TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN - ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - TRUYỀN THÔNG

ĐỀ TÀI:

HỆ THỐNG CẢNH BÁO NHÂN VIÊN NGỦ GẬT

GVHD: TS. TRƯƠNG NGỌC SƠN

SVTH: NGUYỄN VĂN VỸ

MSSV: 15141336

SVTH: LÊ THỊ CẨM THI

MSSV: 15141288

TP. HỒ CHÍ MINH – 06/2019

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ TRUYỀN THÔNG

ĐỀ TÀI:

HỆ THỐNG CẢNH BÁO NHÂN VIÊN NGỦ GẬT

GVHD: TS. TRƯƠNG NGỌC SƠN
SVTH: NGUYỄN VĂN VỸ
MSSV: 15141336
SVTH: LÊ THỊ CẨM THI
MSSV: 15141288

TP. HỒ CHÍ MINH – 06/2019

THÔNG TIN KHÓA LUẬN TỐT NGHIỆP

1. Thông tin sinh viên

Họ và tên sinh viên: Nguyễn Văn Vỹ	MSSV: 15141336
Email: nguyenvanvy112@gmail.com	Điện thoại: 0358121936
Họ và tên sinh viên: Lê Thị Cẩm Thi	MSSV: 15141288
Email: lethicamthii@gmail.com	Điện thoại: 0398791112

2. Thông tin đề tài

- Tên của đề tài: Hệ thống cảnh báo nhân viên ngủ gật.
- Đơn vị quản lý: Bộ môn Kỹ thuật Máy tính - Viễn thông, Khoa Điện - Điện tử, Trường Đại học Sư phạm Kỹ thuật Tp. Hồ Chí Minh.
- Thời gian thực hiện: Từ ngày 18/03/2016 đến ngày 07/06/2016
- Thời gian bảo vệ trước hội đồng: Ngày 20/06/2016

3. Lời cam đoan của sinh viên

Chúng tôi – Nguyễn Văn Vỹ và Lê Thị Cẩm Thi cam đoan KLTN là công trình nghiên cứu của chúng tôi dưới sự hướng dẫn của tiến sĩ Trương Ngọc Sơn. Kết quả công bố trong KLTN là trung thực và không sao chép từ bất kỳ công trình nào khác.

Tp.HCM, ngày ... tháng ... năm 20...
SV thực hiện đồ án
(*Ký và ghi rõ họ tên*)

Nguyễn Văn Vỹ Lê Thị Cẩm Thi

Giảng viên hướng dẫn xác nhận quyển báo cáo đã được chỉnh sửa theo đề nghị được ghi trong biên bản của Hội đồng đánh giá Khóa luận tốt nghiệp.

……

Xác nhận của Bộ Môn

Tp.HCM, ngày ... tháng ... năm 20...
Giáo viên hướng dẫn
(Ký, ghi rõ họ tên và học hàm - học vị)

BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(Dành cho giảng viên hướng dẫn)

Đề tài: **Hệ thống cảnh báo nhân viên ngủ gật**

Sinh viên thực hiện: 1. Nguyễn Văn Vỹ MSSV: 15141336

2. Lê Thị Cẩm Thi MSSV: 15141288

Giảng viên hướng dẫn: Ts. Trương Ngọc Sơn

Nhận xét bao gồm các nội dung sau đây:

1. Tính hợp lý trong cách đặt vấn đề và giải quyết vấn đề; ý nghĩa khoa học và thực tiễn:
Đặt vấn đề rõ ràng, mục tiêu cụ thể; đề tài có tính mới, cấp thiết; đề tài có khả năng ứng dụng, tính sáng tạo.

.....

.....

2. Phương pháp thực hiện/ phân tích/ thiết kế:
Phương pháp hợp lý và tin cậy dựa trên cơ sở lý thuyết; có phân tích và đánh giá phù hợp; có tính mới và tính sáng tạo.

.....

.....

3. Kết quả thực hiện/ phân tích và đánh giá kết quả/ kiểm định thiết kế:
Phù hợp với mục tiêu đề tài; phân tích và đánh giá / kiểm thử thiết kế hợp lý; có tính sáng tạo/ kiểm định chặt chẽ và đảm bảo độ tin cậy.

.....

.....

4. Kết luận và đề xuất:
Kết luận phù hợp với cách đặt vấn đề, đề xuất mang tính cải tiến và thực tiễn; kết luận có đóng góp mới mẻ, đề xuất sáng tạo và thuyết phục.

.....

.....

5. Hình thức trình bày và bố cục báo cáo:
Văn phong nhất quán, bố cục hợp lý, cấu trúc rõ ràng, đúng định dạng mẫu; có tính hấp dẫn, thể hiện năng lực tốt, văn bản trau chuốt.

.....

.....

6. Kỹ năng chuyên nghiệp và tính sáng tạo:
Thể hiện các kỹ năng giao tiếp, kỹ năng làm việc nhóm, và các kỹ năng chuyên nghiệp khác trong việc thực hiện đề tài.

.....

.....

7. Tài liệu trích dẫn
Tính trung thực trong việc trích dẫn tài liệu tham khảo; tính phù hợp của các tài liệu trích dẫn; trích dẫn theo đúng chỉ dẫn APA.

.....

.....

8. Đánh giá về sự trùng lặp của đề tài
Cần khẳng định đề tài có trùng lặp hay không? Nếu có, đề nghị ghi rõ mức độ, tên đề tài, nơi công bố, năm công bố của đề tài đã công bố.

.....

.....

9. Những nhược điểm và thiếu sót, những điểm cần được bổ sung và chỉnh sửa* .....

.....

.....

.....

.....

.....

10. Nhận xét tinh thần, thái độ học tập, nghiên cứu của sinh viên

thái độ tích cực, chăm chỉ

ghi nhận thực hiện tốt đề tài đã đặt ra, nỗ lực học hỏi, nghiên cứu.

Đề nghị của giảng viên hướng dẫn

Ghi rõ: "Báo cáo đạt/ không đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư, và được phép/ không được phép bảo vệ khóa luận tốt nghiệp"

báo cáo đạt, được phép bảo vệ KLTN.

Tp. HCM, ngày 6 tháng 6 năm 20.9

Người nhận xét

(Ký và ghi rõ họ tên)

Trương Ngọc Sơn

* *Giáo viên hướng dẫn có thể ghi tiếp vào mặt sau*

BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(Dùng cho giảng viên phản biện)

Đề tài: **Hệ thống cảnh báo nhân viên ngủ gật**

Sinh viên thực hiện: 1. Nguyễn Văn Vỹ MSSV: 15141336

2. Lê Thị Cẩm Thi MSSV: 15141288

Giảng viên hướng dẫn: Ts. Trương Ngọc Sơn

Nhận xét bao gồm các nội dung sau đây:

1. Tính hợp lý trong cách đặt vấn đề và giải quyết vấn đề; ý nghĩa khoa học và thực tiễn [15/100]:

Đặt vấn đề rõ ràng, mục tiêu cụ thể [5]; đề tài có tính mới, cấp thiết [5]; đề tài có khả năng ứng dụng, tính sáng tạo [5].

.....

.....

2. Phương pháp thực hiện/ phân tích/ thiết kế [25/100]:

Phương pháp hợp lý và tin cậy dựa trên cơ sở lý thuyết [10]; có phân tích và đánh giá phù hợp [10]; có tính mới và tính sáng tạo [5].

.....

.....

3. Kết quả thực hiện/ phân tích và đánh giá kết quả/ kiểm định thiết kế [25/100]:

Phù hợp với mục tiêu [10]; phân tích và đánh giá / kiểm thử thiết kế hợp lý [10]; có tính sáng tạo/ kiểm định chặt chẽ và đảm bảo độ tin cậy [5].

.....

.....

4. Kết luận và đề xuất [10/100]:

Kết luận phù hợp với cách đặt vấn đề, đề xuất mang tính cải tiến và thực tiễn [5]; kết luận có đóng góp mới mẻ, đề xuất sáng tạo và thuyết phục[5].

.....

.....

5. Hình thức trình bày, bố cục và chất lượng báo cáo [15/100]:

Văn phong nhất quán, bố cục hợp lý, cấu trúc rõ ràng, đúng định dạng mẫu [5]; có tính hấp dẫn, thể hiện năng lực tốt, văn bản trau chuốt [15].

.....

.....

6. Tài liệu trích dẫn [10/100]

Tính trung thực trong việc trích dẫn tài liệu tham khảo; tính phù hợp của các tài liệu trích dẫn; trích dẫn theo đúng chỉ dẫn APA.

.....

.....

7. Đánh giá về sự trùng lặp của đề tài

Cần khẳng định đề tài có trùng lặp hay không? Nếu có, đề nghị ghi rõ mức độ, tên đề tài, nơi công bố, năm công bố của đề tài đã công bố.

.....

.....

8. Những nhược điểm và thiếu sót, những điểm cần được bổ sung và chỉnh sửa*

.....

.....

Câu hỏi sinh viên phải trả lời trước hội đồng* *(ít nhất 02 câu)*

.....

.....

Đánh giá chung

- Điểm (Quy về thang điểm 10 không làm tròn): ……. /10.
- Xếp loại chung (Xuất sắc, Giỏi, Khá, Trung bình, Yếu, Kém): …….

Đề nghị của giảng viên phản biện

Ghi rõ: "Báo cáo đạt/ không đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư, và được phép/ không được phép bảo vệ khóa luận tốt nghiệp"

.....

Tp. HCM, ngày ... tháng .... năm 20...

Người nhận xét

(*Ký và ghi rõ họ tên)*

LỜI CẢM ƠN

Trên thực tế không thành công nào mà không gắn liền với sự hỗ trợ, dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian thực hiện đề tài nhóm thực hiện đề tài đã nhận được nhiều sự giúp đỡ, quan tâm từ thầy cô, gia đình và bạn bè. Với lòng biết ơn sâu sắc nhất, nhóm thực hiện xin gửi tới thầy cô ở bộ môn Kỹ thuật Điện tử Viễn thông Trường Đại học Sư phạm Kỹ thuật Tp.HCM đã dùng tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho nhóm trong suốt thời gian thực hiện đề tài cũng như học tập tại trường.

Nhóm thực hiện đề tài xin gửi lời cảm ơn sâu sắc đến Thầy Trương Ngọc Sơn, giảng viên bộ môn Máy tính – Viễn thông đã trực tiếp hướng dẫn và tận tình giúp đỡ nhóm thực hiện tốt đề tài.

Nhóm thực hiện đề tài cũng gửi lời đồng cảm ơn đến các bạn lớp 15141VT2B và 15941VT đã chia sẻ, trao đổi kiến thức, những kinh nghiệm quý báu cũng như luôn động viên tinh thần cho nhóm trong thời gian thực hiện đề tài.

Gửi lời cảm ơn chân thành nhất đến cha mẹ, đã ủng hộ tinh thần cũng như vật chất để nhóm có thể hoàn thành đề tài.

Xin chân thành cảm ơn!

Tp. HCM, ngày 07 tháng 06 năm 2019

Người thực hiện đề tài

Nguyễn Văn Vỹ Lê Thị Cẩm Thi

TÓM TẮT

Với sự phát triển vượt bậc của công nghệ kỹ thuật trong những năm gần đây, đặc biệt là cuộc cách mạng công nghiệp 4.0 bùng nổ thì đời sống của con người dần trở nên tiện nghi hơn khi xung quanh luôn có các thiết bị thông minh hỗ trợ con người trong mọi mặt kể cả sinh hoạt, học tập, giải trí đặc biệt là trong quá trình lao động. Cùng với sự phát triển trên, Artificial Intelligence (AI) đang được xem là một ngành rất có tiềm năng ở tương lai trong việc xây dựng các mô hình, hệ thống thiết bị thông mình phục vụ cho con người.

Trong đề tài này, nhóm thực hiện xây dựng một hệ thống cảnh báo nhân viên ngủ gật bằng mô hình Convolution Neural Network (CNN) là một phần của AI. Để hỗ trợ cho các nhân viên làm việc ở những môi trường yêu cầu sự tập trung cao trong một thời gian dài, chỉ cần sơ suất ngủ gật có thể dẫn đến hậu quả vô cùng nghiêm trọng. Bước đầu, nhóm thực hiện đề tài đi tìm hiểu về thuật toán Adaptive Boost (AdaBoost) cùng với đặc trưng Haar-like và mô hình Cascade cho phép phát hiện khuôn mặt, rút trích ảnh đôi mắt nhanh và chính xác. Sau đó, sẽ đi đến quá trình xây dựng mô hình CNN cho hệ thống cảnh báo nhân viên ngủ gật. Thực hiện cài đặt phần mềm Python và một số thư viện hỗ trợ trong quá trình xây dựng mô hình. Cuối cùng từ kết quả mô phỏng đánh giá và tiến hành chỉnh sửa lại hệ thống để có thể làm việc một cách tốt nhất.

DANH MỤC HÌNH ẢNH

CÁC TỪ VIẾT TẮT

Chữ viết tắt	Chữ tường minh
AdaBoost	Adaptive Boost
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ARM	Acorn RISC Machine
BSD	Berkeley Software Distribution
CBC	Cascade of Boosted Classifiers
CCD	Charge Coupled Device
CNN	Convolution neural network
CNTK	Computational Network Toolkit
Conv	Convolutional layer
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DARPA	Defense Advanced Research Projects Agency
DIY	Do It Yourself
DL	Deep learning
EEPROM	Electrically Erasable Programmable Read-Only Memory
eMMC	embedded Multi-Media Controller
FC	Fully Connected layer
FNN	Feedforward Neural Networks
GPU	Graphics Processing Unit
HD	High Definition
HDMI	High-Definition Multimedia Interface

I2C	Inter-Intergrated Circuit
IBM	International Business Machines
KL	Karhumen Loeve
KNN	K-Nearest Neighbour
LPDDR2	Low Power Double Data Rate 2
ML	Mechine Learning
OEM	Original Equipment Manufacturing
OOP	Object-Oriented Programming
OpenCV	Open Computer Vision
P	Padding
Pool	Pooling layer
PWM	Pulse-width modulation
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RNN	Recurrent Neural Network
S	Stride
SciPy	Science Python
SDIO	Secure Digital Input Output
SDRAM	Synchronous Dynamic Random Access Memory
SoC	System on a chip
SO-DIMM	Small Outline Dual In-line Memory Module
SRAM	Static Random-Access Memory
UART	Universal Asynchronous Receiver – Transmitter
USB	Universal Serial Bus
OpenCL	Open Computing Language

DANH MỤC BẢNG

MỤC LỤC

CHƯƠNG 1

GIỚI THIỆU

1.1 GIỚI THIỆU

Nhận thấy rằng xã hội không ngừng tiến bộ về mọi lĩnh vực, trong đó lĩnh vực khoa học kỹ thuật đang trong thời kỳ phát triển vượt bậc với ảnh hưởng của cuộc cách mạng công nghệ lần thứ tư. Vài năm gần đây lĩnh vực AI thực sự được bùng nổ bởi sự ảnh hưởng và những ứng dụng tuyệt vời được ra đời phục vụ cho mọi mặt đời sống con người.

Tìm hiểu thực tế nhóm thực hiện đề tài nhận thấy, có rất nhiều nhân viên ngủ gật trong giờ làm việc vì nhiều nguyên nhân: mệt mỏi vì làm việc quá sức, đêm qua đi ngủ muộn hoặc do tính chất công việc văn phòng ít di chuyển dẫn đến dễ buồn ngủ. Sẽ ra sao nếu nhân viên ở phòng kiểm soát không lưu ngủ gật? Hậu quả thật sự rất khủng khiếp. Ở Việt Nam, theo báo Người lao động ngày 21/03/2017 đưa tin về sự việc xảy ra vào ngày 09/03/2017, một sự kiện làm cho người dân lo lắng về sự an toàn hàng không khi kiểm soát viên không lưu thuộc đài kiểm soát không lưu sân bay Cát Bi, Hải Phòng đã ngủ gật trong ca làm việc. Cụ thể là: "Trong ca trực chính kiểm soát viên đã ngủ quên từ 21 giờ 40 phút đến 23 giờ 15 phút và không duy trì liên tục canh nghe trên các kênh liên lạc làm cho hoạt động điều hành bay bị gián đoạn suốt hơn 30 phút đồng hồ. Hậu quả là 2 chuyến bay VJ921 và VJ292 thiết lập tổng cộng 39 lần liên lạc nhưng kiểm soát viên không trả lời khiến cho 2 chuyến bay VJ921 cất cánh từ Hải Phòng đi Seoul (Hàn Quốc) và chuyến bay VJ292 từ TP.HCM về Hải Phòng bị trễ do mất liên liên lạc với đài kiểm soát không lưu Cát Bi". Cũng thật may là trong vụ việc trên không xảy ra bất kỳ hậu quả đáng tiếc nào, kiểm soát viên đã chịu khiển trách và tạm ngưng nhiệm vụ 2 tháng.

Ngoài ra, theo một số tài liệu nghiên cứu bí mật của Mỹ thì có tới khoảng 60% kiểm soát viên không lưu thừa nhận họ từng ngủ quên khi phải thực hiện ca trực. Và không chỉ là kiểm soát viên và ngay cả phi công cũng đôi lúc rơi vào trạng thái

ngủ gật trong quá tình điều khiển máy bay. Nhận thấy được sự quan trọng của vấn đề nên nhóm thực hiện đề tài đã quyết định chọn đề tài "Hệ thống cảnh báo nhân viên ngủ gật" để có thể xây dựng một hệ thống ngăn chặn tình trạng ngủ gật trong giờ làm việc của nhân viên, đặc biệt là nhân viên làm việc ở các các lĩnh vực có ảnh hưởng nghiêm trọng đến tính mạng con người cũng như ảnh hưởng đến hoạt động xã hội.

1.2 MỤC TIÊU ĐỀ TÀI

- Tìm hiểu về thuật toán nhận dạng AdaBoost và đặc trưng Haar-like.
- Ứng dụng thuật toán AdaBoost và đặc trưng Haar-like vào nhận dạng khuôn mặt và mắt.
- Tìm hiểu mạng CNN.
- Thiết kế hệ thống cảnh báo nhân viên ngủ gật.

1.3 NỘI DUNG ĐỀ TÀI

Đề tài được thực hiện dựa trên những nội dung sau đây:

- Nội dung 1: Tìm hiểu về các lý thuyết xử lý ảnh, thuật toán nhận dạng AdaBoost và mô hình CNN.
- Nội dung 2: Áp dụng thuật toán AdaBoost và đặc trưng Haar–like xây dựng tập cơ sỡ dữ liệu.
- Nội dung 3: Tiến hành xây dựng mô hình CNN.
- Nội dung 4: Thực hiện quá trình huấn luyện cho mô hình CNN vừa xây dựng và điều chỉnh thông số.
- Nội dung 5: Đánh giá mô hình.
- Nội dung 6: Viết báo cáo.

1.4 GIỚI HẠN ĐỀ TÀI

- Hệ thống phát hiện nhân viên ngủ gật khi nhân viên nhắm cả hai mắt trong khoảng thời gian lớn hơn 3 giây.
- Hệ thống chỉ nhận diện được trạng thái mắt của nhân viên ngồi gần camera của hệ thống nhất nếu camera thu được nhiều khuôn mặt.
- Góc nghiêng mặt của nhân viên so với camera không quá 30°.
- Khoảng cách từ camera đến đối tượng dưới 1 mét, khoảng cách càng xa tỉ lệ chính xác của mô hình càng thấp.
- Hệ thống không xác định được mắt khi nhân viên mang kính râm.
- Hệ thống không phân biệt được khuôn mặt thật và ảnh chụp khuôn mặt.

1.5 BỐ CỤC

- **Chương 1:** Giới thiệu

Trong chương này tìm hiểu về các vấn đề hình thành nên đề tài, những yêu cầu cần thiết nhất đề ra cho hệ thống. Kèm theo đó là một số nội dung và giới hạn của đề tài mà nhóm thực hiện đề tài đã đặt ra.

- **Chương 2:** Nghiên cứu tổng quan

Trong chương này trình bày những kiến thức cơ bản xử lý ảnh, giới thiệu về thuật toán AdaBoost, đặc trưng Haar-like, cái nhìn tổng quan về trí tuệ nhân tạo và cách thức xây dựng một mô hình CNN đơn giản, tìm hiểu về ngôn ngữ Python và ứng dụng, tìm hiểu về Framework Keras.

- **Chương 3**: Thiết kế hệ thống

Từ nghiên cứu tổng quan có được ở chương 2 tiến hành phân tích quá trình xây dựng mô hình CNN cảnh báo nhân viên ngủ gật, thay đổi các thông số liên quan. Phân tích sơ đồ khối và nêu chức năng của từng khối có trong hệ thống. Phân tích các giải thuật và hoàn thiện các khâu còn lại để có một hệ thống cảnh báo nhân viên ngủ gật hoàn chỉnh.

- **Chương 4:** Kết quả nghiên cứu

Sau khi xây dựng hệ thống hoàn chỉnh, nhìn nhận lại kết quả và so sánh lại với mục tiêu ban đầu đã đề ra. Nhận xét về chất lượng của hệ thống, nêu lên ưu điểm và nhược điểm của đề tài.

- **Chương 5:** Kết luận và hướng phát triển

Tóm lại kết quả trong quá trình thực hiện so với mục tiêu đề ra ban đầu. Đề ra hướng phát triển của đề tài để đáp ứng trên thực tế.

CHƯƠNG 2

NGHIÊN CỨU TỔNG QUAN

Phần này sẽ tiến hành đi tìm hiểu một số lý thuyết về kỹ thuật xử lý ảnh, thuật toán AdaBoost, tổng quan về AI, cách thức xây dựng một mô hình CNN đơn giản, giới thiệu về ngôn ngữ lập trình Python cùng các thư viện, tìm hiểu về Keras.

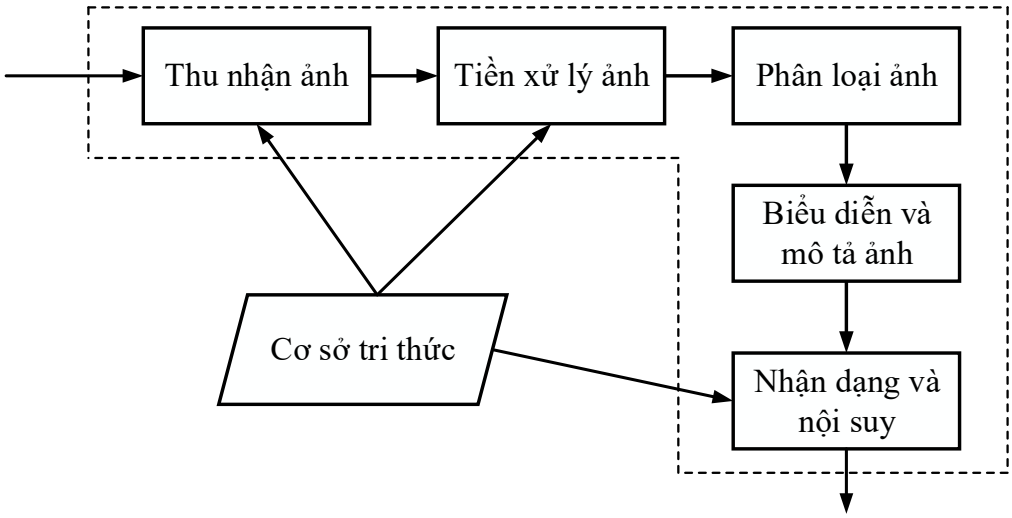
2.1 TỔNG QUAN VỀ XỬ LÝ ẢNH

2.1.1 Khái niệm xử lý ảnh

Xử lý ảnh là một trong những lĩnh vực về khoa học và công nghệ khá mới nhưng tốc độ phát triển của nó rất nhanh, được nhiều trung tâm nghiên cứu, ứng dụng.

Xử lý ảnh được hiểu là "quá trình biến đổi một hình ảnh thành một ảnh khác bằng máy tính điện tử một cách tự động phụ thuộc vào mục đích của người sử dụng" [1]. Để có kiến thức về xử lý ảnh người học phải trang bị cho mình nhiều kiến thức về chuyên môn, nền tảng của xử lý ảnh được xây dựng dựa trên xử lý tín hiệu số. Đây là môn học rất cơ bản về các khái niệm tích chập, các biến đổi Fourier, biến đổi Laplace, các bộ lọc hữu hạn. Ngoài ra cần có kiến thức về toán học như đại số tuyến tính, xác suất thống kê trong quá trình phân tích và nhận dạng ảnh.

Xây dựng một hệ thống xử lý ảnh cần xem xét các bước sau. Đầu tiên, ảnh tự nhiên từ thế giới bên ngoài được thu nhận qua các thiết bị thu (như camera, máy chụp ảnh), thường thì ảnh thu nhập được có thể là ảnh màu hoặc ảnh đen trắng được lấy ra từ camera, sau đó được chuyển trực tiếp thành ảnh số để tạo thuận lợi cho quá trình xử lý tiếp theo sau đó.



Hình 2.1 Các bước cơ bản trong xử lý ảnh [1].

Phân tích sơ đồ này gồm những thành phần chính sau đây:

- **Phần thu nhận ảnh**

Đây là một giai đoạn có ý nghĩa vô cùng quan trọng đối với quá trình xây dựng một hệ thống xử lý ảnh. Ảnh có thể nhận qua camera màu hoặc đen trắng. Thường ảnh nhận qua camera là ảnh tương tự (loại camera ống chuẩn CCIR với tần số 1/25, mỗi ảnh 25 dòng), cũng có loại camera đã số hoá (như loại CCD – Change Coupled Device) là loại photodiot tạo cường độ sáng tại mỗi điểm ảnh. Camera thường dùng là loại quét dòng tạo ra ảnh dạng hai chiều. Chất lượng một ảnh thu nhận được phụ thuộc vào thiết bị thu, vào môi trường (ánh sáng, phong cảnh).

- **Tiền xử lý**

Sau bộ thu nhận, ảnh mà ta thu nhận được có thể sẽ bị nhiễu, có độ tương phản thấp vì thế bắt buộc phải đưa qua bộ tiền xử lý sẽ có chức năng là lọc nhiễu, nâng độ tương phản để làm ảnh rõ hơn, nét hơn.

- **Phân vùng ảnh**

Phân vùng ảnh là tách một ảnh đầu vào thành các vùng thành phần để biểu diễn phân tích, nhận dạng ảnh. Ví dụ: để nhận dạng chữ (hoặc mã vạch) trên phong bì thư cho mục đích phân loại bưu phẩm, cần chia các câu, chữ về địa chỉ hoặc tên người thành các từ, các chữ, các số (hoặc các vạch) riêng biệt để nhận dạng. Đây là phần phức tạp khó khăn nhất trong xử lý ảnh và cũng dễ gây lỗi, làm mất độ chính xác của ảnh. Kết quả nhận dạng ảnh phụ thuộc rất nhiều vào công đoạn này.

Hình 2.2 Phân vùng ảnh miền có dấu vân tay thật sự.

- **Biểu diễn ảnh số**

Đầu ra ảnh sau phân đoạn chứa các điểm ảnh của vùng ảnh cộng với mã liên kết với các vùng lận cận. Việc biến đổi các số liệu này thành dạng thích hợp là cần thiết cho xử lý tiếp theo bằng máy tính. Việc chọn các tính chất để thể hiện ảnh gọi là trích chọn đặc trưng (Feature Selection) gắn với việc tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc làm cơ sở để phân biệt lớp đối tượng này với đối tượng khác trong phạm vi ảnh nhận được. Ví dụ: trong nhận dạng ký tự trên phong bì thư, chúng ta miêu tả các đặc trưng của từng ký tự giúp phân biệt ký tự này với ký tự khác.

- **Nhận dạng và nội suy**

Nhận dạng ảnh là quá trình xác định ảnh. Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được học (hoặc lưu) từ trước. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng. Ví dụ: một loạt chữ số và nét gạch ngang trên phong bì thư có thể được nội suy thành mã điện thoại. Có nhiều cách phân loại ảnh khác nhau về ảnh. Theo lý thuyết về nhận dạng, các mô hình toán học về ảnh được phân theo hai loại nhận dạng ảnh cơ bản:

+ Nhận dạng theo tham số.

+ Nhận dạng theo cấu trúc.

Một số đối tượng nhận dạng khá phổ biến hiện nay đang được áp dụng trong khoa học và công nghệ là: nhận dạng ký tự (chữ in, chữ viết tay, chữ ký điện tử), nhận dạng văn bản, nhận dạng vân tay, nhận dạng mã vạch, nhận dạng mặt người…

- **Cơ sở tri thức**

Như đã nói ở trên, ảnh là một đối tượng khá phức tạp về đường nét, độ sáng tối, dung lượng điểm ảnh, môi trường để thu ảnh phong phú kèm theo đó là nhiễu. Trong nhiều khâu xử lý và phân tích ảnh ngoài việc đơn giản hóa các phương pháp toán học đảm bảo tiện lợi cho xử lý, người ta mong muốn bắt chước quy trình tiếp nhận và xử lý ảnh theo cách của con người. Trong các bước xử lý đó, nhiều khâu hiện nay đã xử lý theo các phương pháp trí tuệ con người. Vì vậy, ở đây các cơ sở tri thức được phát huy.

- **Biểu diễn ảnh**

Ảnh sau khi số hoá sẽ được lưu vào bộ nhớ, hoặc chuyển sang các khâu tiếp theo để phân tích. Nếu lưu trữ ảnh trực tiếp từ các ảnh thô, đòi hỏi dung lượng bộ nhớ cực lớn và không hiệu quả theo quan điểm ứng dụng và công nghệ. Thông thường, các ảnh thô đó được đặc tả (biểu diễn) lại (hay đơn giản là mã hoá) theo các đặc điểm của ảnh được gọi là các đặc trưng ảnh (Image Features) như: biên ảnh (Boundary), vùng ảnh (Region).

Một số phương pháp biểu diễn thường dùng:

+ Biểu diễn bằng mã chạy (Run-Length Code)

+ Biểu diễn bằng mã xích (Chaine -Code)

+ Biểu diễn bằng mã tứ phân (Quad-Tree Code)

2.1.2 Một số ứng dụng của xử lý ảnh

Các ứng dụng cơ bản của xử lý ảnh đã hỗ trợ con người rất nhiều trên từng lĩnh vực và đem lại không ít những cái nhìn tích cực về nó. Sau đây là một số ứng dụng tiêu biểu:

- Khôi phục hình ảnh, chỉnh sửa, điều chỉnh độ phân giải.
- Trong lĩnh vực y tế (chụp X-ray, siêu âm màu...).
- Thị giác máy tính và robot.
- Lĩnh vực nhận dạng.
- Trong do thám, thám hiển và quân sự.

2.2 GIỚI THIỆU VỀ CÁC VẤN ĐỀ TRONG XỬ LÝ ẢNH

2.2.1 Điểm ảnh và ảnh số

Ảnh số được cấu tạo từ một tập hợp nhiều điểm ảnh, mỗi điểm ảnh được gọi là một pixel. Điểm ảnh được hiểu là một phần tử của ảnh số tại tọa độ (x, y), được xem như các phần tử có cấu trúc hạt trên một ảnh thông thường nhưng được sắp xếp theo từng hàng và cột chứa các thông tin khác nhau như: biểu diễn một màu sắc nhất định hay độ sáng đối với ảnh trắng đen. Bằng phương pháp đo lường và thống kê một lượng lớn các điểm ảnh, chúng ta có thể tái cấu trúc các điểm ảnh này thành một ảnh mới gần giống với ảnh gốc.

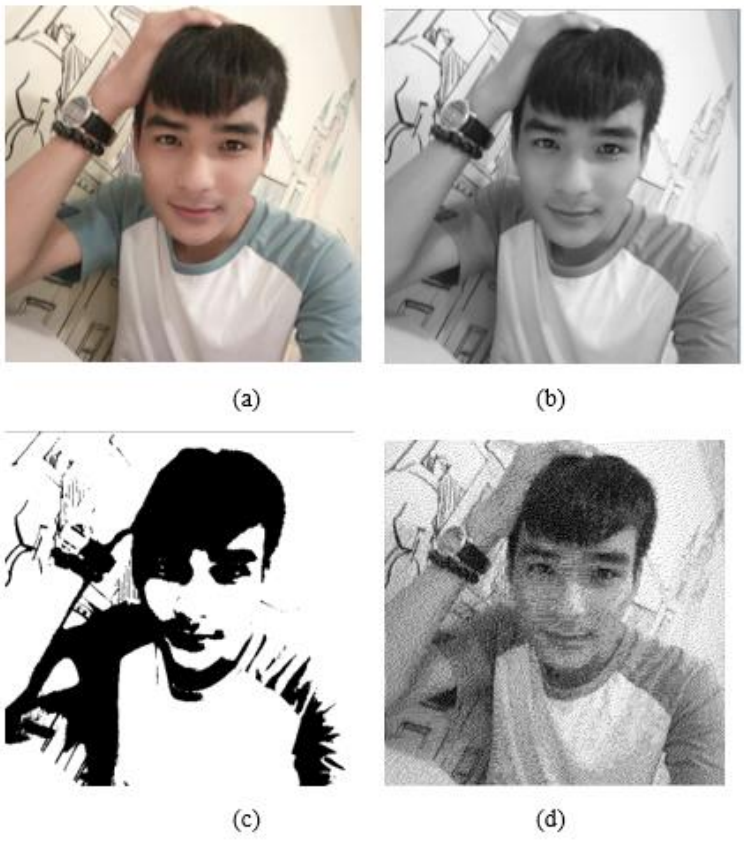
2.2.2 Phân loại ảnh

Có bốn dạng quan trọng trong ảnh số được dùng với nhiều mục đích nhất định:

- Ảnh xám: được xây dựng từ nhiều pixel mà tại đó biểu diễn một giá trị nhất định tương ứng với một mức xám trải dài từ đen sang trắng với bước nhảy mịn, thường là 256 mức xám khác nhau.
- Ảnh màu: được xây dựng từ nhiều pixel mà trong đó mỗi pixel được biểu diễn bởi ba giá trị tương ứng với các mức trong các kênh màu đỏ (Red), xanh lá

(Green) và xanh dương (Blue) trong không gian màu Red Green Blue (RGB) tại một vị trí cụ thể. Mỗi kênh màu có 256 mức, nên từ không gian màu RGB có thể tạo ra 256 x 256 x 256 = 16777216 màu khác nhau bằng phương pháp pha trộn. Từ đó có thể thấy rằng một pixel ảnh xám chỉ cần một byte cho việc lưu trữ, còn ảnh màu cần đến ba byte cho việc lưu trữ trong trường hợp không xử dụng các kỹ thuật nén ảnh.

- Ảnh nhị phân: chỉ sử dụng duy nhất một bit để biểu diễn một pixel. Do một bit chỉ có thể xác định hai trạng thái 0 và 1 tương ứng với hai màu đen và trắng.
- Ảnh chỉ số (indexed): một vài tấm màu (hay xám) được tạo thành từ một bảng màu có sẵn bị giới hạn, điển hình thường dùng là tập 256 màu khác nhau. Những ảnh này được gọi là ảnh màu chỉ số hóa (indexed) do dữ liệu dành cho mỗi pixel bao gồm chỉ số có sẵn chỉ rõ màu trong tập có sẵn ứng với pixel đang xét.



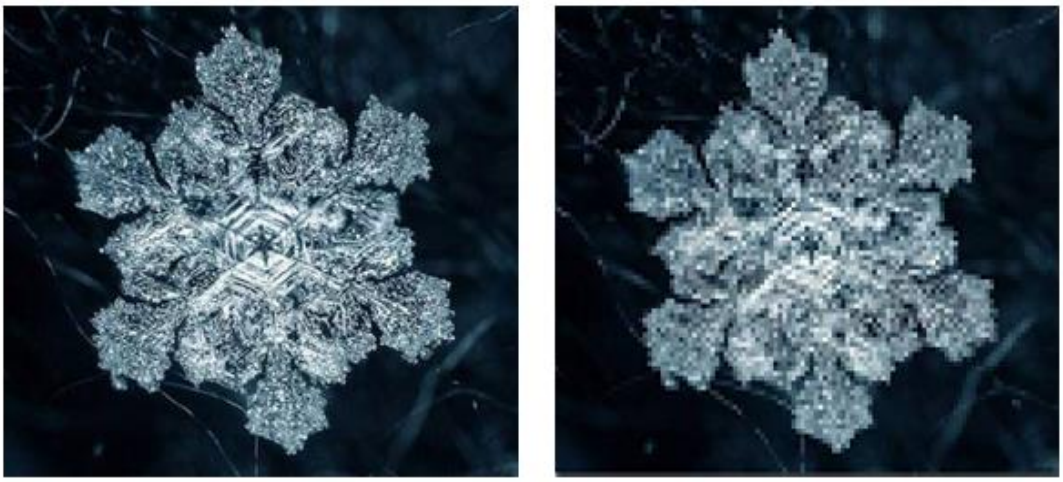
Hình 2.3 Các loại ảnh khác nhau.

(a) ảnh màu, (b) ảnh xám, (c) ảnh nhị phân, (d) ảnh chỉ số.

2.2.3 Độ phân giải của ảnh

"Độ phân giải của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị" [2]. Khoảng cách giữa các điểm ảnh sao cho mắt người vẫn thấy được sự liên tục của ảnh. Độ phân giải được phân bố theo trục x và y trong không gian hai chiều. Với cùng một ảnh, độ phân giải càng cao thì ảnh càng chứa nhiều thông tin và sắc nét hơn.

Ví dụ như hình bên dưới.



Hình 2.4 Độ phân giải của 2 ảnh khác nhau.

Ở hình (a) có độ phân giải là 500x350 pixels, hình (b) có độ phân giải là 117x100 pixels.

Kết quả trên cho thấy rằng, với độ phân giải càng cao thì độ sắc nét của ảnh càng cao và chất lượng hình ảnh càng rõ nét.

2.2.4 Mức xám

Một điểm ảnh có hai đặc trưng cơ bản đó chính là vị trí (x, y) của điểm ảnh và độ xám của ảnh. Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại thời điểm đó. Thông thường thì giá trị của mức xám được biểu diễn bằng 256 mức từ 0 đến 255 và mức xám sử dụng cho mỗi loại ảnh thì khác nhau.

(a) (b)

Hình 2.5 Sự khác nhau giữa ảnh màu và ảnh xám.

Với hình (a) là ảnh màu và hình (b) là ảnh xám.

2.2.5 Biến đổi ảnh

Trong xử lý ảnh do số điểm ảnh lớn hơn, các tính toán nhiều dẫn đến độ phức tạp tính toán cao nên đòi hỏi dung lượng bộ nhớ lớn, thời gian tính toán lâu. Các phương pháp khoa học kinh điển áp dụng cho xử lý ảnh hầu như khó khả thi. Người ta sử dụng các phép toán tương đương hoặc biến đổi sang miền xử lý khác để dễ tính toán, sau khi đã xử lý dễ dàng, dùng biến đổi ngược để đưa về miền xác định ban đầu, các biến đổi thường gặp trong xử lý ảnh bao gồm:

- Biến đổi Fourier, Cosin, Sin.
- Biến đổi ảnh bằng tích chập, tích Kronecker.
- Các biển đổi khác như KL (Karhumen Loeve), Hadamard.

2.3 ĐẶC TRƯNG HAAR-LIKE VỀ NHẬN DIỆN KHUÔN MẶT

2.3.1 Giới thiệu đặc trưng Haar-like

Vào năm 2001, Paul Viola và Michael Jones đã đưa ra một phương pháp phát hiện đối tượng rất hiệu quả sử dụng phân loại tầng dựa trên tính năng Haar-like trong bài báo của họ, "Phát hiện đối tượng nhanh bằng cách sử dụng một tính năng đơn giản của Boosted" [3].

Đặc trưng Haar-like được sử dụng trong việc nhận dạng đối tượng trong ảnh số và gồm 4 đặc trưng cơ bản để xác định một đối tượng trong ảnh. Mỗi đặc trưng Haar-like là sự kết hợp gồm 2 hoặc 3 khối chữ nhật mang giá trị "đen" hoặc "trắng". Những khối chữ nhật này thể hiện sự liên hệ tương quan giữa các bộ phận trong ảnh mà bản thân từng giá trị pixel không thể diễn đạt được.

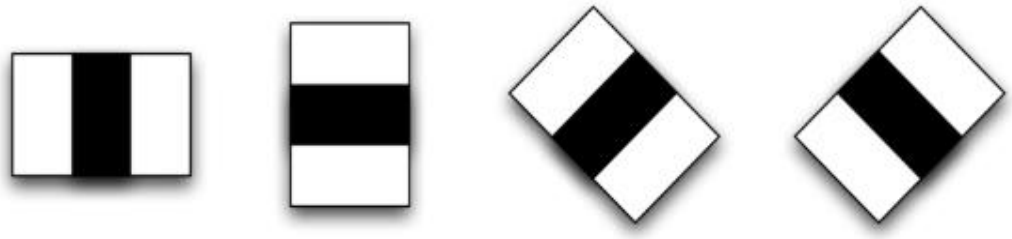
Hình 2.6 Bốn đặc trưng cơ bản nhất của Haar-like.

Để vận dụng vào việc xác định đối tượng thì bốn đặc trưng Haar-like cơ bản trên được mở rộng và được chia làm các nhóm đặc trưng sau:

- Đặc trưng cạnh

Hình 2.7 Đặc trưng cạnh trong Haar-like.

- Đặc trưng đường



Hình 2.8 Đặc trưng đường trong Haar-like.

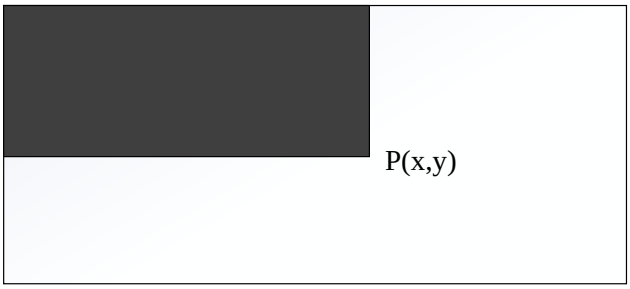
- Đặc trưng xung quanh tâm

Hình 2.9 Đặc trưng xung quanh tâm.

Dùng các đặc trưng trên, có thể tính được các giá trị của đặc trưng Haar-like là sự chênh lệch giữa tổng của các pixel của vùng đen và vùng trắng như trong công thức sau:

$$F(x) = \text{Tổng}_{\text{vùng đen}}(\text{pixel}) - \text{Tổng}_{\text{vùng trắng}}(\text{pixel}). \quad (2.1)$$

Như vậy để tính giá trị đặc trưng Haar-like cần phải thực hiện tính toán tổng các vùng pixel trên ảnh. Điều này làm cho chi phí bài toán lớn không thể đáp ứng các tính năng yêu cầu thời gian thực. Do vậy Viola và Jones đã đề xuất ra khái niệm "Integral Image" để giảm thiểu chi phí cho bài toán tính giá trị của đặc trưng Haar-like để bài toán có thể xử lý với thời gian thực. Tính "Integral Image" bằng cách sử dụng mảng 2 chiều với kích thước bằng kích thước của ảnh cần tính giá trị đặc trưng Haar-like. Ảnh chia nhỏ ở vị trí (x, y) được tính bằng tổng các giá trị pixel của vùng từ vị trí (0,0) đến vị trí (x-1, y-1). Việc tính toán đơn giản là thực hiện phép cộng số nguyên nên tốc độ thực hiện được tối ưu hóa.

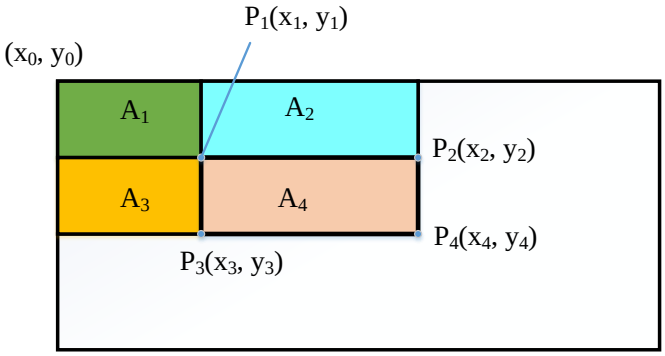


Hình 2.10 Cách tính Integral Image của ảnh.

Công thức tính Integral Image

$$P(x, y) = \sum_{x' \le x,\ y' \le y} i(x', y') \quad (2.2)$$

Kết quả có được sau khi tính Integral Image, việc tính tổng giá trị pixel trong vùng cần tính thực hiện như sau:



Hình 2.11 Cách tính tổng giá trị pixel vùng cần tính.

Gọi vùng cần tính tổng các giá trị pixel là vùng "A4".

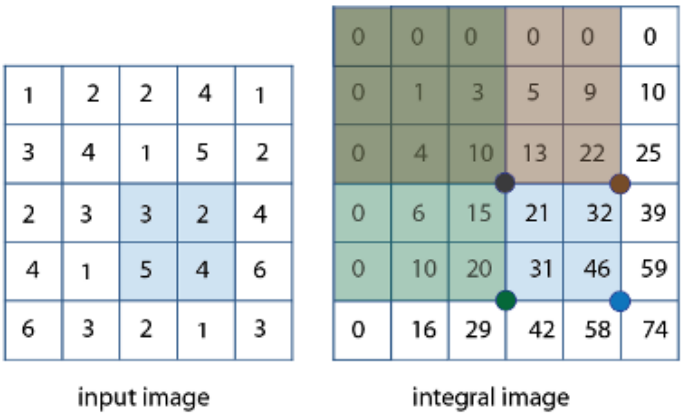
$$P_1(x_1, y_1) = A_1 \quad (2.3)$$

$$P_2(x_2, y_2) = A_1 + A_2 \quad (2.4)$$

$$P_3(x_3, y_3) = A_1 + A_2 + A_3 \quad (2.5)$$

$$P_4(x_4, y_3) = A_1 + A_2 + A_3 + A_4 \quad (2.6)$$

Ta được: $A_4 = P_4 + P_1 - P_2 - P_3$ (2.7)



Hình 2.12 Tạo Integral Image từ ảnh đầu vào.

2.3.2 Thuật toán Adaptive Boost

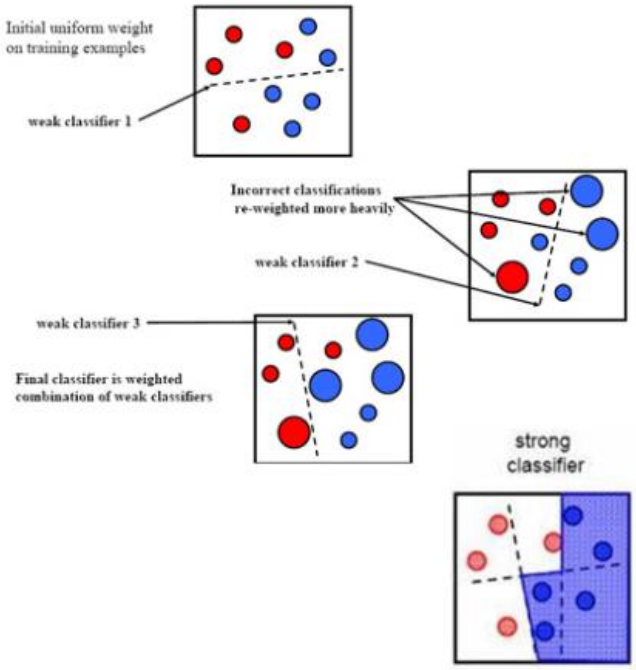
"Adaptive Boost (AdaBoost) là một bộ phân lớp mạnh phi tuyến phức dựa trên hướng tiếp cận Boosting được Freund và Schapire đưa ra vào năm 1995. AdaBoost cũng hoạt động trên nguyên tắc kết hợp tuyến tính các "weak classifiers" để hình thành một "strong classifier". Trước khi tìm hiểu kỹ về thuật toán học máy AdaBoost, cùng tìm hiểu về thuật toán Boosting" [4].

2.3.2.1 Tiếp cận Boosting

Boosting bắt nguồn từ câu hỏi nổi tiếng được đưa ra bởi Kears vào năm 1989: "Liệu có thể tạo ra một strong classifier từ một tập các weak classifiers?". Thì năm 1990, ông Robert Schapire đưa ra thuật toán Boosting đầu tiên, tiếp đến năm 1993 thì nó được Drucker, Schapire và và Simard kiểm nghiệm trong các chương trình nhận dạng. Thuật toán này không dừng lại ở đó, Freund đã tiếp tục về các nghiên cứu của Schapire và đến năm 1995 thì ông cùng với Schapire phát triển Boosting thành thuật toán máy học AdaBoost.

Nguyên lý cơ bản của thuật toán Boosting là sự kết hợp các weak classifiers thành một strong classifiers. Trong quá trình huấn luyện, cứ mỗi weak classifier được xây dựng, thuật toán sẽ tiến hành cập nhật lại trọng số để chuẩn bị cho việc xây dựng weak classifier kế tiếp: tăng trọng số của các mẫu bị nhận dạng sai và giảm trọng số của các mẫu được nhận dạng đúng bởi weak classifier vừa xây dựng. Bằng cách này weak classifer sau có thể tập trung vào các mẫu mà các weak

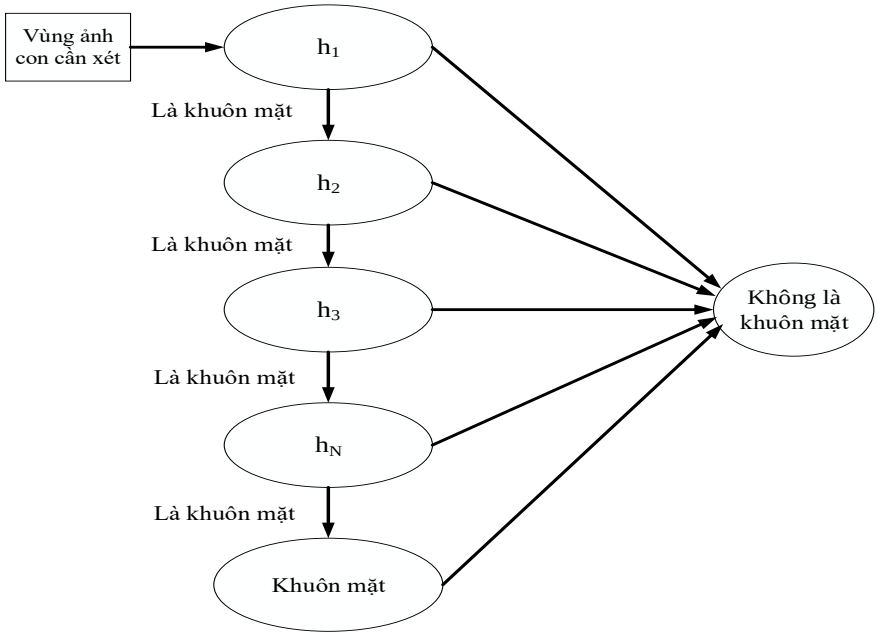
classifiers trước nó làm chưa tốt. Sau cùng, các weak classifers sẽ được kết hợp tùy theo mức độ tốt của chúng để tạo nên strong classifier.



Hình 2.13 Kết hợp các weak classifier thành strong classifier Boosting.

2.3.2.2 Thuật toán AdaBoost

Viola và Jones dùng AdaBoost kết hợp các weak classifier sử dụng các đặc trưng Haar-like theo mô hình Cascade như sau:



Hình 2.14 Mô hình phân tầng kết hợp với bộ phân loại yếu [5].

Trong đó h_k là các bộ phân loại yếu được biểu diễn như sau:

$$h_k(x) = \begin{cases} 1 \text{ nếu } pkfk(x) < pk\theta k \\ 0 \text{ nếu ngược lại} \end{cases} \quad (2.8)$$

x: cửa sổ con cần xét.

θ_k: mức ngưỡng.

f_k: giá trị của đặc trưng Haar-like.

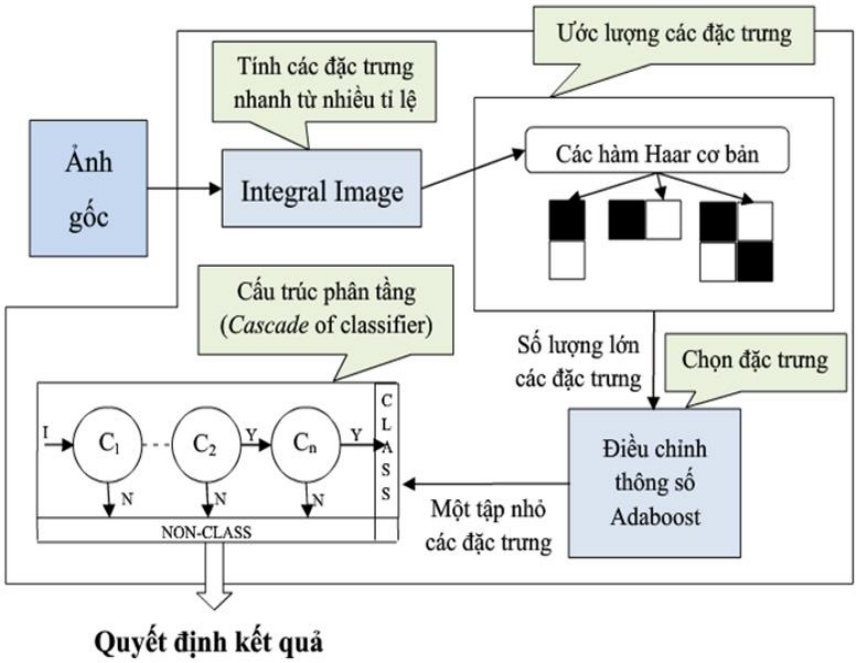
p_k: hệ số quyết định chiều của phương trình.

AdaBoost sẽ kết hợp các weak classifier thành strong classifier như sau:

$$H(x) = \text{sign}(a_1h_1(x) + a_2h_2(x) + ... + a_nh_n(x) \text{ (a = alpha)} \quad (2.9)$$

Với: $a_t >= 0$ là hệ số chuẩn hoá cho các weak classifier.

2.3.3 Hệ thống phát hiện vị trí khuôn mặt



Hình 2.15 Hệ thống xác định vị trí khuôn mặt người [6].

"Từ hệ thống trên, có ảnh gốc ban đầu đưa vào, thực hiện tính các đặc trưng nhanh từ nhiều tỉ lệ (Integral Image), là mảng 2 chiều với phần tử (x, y) sẽ được tính bằng tổng của các phần tử (x', y') với x' < x và y' < y, mục đích là để tính nhanh tổng của các giá trị mức xám của một vùng hình chữ nhật bất kỳ trên ảnh gốc. Các vùng ảnh con này sẽ được đưa qua các hàm Haar-like cơ bản để ước lượng đặc trưng, kết quả ước lượng sẽ được đưa qua bộ điều chỉnh AdaBoost để

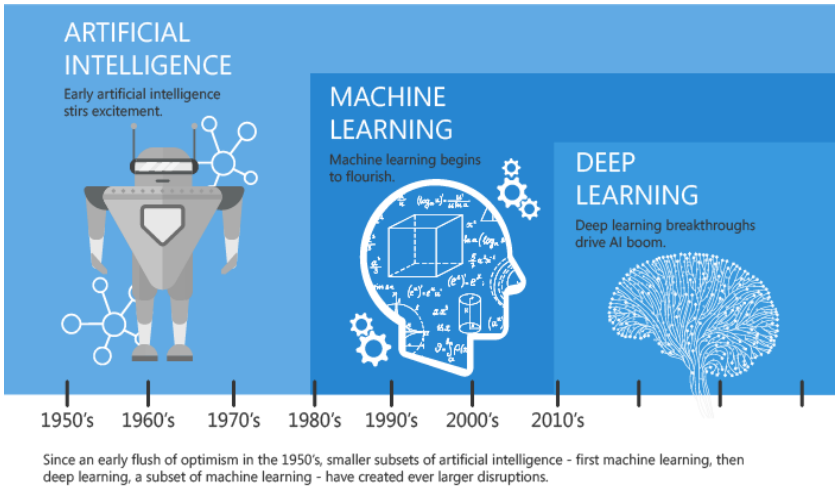
loại bỏ nhanh các đặc trưng không có khả năng là đặc trưng của khuôn mặt người. Chỉ có một tập nhỏ các đặc trưng mà bộ điều chỉnh AdaBoost cho là có khả năng là đặc trưng của khuôn mặt người mới được chuyển sang cho bộ quyết định kết quả. Bộ quyết định sẽ tổng hợp kết quả là khuôn mặt người nếu kết quả của các weak classifer trả về là khuôn mặt người" [7].

Weak classifer sẽ quyết định kết quả cho một đặc trưng Haar – like, được xác định ngưỡng đủ nhỏ sao cho có thể vượt được tất cả các bộ dữ liệu mẫu trong tập dữ liệu huấn luyện (số lượng ảnh khuôn mặt trong tập huấn luyện có thể rất lớn). Trong quá trình xác định khuôn mặt người, mỗi vùng ảnh con sẽ được kiểm tra với các đặc trưng trong chuỗi các đặc trưng Haar-like, nếu có một đặc trưng Haar-like nào cho ra kết quả là khuôn mặt người thì các đặc trưng khác không cần xét nữa. Thứ tự xét các đặc trưng trong chuỗi các đặc trưng Haar-like sẽ được dựa vào trọng số (weight) của đặc trưng đó do AdaBoost quyết định dựa vào số lần và thứ tự xuất hiện của các đặc trưng Haar-like.

2.4 GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO.

2.4.1 Khái niệm về Artificial Intelligence

Artificial Intelligence (AI) được dịch là trí tuệ nhân tạo, được xem như một nhánh của khoa học máy tính liên quan đến sự tự động hóa về hành vi thông minh. Trí tuệ nhân tạo là nghiên cứu về cách làm cho máy tính làm những việc mà hiện tại con người làm rất tốt. Mặt khác, trí tuệ nhân tạo cũng sử dụng một số nền tảng của lĩnh vực xử lý ảnh để phát triển nhằm mục đích phục vụ.



Hình 2.16 Minh họa về các lĩnh vực bên trong của trí tuệ nhân tạo.

AI được chia làm 2 loại là general AI và narrow AI:

- General AI gồm những đặc tính não bộ của con người.
- Narrow AI tức là máy chỉ có một khả năng duy nhất của não bộ người như nhận dạng được hình ảnh.

"Mechine learning (ML) là một tập hợp con của AI sử dụng các phương pháp thống kê để cho phép các máy cải thiện trải nghiệm. Nó cho phép một máy tính hành động và đưa ra các quyết định dựa trên dữ liệu để thực hiện một nhiệm vụ nhất định. Các chương trình hoặc thuật toán này được thiết kế theo cách mà chúng có thể học hỏi và cải thiện theo thời gian khi tiếp xúc với dữ liệu mới" [8].

Deep learning (DL) là một lĩnh vực của ML liên quan đến các thuật toán lấy cảm hứng từ cấu trúc và chức năng của bộ não được gọi Artificial Neural Network (ANN), được dịch là mạng thần kinh nhân tạo. Đó là một loại máy học đặc biệt được lấy cảm hứng từ chức năng của các tế bào não gọi là tế bào thần kinh dẫn đến khái niệm mạng lưới thần kinh nhân tạo. ANN được mô hình hóa bằng cách sử dụng các lớp tế bào thần kinh nhân tạo hoặc các đơn vị tính toán để nhận đầu vào và áp dụng chức năng kích hoạt cùng với ngưỡng.

2.4.2 Một số ứng dụng của AI

Với sự ra đời của AI đã làm ra những sản phẩm phục vụ nhu cầu của con người ngày càng tiên tiến hơn. Sau đây là một số ứng dụng tiêu biểu của AI tính đến thời điểm hiện tại:

- Trò chơi và các bài toán đố.
- Suy luận và chứng minh định lý tự động.
- Xử lý ngôn ngữ tự nhiên.
- Xe ô tô tự lái.
- Robot tự phẫu thuật.

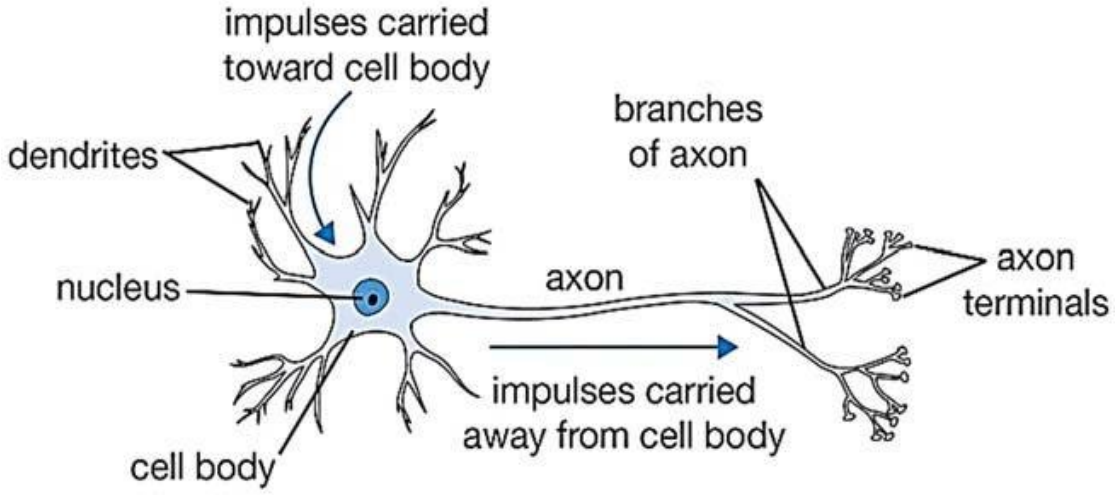
2.5 GIỚI THIỆU MÔ HÌNH CNN

2.5.1 Tìm hiểu về mạng neuron sinh học và mạng neuron nhân tạo

2.5.1.1 Giới thiệu về mạng neuron sinh học

Neuron là một đơn vị cơ bản nhất cấu tạo nên hệ thần kinh của con người và là một phần rất quan trọng đối với não. "Cấu tạo một neuron điển hình gồm: thân chứa nhân, từ thân phát đi nhiều tua ngắn phân nhánh và một tua dài gọi là sợi trục. Tận cùng sợi trục là các đầu mút" [9]. Chức năng cơ bản của neuron là cảm ứng và truyền dẫn xung thần kinh dưới dạng các tín hiệu hóa học trong các hoạt động điều khiển, điều hoà và phối hợp mọi hoạt động của các cơ quan và hệ cơ quan trong cơ thể nhằm tạo sự thích nghi với các thay đổi từ môi trường bên trong và bên ngoài.

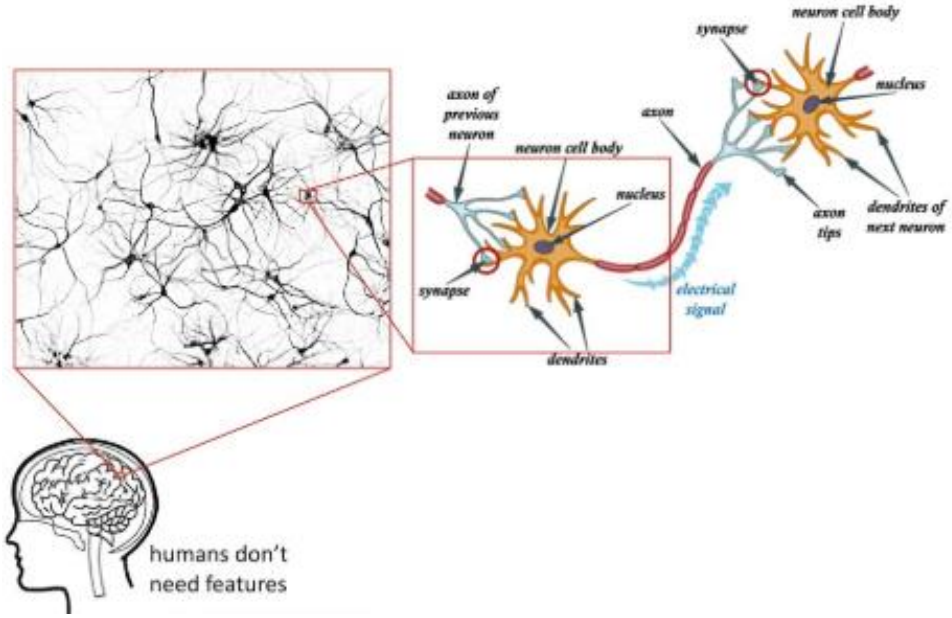
Nguyên lý hoạt động của neuron: tiếp nhận được tín hiệu kích thích từ mô trường bên ngoài từ các đầu mút, khi tín hiệu vượt quá một ngưỡng thì tín hiệu sẽ được dẫn truyền thông tin đến trung tâm sau đó trung tâm xử lý thông tin và các neuron khác theo sợn trục (axon) dẫn truyền thông tin phản hồi đến các cơ quan.



Hình 2.17 Mô tả về cấu tạo của neuron.

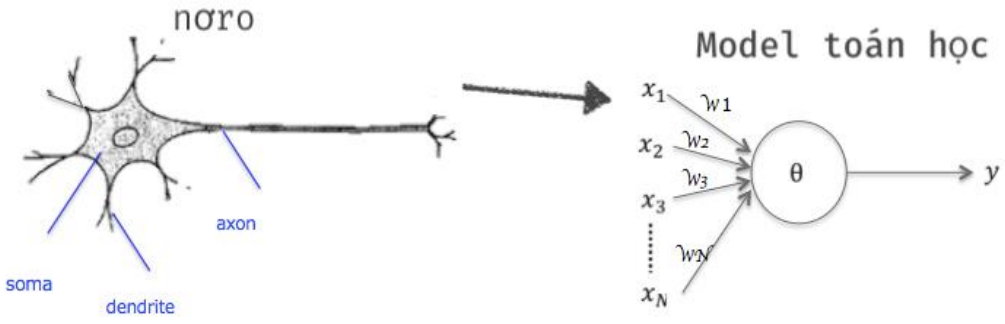
Mạng neuron sinh học là một mạng lưới các neuron có kết nối hoặc có liên quan về mặt chức năng trực thuộc hệ thần kinh ngoại biên hay hệ thần kinh trung ương.

Trong ngành thần kinh học, nó thường được dùng để chỉ một nhóm neuron thuộc hệ thần kinh là đối tượng của một nghiên cứu khoa học nhất định.



Hình 2.18 Mạng neuron sinh học.

2.5.1.2 Giới thiệu về mạng neuron nhân tạo



Hình 2.19 Xây dựng neuron nhân tạo.

Mạng neuron nhân tạo là sự mô phỏng toán học của mạng neuron sinh học. Một mạng neuron nhân tạo được xây dựng từ những thành phần cơ sở là những neuron nhân tạo gồm nhiều đầu vào và một đầu ra. Các đầu vào tiếp nhận kích thích từ đầu ra của những neuron khác hoặc từ môi trường. Mỗi neuron vào có một bộ trọng số nhằm khuếch đại tín hiệu kích thích sau đó tất cả sẽ được cộng lại. Tín hiệu sau đó sẽ được tiếp tục biến đổi nhờ một hàm phi tuyến, thường gọi là hàm kích hoạt.

Và cuối cùng tín hiệu sẽ được đưa đến đầu ra của neuron để lại trở thành đầu vào của các neuron khác hoặc trở thành tín hiệu ra của toàn bộ mạng. Khi kết hợp các neuron lại với nhau sẽ cho ra một mạng neuron nhân tạo.

Công thức tính output y sẽ như sau:

$$Y = f(w_1*x_1 + w_2*x_2 + w_3*x_3 - \theta) \quad (2.10)$$

Với: f được gọi là hàm truyền.

θ là ngưỡng kích hoạt của neuron.

y là tín hiệu đầu ra của một neuron.

Sau khi xây dựng được neuron nhân tạo thì cũng có thể xây dựng được một mô hình mạng neuron nhân tạo dựa vào sự kết nối của các neuron.

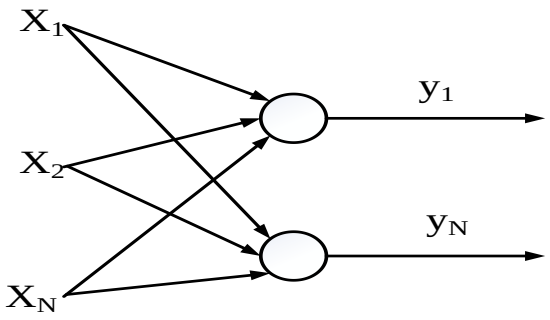
2.5.2 Phân loại Neural Network

Khả năng học, ghi nhớ và khái quát hóa dữ liệu từ các tập huấn luyện làm cho ANN trở thành một phát minh đầy hứa hẹn trong hệ thống thông tin. Để phân loại Neural Network thì xem xét đến cấu trúc hay sự liên kết giữa các lớp trong mạng với nhau, từ đó phân chia thành các nhóm khác nhau.

2.5.2.1 Phân loại theo cấu trúc

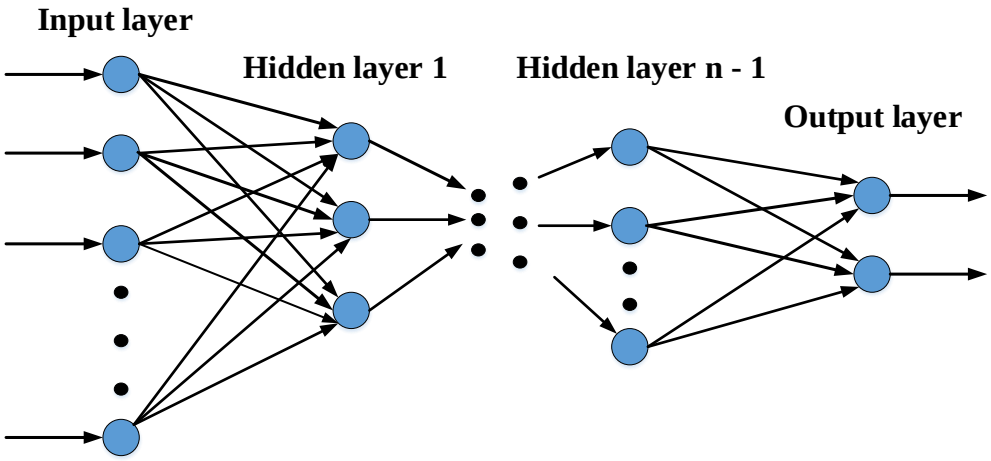
Phân loại theo cấu trúc được chia như sau**:** Neural Network một lớp, Neural Network nhiều lớp.

- Phân loại Neural Network một lớp: Neural Network một lớp cấu thành từ một lớp neuron, nó vừa là lớp vào vừa là lớp ra.



Hình 2.20 Mạng Neural Network một lớp.

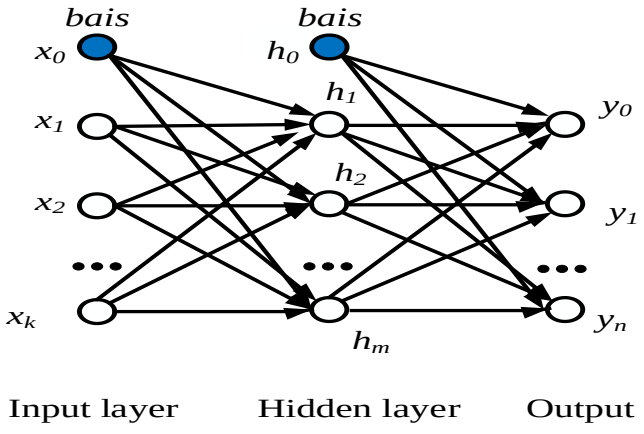
- Phân loại Neural Network nhiều lớp: tổng quát có n lớp (với n > 2), trong đó gồm lớp nhận tín hiệu đầu vào được gọi là lớp đầu vào (input layer). Các tín hiệu đầu ra của mạng được sản sinh bởi lớp ngõ ra của mạng (output layer) lớp thứ n. Các lớp nằm giữa lớp vào và lớp ra được gọi là lớp ẩn (hidden layer) – có (n-1) lớp ẩn (thông thường lớp đầu tiên chỉ có tác dụng chuyển tín hiệu vào lớp tiếp theo).



Hình 2.21 Neural Network nhiều lớp.

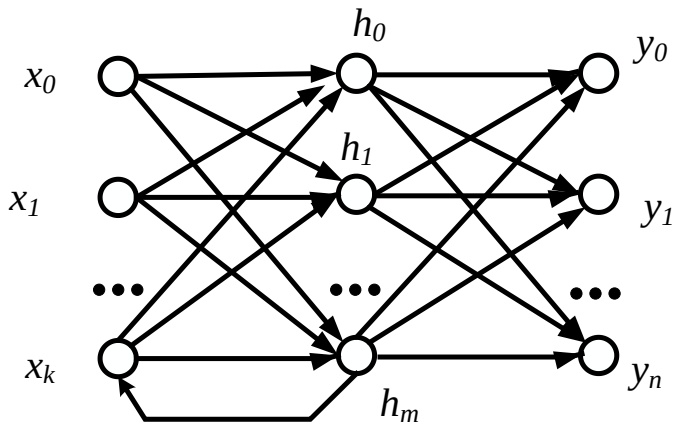
2.5.2.2 Phân loại dựa theo cách thức liên kết

Mạng truyền thẳng - Feedforward Neural Networks: Dòng dữ liệu từ đơn vị đầu vào đến đơn vị đầu ra chỉ được truyền thẳng. Việc xử lý dữ liệu có thể trên nhiều lớp, nhưng không có những liên kết ngược. Cụ thể hơn là không có các liên kết từ các neuron từ các lớp đầu vào và các neuron ở các lớp đầu ra hay các neuron trong cùng một lớp cũng không có liên kết với nhau.



Hình 2.22 Mô hình Neural Network truyền thẳng nhiều lớp.

Mạng hồi quy - Recurrent Neural Network: Có chứa các liên kết ngược. Khác với mạng truyền thẳng, mạng hồi quy chứa các liên kết ngược có sự kết nối giữa neuron đầu ra với neuron đầu vào. Mạng lưu lại các trạng thái trước đó, và trạng thái tiếp theo không chỉ phụ thuộc vào các tín hiệu đầu vào mà còn phụ thuộc vào các trạng thái trước đó của mạng.



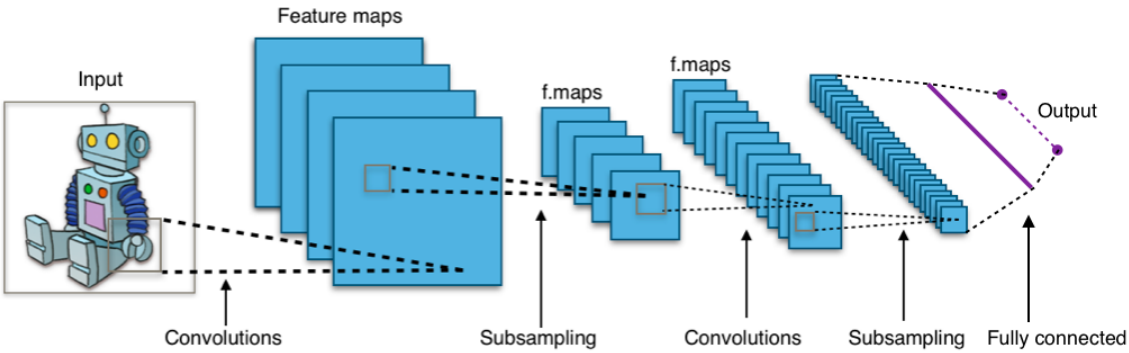
Hình 2.23 Mô hình Neural Network hồi quy.

2.5.3 Giới thiệu Convolutional Neural Network

2.5.3.1 Khái niệm Convolutional Neural Network

"Convolutional Neural Network (CNN) được dịch là "mạng thần kinh tích chập" là một loại mạng thần kinh nhân tạo đã được chứng minh rất hiệu quả trong các lĩnh vực nhận dạng hình ảnh. Do đó, trong hầu hết các trường hợp CNN được áp dụng để xử lý hình ảnh. Đó là một thuật toán Deep Learning, trong đó CNN lấy đầu vào là hình ảnh và đặt trọng số (weight) và độ lệch (bias) một cách hiệu quả cho ảnh và cuối cùng có thể phân biệt hình ảnh với nhau" [10]. Về cơ bản CNN là một kiểu mạng ANN truyền thẳng, trong đó kiến trúc chính gồm nhiều thành phần được ghép nối với nhau theo cấu trúc nhiều lớp đó là: Convolution, Pooling và Fully Connected.

2.5.3.2 Mô hình CNN



Hình 2.24 Mô hình CNN.

Trong một mô hình CNN được xây dựng với 3 lớp cơ bản sau:

- Convolutional Layer
- Pooling Layer
- Fully Connected

Tìm hiểu rõ những lớp cơ bản của CNN có nhiệm vụ và chức năng như sau:

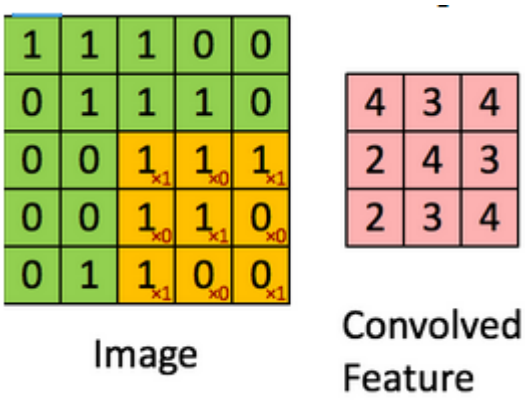
- **Convolutional Layers:**

Cũng giống như các lớp ẩn khác, Convolutional Layer (Conv) lấy dữ liệu đầu vào, thực hiện các phép chuyển đổi để tạo ra dữ liệu đầu vào cho lớp kế tiếp (đầu ra của lớp này là đầu vào của lớp sau). Phép biến đổi được sử dụng là phép tính tích chập. Mỗi Conv chứa một hoặc nhiều bộ lọc - bộ phát hiện đặc trưng (filter - feature detector) cho phép phát hiện và trích xuất những đặc trưng khác nhau của ảnh.

Độ phức tạp của đặc trưng được phát hiện bởi bộ lọc tỉ lệ thuận với độ sâu của Conv mà nó thuộc về. Trong mạng CNN, Conv đầu tiên sử dụng bộ lọc hình học (geometric filters) để phát hiện những đặc trưng đơn giản như cạnh ngang, dọc, chéo của bức ảnh. Conv sau đó được dùng để phát hiện đối tượng nhỏ, bán hoàn chỉnh như mắt, mũi, tóc, v.v. Conv sâu nhất dùng để phát hiện đối tượng hoàn hỉnh như: chó, mèo, chim, ô tô, đèn giao thông, v.v.

Convolution – tích chập được sử dụng đầu tiên trong xử lý tín hiệu số (Signal processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kĩ thuật này vào xử lý ảnh và video số. Để dễ hình dung, có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận.

Đối với một bức ảnh đen trắng thì việc tích chập được thực hiện như sau:



Hình 2.25 Minh họa về tích chập của ảnh trắng đen.

Ma trận bên trái là một bức ảnh trắng đen. Mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel), 0 là màu đen, 1 là màu trắng (nếu là ảnh grayscale thì giá trị biến thiên từ 0 đến 255). Sliding window còn có tên gọi là Kernel, Filter hay Feature detector. Ở đây, dùng một ma trận Kernel 3×3 nhân từng thành phần tương ứng với ma trận ảnh bên trái. Giá trị đầu ra do tích của các thành phần này cộng lại. Kết quả của tích chập là một ma trận (convoled feature) sinh ra từ việc trượt ma trận Kernel và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái.

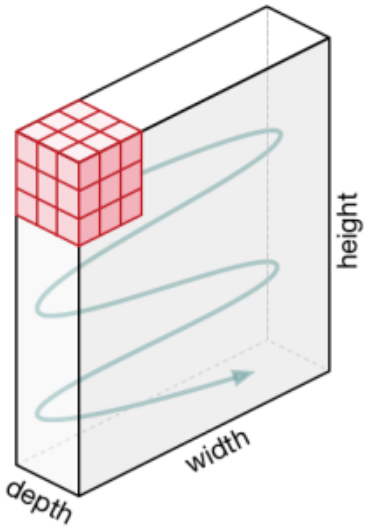
Số chiều của Conv chính là số hướng mà hàm Kernel có thể di chuyển được. Cụ thể như sau:

+ Convolution 1D: Tích chập 1 chiều sẽ chỉ cho phép Kernel di chuyển theo một chiều, (nghĩa là image của ta phải convert về dạng vector) chính là độ dài của vector.

+ Convolution 2D: Tích chập 2 chiều sẽ cho phép Kernel di chuyển theo 2 chiều dài và rộng (Width và Height).

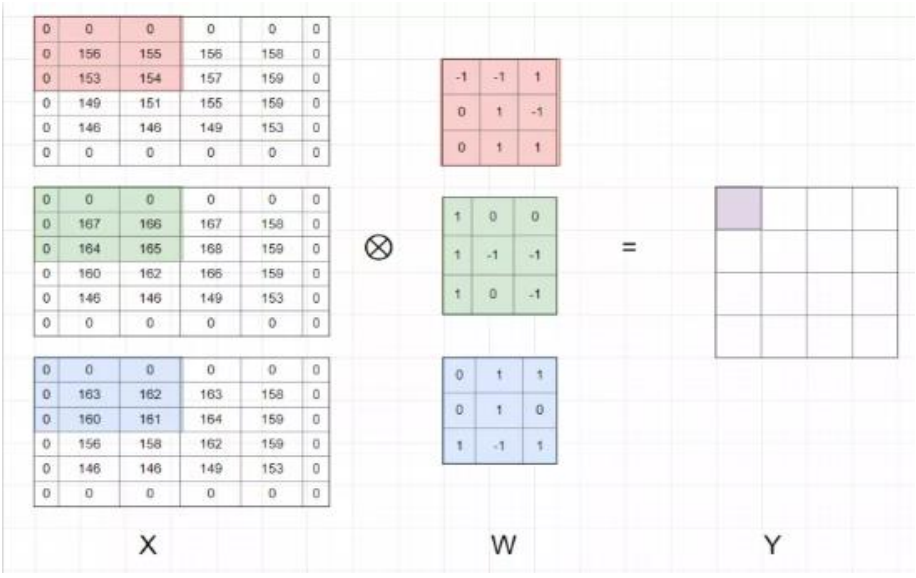
+ Convolution 3D: Tương tự cũng có tích chập 3 chiều là khi Kernel có thể di chuyển thêm cả chiều sâu của ảnh.

Đối với một bức ảnh màu có tới 3 kênh (Red, Green, Blue) nên khi biểu diễn ảnh dưới dạng tensor 3 chiều. Nên cũng sẽ định nghĩa Kernel là một tensor 3 chiều kích thước k*k*3.



Hình 2.26 Ảnh màu biểu diễn ở dạng tensor 3 chiều.

Và số lớp của Kernel phải bằng số kênh của ảnh đầu vào. Sau đó thực hiện di chuyển khối Kernel tương tự như khi thực hiện trên ảnh trắng đen.

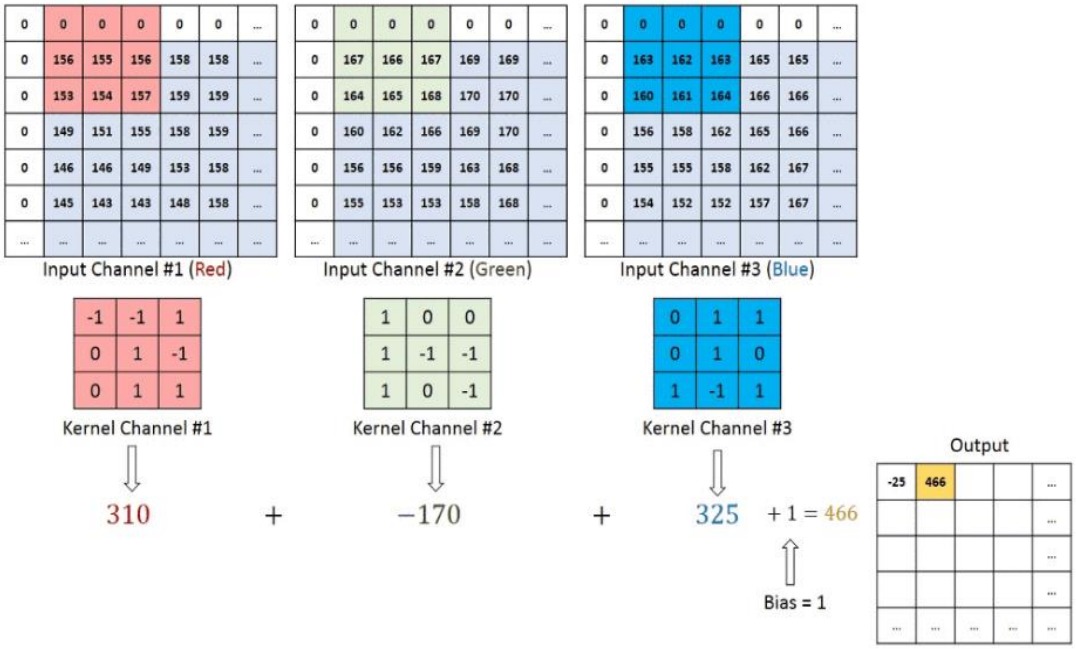


Hình 2.27 Minh họa cách tích chập trên ảnh màu.

Ba lớp của bộ lọc có thể được cấu hình khác nhau để phát hiện đặc trưng trên một, hai hoặc cả ba kênh màu của ảnh đầu vào. Đối với ba kênh màu ở hình 2.18 và xét ở phần tử đầu tiên cột 1 hàng 1, có công thức tính tích chập như sau:

$$Y_{11} = b + (X_{111} * W_{111} + X_{121} * W_{121} + X_{131} * W_{131} + X_{211} * W_{211} + X_{221} * W_{221} + X_{231} * W_{231} + X_{311} * W_{311} + X_{321} * W_{321} + X_{331} * W_{331}) + (X_{112} * W_{112} + X_{122} * W_{122} + X_{132} * W_{132} + X_{212} * W_{212} + X_{222} * W_{222} + X_{232} * W_{232} + X_{312} * W_{312} + X_{322} * W_{322} + X_{332} * W_{332}) + (X_{113} * W_{113} + X_{123} * W_{123} + X_{133} * W_{133} + X_{213} * W_{213} + X_{223} * W_{223} + X_{233} * W_{233} + X_{313} * W_{313} + X_{323} * W_{323} + X_{333} * W_{333}) = -25$$

Khi biểu diễn ma trận cần 2 chỉ số hàng và cột: i và j, thì khi biểu diễn ở dạng tensor 3 chiều cần thêm chỉ số độ sâu k. Nên chỉ số mỗi phần tử trong tensor là X_{ijk}



Hình 2.28 Minh họa thực hiện tích chập trên ảnh màu.

Sau khi thực hiện tích chập thì ngõ ra sẽ được đưa qua Activation function – hàm kích hoạt trước khi trở thành đầu vào của lớp tiếp theo.

+ Activation funtion cực kỳ quan trọng của ANN. Về cơ bản nó quyết định một neuron có nên được kích hoạt hay không. Cho dù thông tin mà neuron nhận được có liên quan đến thông tin đã cho hay không.

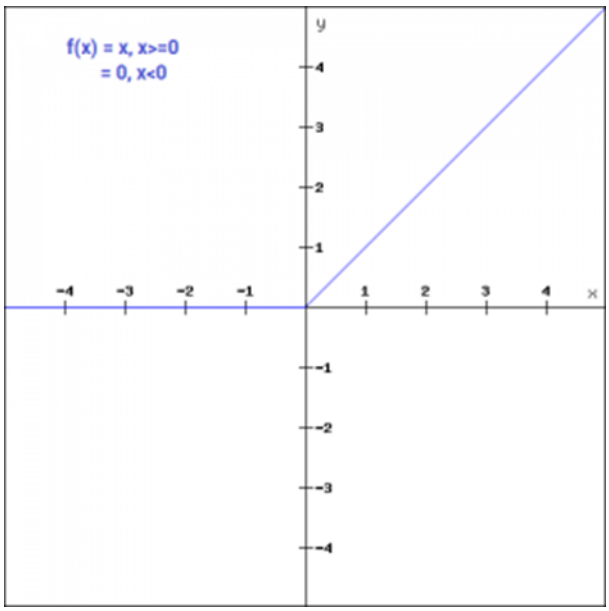
Công thức ngõ ra của Activation:

$$Y = \text{Activation}\ (\Sigma(\text{weight} * \text{input}) + bias) \quad (2.11)$$

Có rất nhiều Activation function được dùng trong bài toán ANN, tuy nhiên trong mô hình cảnh báo nhân viên ngủ gật chỉ sử dụng 2 Activation function sau:

+ Activation function sử dụng hàm Rectified Linear Unit (ReLU) được dựa trên tư tưởng của việc loại bỏ bớt những tham số không quan trọng trong quá trình huấn luyện và điều đó làm cho mạng trở nên nhẹ hơn và việc huấn luyện cũng nhanh chóng và có hiệu quả hơn. Hàm này thực hiện một việc rất đơn giản như sau: giữ nguyên những giá trị đầu vào lớn hơn 0, nếu giá trị đầu vào nhỏ hơn 0 thì coi là 0.

Có thể hình dung kĩ hơn trong hình sau:



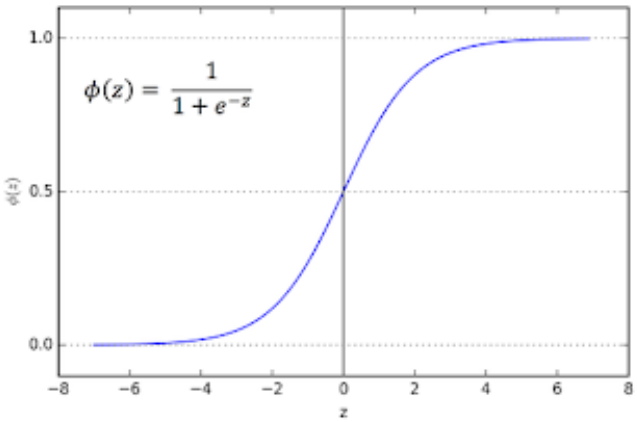
Hình 2.29 Hàm ReLU

+ Activation function sử dụng hàm Sigmod. Hàm Sigmoid (hay còn gọi là hàm số Logistic) là một hàm số toán học có đường cong dạng hình chữ "S" với công thức như sau:

$$f(x) = 1/(1+ e^{(-x)}) \quad (2.12)$$

Mọi giá trị khi đi qua hàm Sigmoid sẽ nằm trong miền giá trị số thực chạy từ 0.0 đến 1.0, có thể thấy giá trị càng âm khi đi qua hàm Sigmoid sẽ cho kết quả

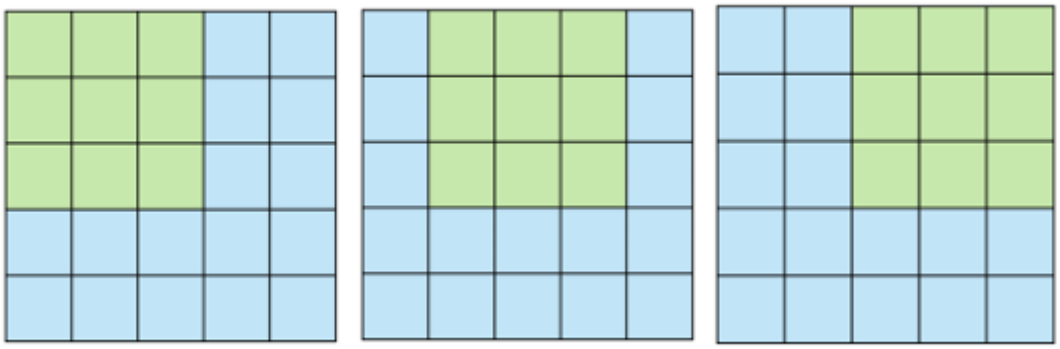
càng tiến về 0 và ngược lại, giá trị càng dương khi đi qua hàm Sigmoid sẽ cho kết quả càng tiến về 1, tại vị trí x = 0 hàm Sigmoid sẽ cho giá trị 0.5.



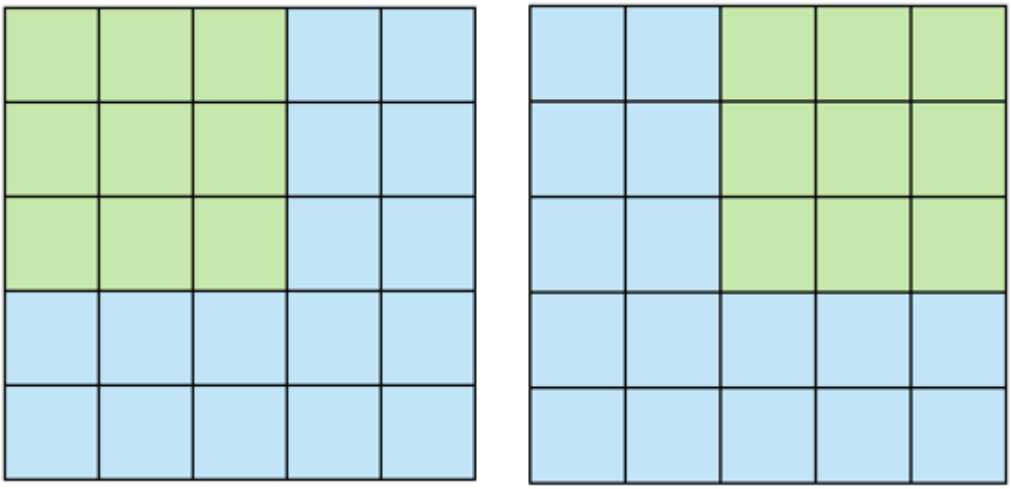
Hình 2.30 Hàm Sigmoid

Ngoài ra, khi xét đến Kernel thì phải xem xét đến Stride và Padding:

- Stride là khoảng cách bước giữa 2 Kernel khi quét. Với Stride = 1, Kernel sẽ quét 2 ô ngay cạnh nhau, nhưng với Stride = 2, Kernel sẽ quét ô số 1 và ô số 3. Bỏ qua ô ở giữa. Điều này nhằm tránh việc lặp lại giá trị ở các ô bị quét. Chọn Stride và kích thước của Kernel càng lớn thì kích thước của Feature map càng nhỏ, một phần lý do đó là bởi Kernel phải nằm hoàn toàn trong ngõ vào.

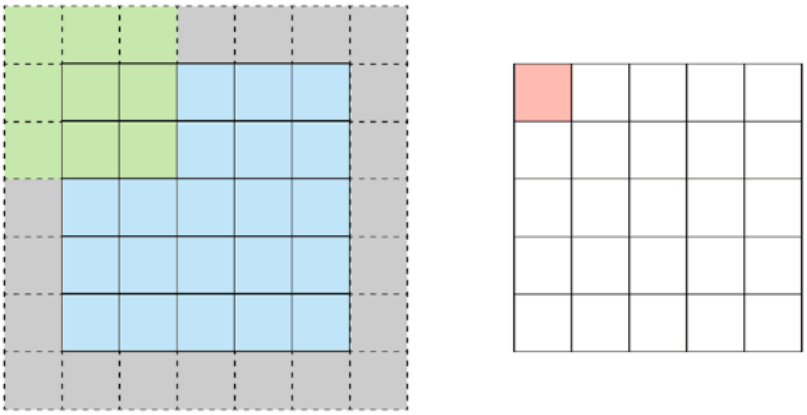


Hình 2.31 Quá trình quét của Kernel khi Stride bằng 1.



Hình 2.32 Quá trình quét của Kernel khi Stride bằng 2.

- Padding: là giữ nguyên kích cỡ của Feature map so với ban đầu. Khi ta điều chỉnh Padding = 1, tức là ta đã thêm 1 ô bọc xung quanh các cạnh của ngõ vào, muốn phần bọc này càng dày thì ta cần phải tăng Padding lên.



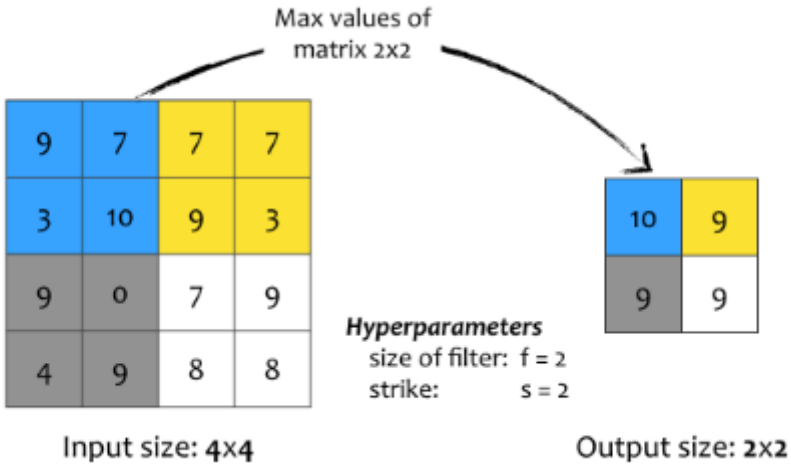
Hình 2.33 Điều chỉnh Padding bằng 1.

- **Pooling Layer**

Ngoài các Conv vừa mô tả, CNN cũng chứa các Pooling layer (Pool). Pool thường được sử dụng ngay sau Conv. Những gì các Pooling làm để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng sau khi thực hiện tích chập. Điều này rất hữu ích khi sử dụng mạng cho ảnh có kích cỡ lớn, giúp giảm việc tính toán trong model. Tuy nhiên nếu lạm dụng loại lớp này cũng có thể khiến dữ liệu đi qua bị mất dữ liệu.

Có nhiều hướng Pooling được sử dụng, trong đó phổ biến nhất là Max Pooling và Average Pooling.

+ Max pooling: "Nếu một đặc tính được phát hiện ở một vùng nào đó bị bao phủ bởi bộ lọc, giá trị cao nhất trong vùng sẽ được giữ lại tuy nhiên chưa ai giải thích được tại sao cách tiếp cận này lại hoạt động tốt trong thực nghiệm" [11].



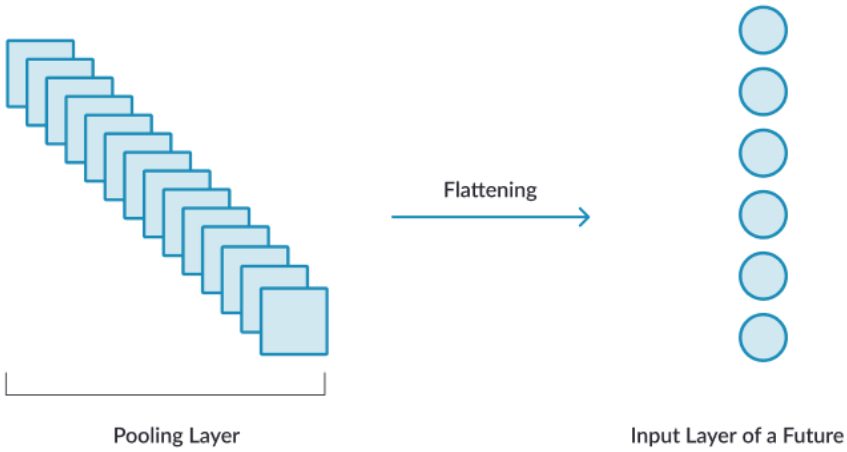
Hình 2.34 Tìm giá trị Max Pooling trong ma trận 4x4.

Trong Max Pooling có 2 tham số Hyperparameters (kích thước của bộ lọc P và giá trị bước sải S) tuy nhiên không có tham số cần huấn luyện trong lớp Max Pooling.

+ Average Pooling: Thay vì lấy giá trị cực đại, Pooling theo giá trị trung bình lấy trung bình của tất cả các giá trị trong vùng bị bao phủ bởi Kernel khi nó trượt trên ma trận đầu vào. Tuy nhiên Pooling theo giá trị trung bình rất ít khi được sử dụng, hầu hết các CNN hiện nay sử dụng Pooling theo giá trị cực đại.

- **Fully Connected Layer:**

Ở giữa Conv và Fully Connected Layer (FC), có một Flatten Layer có chức năng biến đổi tensor có kích thước H*W*D của các thuộc tính ảnh thành vector dọc có kích thước H*W*D hàng có thể đưa vào FC.



Hình 2.35 Chuyển từ tensor sang vector của Flatten Layer.

FC là cách kết nối các neuron ở hai tầng với nhau trong đó tầng sau kết nối đầy đủ với các neuron ở tầng trước nó. Đây cũng là dạng kết nối thường thấy ở ANN, trong CNN tầng này thường được sử dụng ở các tầng phụ cuối của kiến trúc mạng.

2.6 NGÔN NGỮ LẬP TRÌNH PYTHON

2.6.1 Lịch sử ra đời của ngôn ngữ Python

"Python là một ngôn ngữ lập trình thông dịch được phổ biến vào năm 1991, ngôn ngữ này được tạo ra vào cuối những năm 1980 bởi lập trình viên người Hà Lan Guido van Rossum. Ông đã tạo ra ngôn ngữ mới dựa trên nền tảng ngôn ngữ lập trình ABC." [12]. Và đến thời điểm để phổ biến ngôn ngữ này thì Van Rossum muốn có tên gọi sáng tạo dựa trên các tiêu chí như ngắn, độc đáo và một chút bí ẩn. Ông đã tìm thấy nguồn cảm hứng khi xem chương trình của nhóm hài nổi tiếng người Anh: Monty Python.

Python hoàn toàn từ động tạo và sử dụng cơ chế cấp phát bộ nhớ tự động, do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk và Tcl. Python đang được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý. Cú pháp Python gọn gàng và có tính năng gõ động, cùng với tính diễn dịch thân thiện làm Python là một ngôn ngữ lý tưởng dành cho những người mới bắt đầu lập trình và phát triển ứng dụng nhanh chóng trong nhiều lĩnh vực trên nhiều nền tảng khác nhau.

2.6.2 Một số đặc điểm của ngôn ngữ Python

Là một ngôn ngữ lập trình mạnh và dễ học đối với những người mới bắt đầu, có mã nguồn dễ đọc, bố cục đơn giản, cú pháp dễ nhớ, dễ hiểu.

Ngôn ngữ lập trình đơn giản, dễ học: Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như C++, Java, C#. Python làm cho việc lập trình trở nên thú vị, cho phép bạn tập trung vào những giải pháp chứ không phải cú pháp.

Miễn phí, mã nguồn mở: Bạn có thể tự do sử dụng và phân phối Python, thậm chí là dùng cho mục đích thương mại. Vì là mã nguồn mở, bạn không những có thể sử dụng các phần mềm, chương trình được viết trong Python mà còn có thể thay đổi mã nguồn của nó. Python có một cộng đồng rộng lớn, không ngừng cải thiện nó mỗi lần cập nhật.

Khả năng di chuyển: Các chương trình Python có thể di chuyển từ nền tảng này sang nền tảng khác và chạy nó mà không có bất kỳ thay đổi nào. Nó chạy liền mạch trên hầu hết tất cả các nền tảng như Windows, macOS, Linux.

Khả năng mở rộng và có thể nhúng: Giả sử một ứng dụng đòi hỏi sự phức tạp rất lớn, bạn có thể dễ dàng kết hợp các phần code bằng C, C++ và những ngôn ngữ khác (có thể gọi được từ C) vào code Python. Điều này sẽ cung cấp cho ứng dụng của bạn những tính năng tốt hơn cũng như khả năng scripting mà những ngôn ngữ lập trình khác khó có thể làm được.

Ngôn ngữ thông dịch cấp cao: Không giống như C/C++, với Python, bạn không phải lo lắng những nhiệm vụ khó khăn như quản lý bộ nhớ, dọn dẹp những dữ liệu vô nghĩa... Khi chạy code Python, nó sẽ tự động chuyển đổi code sang ngôn ngữ máy tính có thể hiểu. Bạn không cần lo lắng về bất kỳ hoạt động ở cấp thấp nào.

Thư viện tiêu chuẩn lớn để giải quyết những tác vụ phổ biến: Python có một số lượng lớn thư viện tiêu chuẩn giúp cho công việc lập trình của bạn trở nên dễ thở hơn rất nhiều, đơn giản vì không phải tự viết tất cả code.

Hướng đối tượng: Mọi thứ trong Python đều là hướng đối tượng. Lập trình hướng đối tượng (OOP) giúp giải quyết những vấn đề phức tạp một cách trực quan. Với OOP, bạn có thể phân chia những vấn đề phức tạp thành những tập nhỏ hơn bằng cách tạo ra các đối tượng.

2.6.3 Một số ứng dụng

Python được ứng dụng trong nhiều lĩnh vực như sau:

- Xây dựng các tiện ích nhỏ để tự động hóa các công việc nào đó như: tự động tìm kiếm, phân loại tập tin theo tiêu chí riêng, tự động cập nhật các tập tin văn bản theo yêu cầu nào đó.
- Xây dựng ứng dụng web: Python cung cấp nhiều framework để ta có thể lựa chọn để phát triển ứng dụng web tùy theo mô hình của ứng dụng như: Django, Pyramid, Flask.
- Lập trình các tính toán khoa học, số liệu nhờ các công cụ và lớp thư viện được xây dựng sẵn như: SciPy, IPython, Opencv.
- Lập trình ứng dụng desktop (wxWidgets), lập trình màn hình tương tác (Kivy).

Bên cạnh đó, Python còn là ngôn ngữ lập trình được lựa chọn để giảng dạy về lập trình các khóa học nhập môn lập trình ở các trường Đại học lớn trên thế giới.

2.7 MỘT SỐ THƯ VIỆN SỬ DỤNG TRONG PYTHON

2.7.1 Thư viện Opencv.

2.7.1.1 Tổng quan về thư viện OpenCV.

"OpenCV được bắt đầu tại Intel vào năm 1999 bởi Gary Bradsky và bản phát hành đầu tiên ra mắt vào năm 2000. Vadim Pisarevsky đã cùng Gary Bradsky quản lý nhóm OpenCV phần mềm Intel của Nga. Vào năm 2005, OpenCV đã được sử dụng trên Stanley, chiếc xe đã chiến thắng Thử thách lớn DARPA năm 2005. Sau đó, sự phát triển tích cực của nó tiếp tục dưới sự hỗ trợ của Willow Garage, với Gary Bradsky và Vadim Pisarevsky dẫn dắt dự án. OpenCV hiện hỗ trợ vô số thuật toán liên quan đến Thị giác máy tính và máy học và đang mở rộng từng ngày" [13]. Thư viện OpenCV bao gồm nhiều giao diện dành cho C++, C, Python, Java,

MATLAB và hỗ trợ cho các hệ điều hành khác nhau như Windows, Linux, Android, MacOS. Trong phiên bản OpenCV 3.1, giao diện sử dụng cho CUDA và OpenCL cũng đã được phát triển hoàn thiện. OpenCV được viết nguyên bản bằng ngôn ngữ C++.

2.7.1.2 Sơ lược về cấu trúc của thư viện OpenCV

Thư viện OpenCV có thể được chia thành 2 phần chính: Phần căn bản (basic) là mã nguồn được nhóm phát triển xây dựng và kiểm định toàn diện, gồm các thuật toán đã được thế giới công nhận và đánh giá dựa trên cơ sở lý thuyết chắc chắn. Phần mở rộng (contribution) được nhiều tổ chức khoa học khác nhau trên thế giới đóng góp, gồm nhiều thuật toán cập nhật được xây dựng dựa trên các công trình nghiên cứu, bài báo mới đăng trong thời gian gần đây. Do vậy các thuật toán trong phần mở rộng có độ ổn định và tối ưu không cao. Từ phiên bản 3.0, phần mở rộng được tách riêng không còn được gộp chung với thư viện mặc định.

OpenCV là thư viện mã nguồn mở được đóng gói thành tập tin nén. Tùy vào phiên bản dành cho các hệ điều hành khác nhau mà tập tin nén này có định dạng tương ứng. Thư viện OpenCV cung cấp cho người dùng các cấu trúc dữ liệu, đối tượng và hàm bằng cách khai báo nguyên mẫu (prototype) của chúng trong các tập tin thư viện C/C++ (*.h, *.hpp…) và định nghĩa chi tiết trong các tập tin mã nguồn (*.c, *.cpp). Với mức độ sử dụng OpenCV, ta chỉ cần giải nén các tập tin đã được biên dịch sẵn rồi thực hiện các thao tác cài đặt đường dẫn cho thích hợp để hệ điều hành tìm đến đúng vị trí của các tập tin thư viện. Ở mức độ cao hơn, nếu muốn hiệu chỉnh sửa đổi thuật toán hay sử dụng phần mở rộng của OpenCV ta cần phải biên dịch mã nguồn trực tiếp trên máy trước khi cài đặt.

2.7.1.3 Ứng dụng của thư viện OpenCV

Có rất nhiều ứng dụng: nhận dạng ảnh, xử lý ảnh, phục hồi ảnh và video, thực tế ảo và các ứng dụng khác.

2.7.2 Keras

Keras là một framework mã nguồn mở cho Deep Learning được viết bằng Python và là một API được thiết kế cho con người, không phải máy móc. “Keras dựa theo các kinh nghiệm tốt nhất để giảm tải hoạt động nhận thức: nó cung cấp các API đồng nhất và đơn giản, các trường hợp phổ biến và các phản hồi rõ ràng khi người dùng gặp lỗi, từ đó giảm thiểu công sức của con người. Keras có thể chạy trên nền của các Deep Learning framework khác như: Tensorflow, Theano, CNTK. Với các API bậc cao, dễ sử dụng, dễ mở rộng, Keras giúp người dùng xây dựng các Deep Learning model một cách đơn giản” [14].

Điều này làm cho Keras dễ học và dễ sử dụng. Là người dùng Keras, người dùng có năng suất cao hơn, cho phép thử nhiều ý tưởng hơn so với đối thủ cạnh tranh, nhanh hơn - điều này giúp giành chiến thắng trong các cuộc thi máy học. Dễ sử dụng này không phải trả giá: bởi vì Keras tích hợp với các ngôn ngữ Deep Learning cấp độ thấp hơn (đặc biệt là TensorFlow), nó cho phép bạn thực hiện bất cứ điều gì bạn có thể xây dựng bằng ngôn ngữ cơ bản.

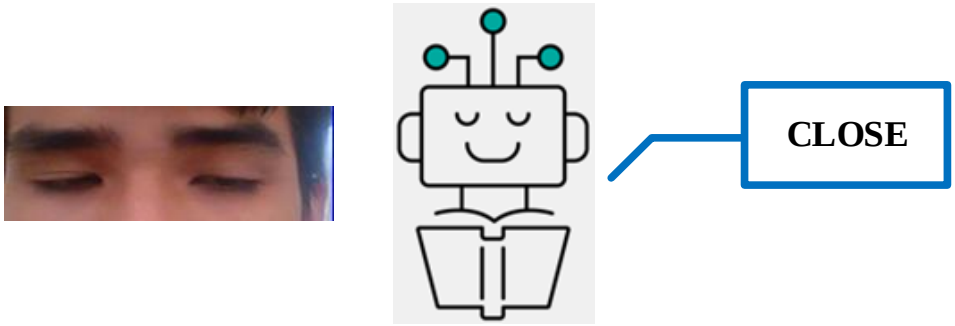
CHƯƠNG 3

THIẾT KẾ HỆ THỐNG

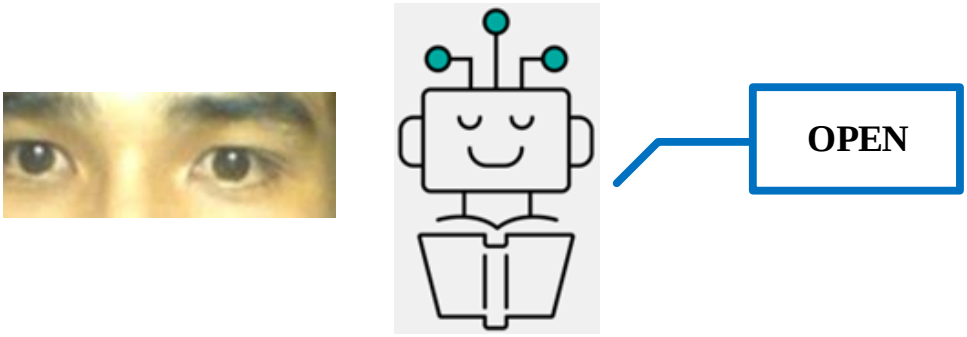
3.1 ỨNG DỤNG KERAS VÀO XÂY DỰNG MÔ HÌNH CNN CẢNH BÁO NHÂN VIÊN NGỦ GẬT.

3.1.1 Ý tưởng xây dựng mô hình

Bản chất của bài toán DL là: người lập trình có dữ liệu, muốn máy tính học theo các model từ dữ liệu mẫu, sau đó dùng mô hình đấy để dự đoán các dữ liệu mới. Trong bài toán cảnh báo nhân viên ngủ gật nhóm nghiên cứu sẽ có đầu vào là bức ảnh trạng thái mắt và mô hình DL dự đoán ảnh đó là ảnh nhắm mắt hay mở mắt.



Hình 3.1 Mô hình CNN với dữ liệu đầu vào ở trạng thái nhắm mắt.



Hình 3.2 Mô hình CNN với dữ liệu đầu vào ở trạng thái mở mắt.

3.1.2 Chuẩn bị dữ liệu

Bộ cơ sở dữ liệu là một tập dataset_eye tự tạo từ thuật toán AdaBoost dựa vào đặc trưng Haar-like và Cascade có hai bộ dữ liệu là close và open với các ảnh chụp đôi mắt ở hai trạng thái nhắm mắt và mở mắt, là các ảnh màu có kích thước bất kỳ và được resize về kích thước chuẩn 40*150*1.

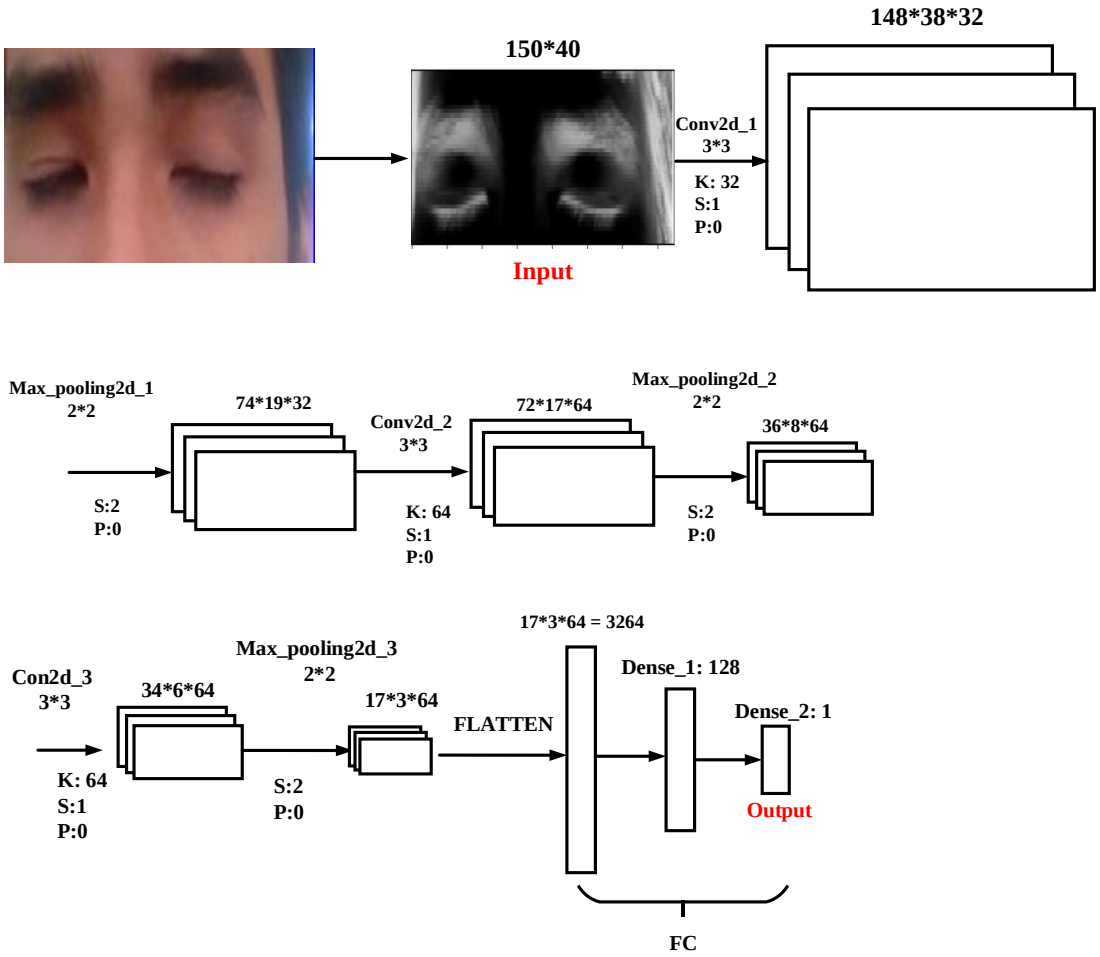
Bộ dataset_eye có 1157 ảnh bao gồm cả ảnh ở trạng thái nhắm mắt và mở mắt. Được chia ngẫu nhiên thành hai tập con là Training set và Test set với tỷ lệ là 0.9. Trong tập Training set chia làm hai tập là Training set và validation set tỷ lệ là 0.9.

- Training set có thể hiểu là dữ liệu dùng để dạy cho model học.
- Validation set là để đánh giá xem model hiện tại có tốt không, thường được dùng để điều chỉnh các tham số của model.
- Test set là để đánh giá xem model hoạt động với dữ liệu thực tế có tốt không.

Như vậy dataset_eye có 1157 dữ liệu, chia ra 936 dữ liệu cho training set và 105 dữ liệu cho validation set. Vẫn giữ nguyên 116 dữ liệu của test set.

3.1.3 Xây dựng mô hình

Mô hình CNN cảnh báo nhân viên ngủ gật:



Hình 3.3 Các khối trong mô hình CNN cảnh báo nhân viên ngủ gật.

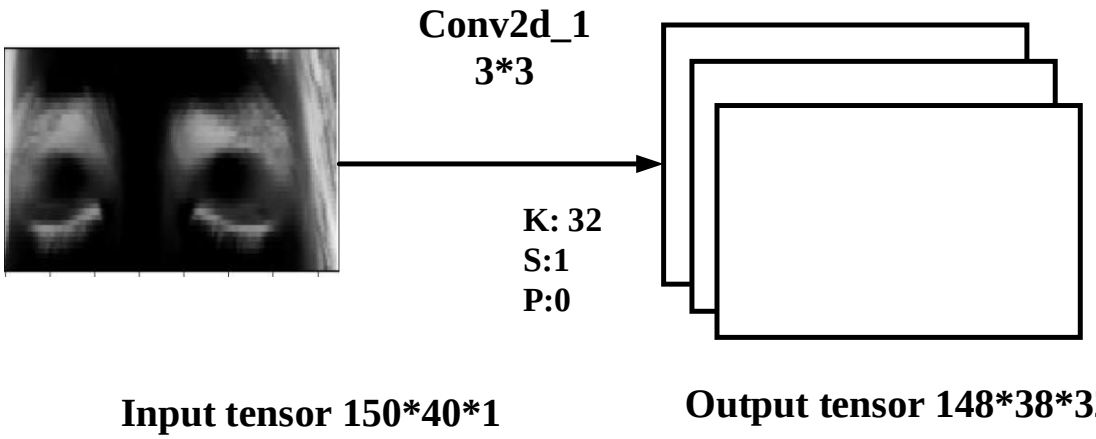
- **Layer thứ nhất: Input layer** là ảnh xám có kích thước 150*40*1.
- **Layer thứ hai: Conv2d_1**

  + Ngõ vào của Conv2d_1 là ma trận ảnh xám có kích thước: 150*40.

  + Conv2d_1 áp dụng 32 kernel có kích thước 3*3 với S = 1, P = 0.

  + Ngõ ra của layer là tenser 3 chiều có kích thước: 148*38*32.

  + Tổng số Parameter của conv2d_1: 32*(3*3*1+1) = 320.



Hình 3.4 Conv2d_1 layer.

+ Ngõ ra của Conv2d_1 sẽ qua hàm Activation trước khi trở thành ngõ vào của layer tiếp theo.

+ Conv2d_1 chọn hàm activation là hàm ReLU.

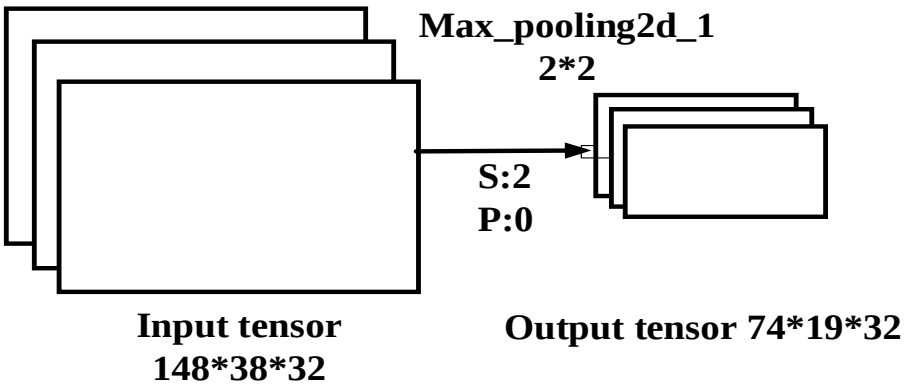
- **Layer thứ ba: max_pooling2d_1**

+ Ngõ vào của max_pooling2d_1 có kích thước: 148*38*32.

+ Max_pooling2d_1 sử dụng pool_size kích thước 2*2 với S = 2, P = 0.

+ Ngõ ra là tensor có kích thước: 74*19*32 với ngõ ra tensor sẽ chứa maximun của ngõ vào tensor.

+ Tổng số Parameter của max_pooling2d_1: 0*(2*2*32+1) = 0.



Hình 3.5 Max_pooling2d_1.

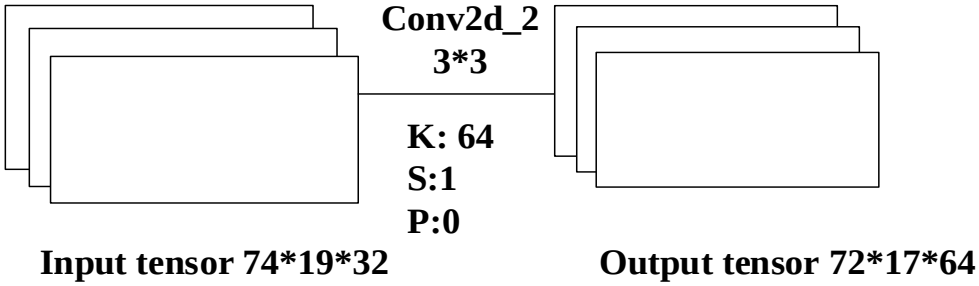
- **Layer thứ tư: Conv2d_2**

  + Ngõ vào của conv2d_2 là tensor có kích thước: 74*19*32.

  + Conv2d_2 áp dụng 64 kernel có kích thước 3*3 với S = 1, P = 0.

  + Ngõ ra của conv2d_2 là tensor chiều có kích thước: 72*17*64.

  + Tổng số Parameter của conv2d_2: 64*(3*3*32+1) = 18496.



Hình 3.6 Conv2d_2 layer.

+ Conv2d_2 chọn hàm activation là hàm ReLU.

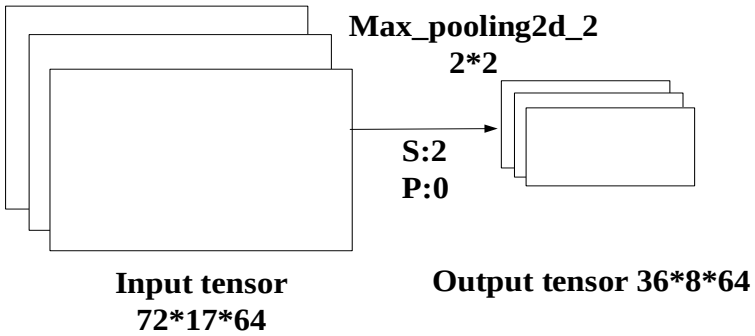
- **Layer thứ năm: max_pooling2d_2**

  + Ngõ vào của max_pooling2d_2 là tensor có kích thước: 72*17*64.

  + Max_pooling2d_2 sử dụng pool_size kích thước 2*2 với S = 2, P = 0.

  + Ngõ ra là tensor có kích thước: 36*8*64 với ngõ ra tensor sẽ chứa maximun của ngõ vào tensor.

  + Tổng số Parameter của max_pooling2d_2: 0*(2*2*64+1) = 0.



Hình 3.7 Max_pooling2d_2.

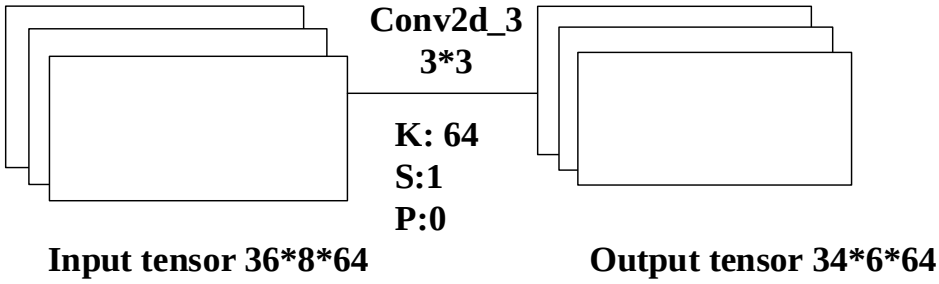
- **Layer thứ sáu: Conv2d_3**

  + Ngõ vào của conv2d_3 là tensor có kích thước: 36*8*64.

  + Conv2d_2 áp dụng 64 kernel có kích thước 3*3 với S = 1, P = 0.

  + Ngõ ra của conv2d_2 là tensor chiều có kích thước: 34*6*64.

  + Tổng số Parameter của conv2d_3: 64*(3*3*64+1) = 36928.



Hình 3.8 Conv2d_3 layer.

+ Conv2d_3 chọn hàm activation là hàm ReLU.

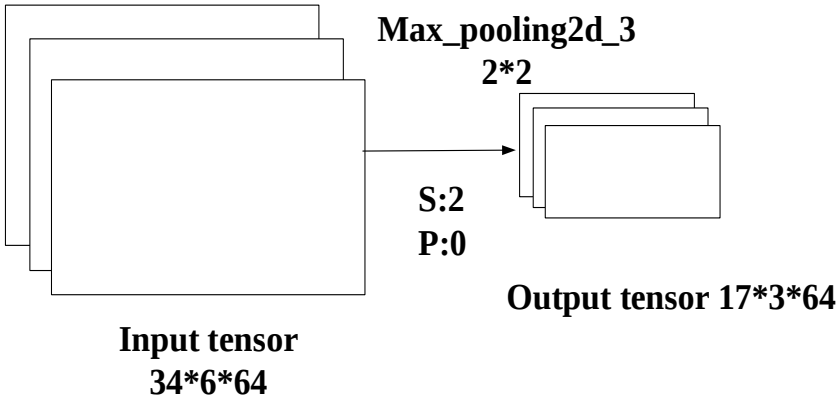
- **Layer thứ bảy: max_pooling2d_3**

  + Ngõ vào của max_pooling2d_3 là tensor có kích thước: 34*6*64.

  + Max_pooling2d_3 sử dụng pool_size kích thước 2*2 với S = 2, P = 0.

  + Ngõ ra là tensor có kích thước: 17*3*64 với ngõ ra tensor sẽ chứa maximun của ngõ vào tensor.

  + Tổng số Parameter của max_pooling2d_3 là 0.

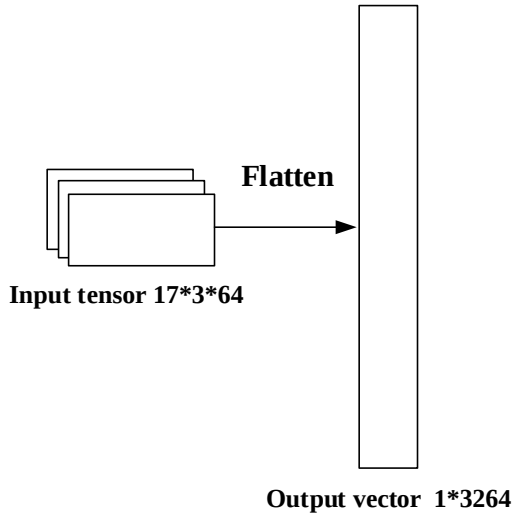


Hình 3.9 Max_pooling2d_3.

- **Layer thứ tám:** flatten_1

  + Ngõ vào của Flatten là tensor 3 chiều có kích thước: 17*3*46.

  + Ngõ ra của layer là vector dọc có 17*3*64 = 3264 hàng.



Hình 3.10 Flatten layer.

- **Layer thứ chín:** FC có Dense_1 và Dense_2

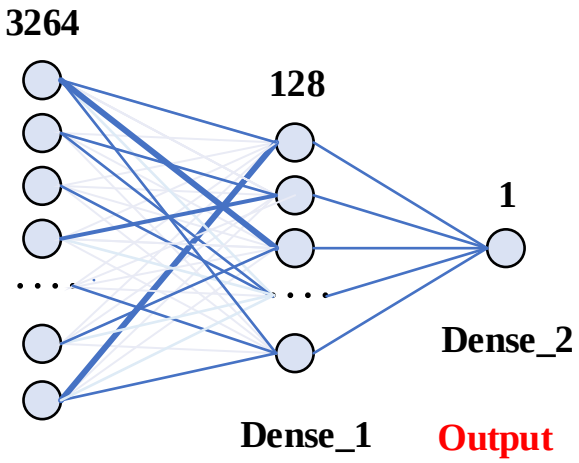
  + Dense_1:

    - Có 128 node.
    - Ngõ vào layer có 3264 node.
    - Số weight giữa input và dense_1 là: 3264*128 = 417729.
    - Số lượng bias là: 128.
    - Tổng số Parameter của dense_1: là 417920.
    - Dense_1 chọn hàm activation là hàm ReLU.

  + Dense_2:

    - Có 1 node.
    - Ngõ vào layer có 128 node.
    - Số weight giữa dense_1 và dense_2 là: 1*128 = 128.
    - Số lượng bias là: 1.

- Tổng số Parameter của dense_2: là 129.
- Dense_2 chọn hàm activation là hàm Sigmoid.
- Cụ thể trong mô hình là:



Hình 3.11 Fully connected layer.

Tóm tắt cấu trúc mô hình:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 38, 32)	320
max_pooling2d_1 (MaxPooling2	(None, 74, 19, 32)	0
conv2d_2 (Conv2D)	(None, 72, 17, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 36, 8, 64)	0
conv2d_3 (Conv2D)	(None, 34, 6, 64)	36928

max_pooling2d_3 (MaxPooling2	(None, 17, 3, 64)	0
flatten_1 (Flatten)	(None, 3264)	0
dense_1 (Dense)	(None, 128)	417920
dense_2 (Dense)	(None, 1)	129

Total params: 473,793

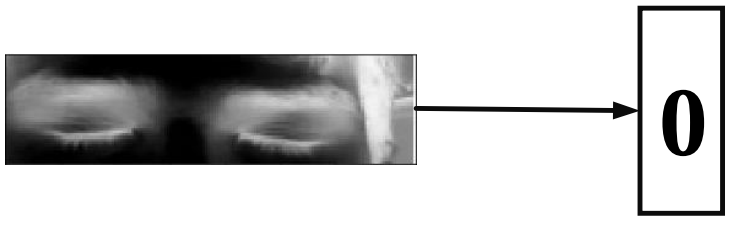
Trainable params: 473,793

Non-trainable params: 0

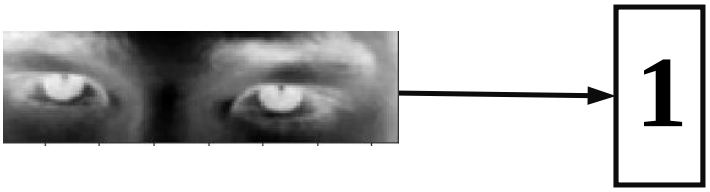
- **Compile model**

Loss function: sẽ tính toán được hiệu năng model trong quá trình huấn luyện.

Như ban đầu đã được định nghĩa, mô hình bài toán chỉ có 2 class là nhắm mắt và mở mắt. Dùng one-hot encoding chuyển đổi label của ảnh từ giá trị số sang vector cùng kích thước với ngõ ra của model, cụ thể như sau:



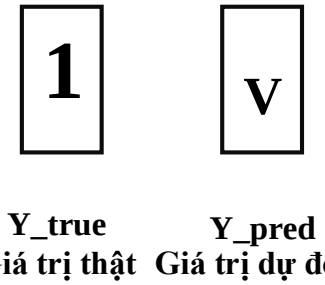
Hình 3.12 Dự đoán trạng thái mắt nhắm.



Hình 3.13 Dự đoán trạng thái mắt mở.

Để ý là label của data là thứ i là vector v kích thước 1*1. Với v = 1 so với quy ước về phần trăm ở trên thì one-hot encoding có ý nghĩa là ta chắc chắn 100% ảnh này là mở mắt.

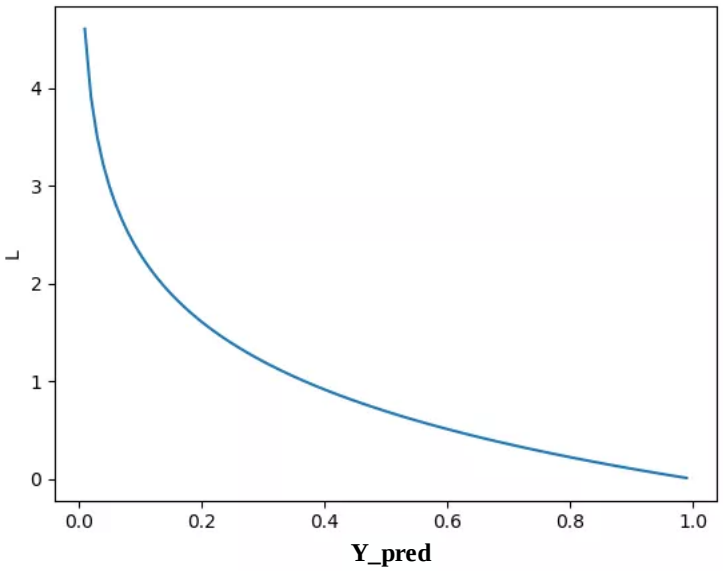
Giờ ta có giá trị thật (label) dạng one-hot encoding giá trị dự đoán ở ngõ ra layer sau sigmoid cùng kích thước 1*1. Cần định nghĩa loss function để đánh giá độ tốt của model.



Hình 3.14 Giá trị dự đoán so với giá trị thật

Mong muốn là V gần với giá trị 1 như thế nghĩa là model dự đoán đúng được ảnh đầu vào là ảnh nhắm mắt. Định nghĩa loss function:

$$L = -log(V). \quad (3.1)$$



Hình 3.15 Đồ thị hàm loss function

- Ta thấy:

+ Hàm L giảm dần từ 0 đến 1

+ Khi model dự đoán y_pred gần 1, tức giá trị dự đoán gần với giá trị thật y_true thì L nhỏ, xấp xỉ 0

+ Khi model dự đoán y_pred gần 0, tức giá trị dự đoán ngược lại giá trị thật y_true thì L rất lớn

Kết luận hàm L nhỏ khi giá trị model dự đoán gần với giá trị thật và rất lớn khi model dự đoán sai, hay nói cách khác L càng nhỏ thì model dự đoán càng gần với giá trị thật.

Như vậy bài toán tìm model trở thành tìm giá trị nhỏ nhất của L.

Hàm loss function định nghĩa trong keras gọi là "**binary_crossentropy**"

Binary_crossentropy

```
keras.losses.brinary_crossentropy(y_true, y_pred)
```

- Nếu y_true = y_pred thì trả về giá trị 1 ngược lại 0.

+ Optimizers: là quá trình tối ưu trong DL, về nguyên tắc là làm giảm tổn thất, mất mát trong quá trình huấn luyện. Vì việc optimizer là điều chỉnh việc cập nhật các trọng số trong neural network.

Chọn phương thức tối ưu hóa Adam: một thuật toán tối ưu cho hàm gradient-based dựa trên các đánh giá phù hợp của trạng thái thấp hơn. Phương pháp này rất đơn giản để thực hiện, tính toán hiệu quả, có ít yêu cầu bộ nhớ, bất biến đối với việc thay đổi kích thước đường chéo của gradient và rất phù hợp cho các vấn đề về dữ liệu lớn và Parameters. Phương pháp này cũng phù hợp cho các đối tượng và vấn đề không cố định với nhiễu hoặc thưa thớt gradient.

3.1.4 Tiến hành huấn luyện model

Train on 936 samples, validate on 105 samples

Epoch 1/10
936/936 [==============================] - 14s 15ms/step - loss: 0.5504 - acc: 0.7382 - val_loss: 0.1840 - val_acc: 0.9619

Epoch 00001: val_loss improved from inf to 0.18402, saving model to D:/nam 4/do_an/model_eye_train/_mini_XCEPTION.01-0.96.hdf5
Epoch 2/10
936/936 [==============================] - 13s 14ms/step - loss: 0.0693 - acc: 0.9765 - val_loss: 0.1103 - val_acc: 0.9714

Epoch 00002: val_loss improved from 0.18402 to 0.11025, saving model to D:/nam 4/do_an/model_eye_train/_mini_XCEPTION.02-0.97.hdf5
Epoch 3/10
936/936 [==============================] - 14s 15ms/step - loss: 0.0442 - acc: 0.9850 - val_loss: 0.0308 - val_acc: 0.9905

Epoch 00003: val_loss improved from 0.11025 to 0.03081, saving model to D:/nam 4/do_an/model_eye_train/_mini_XCEPTION.03-0.99.hdf5
Epoch 4/10
936/936 [==============================] - 14s 15ms/step - loss: 0.0281 - acc: 0.9904 - val_loss: 0.0887 - val_acc: 0.9905

Epoch 00004: val_loss did not improve from 0.03081
Epoch 5/10
936/936 [==============================] - 13s 14ms/step - loss: 0.0321 - acc: 0.9893 - val_loss: 0.0450 - val_acc: 0.9905

Epoch 00005: val_loss did not improve from 0.03081
Epoch 6/10
936/936 [==============================] - 14s 15ms/step - loss: 0.0273 - acc: 0.9882 - val_loss: 0.0925 - val_acc: 0.9714

Epoch 00006: val_loss did not improve from 0.03081
Epoch 7/10
936/936 [==============================] - 14s 15ms/step - loss: 0.0220 - acc: 0.9936 - val_loss: 0.0435 - val_acc: 0.9905

Epoch 00007: val_loss did not improve from 0.03081

Epoch 8/10
936/936 [==============================] - 13s 14ms/step - loss: 0.0167 - acc: 0.9957 - val_loss: 0.0117 - val_acc: 0.9905

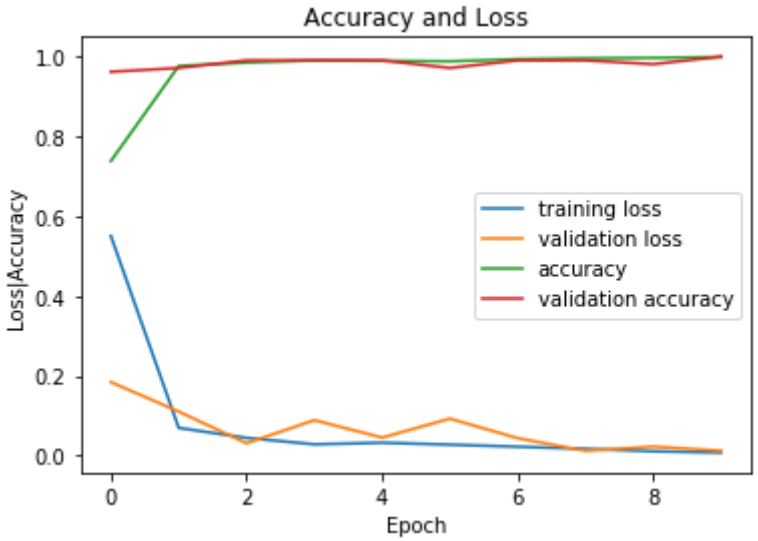
Epoch 00008: val_loss improved from 0.03081 to 0.01172, saving model to D:/nam 4/do_an/model_eye_train/_mini_XCEPTION.08-0.99.hdf5
Epoch 9/10
936/936 [==============================] - 13s 14ms/step - loss: 0.0106 - acc: 0.9968 - val_loss: 0.0224 - val_acc: 0.9810

Epoch 00009: val_loss did not improve from 0.01172
Epoch 10/10
936/936 [==============================] - 14s 15ms/step - loss: 0.0072 - acc: 0.9989 - val_loss: 0.0121 - val_acc: 1.0000

Epoch 00010: val_loss did not improve from 0.01172

3.1.5 Kết quả huấn luyện dữ liệu với model

Kết quả huấn luyện được biểu qua biểu đồ:



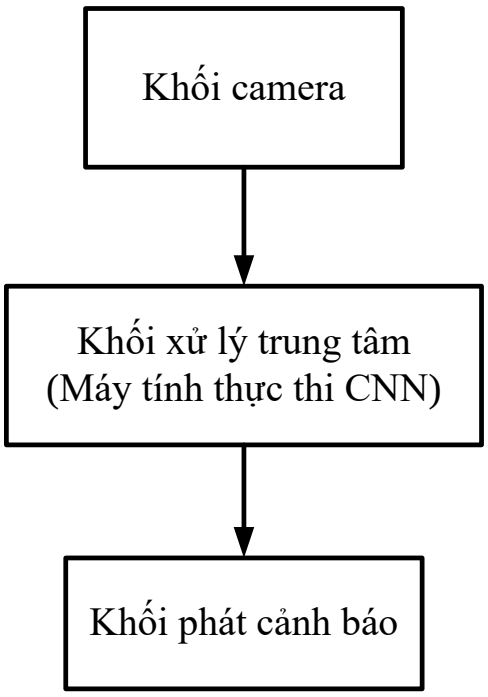
Hình 3.16 Biểu đồ các giá trị khi huấn luyện.

Như trên biểu đồ có thể thấy:

- Ở epoch 1/10: training loss = 0.5504, validation loss = 0.1840 => loss còn lớn nên giá trị dự đoán và giá trị thực chưa gần nhau, cụ thể: accuracy = 0.7382; validation accuracy = 0.9619.
- Ở epoch 10/10: training loss = 0.0072, validation loss = 0.0121=> loss xấp xỉ bằng 0 nên giá trị dự đoán và giá trị thực xấp xỉ trùng nhau, cụ thể: accuracy = 0.9989; validation accuracy = 1.0000.

=> Giá trị loss càng nhỏ kết quả giá trị dự đoán càng gần giá trị thật.

3.2 SƠ ĐỒ KHỐI HỆ THỐNG



Hình 3.17 Sơ đồ khối hệ thống.

Chức năng của từng khối:

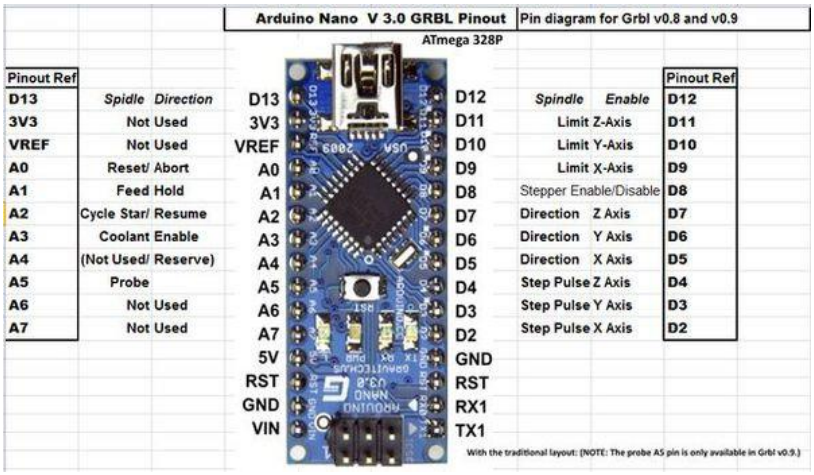
- Khối Camera: những loại Camera khác nhau thì có chức năng khác nhau, trong mô hình sử dụng Camera của máy tính để ghi nhận lại hình ảnh của nhân viên và gửi về khối xử lý trung tâm.

- Khối xử lý trung tâm (máy tính thực thi): nhận hình ảnh từ Camera ghi lại tiến hành phân tích và đưa ra tín hiệu cho khối cảnh báo.
- Khối phát cảnh báo: bao gồm Arduino nano và còi buzzer. Thực hiện chức năng nhận dữ liệu từ máy tính thực thi và tiến hành phát tín hiệu cảnh báo.

3.2.1 Giới thiệu về Arduino Nano

Board Arduino Nano là một trong những phiên bản nhỏ gọn của board Arduino. Arduino Nano có đầy đủ các chức năng và chương trình có trên Arduino Uno do cùng sử dụng MCU ATmega328P. Nhờ việc sử dụng IC dán của ATmega328P thay vì IC chân cắm nên Arduino Nano có thêm 2 chân Analog so với Arduino Uno.

Hình 3.18 Hình ảnh thực tế Arduino Nano.



Hình 3.19 Sơ đồ chân của Arduino Nano.

Đặc điểm của Arduino Nano:

- Vi điều khiển: ATmega328P (họ 8 bit).
- Điện áp hoạt động: 5VDC.
- Điện áp khuyên dùng: 7 – 12 VDC.
- Điện áp giới hạn: 6 – 20 VDC.
- Tần số hoạt động: 16MHz.
- Dòng tiêu thụ: 30mA.
- Số chân Digital I/O: 14 chân (6 chân PWM).
- Số chân analog: 8 chân (độ phân giải 10 bit).
- Dòng tối đa trên mỗi chân I/O: 40mA.
- Dòng ra tối đa (5V): 500mA.
- Dòng ra tối đa (3.3V): 50mA.
- Bộ nhớ Flash: 32KB (ATmega328) với 2KB dùng bởi bootloader.
- SRAM: 2KB (ATmega328).
- EEPROM: 1 KB (ATmega328).
- Kích thước: 1.85cm x 4.3cm.

3.2.2 Giới thiệu Buzzer

Còi Buzzer 5VDC có tuổi thọ cao, hiệu suất ổn định, chất lượng tốt, được sản xuất nhỏ gọn phù hợp thiết kế với các mạch còi buzzer nhỏ gọn, mạch báo động.

Thông số kỹ thuật:

- Nguồn: 3.5V - 5.5V.
- Dòng điện tiêu thụ: <25mA.
- Tần số cộng hưởng: 2300Hz ± 500Hz.

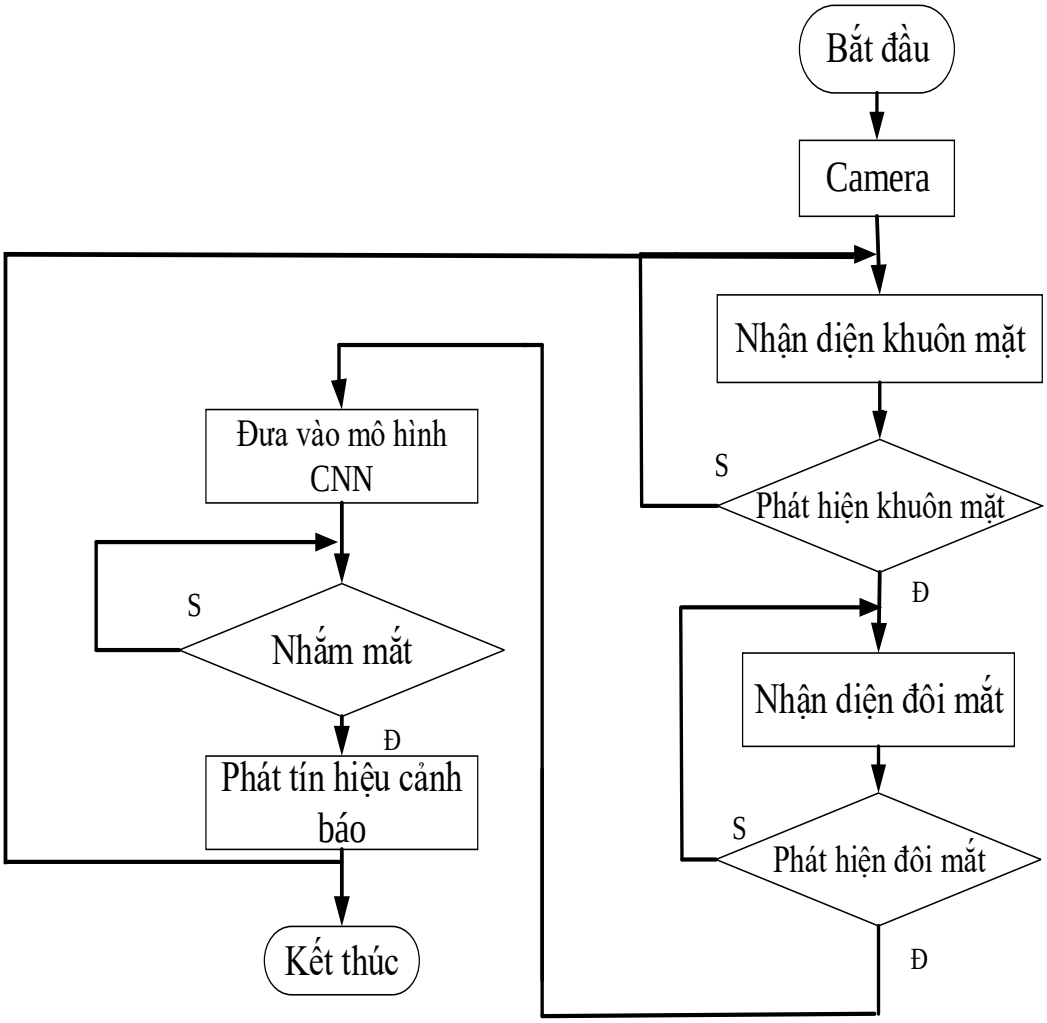
- Biên độ âm thanh: >80 dB.

- Nhiệt độ hoạt động: -20 °C đến +70 °C.

- Kích thước: Đường kính 12mm, cao 9,7mm.

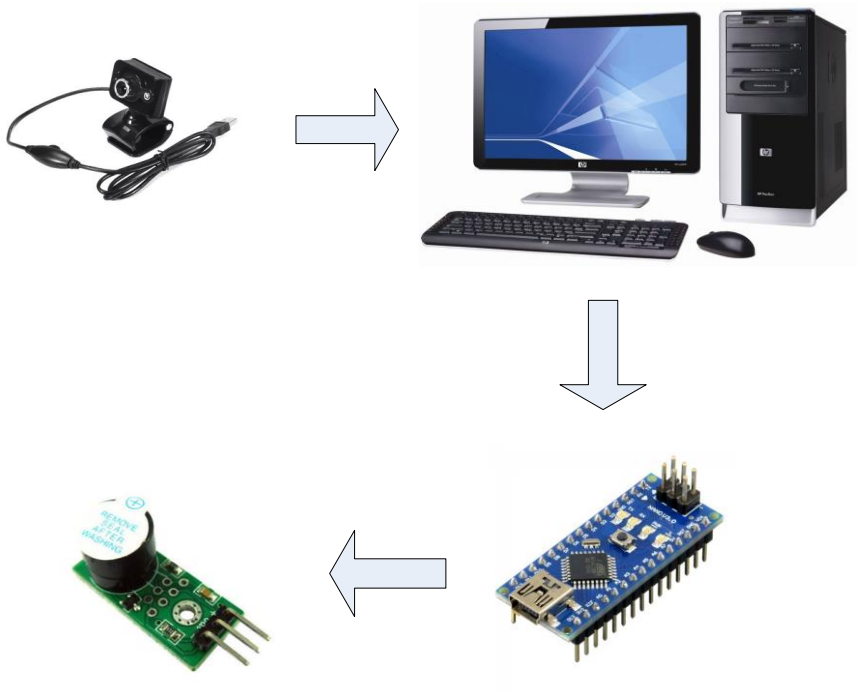
Hình 3.20 Hình ảnh về còi Buzzer.

3.3 LƯU ĐỒ THUẬT TOÁN



Hình 3.21 Lưu đồ thuật toán của hệ thống.

3.4 SƠ ĐỒ KẾT NỐI PHẦN CỨNG



Hình 3.22 Sơ đồ khối hệ thống.

3.5 CÁC BƯỚC THỰC HIỆN ĐỀ TÀI

Trong quá trình thực hiện và nghiên cứu đề tài nhóm đã thực hiện theo các bước sau:

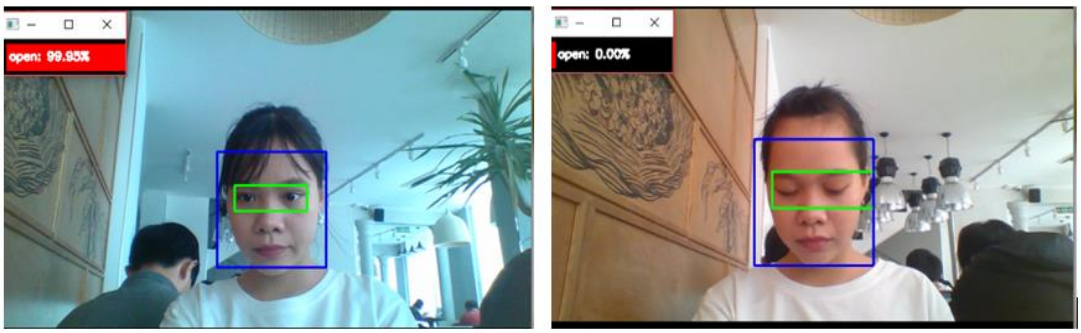
- Bước 1: Tạo tập dữ liệu dataset_eye.
- Bước 2: Tiến hành phân chia tập dataset_eye và phân tích dữ liệu để làm tín hiệu đầu vào cho mô hình CNN.
- Bước 3: Xây dựng mô hình CNN và tiến hành huấn luyện.
- Bước 4: Đánh giá mô hình.
- Bước 5: Thiết kế hệ thống.

CHƯƠNG 4

KẾT QUẢ NGHIÊN CỨU

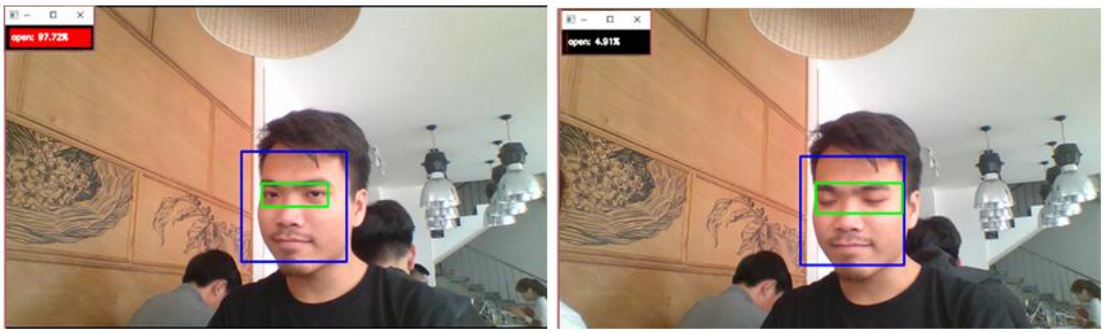
4.1 KẾT QUẢ

Sau khi thiết kế hệ thống thì chúng tôi đã thu nhận được một số kết quả của quá trình thử nghiệm như sau:



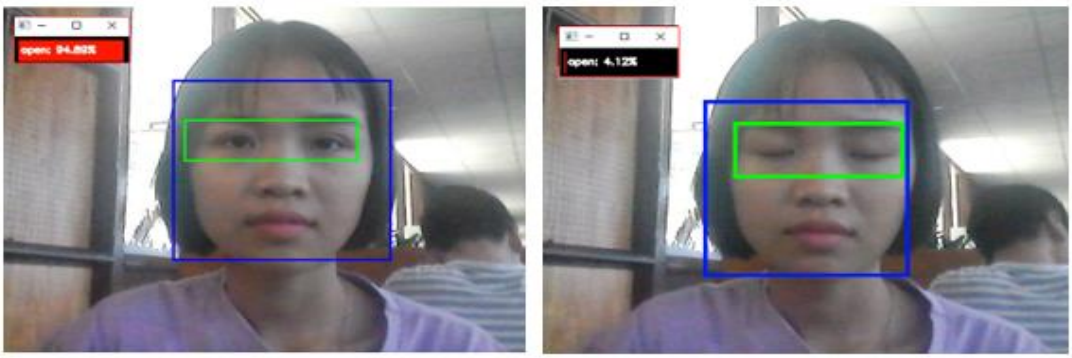
Hình 4.1 Kết quả thử nghiệm ở người số 1.

Kết quả thử nghiệm trên người thứ nhất: với điều kiện đầy đủ ánh sáng thì khi mở mắt kết quả open = 99.95%; khi nhắm mắt kết quả open = 0.00%.



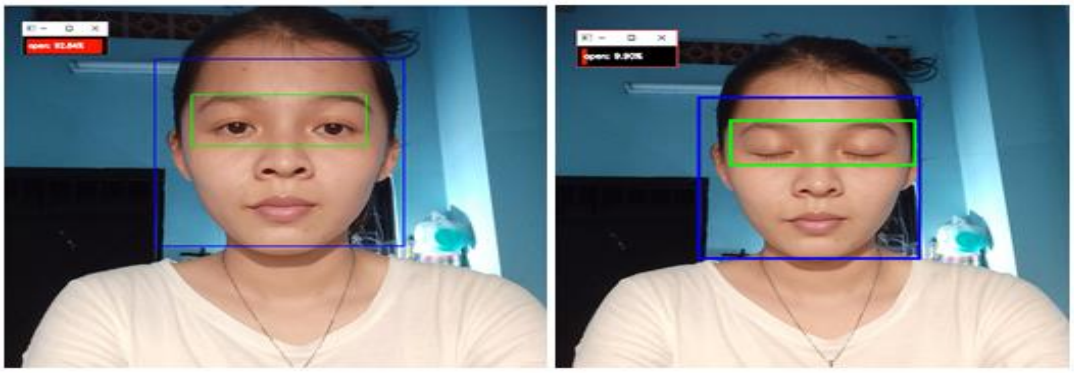
Hình 4.2 Kết quả thử nghiệm ở người số 2.

Kết quả thử nghiệm trên người thứ hai: với điều kiện đầy đủ ánh sáng thì khi mở mắt kết quả open = 97.72%; khi nhắm mắt kết quả open = 4.91%.



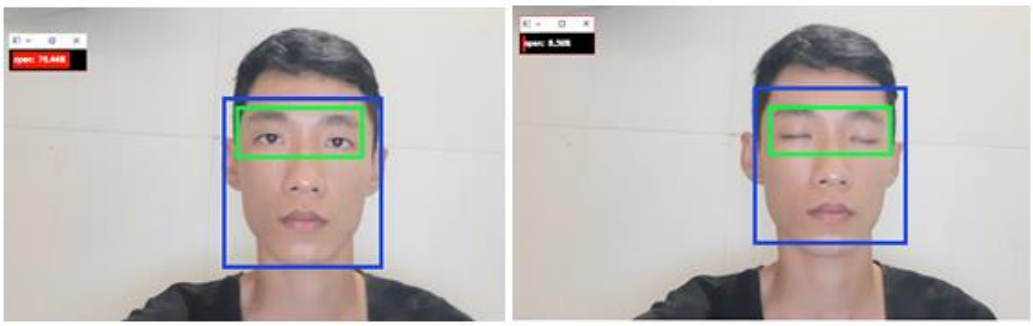
Hình 4.3 Kết quả thử nghiệm ở người số 3.

Kết quả thử nghiệm trên người thứ ba: với điều kiện ánh sáng yếu thì khi mở mắt kết quả open = 94.00%; khi nhắm mắt kết quả open = 4.12%



Hình 4.4 Kết quả thử nghiệm ở người số 4.

Kết quả thử nghiệm trên người thứ tư: với điều kiện ban đêm có đèn đủ sáng thì khi mở mắt kết quả open = 92.54%; khi nhắm mắt kết quả open = 9.92%



Hình 4.5 Kết quả thử nghiệm ở người số 5.

Kết quả thử nghiệm trên người thứ năm: với điều kiện ban đêm có đèn đủ sáng thì khi mở mắt kết quả open = 74.48%; khi nhắm mắt kết quả open = 9.58%

4.2 NHẬN XÉT

Qua quá trình thử nghiệm, cho thấy hệ thống hoạt động đúng với mục tiêu yêu cầu đặt ra từ ban đầu. Khả năng cảnh báo nhân viên ngủ gật đạt hiệu quả trên 90% tổng số người thực hiện thử nghiệm. Kết quả thể hiện ở bản sau:

Bảng 4.1 Kết quả nhận dạng trạng thái mắt

Tên	Số lần thử nghiệm	Điều kiện ánh sáng	Trạng thái	Kết quả
Thi	20 lần nhắm và mở mắt	Đủ ánh sáng	Nhắm mắt	95%
			Mở mắt	94%
		Thiếu ánh sáng	Nhắm mắt	92%
			Mở mắt	89%
Đạt	20 lần nhắm và mở mắt	Đủ ánh sáng	Nhắm mắt	96%
			Mở mắt	88%
		Thiếu ánh sáng	Nhắm mắt	92%
			Mở mắt	85%
Tinh	20 lần nhắm và mở mắt	Đủ ánh sáng	Nhắm mắt	87%
			Mở mắt	81%
		Thiếu ánh sáng	Nhắm mắt	83%
			Mở mắt	85%
Sáu	20 lần nhắm và mở mắt	Đủ ánh sáng	Nhắm mắt	98%
			Mở mắt	93%
		Thiếu ánh sáng	Nhắm mắt	91%
			Mở mắt	89%
Tân	20 lần nhắm và mở mắt	Đủ ánh sáng	Nhắm mắt	90%
			Mở mắt	98%
		Thiếu ánh sáng	Nhắm mắt	88%
			Mở mắt	92%

4.3 ĐÁNH GIÁ HỆ THỐNG

4.3.1 Ưu điểm

- Tốc độ xử lý dữ liệu của hệ thống nhanh.
- Hệ thống phát hiện được đôi mắt trong bức ảnh và video real-time, với độ chính xác trên 90%.
- Hệ thống dễ dàng cài đặt ở laptop cá nhân của từng nhân viên.
- Hệ thống có thể hoạt động liên tục trong quá trình nhân viên làm việc.
- Có hệ thống cảnh báo cho nhân viên biết khi ngủ gật, có thể đảm báo an toàn đối với công việc của nhân viên mang tính chất quyết định đến cuộc sống của người khác.
- Đề tài có tính ứng dụng cao trong thực tế phục vụ một số lĩnh vực như kiểm soát các nhân viên làm viên ở văn phòng, hỗ trợ cho các nhân viên làm việc ở chạm chốt tàu hỏa…

4.3.2 Nhược điểm

- Chất lượng hoạt động của hệ thống phụ thuộc vào chất lượng Camera trên máy tính của nhân viên.
- Hệ thống không phát hiện được đối với nhân viên mang kính râm trong giờ làm việc.
- Không phân biệt được ảnh và người thật.
- Hệ thống bị hạn chế về góc nhận diện: khi nhân viên xoay mặt đi nghiêng so với Camera.

CHƯƠNG 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT LUẬN

Hệ thống cảnh báo nhân viên ngủ gật được thiết kế với mục đích cảnh báo nhân viên làm việc ở phòng kiểm soát không lưu ngủ gật để tránh các hậu quả đáng tiếc do ngủ gật trong quá trình làm việc. Kết thúc nghiên cứu hệ thống đạt được mục đích ban đầu với độ chính xác cao và dễ dàng áp dụng cho các cơ quan công ty mà nhân viên thường xuyên làm việc trên máy tính với tính chất công việc dễ gây buồn ngủ.

5.2 HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

Với những ứng dụng về công nghệ thì không bao giờ là lỗi thời mà ngày càng phải tân tiến hơn nữa. Như vậy khi đề tài thành công thì cũng phải đưa ra một số phương pháp cải tiến hoặc hướng phát triển cho chúng trong tương lai gần và sau đây là một số hướng phát triển mà nhóm thực hiện đề tài đã đề ra như sau:

- Áp dụng hệ thống vào những văn phòng làm việc của nhân viên, các trạm chốt trực tàu hỏa, các trạm kiểm soát không lưu hoặc buồng điều khiển máy bay và thậm chí là các tài xế lái xe.
- Phát triển thêm chức năng cảnh báo như gửi tin nhắn, thực hiện cuộc gọi khi nhân viên rơi vào tình trạng ngủ gật, gửi cánh báo về trung tâm qua module Wifi/GSM và tự động cập nhật danh sách số lần nhân viên ngủ gật trên hệ thống công ty.
- Tiếp tục nghiên cứu một số thuật toán có thể đáp ứng việc nhận dạng tốt hơn để đáp ứng được nhu cầu của xã hội.
- Sử dụng một số kit có sẵn để thực thi hệ thống:

  + Kit Raspberry Pi

Hình 5.1 Hình ảnh về kit Raspberry Pi 3 model B

Phiên bản Raspberry Pi đầu tiên được phát hành tháng 2 năm 2012, và tới nay đã có nhiều phiên bản khác nhau, với sự nâng cấp của phần cứng, cũng như hướng tới những mục tiêu khác nhau. Phiên bản theo thứ tự ra mắt là: Pi A → Pi A+ → Pi 1 B → Pi 1B+ → Pi 2B → Pi Zero → Pi 3B.

Bảng 5.1 Bảng thông số phần cứng các dòng Raspberry Pi

Raspberry Pi	Model A	Model B+	Pi 2, Model B	Pi 3, Modle B	Pi Zero
Vi xử lý	Boardcom BCM2835, ARMv6 (32 bit) single core		Boardcom BCM2836, ARMv7 (32 bit) quard core	Boardcom BCM2837, ARMv8 (64 bit) quad core	Boardcom BCM2835, ARM11 (32 bit) single core
GPU	Boardcom VideoCore IV, OpenVG 1080p20, 250 MHZ			Boardcom VideoCore IV, OpenGL ES 2.0 OpenVG 1080p60, 400 MHZ	Boardcom VideoCore IV, OpenVG 1080p30, 250 MHZ

Tốc độ xử lý	700 Mhz		900 Mhz	1,2 Ghz	1,0 Ghz
Power Rattings	200mA @ 5V	600mA @ 5V	800mA @ 5V	800mA @ 5V	160mA @ 5V
RAM (chia sẻ với GPU)	256 MB SDRAM (400Mhz)	512 MB SDRAM (400Mhz)	1GB SDRAM (400Mhz)	1GB LPDDR2 (900Mhz)	512 MB SDRAM (400Mhz)
Bộ nhớ	Micro SD				
GPIO	40				
Video & audio	1080p HDMI, stero audio 3.5mm jack				1080p mini HDMI stero audio through PWM on GPIO
Kết nối	1xUSB 2.0 CSI – cổng camera DSI – cổng kết nối màn hình cảm ứng	4xUSB 2.0 10/100mb Ethernet CSI,DSI		4xUSB 2.0 10/100mb Ethernet Wifi 802.11 n Bluetooth 4.1 CSI,DSI	1 micro USB
Kích thước	65x56mm	85x56mm			65x30mm

- Ưu điểm: Giá thành rẻ, thiết kế nhỏ gọn, siêu tiết kiệm điện, GPU mạnh, phục vụ được nhiều mục đích và có khả năng hoạt động liên tục 24/7.

- Nhược điểm: CPU có cấu hình thấp, đòi hỏi người lập trình phải có kiến thức về Linux, điện tử.

+ Kit Jetson Nano

Hình 5.2 Hình ảnh về Kit Jetson Nano của Nvidia

Bảng 5.2 Thông số về Kit Jetson Nano

Feature	Specification
GPU	128-core Nvidia Maxwell ™
CPU	64-bit Quad- ARM® Cortex-A57 MPCore (1.43GHz)
Memory	4GB, 64-bit LPDDR4 (25,6 GB/s bandwidth)
networking	10/100/1000 Mbit Ethernet
Camera	12 MIPI CSI-2 DPHY 1.1 lanes (1.5 Gbps), Dual ISB (1.5 Gbps),
Model mechanical specification	69,6 mm x 45 mm, 260 pin SO-DIMM connect
Model Interfaces	
USB	1x USB 3.0, 3x USB 2.0
PCIe	1 (x1 / x2/ x4)
Display	HDMI 2.0 or DP1.2 | eDP 1.4 | DSI (1 x2) 2 simultaneous displays

Audio	2x
UART	3x
SPI	3x
I2C	3x
Model mechanical specification	69,6 mm x 45 mm, 260 pin SO-DIMM connect
Fan	PWM and tach input

- Ưu điểm: giá thành rẻ, thiết kế nhỏ gọn, CPU và GPU mạnh. Thích hợp cho các dự án về AI. Có RAM 4GB nhiều hơn Raspberry Pi (RAM 1GB) và hỗ trợ nhiều ngôn ngữ.
- Nhược điểm: không có sẵn Wifi và Bluetooth tích hợp, đòi hỏi người lập trình phải có kiến thức về Linux, điện tử.

PHỤ LỤC A

MÃ NGUỒN CHƯƠNG TRÌNH

- Mã nguồn chương trình train

```
from imutils import paths
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import os
import cv2
from sklearn.preprocessing import LabelBinarizer
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
import numpy as np
from keras.optimizers import Adam
image_path = list(paths.list_images('./dataset_eyes'))
labels = [p.split(os.path.sep)[-2] for p in image_path]
le = LabelEncoder()
labels = le.fit_transform(labels)
list_image = []
for (j, imagePath) in enumerate(image_path):
    image = cv2.imread(imagePath,cv2.IMREAD_GRAYSCALE)
    image = cv2.resize(image,(40,150))
    list_image.append(np.array(image))
list_image = np.asarray(list_image)
list_image = list_image.reshape((list_image.shape[0], list_image.shape[1],
list_image.shape[2], 1))
list_image = list_image/255
```

```
 X_train, X_test, y_train, y_test = train_test_split(list_image , labels,
test_size=0.1, random_state=41)
 X_train, X_val, y_train, y_val = train_test_split(X_train , y_train, test_size=0.1,
random_state=41)
print(X_train[0])
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150,40,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))\
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
  optimizer=Adam(0.001),
  metrics=['accuracy'])
model.summary()
H = model.fit(X_train, y_train, validation_data=(X_val, y_val),
          batch_size=32, epochs=10, verbose=1)
import matplotlib.pyplot as plt
score = model.evaluate(X_test, y_test, verbose=0)
print(score)
i=2
plt.imshow(X_test[i].reshape(150,40), cmap='gray')
y_predict = model.predict(X_test[i].reshape(1,150,40,1))
print('Giá trị dự đoán: ', y_predict > 0.5)
```

- Mã nguồn chương trình test_data

```
from keras.preprocessing.image import img_to_array
import imutils
import cv2
import numpy as np
from keras.models import load_model
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
emotion_model_path = 'model_eye/_mini_XCEPTION.10-1.00.hdf5'
model = load_model(emotion_model_path, compile=False)
EMOTION = ["open","close"]
cap = cv2.VideoCapture(0)
try:
  if not os.path.exists('dataset_eyes_opencv'):
     os.makedirs('dataset_eyes_opencv')
except OSError:
  print ('Error: Creating directory of data')

while 1:
  ret, img = cap.read()
  ret, Khuon_mat = cap.read()
  ret, Mat = cap.read()

  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  gray_Khuon_mat = cv2.cvtColor(Khuon_mat, cv2.COLOR_BGR2GRAY)
  faces = face_cascade.detectMultiScale(gray,
scaleFactor=1.3,minNeighbors=5,minSize=(30,30),flags=cv2.CASCADE_SCAL
E_IMAGE)
```

```
  for (x,y,w,h) in faces:
      cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
      roi_gray_Khuon_mat = gray[y:y+h, x:x+w]
      roi_color_Khuon_mat = img[y:y+h, x:x+w]
      eyes = eye_cascade.detectMultiScale(roi_gray_Khuon_mat,
scaleFactor=1.1,minNeighbors=5,minSize=(5,5),flags=cv2.CASCADE_SCALE
_IMAGE)
      canvas = np.zeros((40, 150, 3), dtype="uint8")
      if len(eyes) > 0:
         eyes = sorted(eyes, reverse=True,
         key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
         (ex, ey, ew, eh) = eyes
         roi_gray_eye = roi_gray_Khuon_mat[ey:ey+eh, ex:ex+3*ew]
         roi = cv2.resize(roi_gray_eye, (40, 150))
         print (roi.shape)
         frameClone = roi_gray_eye.copy()
         roi = roi.astype("float") / 255.0
         roi = img_to_array(roi)
         roi = np.expand_dims(roi, axis=0)
 cv2.rectangle(roi_color_Khuon_mat,(ex,ey),(ex+3*ew,ey+eh),(0,255,0),2)
      preds = model.predict(roi)[0]
      print('Giá trị dự đoán: ', preds > 0.5)
      emotion_probability = np.max(preds)
      label = EMOTION[preds.argmax()]
  for (i, (emotion, prob)) in enumerate(zip(EMOTION, preds)):
      text = "{}: {:.2f}%".format(EMOTION, prob * 100)
          time.sleep (0.1)
       if ((len(eyes) > 0)&(len(faces) > 0) ):
        if ((preds > 0.5) == True) :
```

```
            dem = 10
            print ('Giá trị dem: ', dem)
        elif (((prob*100) < 15) & (dem > 0) & (( preds > 0.5) == False)) :
            dem = dem - 1
            print('Giá trị dem: ', dem)
        elif (((prob*100) < 15) & (dem == 0) & ((preds > 0.5) == False)):
            for i in range (3):
                with serial.Serial ('COM4', 9600, timeout=1) as ser:
                    time.sleep (0.2)
                    ser.write (b'H')
                    time.sleep (0.1)
                    ser.write (b'L')
            dem = 10
        w = int(prob * 150)
        cv2.rectangle(canvas, (7, (i * 35) + 5),
        (w, (i * 35) + 35), (0, 0, 255), -1)
        cv2.putText(canvas, text, (10, (i * 35) + 23),
        cv2.FONT_HERSHEY_SIMPLEX, 0.45,
        (255, 255, 255), 2)
        cv2.putText(frameClone, label, (ex, ey - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
        cv2.rectangle(frameClone, (ex, ey), (ex + 3*ew, ey + eh), (0, 0, 255), 2)
    cv2.imshow('img',img)
    cv2.imshow('Mat',roi_gray_eye)
    cv2.imshow('Mat', frameClone)
    cv2.imshow("Probabilities", canvas)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

PHỤ LỤC B

HƯỚNG DẪN SỬ DỤNG HỆ THỐNG

Yêu cầu cơ bản để sử dụng hệ thống phải cài phần mềm Python và một số thư viện hỗ trợ như:

- Thư viện opencv.
- Thư viện tensorflow.
- Thư viện numpy.
- Thư viện os.
- Thư viện imutils.
- Thư viện sklearn và một số thư viện đính kèm khác.
- Thư viện Keras.

Sau khi hoàn tất quá trình cài đặt các gói thư viện trên Python, chúng ta sẽ tải thư mục model_eye.rar từ địa chỉ sau:

https://drive.google.com/file/d/1DGRrOyxR0FPoPc9SqbCxBfjuwc0z3YSD/view?usp=sharing

Tiếp theo, giải nén file model_eye.rar thành thư mục model_eye và chạy test_data. Nhưng vậy là đã hoàn thành hướng dẫn sử dụng hệ thống.

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Quang Hoan, *Xử Lý Ảnh*, Hà Nội: Học Viện Công Nghệ Bưu Chính Viễn Thông, 2006.

[2]. TS. Nguyễn Thanh Hải, *Xử Lý Ảnh*, Tp Hồ Chí Minh: Đại học Quốc Gia TP. Hồ Chí Minh, 2014.

[3]. Viola, Paul and Michael J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001. Volume: 1, pp.511–518.

[4]. M.Gopi Krishna, A. Srinivasulu, " Face Detection System On AdaBoost Algorithm Using Haar Classifiers", *International Journal of Modern Engineering Research (IJMER)*, Vol. 2, Issue. 5, Sep.-Oct. 2012 pp-3556-3560.

[5]. Nguyễn Thái Nghe, Nguyễn Văn Đồng, Võ Hùng Vĩ. "Một giải pháp trong xây dựng hệ thống hỗ trợ giữ xe thông minh", *Tạp chí Khoa học Trường Đại học Cần Thơ*, số 35, pp.17-30, 2014.

[6]. Lê Mạnh Tuấn. *Phát hiện mặt người trong ảnh và ứng dụng*. Đồ án tốt nghiệp, trường đại học công nghệ, 2009.

[7]. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", *In Computer Vision and Pattern Recognition*, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I– 511. IEEE, 2001.

[8]. Nils J. Nilsson, *Introduction To Machine Learning*. Stanford, California: November 3, 1998.

[9]. Nguyễn Quang vinh, Trần Đăng Cát, Đỗ Mạnh Hùng, *Sinh học 8*. Việt Nam: nhà xuất bản giáo dục việt nam, năm 2011.

[10]. LeCun Y, Jackel L, Boser B, Denker J, Graf H, Guyon I, Henderson D, Howard R, and Hubbard W. Handwritten digit recognition : Applications of neural networks chipsand automatic learning. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11]. Nagi, J., F. Ducatelle, G. A. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, L. M. Gambardella. "Max-Pooling Convolutional Neural Networks for Vision-based Hand Gesture Recognition". *IEEE International Conference on Signal and Image Processing Applications (ICSIPA2011),* 2011.

[12]. Kalyani Adawadkar, "Python Programming-Applications and Future", *International Journal of Advance Engineering and Research Development (IJAERD) Special Issue SIEICON-2017*, e-ISSN: 2348 - 4470, April -2017.

[13]. Bhor.Dipali.A, Dr.G.U.Kharat, "Object detection & tracking with Particle filter based video surveillance system using raspberry pi*," International Journal of Advanced Scientific and Technical Research*, Issue 6 volume 6, November-December 2016.

[14]. François Chollet, *Deep Learning with Python*. Shelter Island, NY 11964: Manning Publications, 2018.