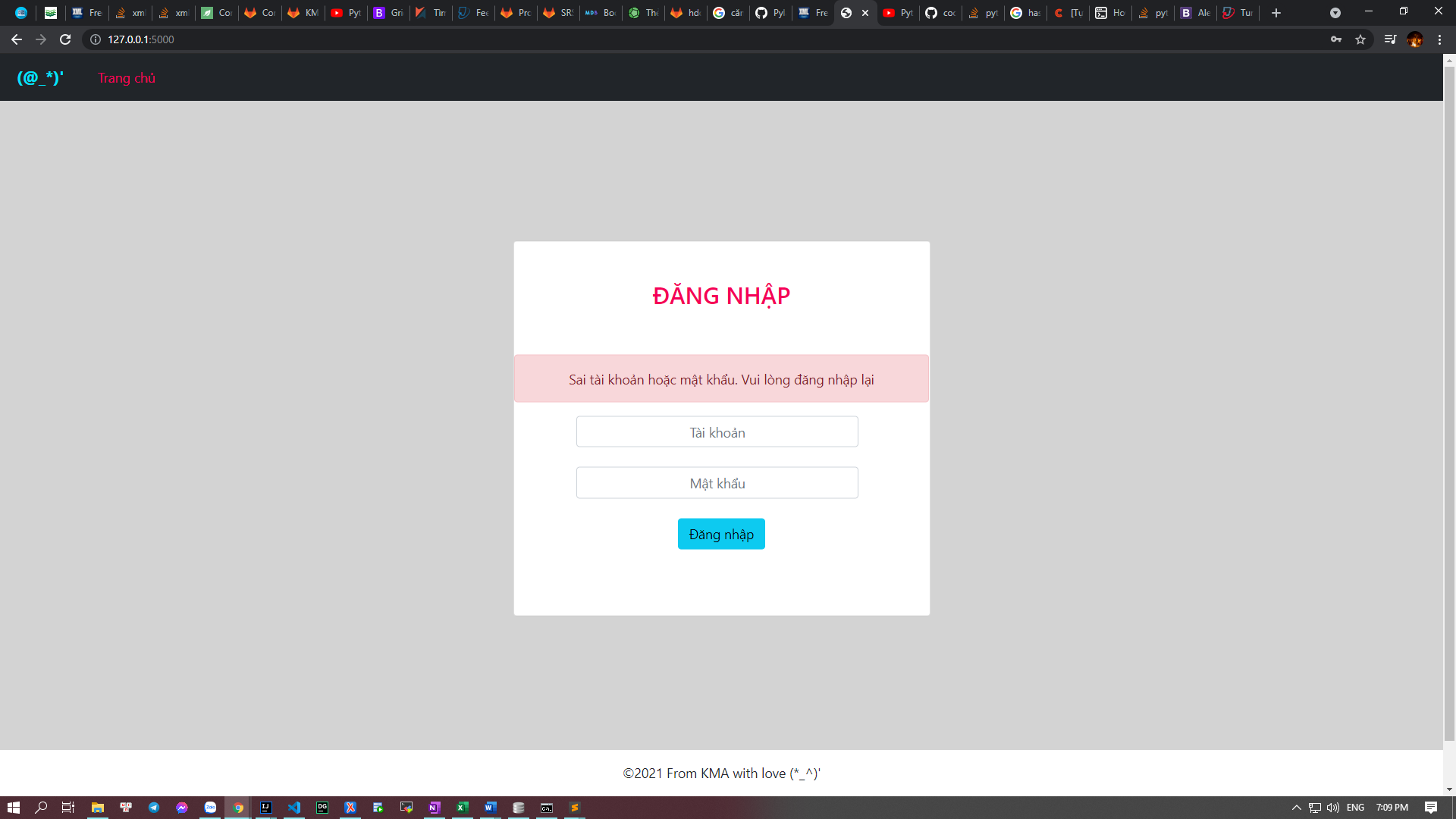
Xem lại được tất cả video đã lưu khi phát hiện bất thường.



Hình 3.10 Các video đã lưu khi phát hiện người

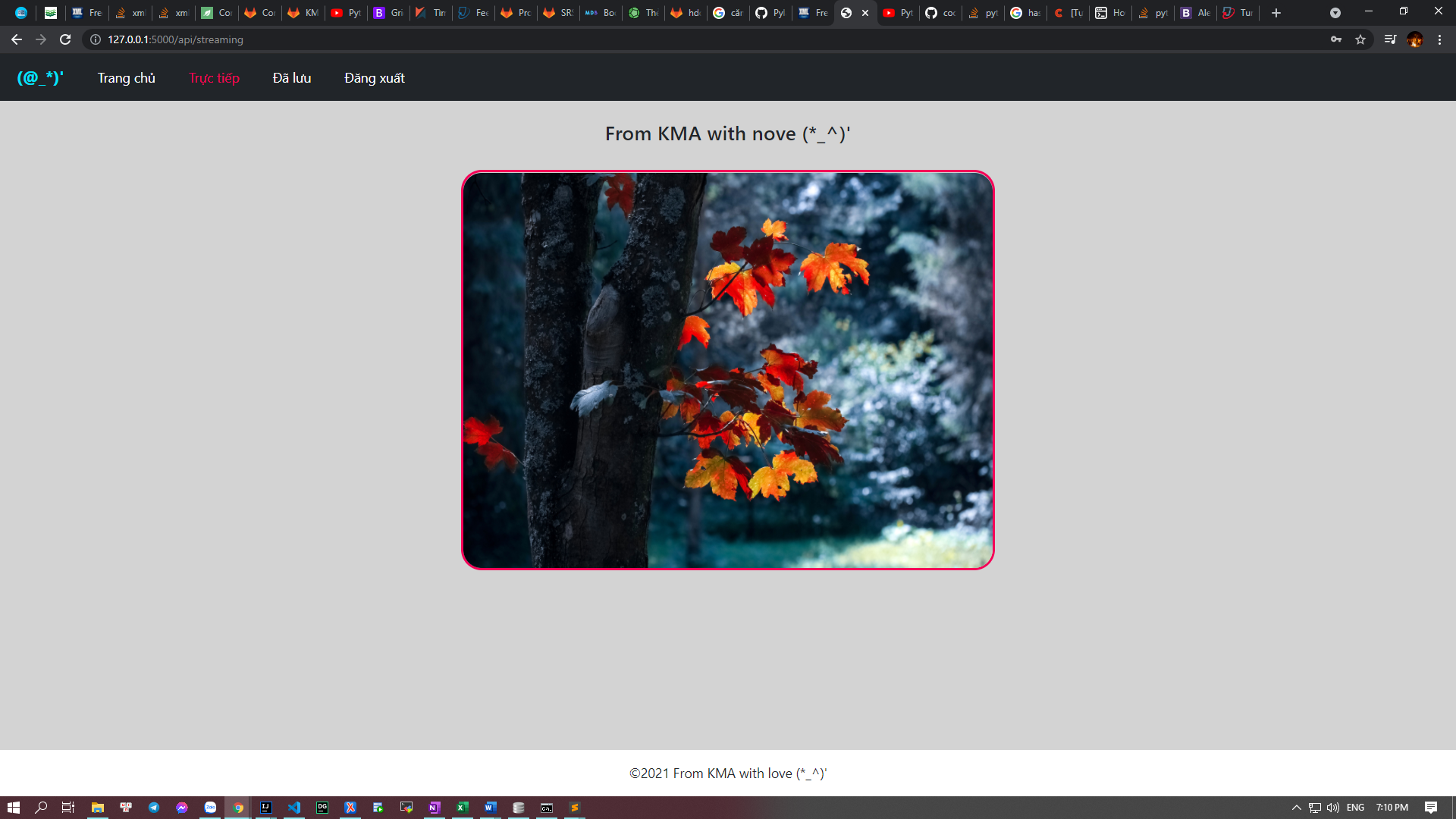
#### **3.4.2.5 Đăng nhập**

Khi truy cập vào hệ thống, hệ thống sẽ yêu cầu đăng nhập. Nếu người dùng nhập sai thông tin đăng nhập, hệ thống sẽ cảnh báo và yêu cầu người dùng đăng nhập lại.



Hình 3.11 Thông báo khi nhập sai thông tin đăng nhập

Nếu nhập đúng thông tin đăng nhập, hệ thống sẽ hiển thị hình ảnh trực tiếp từ camera và các chức năng khác cho người dùng.



Hình 3.12 Giao diện khi nhập đúng thông tin đăng nhập

* Như vậy, hệ thống hoạt động ổn định và chính xác đúng như các yêu cầu đã đặt ra.

# **KẾT LUẬN**

Sau thời gian nỗ lực thực hiện theo tiến độ, đề tài đã hoàn thành theo đúng mục tiêu đã đề ra:

* Train xong model Yolo để phát hiện bất thường trong hệ thống camera an ninh thông minh.
* Xây dựng hệ thống lưu video và gửi cảnh báo đến người dùng khi phát hiện bất thường.
* Xây dựng web server để người dùng truy cập giám sát trực tiếp camera an ninh cũng như xem lại các video đã lưu.
* Tìm hiểu về mã hóa và áp dụng vào bảo mật hệ thống camera an ninh thông minh.

Tuy nhiên vì kiến thức cũng như thời gian còn hạn chế nên đề tài còn một số thiếu sót như:

* Hệ thống chưa có tính năng phát hiện người dùng đăng kí.
* Giao diện webserver còn đơn giản.
* Chưa tích hợp nhiều camera

Do đó, trong tương lai, tôi sẽ tiếp tục nghiên cứu, phát triển để hoàn thiện và có thể áp dụng vào thực tế. Rất mong nhận được các ý kiến đóng góp của thầy cô và các bạn.

# **TÀI LIỆU THAM KHẢO**

[1] Tìm Hiểu Convolutional Neural Networks Cho Phân Loại Ảnh, https://pbcquoc.github.io/cnn/?fbclid=IwAR3xIviL4NS7Qe7eDtN8bNuomxgZeq1WnYYkxz3EYdlxdM9gKXSy7xTk330

[2]Tìm hiểu region based object detectors, https://www.phamduytung.com/blog/2018-12-05-what-do-we-learn-from-object-detection-p1/?fbclid=IwAR1Lrgqvd1sAgm4ZUUcYfW7s9ueQ8QVf8ZW2aPanigSjB0LttaznvcM48m8

[3] Tìm Hiểu Mô Hình YOLO Cho Bài Toán Object Detection, https://pbcquoc.github.io/yolo/?fbclid=IwAR0aZy8yzAQM676VeIKYdtayzGFsDZNXCaiaYvH\_knEjtDcJJARF7kygvNs

[4] Object detection : YOLO, https://forum.machinelearningcoban.com/t/object-detection-yolo/503?fbclid=IwAR27L7Iso\_EtKkwSlgVVziJnwdGz6aHsjKu6SCfQGqpx0to0KH9u9Vxt6E4

[5] Cách tạo ra bounding box trong YOLO?, https://viblo.asia/q/cach-tao-ra-bounding-box-trong-yolo-QqKLQ0wMZ7z?fbclid=IwAR0OuZan\_PSkW9\_ibLPK\_nP1RAHcIGjB0EMOeB1PpDKRBCX5JgUcSh2zblI

[6] Tìm hiểu mô hình YOLO cho phát hiện vật - Từ YOLOv1 đến YOLOv5, https://aicurious.io/posts/tim-hieu-yolo-cho-phat-hien-vat-tu-v1-den-v5/?fbclid=IwAR3HYgzzLIPF6-fpQ2elK1jrbPGLs4hojhSMCr4OBs7rjTb-kPRDcM9d9iE#iv-yolov4

[7] ZeroMQ documentation, https://zeromq.org/get-started/

[8] Learning ØMQ with pyzmq,

https://learning-0mq-with-pyzmq.readthedocs.io/en/latest/index.html

[9] Xây dựng RESTful API Với Flask: Theo hướng Tự làm, https://code.tutsplus.com/vi/tutorials/building-restful-apis-with-flask-diy--cms-26625

[10] Flask API, https://www.flaskapi.org/

[11] Vũ Hữu Tiệp, “*Machine Learning Cơ Bản*”, 2018.

# **PHỤ LỤC**

* + - 1. **Code phía client (Raspberry Pi 3)**

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Thêm thư viện\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import socket

import imagezmq

from picamera.array import PiRGBArray

from picamera import PiCamera

import cv2

import requests

import json

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1\_OAEP

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

import base64

import ast

import zmq

import numpy as np

import time

import hashlib

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Khai báo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

camera = PiCamera()

camera.resolution = (640, 480)

camera.framerate = 8

rawCapture = PiRGBArray(camera, size=(640, 480))

context = zmq.Context()

socket = context.socket(zmq.PUB)

socket.bind("tcp://169.254.99.118:3021")

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Tạo và trao đổi khóa RSA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

addr = 'http://169.254.5.211:5000'

url = addr + '/api/rsa'

key = RSA.generate(1024)

pubKey = key.publickey().exportKey("PEM")

response = requests.post(url, data=pubKey)

data = response.json().get('key')

rsa\_private\_key = PKCS1\_OAEP.new(key)

key = rsa\_private\_key.decrypt(ast.literal\_eval(str(data)))

sha = hashlib.sha1(key).hexdigest()

keyAES = sha[0:16].encode("utf-8")

iv = b'0123456789abcdef'

cipher = AES.new(keyAES, AES.MODE\_CBC,iv)

cipherDecrypto = AES.new(keyAES, AES.MODE\_CBC, iv)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Mã hóa và gửi hình ảnh\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

for frame in camera.capture\_continuous(rawCapture, format="bgr", use\_video\_port=True):

image = frame.array

\_,image = cv2.imencode('.jpg',image)

imgBytes = image.tobytes()

imgEncrypto = cipher.encrypt(pad(imgBytes, AES.block\_size))

socket.send(imgEncrypto)

rawCapture.truncate(0)

* + - 1. **Server**
* File server.py (Flask server)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Thêm thư viện\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

from flask import Flask, render\_template, Response, request, redirect, flash

import cv2

import threading

import base64

from flask\_login.utils import login\_user, logout\_user

from flask\_socketio import SocketIO,emit

import jsonpickle

import imagezmq

from flask\_sqlalchemy import SQLAlchemy

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1\_OAEP

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

from flask\_login import LoginManager, UserMixin

import hashlib

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Khai báo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = '8f42a73054b1749f8f58848be5e6502c'

#login

login = LoginManager(app=app)

#Stream

socketio = SocketIO(app,async\_mode='threading')

countConnect = 0

imageRecv = imagezmq.ImageHub(open\_port = 'tcp://127.0.0.1:1998' , REQ\_REP= False)

#Database

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///video.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

user = None

class Video(db.Model):

ten = db.Column(db.String(50), primary\_key=True)

thoigian = db.Column(db.String(20), nullable=False)

mahoa = db.Column(db.String(2), nullable=False)

class User(db.Model, UserMixin):

\_\_tablename\_\_ = "user"

id = db.Column(db.Integer, primary\_key = True, autoincrement=True)

username = db.Column(db.String(50), nullable=False)

password = db.Column(db.String(50), nullable=False)

role = db.Column(db.String(20), nullable=False)

def \_\_init\_\_(self, username, password, role):

self.username = username

self.password = password

self.role = role

#db.create\_all()

key = b'keygenerateaes'

keyAES = b'\xe8\'\x7fFOP\xa7\xd7f"\xffw\xd3\xb5\xe8\x04'

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_API Admin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Login

@login.user\_loader

def user\_load(user\_id):

return User.query.get(user\_id)

@app.route('/login', methods = ["POST"])

def login():

global user

username = request.form.get("username")

password = str(request.form.get("password"))

password = str(hashlib.md5(password.strip().encode("utf-8")).hexdigest())

user = User.query.filter(User.username == username, User.password == password).first()

if user:

login\_user(user=user)

return redirect("/api/streaming")

else:

flash('Sai tài khoản hoặc mật khẩu. Vui lòng đăng nhập lại', 'danger')

return redirect("/")

@app.route('/logout')

def logout():

logout\_user()

return redirect("/")

@app.route('/admin')

def admingg():

global user

users = User.query.all()

return render\_template('admin.html',role = user.role, users = users)

@app.route('/deleteuser/<int:id>')

def deleteuser(id):

global user

User.query.filter\_by(id=id).delete()

db.session.commit()

users = User.query.all()

return render\_template('admin.html',role = user.role, users = users)

@app.route('/register', methods = ["POST"])

def register():

global user

username = request.form.get("username")

password = str(request.form.get("password"))

password = str(hashlib.md5(password.strip().encode("utf-8")).hexdigest())

role = request.form.get("role")

re = User(username,password,role)

db.session.add(re)

db.session.commit()

users = User.query.all()

return render\_template('admin.html',role = user.role, users = users)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_API xem video trực tiếp\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

@app.route('/api/streaming')

def streaming():

global user

if user != None:

return render\_template('videostream.html',role=user.role)

else:

return render\_template('videostream.html',role='nomarl')

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_API xme lại video đã lưu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

@app.route('/api/videoplayback')

def videoplayback():

global user

videos = Video.query.filter\_by(mahoa='1')

if user != None:

return render\_template('videoplayback.html',role=user.role,videos=videos)

else:

return render\_template('videoplayback.html',role='normal',videos=videos) #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_API trao đổi khóa RSA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

@app.route('/api/rsa', methods=['POST'])

def rsa():

global key

data = request.data

pubKey = RSA.importKey(data)

rsa\_public\_key = PKCS1\_OAEP.new(pubKey)

aesKey = rsa\_public\_key.encrypt(key)

response = {'key': "{}".format(aesKey)}

response\_pickled = jsonpickle.encode(response)

return Response(response=response\_pickled , status=200 , mimetype ="application/json")

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_API load video đã lưu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

@app.route('/api/getVideo/<string:name>',methods=['GET','POST'])

def getVideo(name):

global cipherDecrypto

videoDir = "static/video/{}".format(name)

try:

with open(videoDir,'rb') as f:

data = f.read()

iv = data[:AES.block\_size]

cipherDecrypto = AES.new(keyAES, AES.MODE\_CBC, iv)

videoDecrypt = unpad(cipherDecrypto.decrypt(data[AES.block\_size:]), AES.block\_size)

return base64.b64encode(videoDecrypt).decode('ascii')

except IOError:

print("File not accessible")

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Commit ảnh \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#SocketIO

#Nếu có client connect

@socketio.on('connect')

def connectChannel():

global countConnect

countConnect+=1

#Nếu client disconnect

@socketio.on('disconnect')

def disconnectChannel():

global countConnect

if countConnect >0:

countConnect-=1

elif countConnect<0:

countConnect=0

#Hàm commit ảnh

def thSendImg():

    global countConnect

    while True:

        #nhận hình ảnh từ Yolo

        msg,img = imageRecv.recv\_image()

        #encode ảnh

        \_,img\_encode = cv2.imencode('.jpg',img)

        #encode base64

        img\_64 = base64.b64encode(img\_encode).decode('ascii')

        #Nếu có client connect thì gửi ảnh

        if countConnect>0 :

            socketio.emit('imgChannel', {'data': img\_64}, broadcast=True)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Main()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if \_\_name\_\_ == '\_\_main\_\_':

threading.Thread(target=thSendImg, daemon=True).start()

socketio.run(app,host='0.0.0.0',port=5000)

* File index.html và videostream.html (Hiển thị video trực tiếp)

<!DOCTYPE html>

<head>

<title>Video Stream</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/bootstrap.min.css') }}">

<style>

body {

background-color: lightgray;

}

img {

border: 3px solid #f50057;

border-radius: 25px;

}

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

text-align: center;

background-color: white;

}

</style>

</head>

<body>

<div>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand ms-2" style="color: #00e5ff" href="#"><b>(@\_\*)'</b></a>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto ms-3 mb-2 mb-lg-0">

<li class="nav-item me-4">

<a class="nav-link active" aria-current="page" href="{{url\_for('index')}}">Trang

chủ</a>

</li>

{% if current\_user.is\_authenticated %}

<li class="nav-item me-4">

<a class="nav-link active" style="color: #f50057" href="{{url\_for('streaming')}}">Trực tiếp</a>

</li>

<li class="nav-item me-4">

<a class="nav-link active" href="{{url\_for('videoplayback')}}">Đã lưu</a>

</li>

{% if role == 'admin' %}

<li class="nav-item me-4">

<a class="nav-link active" href="{{url\_for('admingg')}}">Quản lí</a>

</li>

{%endif%}

<li class="nav-item me-4">

<a class="nav-link active" href="{{url\_for('logout')}}">Đăng xuất</a>

</li>

{% endif %}

</ul>

</div>

</div>

</nav>

<br>

<div>

<h4 class="text-center">From KMA with nove (\*\_^)'</h4>

<br>

<div class="text-center">

<img id="imgDiv" src="{{ url\_for('static', filename='img/default.jpg') }}" height="480" width="640">

</div>

</div>

</div>

</body>

<script type="text/javascript" src="{{ url\_for('static', filename='js/bootstrap.min.js') }}"></script>

<script src="{{ url\_for('static', filename='js/socket.io.min.js') }}" type="text/javascript"></script>

<script type="text/javascript" src="{{ url\_for('static',filename='js/source.js') }}"></script>

<br><br>

<footer class="footer">

<div class="text-center p-3">

©2021 From KMA with love (\*\_^)'

</div>

</footer>

</html>

* File videoplayback.html (Xem lại video đã lưu)

!DOCTYPE html>

<head>

<title>Video Playback</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/bootstrap.min.css') }}">

<style>

body {

background-color: lightgray;

}

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

text-align: center;

background-color: white;

}

</style>

</head>

<body>

<div>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand ms-2" style="color: #00e5ff" href="#"><b>(@\_\*)'</b></a>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto ms-3 mb-2 mb-lg-0">

<li class="nav-item me-4">

<a class="nav-link active" aria-current="page" href="{{url\_for('index')}}">Trang

chủ</a>

</li>

{% if current\_user.is\_authenticated %}

<li class="nav-item me-4">

<a class="nav-link active" href="{{url\_for('streaming')}}">Trực tiếp</a>

</li>

<li class="nav-item me-4">

<a class="nav-link active" style="color: #f50057" href="{{url\_for('videoplayback')}}">Đã lưu</a>

</li>

{% if role == 'admin' %}

<li class="nav-item me-4">

<a class="nav-link active" href="{{url\_for('admingg')}}">Quản lí</a>

</li>

{%endif%}

<li class="nav-item me-4">

<a class="nav-link active" href="{{url\_for('logout')}}">Đăng xuất</a>

</li>

{% endif %}

</ul>

</div>

</div>

</nav>

<br>

<div class="container">

{% if current\_user.is\_authenticated %}

<h4 class="text-center">From KMA with nove (\*\_^)'</h4>

<div class="row align-items-center">

{% for video in videos %}

<div class="col-md-4 p-4">

<div class="card" style="width: 323px;">

<video value="{{video.ten}}" class="srcVideo" width="320" height="240" controls>

</video>

<div class="card-body text-center">

<b class="card-text">{{video.thoigian}}</b>

</div>

</div>

</div>

{% endfor %}

</div>

{% else %}

<br>

<div class="card"

style="width: 500px;height: 450px;margin-left: 50%;margin-top: 20%; transform: translate(-50%,-50%);">

<form action="{{url\_for('login')}}" method="POST">

<br>

<br>

<h3 class="text-center" style="color:#f50057">ĐĂNG NHẬP</h3>

<br>

<br>

<div class="form-group text-center">

<input style="max-width: 68%; margin-left: 15%; margin-right: 15%;" type="text" name="username"

class="form-control text-center" placeholder="Tài khoản" />

</div>

<br>

<div class="form-group text-center">

<input style="max-width: 68%; margin-left: 15%; margin-right: 15%;" type="password" name="password"

class="form-control text-center" placeholder="Mật khẩu" />

</div>

<br>

<div class="form-group text-center">

<input type="submit" value="Đăng nhập" class="btn btn-info text-center" />

</div>

</form>

</div>

{%endif%}

</div>

</div>

</body>

<script type="text/javascript" src="{{ url\_for('static', filename='js/bootstrap.min.js') }}"></script>

<script type="text/javascript" src="{{ url\_for('static', filename='js/jquery-3.5.1.min.js') }}"></script>

<script type='text/javascript'>

$(document).ready(() => {

var sources = document.getElementsByClassName("srcVideo");

for (let srcVideo of sources) {

$.ajax({

url: "/api/getVideo/" + srcVideo.getAttribute("value"),

type: 'get',

dataType: 'text',

success: (result) => {

srcVideo.innerHTML = '<source src="data:video/webm;base64,' + result + '" type="video/mp4" />';

}

});

}

})

</script>

<br>

<footer class="footer">

<div class="text-center p-3">

©2021 From KMA with love (\*\_^)''

</div>

</footer>

</html>

* File source.js

const socket = io();

var img = document.getElementById('imgDiv');

socket.on('imgChannel',(data)=>{

img.src = "data:image/jpg;base64, " + data.data;

})

* File yolov4.py (Worker Yolo)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Thêm thư viện\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import imagezmq

import cv2

import numpy as np

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

import zmq

import base64

import hashlib

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Khai báo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#yolo

classes = ["nguowfi"]

net = cv2.dnn.readNetFromDarknet("yolov4-tiny-custom.cfg", "yolov4-tiny-custom\_final.weights")

net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

colors = np.random.uniform(0, 255, size=(len(classes), 3))

#send, recv image

context = zmq.Context()

socket = context.socket(zmq.SUB)

socket.connect("tcp://192.168.0.106:3021")

socket.subscribe("")

imageSenderStream = imagezmq.ImageSender(connect\_to = 'tcp://127.0.0.1:1998',REQ\_REP = False)

imageSenderControl = imagezmq.ImageSender(connect\_to = 'tcp://127.0.0.1:1203',REQ\_REP = False)

#Crypto

key = b'keygenerateaes'

sha = hashlib.sha1(key).hexdigest()

keyAESI = sha[0:16].encode("utf-8")

iv = b'0123456789abcdef'

cipherDecrypto = AES.new(keyAESI, AES.MODE\_CBC, iv)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Main()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if \_\_name\_\_ == '\_\_main\_\_':

while True:

imgBytes = socket.recv()

encrypto = unpad(cipherDecrypto.decrypt(imgBytes), AES.block\_size)

nparr = np.frombuffer(encrypto, np.uint8)

imgYolo = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

height, width, channels = imgYolo.shape

class\_ids = []

confidences = []

boxes = []

blob = cv2.dnn.blobFromImage(imgYolo, 1/255, (416, 416), (0, 0, 0), True, crop=False)

net.setInput(blob)

outs = net.forward(output\_layers)

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0:

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

x = int(center\_x - w / 2)

y = int(center\_y - h / 2)

boxes.append([x, y, w, h])

confidences.append(float(confidence))

class\_ids.append(class\_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

font = cv2.FONT\_HERSHEY\_PLAIN

havePerson = 0

for i in range(len(boxes)):

if i in indexes:

havePerson = 1

x, y, w, h = boxes[i]

color = colors[class\_ids[i]]

cv2.rectangle(imgYolo, (x, y), (x + w, y + h), color, 2)

imageSenderStream.send\_image("yolo",imgYolo)

if havePerson == 1:

imageSenderControl.send\_image(1,imgYolo)

else:

imageSenderControl.send\_image(0,imgYolo)

1. File crypto.py (Mã hóa video)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Thêm thư viện\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

from flask import Flask

import cv2

import numpy as np

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

import threading

import zmq

import os

import base64

from flask\_sqlalchemy import SQLAlchemy

from Crypto import Random

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Khai báo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#send, recv name video

context = zmq.Context()

socket = context.socket(zmq.SUB)

socket.connect("tcp://127.0.0.1:2456")

socket.subscribe("")

#crypto

keyAES = b'\xe8\'\x7fFOP\xa7\xd7f"\xffw\xd3\xb5\xe8\x04'

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///video.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

class Video(db.Model):

ten = db.Column(db.String(50), primary\_key=True)

thoigian = db.Column(db.String(20), nullable=False)

mahoa = db.Column(db.String(2), nullable=False)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Funtion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def decrypt(name):

try:

videoDir = "static/video/"+name

videoBinary = open(videoDir, "rb").read()

iv = videoBinary[:AES.block\_size]

cipherDecrypto = AES.new(keyAES, AES.MODE\_CBC, iv)

videoDecrypt = unpad(cipherDecrypto.decrypt(videoBinary[AES.block\_size:]), AES.block\_size)

os.remove(videoDir)

with open(videoDir,'wb') as file:

file.write(videoDecrypt)

except:

print(name)

def encrypt(name):

videoDir = "static/video/"+name

videoBinary = open(videoDir, "rb").read()

iv = Random.new().read(AES.block\_size)

cipher = AES.new(keyAES, AES.MODE\_CBC,iv)

videoEncrypt = cipher.encrypt(pad(videoBinary, AES.block\_size))

data = iv + videoEncrypt

os.remove(videoDir)

with open(videoDir,'wb') as file:

file.write(data)

video = Video.query.filter\_by(ten= name).first()

video.mahoa = '1'

db.session.commit()

print("oki")

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Main()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if \_\_name\_\_ == '\_\_main\_\_':

while True:

nameVideo = socket.recv\_string()

encrypt(nameVideo)

* File control.py (worker Control)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Thêm thư viện\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

from flask import Flask

import imagezmq

import cv2

import numpy as np

import time

import threading

import smtplib

from email.mime.text import MIMEText

from flask\_sqlalchemy import SQLAlchemy

import zmq

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Khai báo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Recv image

imageRecv = imagezmq.ImageHub(open\_port = 'tcp://127.0.0.1:1203' , REQ\_REP= False)

#Save video

fourcc = cv2.VideoWriter\_fourcc(\*'H264')

saveVideo = None

#send name video

context = zmq.Context()

socket = context.socket(zmq.PUB)

socket.bind("tcp://127.0.0.1:2456")

#Database

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///video.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

class Video(db.Model):

ten = db.Column(db.String(50), primary\_key=True)

thoigian = db.Column(db.String(20), nullable=False)

mahoa = db.Column(db.String(2), nullable=False)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Funtion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def firstBlood(ten,now):

thoigian = time.strftime("%H:%M:%S--%d/%m/%Y", now)

message = "Có người xuất hiện lúc {} ".format(thoigian)

msg = MIMEText(message, 'plain')

msg['Subject']= "Cảnh báo!!!"

server = smtplib.SMTP('smtp.gmail.com:587')

server.starttls()

server.login('kmafhnib@gmail.com','Lucifer98@')

server.sendmail('kmafhnib@gmail.com', 'fhnibkma@gmail.com', msg.as\_string())

server.quit()

video = Video(ten = ten,thoigian = thoigian, mahoa = '0')

db.session.add(video)

db.session.commit()

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Main()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if \_\_name\_\_ == '\_\_main\_\_':

sendMail = 0

countOut = 0

while True:

msg,img = imageRecv.recv\_image()

if msg == 1:

if sendMail == 1:

saveVideo.write(img)

else:

now = time.localtime()

timeString = time.strftime("%d%m%Y%H%M%S", now)

nameVideo = timeString + ".mp4"

saveVideo = cv2.VideoWriter('static/video/{}'.format(nameVideo), fourcc, 12, (640, 480))

saveVideo.write(img)

sendMail = 1

countOut = 120

threading.Thread(target=firstBlood, args=(nameVideo, now)).run()

else:

countOut -= 1

"""

if countOut == 0:

saveVideo.write(img)

socket.send\_string(nameVideo)

"""

if countOut > 0:

saveVideo.write(img)

else:

if saveVideo is not None:

saveVideo.release()

saveVideo = None

sendMail = 0

socket.send\_string(nameVideo)