HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

NGUYỄN QUANG HUY

HỆ THỐNG NHẬN DIỆN KHUÔN MẶT QUA CAMERA

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI - 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

NGUYỄN QUANG HUY

HỆ THỐNG NHẬN DIỆN KHUÔN MẶT QUA CAMERA

CHUYÊN NGÀNH: KHOA HỌC MÁY TÍNH

MÃ SỐ: 8.48.01.01

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN ĐÌNH HÓA

HÀ NỘI - 2020

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn này là công trình nghiên cứu của cá nhân tôi, được thực hiện trên cơ sở nghiên cứu lý thuyết, thực tế dưới sự hướng dẫn của TS. Nguyễn Đình Hóa.

Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Hà Nội, ngày 16 tháng 11 năm 2020

Học Viên Thực Hiện

Nguyễn Quang Huy

LỜI CẢM ƠN

Em xin chân thành cảm ơn TS. Nguyễn Đình Hóa đã tận tình chỉ dạy và hướng dẫn cho em trong việc lựa chọn đề tài, thực hiện đề tài và viết báo cáo luận văn, giúp em hoàn thành tốt luận văn này.

Em xin cám ơn các thầy cô giáo trường Học viện Công nghệ Bưu chính Viễn thông đã tận tình dạy dỗ và chỉ bảo em trong suốt 2 năm học.

Cuối cùng em xin cám ơn gia đình, bạn bè, đồng nghiệp, những người đã luôn bên cạnh động viên em những lúc khó khăn và giúp đỡ em trong suốt thời gian học tập và làm luận văn, tạo mọi điều kiện tốt nhất cho em để có thể hoàn thành tốt luận văn của mình.

Mặc dù đã cố gắng hoàn thành nghiên cứu trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Em kính mong nhận được sự góp ý, thông cảm của thầy cô và các bạn. Em xin chân thành cảm ơn!

Hà Nội, ngày 12 tháng 11 năm 2020

Sinh viên

NGUYỄN QUANG HUY

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

Tên viết tắt	Tiếng Anh	Tiếng Việt
DT	Decision Tree	Cây quyết định
ID3	Iterative Dichotomiser 3	Thuật toán ID3
RF	Random Forest	Rừng ngẫu nhiên
MLP	Multi layer perceptron	Mạng nơ-ron truyền thẳng nhiều lớp
CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
PCA	Principal Components Analysis	Phân tích thành phần chính
LDA	Linear Discriminant Analysis	Phân tích sự khác biệt tuyến tính
MLP	Multilayer perceptron	Mạng nơron truyền thẳng nhiều lớp
ANN	Artificial Neural network	Mạng nơron nhân tạo

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

MỞ ĐẦU

Công nghệ thông tin ngày càng phát triển và đã là một thành phần không thể thiếu trong hầu hết mọi lĩnh vực trên thế giới. Những người máy thông minh được con người tạo ra đã có khả năng phân tích và xử lý được các công việc của con người một cách tự động và đem lại lợi ích kinh tế rất lớn. Trong thời gian gần đây, một trong những bài toán được nghiên cứu, ứng dụng nhiều nhất vào trong cuộc sống đó là bài toán nhận diện. Tuy mới xuất hiện chưa lâu nhưng nó đã rất được quan tâm vì tính ứng dụng thực tế của bài toán như nhận dạng chữ viết, nhận dạng giọng nói, nhận dạng hình dáng, nhận diện khuôn mặt. Trong đó, bài toán nhận diện khuôn mặt là một chủ đề đang được khá nhiều nhà đầu tư, doanh nghiệp quan tâm đến. Dù đã được nghiên cứu từ rất lâu nhưng bài toán nhận diện khuôn mặt vẫn đang gặp phải nhiều thách thức và vẫn chưa có phương pháp cụ thể nào có thể giải quyết hết các vấn đề trong bài toán này.

Bài toán nhận diện khuôn mặt là một trong những chủ đề đang được quan tâm nhiều nhất. Ứng dụng từ bài toán này được áp dụng trong rất nhiều lĩnh vực khác nhau. Các ứng dụng liên quan đến nhận diện khuôn mặt có thể kể như: tra cứu thông tin tội phạm, phát hiện tội phạm tại các nơi công cộng, tìm người lạc, điểm danh học sinh ...

Từ những phân tích và khảo sát ở trên, em nhận thấy hệ thống nhận diện khuôn mặt rất có ý nghĩa trong thực tiễn cuộc sống và em xin chọn đề tài nghiên cứu "Hệ thống nhận diện khuôn mặt qua camera". Kết quả của luận văn hướng tới việc xây một hệ thống nhận diện khuôn mặt có khả năng mở khả năng mở rộng cao, dễ dàng tích hợp.

Nội dung luận văn được trình bày trong ba chương với các nội dung chính như sau:

Chương 1. Tổng quan về nhận diện khuôn

Chương này sẽ trình bày một số nội dung nền tảng về bài toán nhận diện khuôn mặt, các ứng dụng tương tác người máy liên quan đến nhận diện khuôn mặt, và một số kỹ thuật hay được sử dụng trong bài toán nhận diện khuôn mặt. Nội dung của

chương bao gồm ba phần chính. Phần đầu tiên giới thiệu tổng quan về bài toán nhận diện khuôn mặt cùng với các ứng dụng thực tế. Phần thứ hai giới thiệu một số phương pháp trong nhận diện khuôn mặt thường được áp dụng trong thực tế và nghiên cứu. Phần cuối cùng giới thiệu một số mạng tích chập thường được sử dụng trong bài toán nhận diện khuôn mặt.

Chương 2. Hệ thống nhận diện khuôn mặt dựa trên mạng nơ ron tích chập

Các kỹ thuật cơ bản được sử dụng để xây dựng hệ thống nhận diện khuôn mặt của luận văn được trình bày trong chương này. Nội dung của chương trình bày về các phương pháp trích chọn đặc trưng phục vụ quá trình nhận diện khuôn mặt, phương pháp định danh khuôn mặt và mô hình học máy được sử dụng để phân loại dữ liệu nhận diện khuôn mặt. Chương này cũng bao gồm các thông tin về mô hình, kiến trúc mạng nơ ron tích chập Inception-ResNet sử dụng cho việc trích chọn đặc trưng khuôn mặt của luận văn.

Chương 3. Thử nghiệm và đánh giá

Chương này mô tả chi tiết về bộ dữ liệu được sử dụng, cùng các kịch bản và kết quả các quá trình huấn luyện mô hình. Các kết quả thực nghiệm kèm theo đánh giá mô hình sau khi huấn luyện cũng được trình bày trong chương này.

Nội dung của luận văn được kết thúc bằng phần Kết luận, trong đó trình bày tóm lược các nội dung và kết quả đã đạt được trong luận văn, từ đó đề xuất các hướng phát triển trong tương lai.

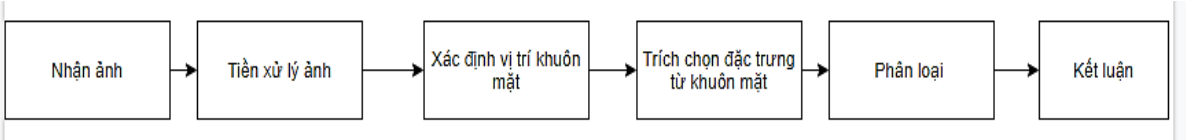
CHƯƠNG 1. TỔNG QUAN VỀ NHẬN DIỆN KHUÔN MẶT

1.1 Tổng quan về nhận diện khuôn mặt cùng với các ứng dụng thực tế của các kỹ thuật nhận dạng khuôn mặt.

1.1.1 Tổng quan

Nhận diện khuôn mặt là một bài toán tổng hợp. Trong đó ta cần các mô đun quan trọng như như xác định vị trí khuôn mặt, trích chọn đặc trưng rồi phân loại. Từ đó ta có thể xác định danh tính người trong ảnh.

1.1.2 Kiến trúc tổng quát hệ thống nhận diện



Hình 1.1. Kiến trúc tổng quát về hệ thống nhận diện

Nhận ảnh là bộ phận thu nhận ảnh. Ảnh ở đây có thể nhận được qua camera màu hoặc đen trắng. Tiền xử lý ảnh là bước tiền xử lý để nâng cao chất lượng ảnh đầu vào. Vì ảnh thu nhận được có thể bị nhiễu hoặc độ tương phản thấp gây ảnh hưởng đến việc trích chọn đặc trưng cũng như xác định vị trí khuôn mặt. Tiếp đến là xác định vị trí khuôn mặt. Ở bước này hệ thống sẽ xác định vị trí khuôn mặt và các điểm mắt, mũi, miệng. Trích chọn đặc trưng từ khuôn mặt sẽ thực hiện lấy khuôn mặt trong ảnh gốc để thực hiện trích chọn đặc trưng. Phân loại là bước thực hiện phân loại đặc trưng từ đó sẽ định danh được khuôn mặt đầu vào là ai. Kết luận là từ kết quả phân loại sẽ đưa ra kết quả nhận diện.

1.1.3 Ứng dụng

Bài toán nhận diện khuôn mặt có rất nhiều ứng dụng trong cuộc sống. Trong đó, một số ứng dụng tiêu biểu không thể không kể đến của bài toán này là hệ thống phát hiện, truy vết tội phạm, hệ thống tìm trẻ lạc, hệ thống điểm danh, chấm công hay ứng dụng nhận diện đối tác, khách hàng VIP. Các bài toán trên hiện đang được sử dụng rất nhiều và thành một phần không thể thiếu trong cuộc sống của mỗi người.

1.2 Một số phương pháp trong nhận diện khuôn mặt thường được áp dụng trong thực tế và nghiên cứu

1.2.1 Phân tích thành phần chính (PCA)

a. Giới thiệu

PCA (Principal Components Analysis) [1] là một thuật toán được sử dụng để tạo ra một ảnh mới từ ảnh ban đầu. Ảnh mới này có kích thước nhỏ hơn nhiều so với ảnh ban đầu nhưng vẫn mang những đặc trưng cơ bản nhất của ảnh cần nhận dạng. Trong nghiên cứu [2], thuật toán PCA thường được sử dụng cho việc trích chọn đặc trưng khuôn mặt. PCA không cần quan tâm đến việc tìm ra các đặc điểm cụ thể của thực thể cần nhận dạng và mối quan hệ giữa các đặc điểm đó. Tất cả các chi tiết đó đều được thể hiện ở ảnh mới được tạo ra từ PCA.

b. Thuật toán PCA

Không gian mới được tạo bởi PCA được cấu thành từ k vectơ đơn vị có chiều là N. Mỗi vectơ được gọi là một Eigenface. Phép biến đổi :

$$A=\begin{bmatrix} a1 \\ a2 \\ \vdots \\ an \end{bmatrix} \rightarrow W = \begin{bmatrix} w1 \\ w2 \\ \vdots \\ wk \end{bmatrix} \quad \text{với } K<<N$$

$$W=T.A \tag{1.1}$$

Với T là ma trận chuyển đổi, T có kích thước K x N. Gọi M là số ảnh đầu vào, mỗi ảnh được chuyển thành vectơ N chiều. Từ (1.1) ta có tập hợp đầu vào

$$X=\{x_1, x_2,\ldots,x_M\} \ (x_i \in R^N) \tag{1.2}$$

Trung bình của các vectơ đầu vào :

$$X_{tb} = \frac{1}{M} \sum_{i=1}^{M} xi \tag{1.3}$$

Sai lệch so với tâm:

$$\Phi_i = x_i - x_{tb} \tag{1.4}$$

Gọi A=[$\Phi_1, \Phi_2, \ldots, \Phi_M$] ta có ma trận tương quan của A là :

$$C=\frac{1}{M}\sum_{i=1}^{M}\Phi i\Phi_{jT} = A.A^T \quad (1.5)$$

Gọi các giá trị riêng của C là: $\lambda_1, \lambda_2, \ldots, \lambda_n$ sắp xếp theo thứ tự giảm dần, tương ứng với N vectơ riêng $u_1, u_2, \ldots, u_N$. Các vectơ riêng này trực giao từng đôi một, Mỗi vectơ riêng u_i được gọi là một eigenface. Tập hợp các vectơ ban đầu được biểu diễn trong không gian tạo bởi n eugenface theo mô tả:

$$x\text{-}x_{tb} = w_1u_1 + w_2u_2 + \ldots + w_Nu_N = \sum_{i=1}^{N} wiui \quad (1.6)$$

Chọn lấy K vectơ riêng u tương ứng với K giá trị riêng λ lớn nhất, ta có:

$$x\text{-}x_{tb} = w_1u_1 + w_2u_2 + \ldots + w_Nu_N = \sum_{i=1}^{K} wiui \quad \text{với } K<<N \quad (1.7)$$

Vectơ các hệ số khai triển [*w1, w2, ..., wk*] chính là biểu diễn mới của ảnh được tạo ra trong không gian PCA. Ảnh mới vẫn giữ được các đặc điểm chính của ảnh đầu vào. Vectơ [$w_1, w_2, \ldots, w_K$] được tính theo công thức:

$$\begin{bmatrix} w1 \\ w2 \\ \vdots \\ wk \end{bmatrix} = \begin{bmatrix} u1T \\ u2T \\ \vdots \\ uKT \end{bmatrix} (x\text{-}x_{tb}) = U^T.(x\text{-}x_{tb}) \quad (1.8)$$

Vấn đề cần giải quyết ở đây là ma trận tương quan $C=A.A^T$ có kích thước N^2. Với N=180x200=36000, khối lượng tính toán sẽ rất lớn. Do đó, để tính được các eigenface mà không cần tính cả ma trận C, người ta đưa ra phương pháp tính nhanh dựa vào vectơ riêng và giá trị riêng của ma trận $L=A^T.A$ có kích thước MxM với M là số ảnh đầu vào. Gọi v_i, μ_i lần lượt là vectơ riêng và giá trị riêng của ma trận L:

$$A^T.A.v_i = \mu_i.\ v_i \quad (1.9)$$

Nhân cả 2 vế với A, ta có :

$$A.A^T.A.v_i = \mu_i.\ A.v_i \quad (1.10)$$

Ta thấy $A.v_i$ chính là vectơ riêng của $C=A.A^T$ ứng với giá trị riêng μ_i. Thuật toán PCA thường được sử dụng để trích chọn vectơ đặc trưng. Không gian chứa vectơ này có số chiều là N=w*h với mỗi bức ảnh có kích thước là w*h pixels. Các bước để trích chọn đặc trưng là tạo một tập X gồm M ảnh (ảnh học), mỗi ảnh có kích thước N, các ảnh được chuyển thành vectơ N chiều.

$$X = \{x_1, x_2, \ldots, x_M\} \quad (1.11)$$

Từ đó ta sẽ tính trung bình của tập trên:

$$X_{tb} = \frac{1}{M} \sum_{i=1}^{M} xi \quad (1.12)$$

Bước tiếp theo là tính sai lệch của ảnh đầu vào với giá trị trung bình trên:

$$\Phi_i = x_i - x_{tb} \quad (1.13)$$

Cuối cùng là tìm một tập M vectơ trực giao u biểu diễn phân bố mạnh nhất của tập dữ liệu X. Tập các vectơ u được gọi là eigenface của tập dữ liệu học.Xây dựng các ảnh mới v_i theo M vectơ u :

$$v_i = u_i^t \Phi_i$$

$$\Omega=[v_1, v_{2,\ldots}, v_M]^T \quad (1.14)$$

Trong đó, $v_i = u_i^t \Phi_i$ là vectơ đặc tính của ảnh thứ I trong không gian mới. Ω ở đây là tập các eigenface, các thành phần cơ bản cho bức ảnh cần nhận dạng. Sau khi trích chọn được các vectơ đặc tính, cần đối chiếu vectơ này với cơ sở dữ liệu, từ đó đưa ra kết quả nhận dạng. Trong bài toán, kết quả nhận dạng sẽ là nhận biết được hoặc chưa nhận biết được.

1.2.2 Phân tích sự khác biệt tuyến tính(LDA)

a. Giới thiệu

LDA được coi là một phương pháp giảm chiều dữ liệu (dimensionality reduction), và cũng có thể được coi là một phương pháp phân lớp (classification), và cũng có thể được áp dụng đồng thời cho cả hai, tức giảm chiều dữ liệu sao cho việc

phân lớp hiệu quả nhất. Trong nghiên cứu [3], [4] cũng chỉ rõ đây là một thuật toán tốt được sử dụng cùng với các phương pháp khác như mạng nơ-ron nhân tạo hay PCA trong bài toán nhận diện khuôn mặt.

b. Thuật toán LDA

Ý tưởng cơ bản của LDA là tìm một không gian mới với số chiều nhỏ hơn không gian ban đầu sao cho hình chiếu của các điểm trong cùng 1 class lên không gian mới này là gần nhau trong khi hình chiếu của các điểm của các lớp khác nhau là khác nhau. Phương pháp LDA phân loại các lớp chưa biết thành các lớp đã biết, mà ở đó các khuôn mặt tạo thành một lớp và sự khác biệt giữa các khuôn mặt trong một lớp là rất nhỏ. Cả PCA chọn cách thống kê lấy mẫu, chọn lọc để nhận diện khuôn mặt.

Thuật toán LDA dựa trên phân tích phân loại phi tuyến của Fisher là phương pháp tính toán chuyển đổi tối đa hóa sự phân tán giữa các lớp trong khi giảm thiểu phân tán trong lớp. Giải sử ta có các lớp C với μ_i là vectơ trung bình của các lớp i với i = 1, 2,…C. M_i là số lượng mẫu trong lớp i.

$$\boldsymbol{\mu} = \frac{1}{c}\sum_{y=1}^{C} \boldsymbol{\mu}_y \tag{1.15}$$

Gọi S_w là ma trận tán xạ nội lớp (các phần tử trong lớp) và S_B là ma trận tán xạ tương hổ của các lớp thuộc C.

$$S_w = \sum_{i=1}^{C}\sum_{j=1}^{Mi} (y_j - \mu_i)(y_j - \mu_i)^T$$

$$S_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T \tag{1.16}$$

Phương pháp LDA sẽ tìm giá trị W để cực đại hóa hàm mục tiêu $H_{(W)}$:

$$H_{(W)} = \frac{WS_BW^T}{WS_WW^T} \tag{1.17}$$

LDA tính toán chuyển đổi tối đa hóa sự phân tán giữa các lớp trong khi giảm thiểu phân tán trong lớp.

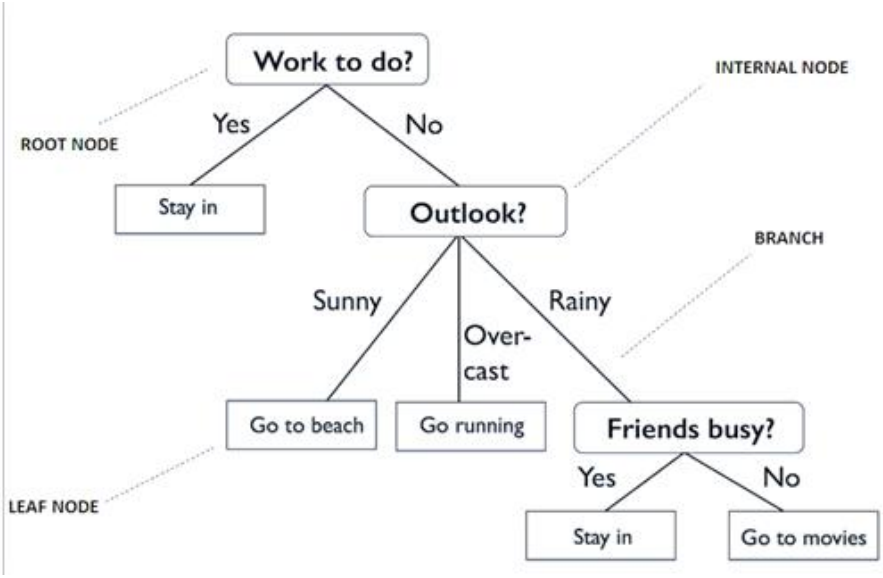
1.2.3 Cây quyết định (Decision Tree)

a. Giới thiệu

Việc quan sát, suy nghĩ và đưa ra các quyết định của con người thường được bắt đầu từ các câu hỏi. Trong học máy cũng có mô hình đưa ra quyết định dựa vào các câu hỏi như cây quyết định. Cây quyết định (Decision Tree) là một trong những thuật toán phổ biến của học máy thuộc nhánh học có giám sát. Decision Tree ra đời từ những năm 1975 từ một tác giả có tên Ross Quinlan. Thuật toán này là tiền đề để ra đời những phương pháp dự báo theo dòng Tree-based method như là: Random Forest, Bagging, AdaBoost, Gradient Boosting Machine. Mô hình cây quyết định thuộc nhóm các bài toán học có giám sát (supervised learning). Mô hình này có thể sử dụng vào cả hai loại bài toán phân loại (classification) và hồi quy (regression) theo [5]. Hiện nay, mô hình cây quyết định vẫn còn được sử dụng rất nhiều trong các nghiên cứu cũng như ứng dụng [6].

b. Thành phần

Một cây quyết định được bao gồm 4 thành phần như sau: root node, internal node, leaf node, dept. Trong đó root node là nhánh chia đầu tiên của cây quyết định. Internal node là các nhánh chia tiếp theo của cây quyết định. Leaf node là các nhánh cuối cùng của một quyết định. Dept sẽ quy định tầng của cây



Hình 1.2. Thành phần cây quyết định

c. Hàm số entropy

Trên thực tế ta sẽ sẽ có một bảng dữ liệu với rất nhiều biến. Decision Tree sẽ sử dụng một vài chỉ số để đưa ra việc xác định câu hỏi và thứ tự các biến nào chia dữ liệu để tạo ra Decision Tree có khả năng phân loại tốt nhất. Các hệ số này là Gini và Cross-Entropy.

Để tìm nghiệm cho các bài toán có nhiều thuộc tính và mỗi thuộc tính có nhiều giá trị khác nhau thì ta sẽ sử dụng một phương pháp đơn giản thường được sử dụng là tại mỗi bước, một thuộc tính tốt nhất sẽ được chọn ra dựa trên một tiêu chuẩn nào đó. Với mỗi thuộc tính được chọn, ta chia dữ liệu vào các child node tương ứng với các giá trị của thuộc tính đó rồi tiếp tục áp dụng phương pháp này cho mỗi child node. Trong đó, hành động chọn ra thuộc tính tốt nhất ở mỗi bước như trên gọi là cách chọn tham lam (greedy). Cách chọn tham lam này có thể không phải là tối ưu nhưng nó đem lại kết quả cũng khá tốt cho bài toán này. Child node sẽ chứa những câu trả lời tương ứng với dữ liệu sau mỗi câu hỏi. Câu hỏi ở đây được coi như là một thuộc tính và câu trả lời sẽ là giá trị của thuộc tính đó. Để đánh giá chất lượng của một cách phân chia, chúng ta cần đi tìm một phép đo. Và đó là hàm entropy.

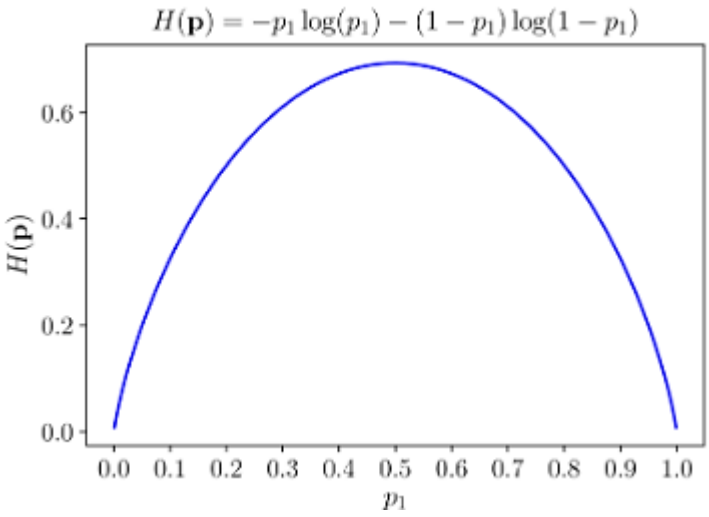
Giả sử ta có một dãy các giá trị $(x_1, x_2, \dots, x_n)$. Công thức để tính xác suất sao cho x nhận các giá trị này là

$$p_i = p(x = x_i) \quad \text{với} \sum_{i=1}^{n} p_i = 1 \geq p_i \geq 0 \tag{1.18}$$

Ký hiệu hàm này là $p = (p_1, p_2, \dots, p_n)$. Từ công thức (1.18) ta có công thức Entropy của p là:

$$H(p) = -\sum_{i=1}^{n} p_i \log(p_i) \tag{1.19}$$

Trong đó log là logarit tự nhiên, $0\log(0) = 0$ là sự quy ước từ trước. Ví dụ ta có n=2 được cho trên.Trường hợp p là tinh khiết nhất khi entropy được tính của phân phối này là (p)=0 khi tức hai giá trị p_i lần lượt bằng 0 và 1. Khi p là có tỉ lệ lẫn lớn nhất nhất, tức cả hai giá trị p_i=0.5 và hàm entropy sẽ đạt giá trị tối đa.



Hình 1.3. Đồ thị hàm entropy

Tổng quát khi với n > 2, khi p_i= 1 thì hàm entropy đạt giá trị nhỏ nhất,khi tất cả các p_i bằng nhau thì entropy đạt giá trị tối đa. Những tính chất này của hàm entropy khiến nó được sử dụng trong việc đo độ vẩn đục của một phép phân chia của ID3. Vì lý do này, ID3 còn được gọi là entropy-based decision tree.

d. Thuật toán ID3

Trong ID3, tổng có trọng số của entropy tại các leaf-node sau khi xây dựng decision tree được coi là hàm mất mát của decision tree đó. Các trọng số ở đây được tính trên với số điểm dữ liệu tại mỗi node. Thuật toán ID3 sẽ tìm ra cách phân chia

hợp lý trong đó việc chọn thứ tự thuộc tính cũng khá quan trọng sao cho hàm mất mát cuối cùng đạt giá trị thấp nhất có thể. Việc này được thực hiện bằng cách chọn ra thuộc tính tốt nhất. Sau đó sử dụng thuộc tính đó để phân sẽ giúp cho việc giá trị entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một cây quyết định bằng thuật toán ID3 có thể giải bằng cách chia thành các bài toán nhỏ, trong mỗi bài toán, ta sẽ chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ ở đây tương ứng với phân chia dữ liệu trên node gốc.

Giả sử ta có C lớp khác nhau và ta đang làm việc với một non-leaf node với các điểm dữ liệu tạo thành một tập S với số phần tử là: |S|=N với N_c điểm sẽ thuộc vào lớp c. Xác suất để mỗi điểm dữ liệu rơi thuộc lớp c được tính xấp xỉ bằng $\frac{N_c}{N}$. Qua đó, ta tính được entropy tại node này với công thức:

$$H(\mathcal{S}) = -\sum_{c=1}^{C} \frac{N_c}{N} \log(\frac{N_c}{N}) \tag{1.20}$$

Coi như thuộc tính phân chia tốt nhất được chọn là x thì dựa trên x, các điểm dữ liệu trong S được phân ra thành K child node $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$. Trong đó với số điểm tại mỗi child node lần lượt là $m_1, m_2, \dots, m_K$ và ta có công thức:

$$H(x, \mathcal{S}) = \sum_{k=1}^{K} \frac{m_k}{N} H(\mathcal{S}_k) \tag{1.21}$$

Tiếp đến là information gain dựa trên thuộc tính x:

$$G(x, \mathcal{S}) = H(\mathcal{S}) - H(x, \mathcal{S}) \tag{1.22}$$

Trong ID3, trên các node, ta có công thức để chọn ra các thuộc tính tốt nhất:

$$x^* = \arg\max_x G(x, \mathcal{S}) = \arg\min_x H(x, \mathcal{S}) \tag{1.23}$$

Thuộc tính này sẽ đem lại information gain đạt giá trị tối đa.

e. Thuật toán C4.5

C4.5 là thuật toán phân lớp dữ liệu dựa trên cây quyết định được sử dụng phổ biến và đem lại hiệu quả cao trong những bài toán khai phá dữ liệu có kích thước trung bình, nhỏ. Lý do C4.5 thích hợp với những dữu liệu vừa và nhỏ vì thuật toán này sử dụng bộ nhớ để lưu trữ dữ liệu trên và sắp xếp lại dữ liệu tại mỗi node trong

quá trình phát triển cây quyết định. C4.5 có khả năng biểu diễn lại cây quyết định dưới dạng một danh sách điều kiện if-else. Thuật toán này được xây dựng giúp cho việc giảm bớt kích thước tập luật, khiến cho các tập luật trở nên đơn giản hơn mà độ chính xác so với nhánh tương ứng cây quyết định là tương đương.

Mã giả của thuật toán C4.5:

Bước 1: Tìm tần số tương đối của lớp và kiểm tra các case cơ bản

Bước 2: Từ các thuộc tính A tìm information Gain

Bước 3: Chọn thuộc tính B tốt nhất với độ đo lựa chọn thuộc tính tối đa

Bước 4: Dùng thuộc tính B tìm được đó làm thuộc tính cho node chia cắt cây.

Bước 5: Đệ quy, lặp lại các hành động trên với các danh sách phụ. Danh sách này là danh sách được tạo ra sau khi phân chia danh sách theo thuộc tính B.

Thuật toán C4.5 có sử dụng cơ chế chọn thuộc tính để kiểm tra trên mỗi node. Thuật toán cũng có cơ chế xử lý riêng với các trường hợp dữ liệu ít hoặc thiếu giúp tránh khỏi việc quá khớp và thêm vào đó C4.5 còn có cơ chế cắt tỉa giúp đem lại hiệu quả cao hơn. Trong thuật toán ID3, Information Gain được sử dụng làm độ đo. Tuy nhiên, phương pháp này lại ưu tiên những thuộc tính có số lượng lớn các giá trị mà ít xét tới những giá trị nhỏ hơn. Do vậy, để khắc phục nhược điểm trên, ta sử dụng độ đo Gain Ratio (trong thuật toán C4.5).

Hai độ đo được sử dụng trong C4.5 là information gain và gain ratio. Ta có biểu Tần số tương đối (Relative Frequency) đối với các trường hợp S thuộc về lớp Cj kí hiệu là RF(Cj,S):

$$\mathrm{RF(Cj,S)} = \frac{|\mathrm{Sj}|}{|\mathrm{S}|} \tag{1.24}$$

Trong đó kí hiệu |Sj| là kích thước tập các trường hợp mà giá trị phân lớp là Cj với kích thước tập dữ liệu huấn luyện là |S| . Công thức để tính chỉ số thông tin cần thiết cho sự phân lớp cho tập S là:

$$\mathrm{I(S)} = -\sum_{\mathrm{j=1}}^{\mathrm{x}} \log(\mathrm{RF(Cj,S)})\mathrm{RF\,(Cj,S)} \tag{1.25}$$

Sau khi S được phân chia thành các tập con S1, S2, …, St bởi test B, ta có công thức để tính information gain sau khi chia S thành các tập con S1, S2, …, Si bởi tập B là:

$$G(S, B) = -\sum \frac{|Si|}{|S|} I(Si) + I(S) \quad (1.26)$$

Tiếp đến ta chuẩn hóa information gain với trị thông tin phân tách (split information):

$$\text{Gain Ratio} = \frac{\text{Information Gain}}{\text{Split Info}} \quad (1.27)$$

Trong đó: Split Info được tính như sau:

$$-\sum_{i=1}^{n} \text{Dilog}_2 \text{Dt} \quad (1.28)$$

Giả sử chúng ta phân chia biến thành n nút con và Di đại diện cho số lượng bản ghi thuộc nút đó. Do đó, hệ số Gain Ratio sẽ xem xét được xu hướng phân phối khi chia cây.

f. Điều kiện cơ sở để dừng

Một trong số các phương pháp được sử dụng trong cây quyết định để tránh việc quá khớp diễn ra là: Trong một node, trường hợp các điều kiện sau xảy ra thì ta sẽ coi đó là một một leaf node và dừng, không tiếp tục phân chia tiếp node đó:

Điều kiện 1: Giá trị entropy tại node bằng 0 và mọi điểm cùng thuộc một lớp

Điều kiện 2: Ta sẽ sử dụng ngưỡng để quyết định có tiếp tục phân chia node. Nếu node đang xét có số phần tử nhỏ hơn ngưỡng thì sẽ được coi là thỏa mãn điều kiện. Với trường hợp này, chúng ta chấp nhận việc một số phần tử sẽ bị phân lớp sai(sai số) nhưng đổi lại ta thu được một mô hình cây tốt hơn, ít bị quá khớp hơn. Với trường hợp này ta sẽ coi các điểm trong node có lớp là lớp chiếm đa số trong node.

Điều kiện 3: Ta sẽ sử dụng ngưỡng quy định số tầng tối đa. Nếu số tầng tính từ node đó tới root node bằng ngưỡng thì sẽ coi là thỏa mãn. Điều này giúp giảm độ sâu, độ phức tạp của tree và cũng tránh việc quá khớp

Điều kiện 4: Ta sẽ sử dụng ngưỡng quy định tổng số leaf node tối đa. Nếu tổng số leaf node vượt quá ngưỡng thì sẽ coi là thỏa mãn.

Điều kiện 5: Ta sẽ kiểm tra entropy sau khi phân chia. Nếu việc phân chia không còn làm cho các giá trị entropy giảm nữa tính trên một ngưỡng nào đó thì sẽ thỏa mãn điều kiện

g. Phương pháp Pruning

Pruning là một kỹ thuật regularization để tránh overfitting cho decision tree nói chung. Kỹ thuật pruning sẽ xây dựng một decision tree cho đến khi mà mọi điểm trong tập huấn luyện đều được phân lớp chính xác. Tiếp đến non-leaf node sẽ tiến hành cắt tỉa các non-leaf node sinh ra từ nó biến chúng thành một leaf-node, với lớp tương ứng với lớp chiếm đa số trong số mọi điểm được phân vào node đó.

Phương pháp cắt tỉa cây thường được xác định dựa trên các yếu tố sau:

Dựa vào một tập kiểm định. Dữ liệu huấn luyện ban đầu sẽ được chia thành một tập huấn luyện và một tập kiểm định. Cây quyết định được xây dựng trên tập huấn luyện cho tới khi mọi điểm trong tập huấn luyện được phân lớp đúng. Sau đó, đi ngược từ các leaf node, cắt tỉa các sibling node của nó và giữ lại node bố mẹ nếu độ chính xác trên validation set được cải thiện. Việc cắt tỉa sẽ dừng lại khi độ chính xác trên tập kiểm định không còn được cải thiện nữa. Phương pháp này còn được gọi là reduced error pruning.

Dựa vào toàn bộ tập dữ liệu. Với cách chọn này, ta sẽ sử dụng toàn bộ tập dữ liệu này để xây dựng, huấn luyện cho cây quyết định. Trong đó, ta sẽ vào hàm mất mát một đại lượng. Đại lượng regularization này sẽ tỉ lệ thuận với leaf node. Cụ thể, giả sử decision tree cuối cùng có K leaf node, tập hợp các điểm huấn luyện rơi vào mỗi leaf node lần lượt là $\mathcal{S}_1, \dots, \mathcal{S}_\mathrm{K}$, hi đó, regularized loss sẽ được tính là:

$$\mathcal{L} = \sum_{\mathrm{k}=1}^{\mathrm{K}} \frac{|\mathcal{S}_\mathrm{k}|}{|\mathcal{S}|} \mathrm{H}(\mathcal{S}_\mathrm{k}) + \lambda \mathrm{K} \tag{1.29}$$

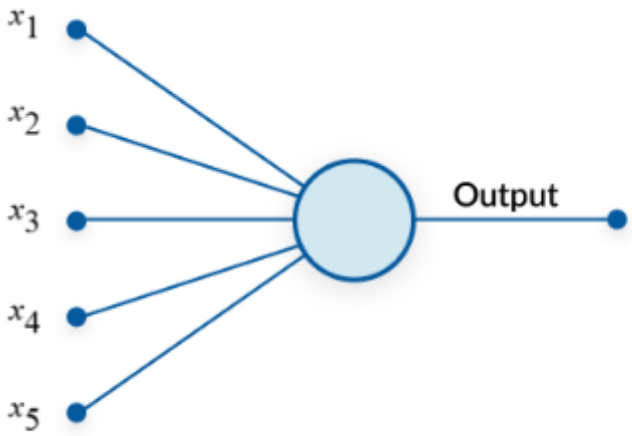
Trong đó, ký hiệu số phần tử của tập hợp $\mathcal{S}_k$ là : $|\mathcal{S}_k|$ và $H(\mathcal{S}_k)$ chính là entropy của leaf node tương ứng với $\mathcal{S}_k$.

1.2.4 Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo [7] là một mô hình lập trình rất đẹp lấy cảm hứng từ mạng nơ-ron thần kinh và nó đang trở thành một công cụ rất mạnh mẽ đem lại hiệu quả rất tốt trong các bài toán khó có thể kể đến như xử lý ngôn ngữ, phân loại ảnh, giọng nói.

a. Perceptrons

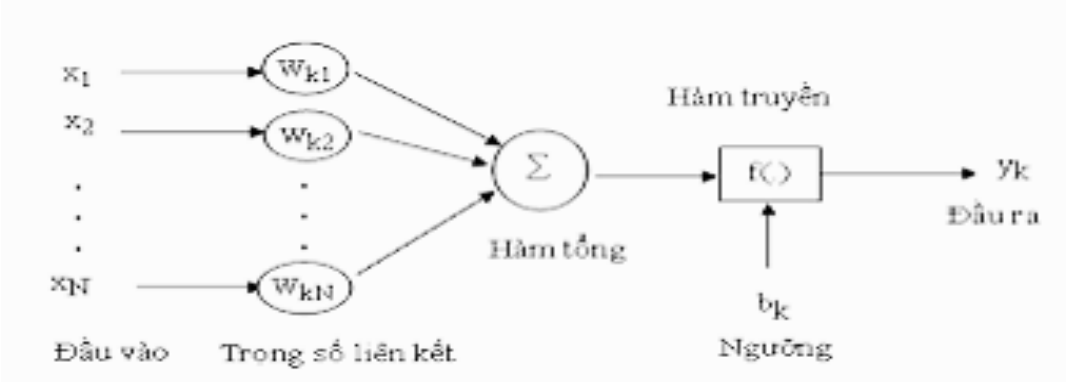
Perceptron là một mạng nơ-ron được cấu tạo dựa trên các nơ-ron đơn lẻ. Mô hình của perceptron cũng giống như một nơ-ron. Chúng đều nhận nhiều đầu vào và cho ra một kết quả duy nhất:



Hình 1.4. Cấu tạo của Perceptrons

Một perceptron sẽ nhận một hoặc nhiều đầu vào dạng nhị phân và cho ra một kết quả dưới dạng nhị phân duy nhất. Dữ liệu đầu vào đều chịu ảnh ảnh hưởng bởi các trọng số tương ứng của nó. Kết quả đầu ra quyết định dựa vào một ngưỡng quyết định.

b. Cấu trúc của nơ-ron nhân tạo

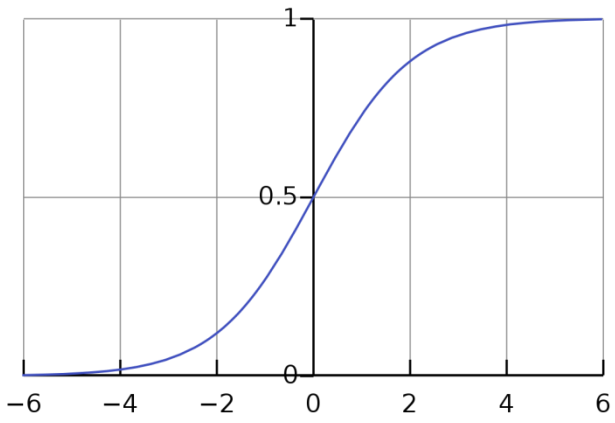
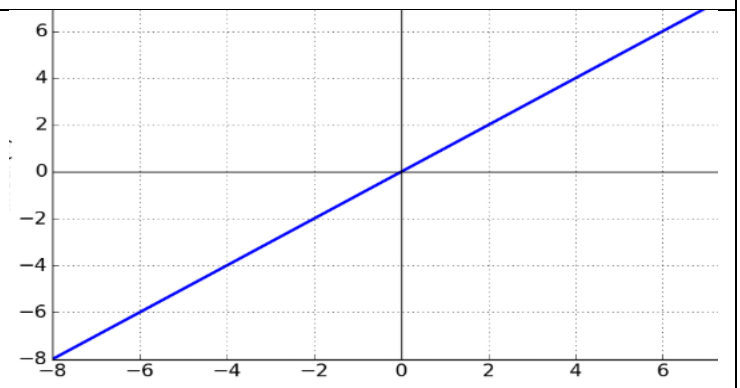
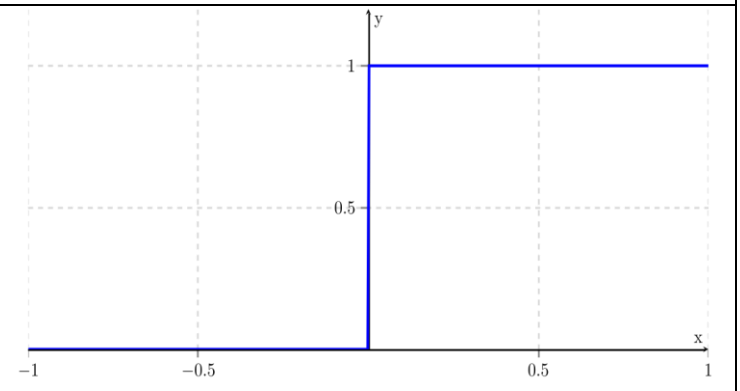


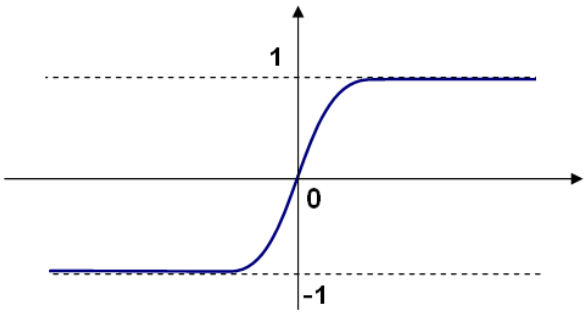
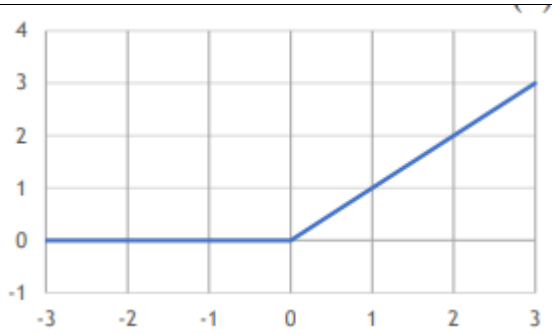
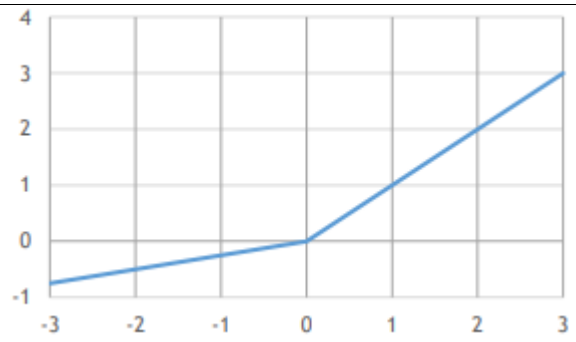
Hình 1.5. Cấu trúc của nơ-ron nhân tạo

Trong đó. dữ liệu đầu vào là các tín hiệu dưới dạng dưới dạng một vectơ N chiều được truyền vào. Trọng số liên kết là thường được gọi là weight. Chúng được khơi tạo một cách ngẫu nhiên. Hàm tổng là sử dụng để tính tổng các tích của dữ liệu đầu vào với trọng số liên kết của nó. Ngưỡng là được đưa vào để sử dụng trong hàm kích hoạt. Hàm kích hoạt là đầu và là kết quả của hàm tổng và ngưỡng. Hàm kích hoạt được sử dụng để giới hạn đầu ra của các nơ-ron. Các hàm kích hoạt là các hàm tuyến tính hay phi tuyến và chúng rất đa dạng. Để chọn lựa được hàm kích hoạt tốt sẽ tùy thuộc vào kinh nghiệm của người thiết kế mạng. Đầu ra là tín hiệu đầu ra duy nhất của mỗi nơ-ron.

c. Các hàm kích hoạt phổ biến

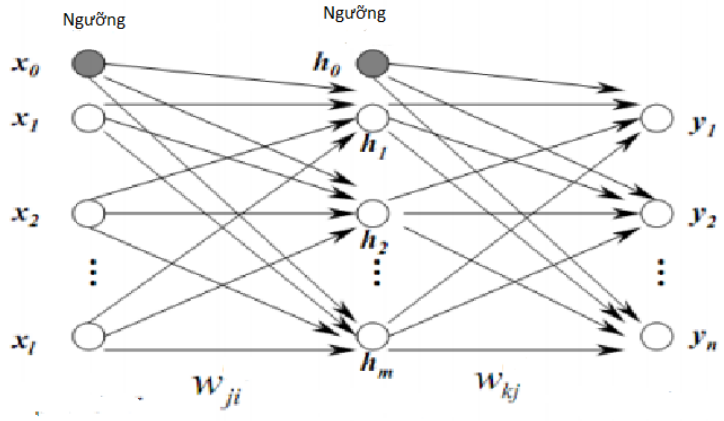
Bảng 1.1. Các hàm kích hoạt

Tên hàm	Mô tả hàm	Đồ thị
Sigmoid	$f = \frac{1}{1 + \exp(-x)}$	
Linear	$f = x$	
Hard Limit	$f = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	

Tên hàm	Mô tả hàm	Đồ thị
Bipolar sigmoid function	$f = \frac{1 - e^{-x}}{1 + e^{-x}}$	
ReLU	$f(x) = \max(0, x)$	
PReLU	$f = \begin{cases} x & x > 0 \\ ax & x \leq 0 \end{cases}$	

d. Mạng truyền thẳng

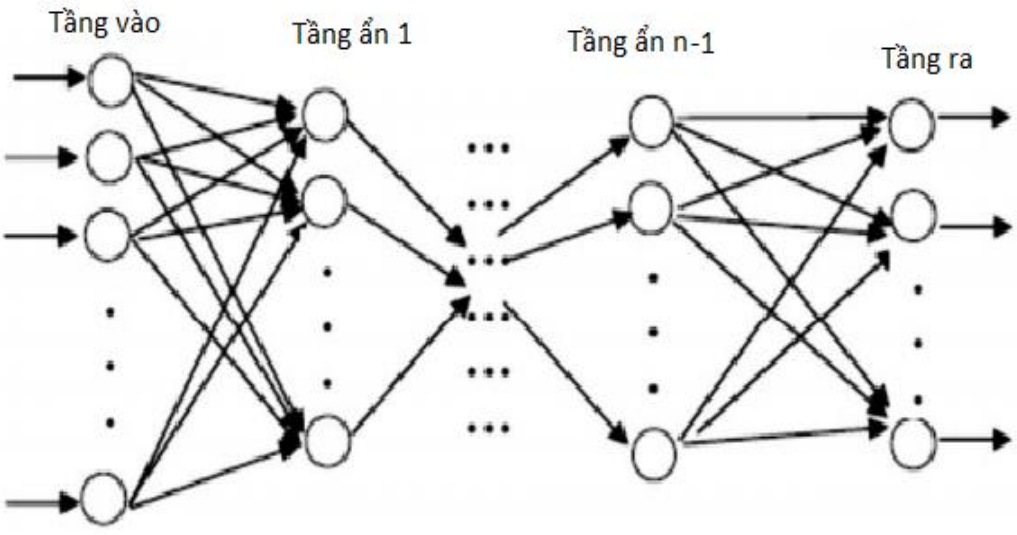
Dữ liệu đầu vào sẽ được truyền đến đơn vị đầu ra bằng cách truyền thẳng. Việc xử lý dữ liệu có thể mở rộng ra nhiều lớp nhưng ở đây chúng không có các liên kết để phản hồi lại.



Hình 1.6. Cấu tạo của mạng truyền thẳng

e. Mạng Multi Layer Perceptron

Mô hình mạng nơ-ron được sử dụng rộng rãi nhất là mô hình mạng nhiều tầng truyền thẳng. Một mạng MLP tổng quát là mạng có n (n≥2) tầng (thông thường tầng đầu vào không được tính đến): trong đó gồm một tầng đầu ra (tầng thứ n) và (n-1) tầng ẩn.



Hình 1.7. Cấu tạo của mạng MLP

f. Thuật toán lan truyền ngược

Phương pháp Gradient Descent là một trong những phương pháp phổ biến nhất để tối ưu mạng Multi Layer Perceptron.

Lan truyền ngược là phương pháp tính gradient của các tham số mạng nơ-ron. Nói một cách đơn giản, phương thức này duyệt qua mạng nơ-ron theo chiều ngược lại, từ đầu ra đến đầu vào, tuân theo quy tắc dây chuyền trong giải tích. Thuật toán lan truyền ngược lưu trữ các biến trung gian (là các đạo hàm riêng) cần thiết trong quá trình tính toán gradient theo các tham số. Giả sử chúng ta có hàm $\mathrm{Y} = \mathrm{f}(\mathrm{X})$ và $\mathrm{Z} = \mathrm{g}(\mathrm{Y})$. Trong đó đầu vào và đầu ra $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ là các tensor có kích thước bất kỳ. Bằng cách sử dụng quy tắc dây chuyền, chúng ta có thể tính đạo hàm của X, Z như sau:

$$\frac{\partial \mathrm{Z}}{\partial \mathrm{X}} = \mathrm{prod}\left(\frac{\partial \mathrm{Z}}{\partial \mathrm{Y}}, \frac{\partial \mathrm{Y}}{\partial \mathrm{X}}\right) \tag{1.30}$$

Ở đây, chúng ta sử dụng toán tử prod để nhân các đối số sau khi các phép tính cần thiết như là chuyển vị và hoán đổi đã được thực hiện. Với vectơ, điều này khá đơn giản: nó chỉ đơn thuần là phép nhân ma trận. Với các tensor nhiều chiều thì sẽ có các phương án tương ứng phù hợp. Toán tử prod sẽ đơn giản hoá việc ký hiệu.

Các tham số của mạng nơ-ron đơn giản với một tầng ẩn là $\mathrm{W}^{(1)}$ và $\mathrm{W}^{(2)}$. Mục đích của lan truyền ngược là để tính gradient $\partial \mathrm{J}/\partial \mathrm{W}^{(1)}$ và $\partial \mathrm{J}/\partial \mathrm{W}^{(2)}$. Để làm được điều này, ta áp dụng quy tắc dây chuyền và lần lượt tính gradient của các biến trung gian và tham số. Các phép tính trong lan truyền ngược có thứ tự ngược lại so với các phép tính trong lan truyền xuôi, bởi ta muốn bắt đầu từ kết quả của đồ thị tính toán rồi dần đi tới các tham số. Bước đầu tiên đó là tính gradient của hàm mục tiêu $\mathrm{J} = \mathrm{L} + \mathrm{s}$ theo mất mát L và điều chuẩn s:

$$\frac{\partial \mathrm{J}}{\partial \mathrm{L}} = 1 \text{ và } \frac{\partial \mathrm{J}}{\partial \mathrm{s}} = 1 \tag{1.31}$$

Tiếp theo, ta tính gradient của hàm mục tiêu theo các biến của lớp đầu ra o, sử dụng quy tắc dây chuyền

$$\frac{\partial \mathrm{J}}{\partial \mathrm{o}} = \mathrm{prod}\left(\frac{\partial \mathrm{J}}{\partial \mathrm{L}}, \frac{\partial \mathrm{L}}{\partial \mathrm{o}}\right) = \frac{\partial \mathrm{L}}{\partial \mathrm{o}} \in \mathbb{R}^{\mathrm{q}} \tag{1.32}$$

Kế tiếp, ta tính gradient của điều chuẩn theo cả hai tham số.

$$\frac{\partial s}{\partial W^{(1)}} = \lambda W^{(1)} \text{ và } \frac{\partial s}{\partial W^{(2)}} = \lambda W^{(2)} \tag{1.33}$$

Bây giờ chúng ta có thể tính gradient $\partial J/\partial W^{(2)} \in \mathbb{R}^{q\times h}$ của các tham số mô hình gần nhất với tầng đầu ra. Áp dụng quy tắc dây chuyền, ta có:

$$\frac{\partial J}{\partial W^{(2)}} = \text{prod}\left(\frac{\partial J}{\partial o}, \frac{\partial o}{\partial W^{(2)}}\right) + \text{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial W^{(2)}}\right) = \frac{\partial J}{\partial o} h^{\top} + \lambda W^{(2)} \tag{1.34}$$

Để tính được gradient theo $W^{(1)}$ ta cần tiếp tục lan truyền ngược từ tầng đầu ra đến các tầng ẩn. Gradient theo các đầu ra của tầng ẩn $\frac{\partial J}{\partial h} \in \mathbb{R}^{h}$được tính như sau:

$$\frac{\partial J}{\partial h} = \text{prod}\left(\frac{\partial J}{\partial o}, \frac{\partial o}{\partial h}\right) = W^{(2)} \frac{\top}{\partial o} \tag{1.35}$$

Vì hàm kích hoạt ϕ áp dụng cho từng phần tử, việc tính gradient của biến trung gian z cũng yêu cầu sử dụng phép nhân theo từng phần tử, kí hiệu bởi $\odot$.

$$\frac{\partial J}{\partial z} = \text{prod}\left(\frac{\partial J}{\partial h}, \frac{\partial h}{\partial z}\right) = \frac{\partial J}{\partial h} \odot \phi'(z) \tag{1.36}$$

Cuối cùng, ta có thể tính gradient $\partial J/\partial W^{(1)} \in \mathbb{R}^{h\times d}$ của các tham số mô hình gần nhất với tầng đầu vào. Theo quy tắc dây chuyền, ta có:

$$\frac{\partial J}{\partial W^{(1)}} = \text{prod}\left(\frac{\partial J}{\partial z}, \frac{\partial z}{\partial W^{(1)}}\right) + \text{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial W^{(1)}}\right) = \frac{\partial J}{\partial z} x^{\top} + \lambda W^{(1)} \tag{1.37}$$

1.2.5 Mạng nơ-ron tích chập

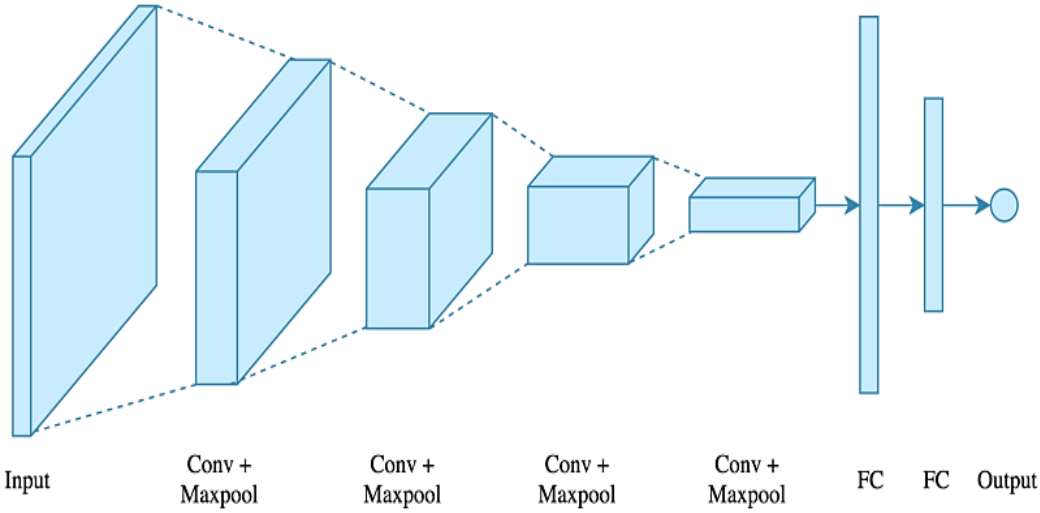
a. Khái niệm

Mạng Nơ-ron Tích chập (Convolutional Neural Network - CNN) [8] là một họ các mạng nơ-ron ưu việt. Các kiến trúc dựa trên CNN hiện nay xuất hiện trong mọi ngóc ngách của lĩnh vực thị giác máy tính, và đã trở thành kiến trúc chủ đạo mà hiếm ai ngày nay phát triển các ứng dụng thương mại hay tham gia một cuộc thi nào đó liên quan tới nhận dạng ảnh, phát hiện đối tượng, hay phân vùng theo ngữ cảnh mà không xây nền móng dựa trên phương pháp này.

b. Kiến trúc mạng CNN

Thiết kế của mạng ConvNets đã vay mượn rất nhiều ý tưởng từ ngành sinh học, lý thuyết nhóm và lượng rất nhiều những thí nghiệm nhỏ lẻ khác. Bên cạnh hiệu

năng cao trên số lượng mẫu cần thiết để đạt được đủ độ chính xác, các mạng nơ-ron tích chập thường có hiệu quả tính toán hơn, bởi đòi hỏi ít tham số hơn và dễ thực thi song song trên nhiều GPU hơn các kiến trúc mạng dày đặc. Theo [9][10] thì kiến trúc mạng CNN sẽ đem lại hiệu quả rất tốt trong việc trích chọn đặc trưng.



Hình 1.8. Kiến trúc mạng CNN

Kiến trúc mạng CNN bao gồm các thành phần chính là: lớp tích chập, lớp pooling, lớp kết nối đầy đủ, hàm kích hoạt.

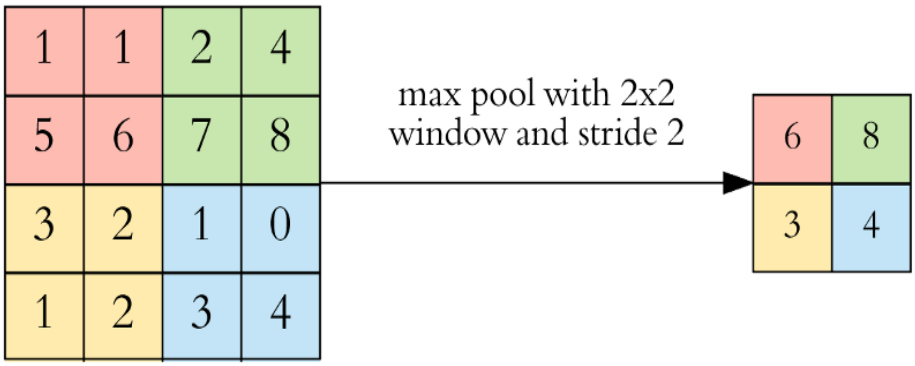
c. Lớp tích chập

Lớp tích chập bao gồm 2 khái niệm khác đó là bộ lọc tích chập và lớp tích chập. Với CNN, lớp tích chập là các lớp ẩn, khác ở chỗ, lớp tích chập là một tập các ma trận đặc trưng và mỗi ma trận đó là một bản thông tin đầu vào, nhưng được trích rút ra các đặc tính cụ thể. bộ lọc tích chập sẽ quyết định các đặc tính sẽ được scan như thế nào, đây sẽ là một ma trận quét qua ma trận dữ liệu đầu vào, từ trái qua phải, từ trên xuống dưới, và nhân tương ứng giá trị của ma trận đầu vào với bộ lọc tích chập rồi cộng tổng lại, tập hợp các con số này gọi là ma trận đặc trưng.

d. Lớp gộp

Lớp gộp sinh ra để làm những điều đó. Cụ thể lớp gộp sẽ làm giảm số tham số sử dụng trong việc tính toán. Điều này sẽ giúp giảm hiệu quả thời gian trong việc tính toán và hạn chế tình trạng quá khớp. Có rất nhiều loại gộp nhưng gộp cực đại thường được sử dụng nhiều. Gộp cực đại sẽ lấy giá trị lớn nhất trong một cửa sổ pool. Pooling

cũng quét ma trận đầu vào như lớp tích với 1 cửa sổ trượt. Từ đó sẽ chọn ra một giá trị từ các giá trị nằm trong cửa sổ trượt Với gộp cực đại thì sẽ chọn ra giá trị lớn nhất trong cửa sổ



Hình 1.9. Ví dụ về lớp gộp cực đại

e. Các tham số trong mạng nơ-ron tích chập

Filter size: là kích thước của cửa sổ trượt dùng để chia nhỏ bức ảnh ra. Thường sẽ là ma trận 3x3 hoặc 4x4.

Epochs: Số lượng chu kì lặp lại của mạng

Batch: số lượng phần tử trong một nhóm để đưa vào tính toán trong mạng.

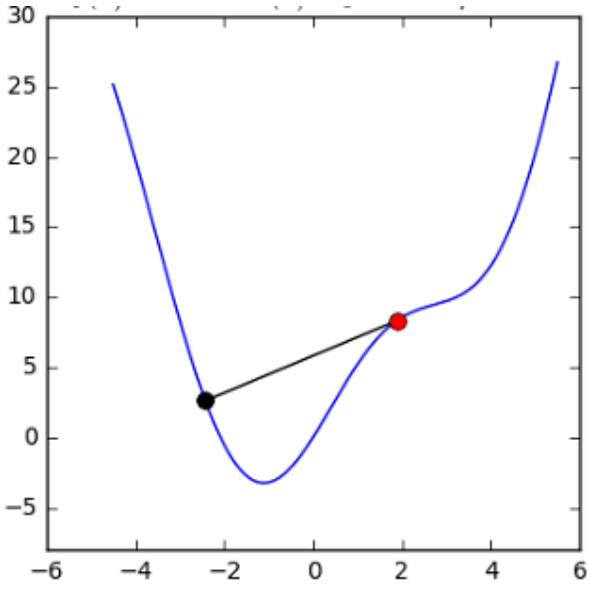
Stride là khoảng cách giữa 2 nhân khi quét. Ví dụ với stride = 1, nhân sẽ quét 2 ô ngay cạnh nhau

Pooling: Lớp thườngđược sử dụng để giảm chiều và nằm ngay phía sau các lớp convolational Pooling phổ biến là max-pooling. Thông thường thi sẽ chỉ cần sử dụng kích thước 2x2, còn nếu đầu vào là ảnh lớn thì ta có thể sử dụng 4x4.

f. Các thuật toán tối ưu

Các thuật toán tối ưu là nền tảng cơ sở giúp cho các mô hình mạng nơ-ron có thể "học " được các đặc trưng của dữ liệu đầu vào. Điều này giúp cho mạng nơ-ron có thể tìm được cặp trọng số và ngưỡng phù hợp cho mô hình. Với mỗi bài toán ta cần tìm một thuật toán tối ưu phù hợp để cải thiện trọng số và bias.

- Thuật toán Gradient Descent



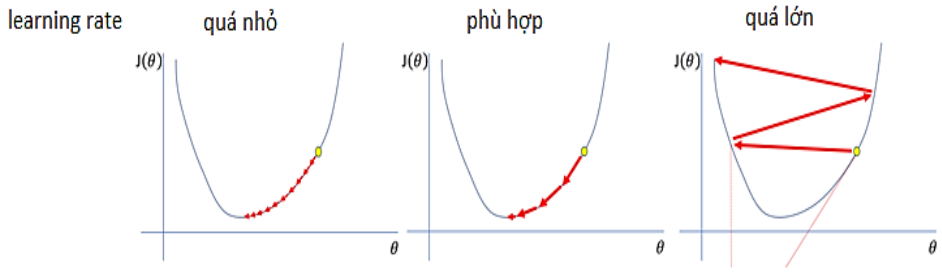
Hình 1.10. Đồ thị hàm f(θ) của thuật toán Gradient Descent

Để tìm được cực tiểu trong hàm f(θ). Kí hiệu của đạo hàm của f tại một điểm theta bất kì là $\nabla_\theta f(\theta)$. Quy tắc cập nhật theta khi ở vòng lặp t có công thức là:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta f(\theta_t) \quad (1.38)$$

Vấn đề của Gradient Descent truyền thống là nếu tốc độ học quá nhỏ thì sẽ cần phải huấn luyện rất nhiều epochs và tốn rất nhiều thời gian. Trong khi tốc độ học quá lớn thì có thể biến cập nhật sẽ nhảnh quanh vùng tối ưu vào không hội tụ.

- Thuật toán Momentum



Hình 1.11. Mối liên hệ giữa tốc độ huấn luyện và hàm $J(\theta)$ trong thuật toán Momentum

Thuật toán được xây dựng với ý tưởng chính là tăng gia tốc khi hướng cập nhật cùng với hướng của gradient descent và giảm gia tốc khi hướng cập nhật ngược với hướng gradient descent. Công thức:

$$\theta = \theta - v_t v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \quad (1.39)$$

Trong đó, v_t là đại lượng đặt trưng cho sự thay đổi vị trí trong không gian. Khi ta đi một bước theo GD và thu được vectơ v_{t-1}. Tại bước t, ta sẽ thực hiện tính vectơ gradient, ta sẽ tăng tốc độ lên khi nếu vectơ gradient vẫn giữ nguyên hướng với điều kiện v_{t-1} phải chung hướng đi với gradient, điểm thu được sẽ có khả năng cao là đến gần với optimal point hơn là chỉ bước theo GD. Ngược lại, nếu gradient tại bước t đổi chiều (tức là ta đã đi quá điểm optimal point), ta cần quay đầu lại nhưng không bước bước lớn như vừa rồi mà phải bước bước nhỏ hơn thì mới hi vọng tới gần hơn với điểm optimal point.

- Adagrad

Adagrad được đề xuất năm 2011 đã giải quyết vấn đề này bằng cách thay đổi bộ đếm thô $s(i, t)$ bởi tổng bình phương của tất cả các gradient được quan sát trước đó. Cụ thể, nó sử dụng $s(i, t + 1) = s(i, t) + (\partial_i f(x))^2$ làm công cụ để điều chỉnh tốc độ học. Việc này đem lại hai lợi ích: trước tiên ta không cần phải quyết định khi nào thì gradient được coi là đủ lớn. Thứ hai, nó tự động thay đổi giá trị tuỳ theo độ lớn của gradient.

Tốc độ học trong Adagrad được coi là 1 tham số và được thuật toán biến thiên tại các thời điểm t. Đó cũng là một lợi ích của thuật toán khi chỉ cần chọn tốc độ học mặc định và thuật toán sẽ tự động điều chỉnh. Quy tắc cập nhật theta khi ở vòng lặp t có công thức là:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \quad (1.40)$$

g. Hàm mất mát

Hàm mất mát (Loss function) là hàm thể hiện sự tương quan, mối quan hệ giữa giá trị thực tế và giá trị mà mô hình dự đoán. Và khi hiệu hai giá trị trên = 0 tức là đoán chính xác. Mục tiêu của việc xây dựng hàm mất mát để có một độ đo lỗi. Loss

càng nhỏ thì lỗi càng ít từ đó hàm tối ưu được sử dụng để tối ưu hàm mất mát sao cho giá trị hàm mất mát là nhỏ nhất.

- Cross-Entropy Loss

Đối với bài toán phân loại thì đây là phương pháp phổ biến và được sử dụng nhiều nhất. Hàm mất mát sẽ tăng khi mà nhãn dự đoán sai tăng. Công thức của phương pháp này là:

$$\text{Loss} = -(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{1.41}$$

- Mean Absolute Error

Để đánh giá mối quan hệ giữa đầu ra dự đoán và thực tế, một trong những cách đơn giản nhất đó là lấy trung bình cộng của trị tuyệt đối của giá trị dự đoán trừ giá trị thực tế. Kí hiệu của hàm mất mát này là MAE:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{1.42}$$

Ưu điểm của phương pháp này là giá trị không bị lệch quá nhiều do hai giá trị dự đoán và giá trị thực tế quá chênh lệch vì nó không sử dụng bình phương.

- Mean Square Error

Hàm mất mát này được tính bằng trung bình của bình phương các sai số. Công thức là

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 \tag{1.43}$$

Đây là phương pháp dễ dàng tính với Y là một vectơ quan sát và $\hat{Y}$ là vectơ dự đoán.

Ưu điểm của phương pháp này là nó rất dễ tính đạo hàm.

h. Một số mạng tích chập thường được sử dụng trong bài toán nhận diện khuôn mặt

Hiện nay có rất nhiều mạng tích chập được sử dụng trong bài toán nhận diện khuôn mặt có thể kể đến như AlexNet, VGG16, DenseNet, GoogleNet hay ResNet. Trong đó mạng được cho là tiêu biểu và có sức ảnh hưởng lớn nhất là GoogleNet hay ResNet. Năm 2014, Các kỹ sư của google giới đã sử dụng một mô hình mạng rất đặc

biệt có tên là GoogleNet để tham gia cuộc thi ImageNet 2014 và được cộng đồng đánh giá rất cao. Resnet được biết đến rộng rãi vào những năm 2015 khi giành giải nhất các cuộc thi như ILSVRC 2015 và COCO 2015. Trong đó, resnet đã lần lượt vượt qua các mạng được coi là tốt nhất lúc bấy giờ như ImageNet Detection, Coco segmentationResNet.

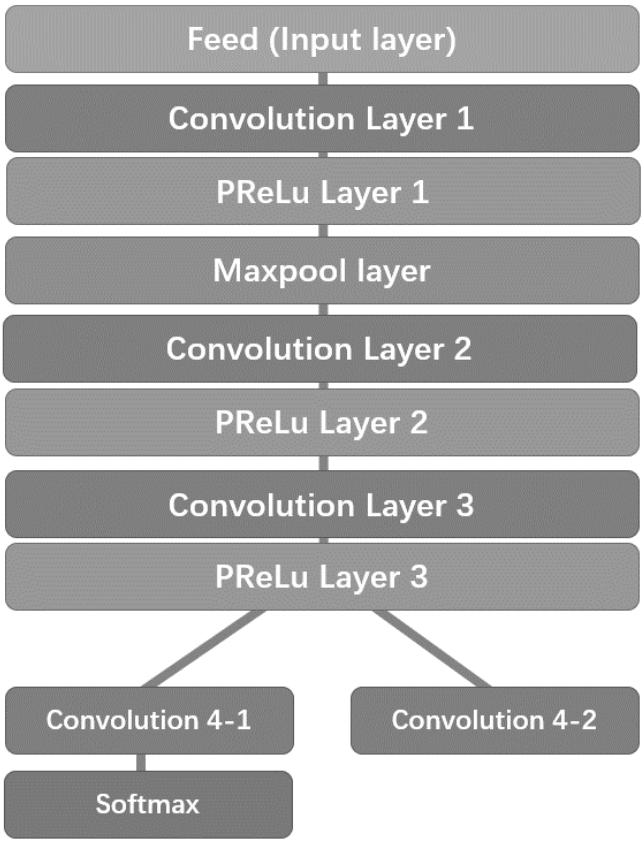
1.3 Phương pháp xác định vị trí khuôn mặt với mạng tích chập MTCNN

1.3.1 Giới thiệu

Hiện tại, có rất nhiều các phương pháp xác định vị trí khuôn mặt qua ảnh có thể nói đến như HOG, MTCNN hay Haar Cascade. Trong đó nhiều nghiên cứu chỉ ra MTCNN là một trong những mạng hiểu quả nhất. Dù còn nhiều hạn chế có thể nói đến như tốc độ xử lý, mạng phức tạp nhưng MTCNN ngày càng được sử dụng rộng rãi trong thực tiễn. MTCNN một mạng lớn gồm nhiều mạng con. Mỗi mang con đều xử lý một bài toán nhỏ trong việc xác định vị trí của khuôn mặt. Trong kiến trúc, MTCNN sử dụng lần lượt ba mạng P-Net, R-Net, O-Net để xác định khuôn mặt.

1.3.2 Cấu trúc mạng P-Net

MTCNN một mạng lớn gồm nhiều mạng con. Mỗi mang con đều xử lý một bài toán nhỏ trong việc xác định vị trí của khuôn mặt. Trong kiến trúc, MTCNN sử dụng lần lượt ba mạng P-Net, R-Net, O-Net để xác định khuôn mặt.

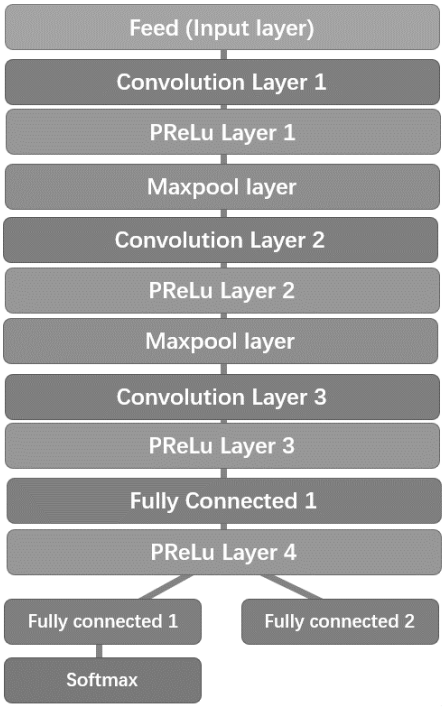


Hình 1.12. Kiến trúc mạng P-Net

Tầng đầu tiên là một cụm gồm có một lớp tích chập 3x3 với sải là 1, hàm kích hoạt là PReLu và lớp gộp cực đại với cửa sổ là 2x2 với sải là 2 để giảm một nửa kích thước đầu ra. Tầng thứ hai có một lớp tích chập 3x3 với sải là 1, hàm kích hoạt PReLu. Tầng thứ ba có một lớp tích chập 3x3 với sải là 1 và hàm kích hoạt là Softmax. Tại tầng thứ ba đầu ra của mạng gồm 3 phần là xác suất của khuôn mặt trong hình, 4 điểm tọa độ khuôn mặt và 10 điểm vị trí của khuôn mặt. Hai bộ lọc có kích thước 1x1 để xác định vị trí khuôn mặt sẽ được cho vào softmax. Còn lại thông tin về tọa độ, các điểm trên khuôn mặt sẽ được gửi tới mạng R-net

1.3.3 Cấu trúc mạng R-Net

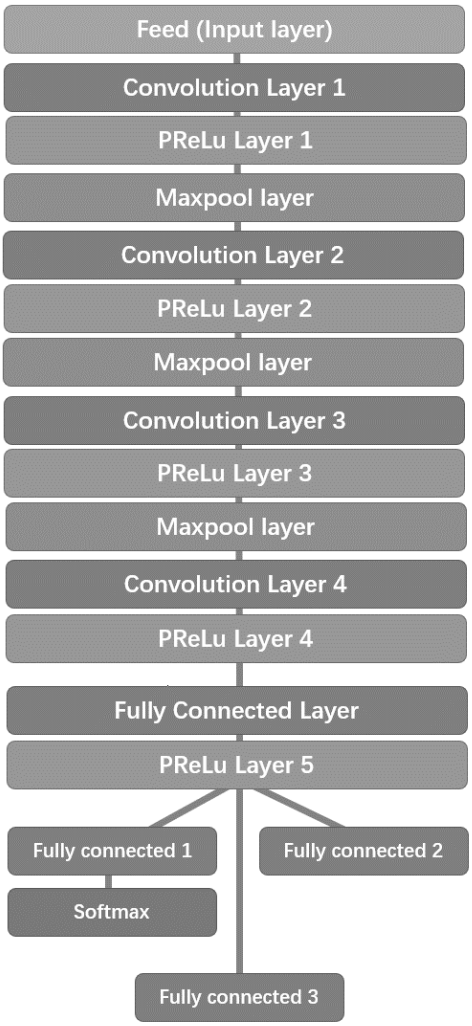
MTCNN một mạng lớn gồm nhiều mạng con. Mỗi mang con đều xử lý một bài toán nhỏ Cấu trúc mạng R-Net



Hình 1.13. Kiến trúc mạng R-Net

Tầng thứ nhất sử dụng lớp tích chập 3x3 với sải là 1, hàm kích hoạt là PReLu và sau đó là một lớp gộp cực đại với cửa sổ trượt là 3x3 với sải là 2 để kích thước của dữ liệu đầu ra. Tầng thứ hai sử dụng lớp tích chập 3x3 với sải là 1, hàm kích hoạt là PReLU và sau đó cũng là một lớp gộp cực đại với cửa sổ trượt là 3x3 với sải là 2 để kích thước của dữ liệu đầu ra. Tầng thứ ba sẽ là một lớp tích chập 3x3 với sải là 1, hàm kích hoạt là PReLu và sau đó là một lớp kết nối đầy đủ với hàm kích hoạt là softmax. Tại tầng thứ ba đầu ra sẽ chia làm hai phần đó là cung cấp tọa độ mới của vị trí khuôn mặt cùng với xác suất của nó. O-Net sẽ tiếp tục lấy tọa độ khuôn mặt mới mà R-net cung cấp làm đầu vào để xác định các mốc trên khuôn mặt.

1.3.4 Cấu trúc mạng O-Net



Hình 1.14. Kiến trúc mạng O-Net

O-Net sử dụng 4 lớp tích chập, 3 tầng gộp cực đại và 1 lớp kết nối đầy đủ. Mạng trong O-Net chia làm năm tầng. Tầng thứ nhất gồm một lớp cnn 3x3 với sải là 1, hàm kích hoạt là PReLu và một tầng gộp cực đại 3x3 với sải là 2 để giảm kích thước của dữ liệu. Tầng thứ hai gồm một lớp cnn 3x3 với sải là 1, hàm kích hoạt là PReLu và một tầng gộp cực đại 3x3 với sải là 2 để giảm kích thước của dữ liệu. Tầng thứ ba gồm một lớp cnn 3x3 với sải là 1, hàm kích hoạt là PReLu và một tầng gộp cực đại 2x2 với sải là 2 để giảm kích thước của dữ liệu. Tầng thứ tư gồm một lớp cnn 2x2 với sải là 1, hàm kích hoạt là PReLu. Tầng thứ năm là lớp kết nối đầy đủ, hàm kích hoạt là softmax. Đầu ra của MTCNN là 3 cụm là xác xuất của khuôn

mặt nằm trong tọa độ, tọa độ của khuôn mặt và tọa độ của vị trí các mốc trên khuôn mặt

1.3.5 Đánh giá

Mô hình MTCNN có khả năng trích rút đặc trưng và xác đinh vị trí khuôn mặt rất tốt và có khả năng phát hiện khuôn mặt trong các trường hợp có nhiễu, ánh sáng quá tối, quá sáng, mờ hay như ảnh chỉ chứa một phần khuôn mặt.

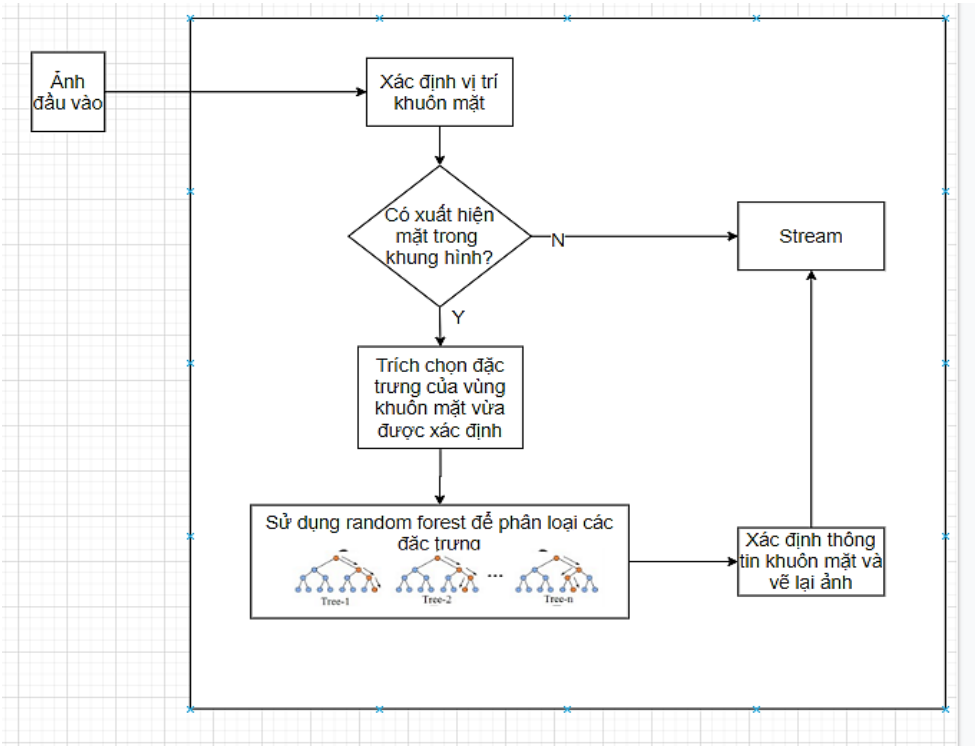
1.4 Kết luận

Chương này đã giới thiệu về tổng quan về nhận diện khuôn mặt, kiến trúc hệ thống nhận diện khuôn mặt cũng như ứng dụng của hệ thống trong thực tế. Bên cạnh đó nội dung của chương cũng trình bày về một số phương pháp hay được sử dụng trong bài toán nhận diện khuôn mặt như PCA, LDA, cây quyết định, mạng nơ ron nhân tạo, nơ ron tích chập. Từ đó có thể đánh giá được ưu nhược điểm của từng phương pháp trong bài toán nhận diện khuôn mặt.

CHƯƠNG 2. HỆ THỐNG NHẬN DIỆN KHUÔN MẶT DỰA TRÊN MẠNG NƠ RON TÍCH CHẬP

2.1 Sơ đồ thiết kế hệ thống nhận diện khuôn mặt

Trong hệ thống có ba bước xử lý chính để có thể nhận diện khuôn mặt. Đó là xác định vị trí khuôn mặt, trích chọn đặc trưng của khuôn mặt và phân loại xác định khuôn mặt.



Hình 2.1. Sơ đồ hoạt động của hệ thống nhận diện khuôn mặt

Trong hệ thống có ba bước xử lý chính để có thể nhận diện khuôn mặt. Đó là xác định vị trí khuôn mặt, trích chọn đặc trưng của khuôn mặt và phân loại xác định khuôn mặt.

Các bước để huấn luyện mô hình phân loại:

Bước 1: Chuẩn bị dữ liệu ảnh khuôn mặt để huấn luyện

Bước 2: Xử lý xác định vị trí và trích chọn đặc trưng khuôn mặt từ tập huấn luyện

Bước 3: Huấn luyện rừng ngẫu nhiên để phân loại đặc trưng

2.2 Mạng Inception-ResNet sử dụng cho việc trích chọn đặc trưng khuôn mặt

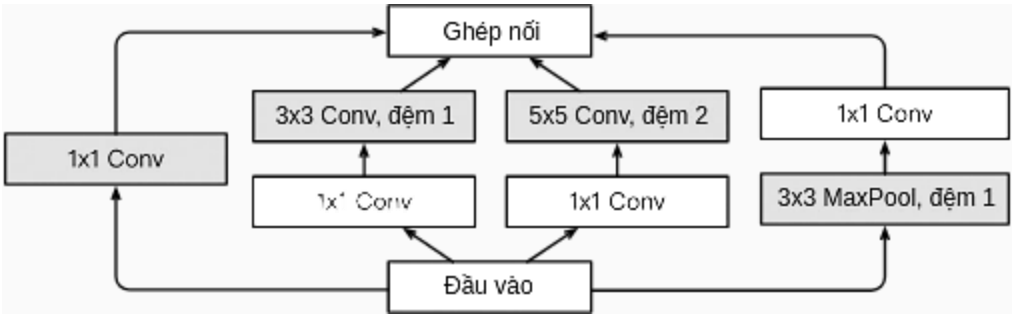
2.2.1 *Giới thiệu*

Các mạng tích chập rất sâu hiện nay đang được coi là trung tâm của những tiến bộ lớn nhất trong hiệu suất nhận dạng hình ảnh trong những năm gần đây. Theo nghiên cứu gần đây thì các mạng CNN hoạt động rất tốt trong việc trích chọn đặc trưng và phân loại khuôn mặt. Trong đó, ta có thể kể đến một số kiến trúc mạng học sâu như VGGNET, ZFNET hay AlexNET trong bài toán nhận diện khuôn mặt. Mạng VGGNET được giới thiệu bởi google năm 2014 đã cho thấy bước ngoặt rất lớn trong thiết kế mạng học sâu với kiến trúc xếp chồng liên tiếp nhiều lớp tích chập đem lại hiệu quả rất tốt. Tuy nhiên với một mạng học quá sâu, quá nhiều lớp tích chập xếp chồng lên nhau sẽ khiến khiến độ chính xác bị bão hòa, mạng học lâu hơn và tỉ lệ lỗi đôi khi còn cao hơn mạng cũ. Vì vậy ta có thể thấy không phải với bài toán nào ta cũng có thể sử dụng những kiến trúc mạng có sẵn như VGGNET, ZFNET hay AlexNet rồi thêm những tầng tích chập phía sau thì sẽ đem lại kết quả tốt. Từ những vấn đề gặp phải trên, nhóm nghiên cứu Microsoft đã cho ra mắt mạng Resnet và đội ngũ Google giới thiệu mạng Inception để khắc phục nhưng vấn đề về mạng học quá sâu như giảm thiểu thời gian, tài nguyên cho việc tính toán và tăng độ chính xác. Điều này khiến chúng ta phải đặt ra câu hỏi liệu có bất kỳ lợi ích, cải tiến nào khi kết hợp kiến trúc mạng inception và mạng kết nối tắt.

2.2.2 *Mạng GoogleNet*

a. Khối Inception

Khối tích chập cơ bản trong mô hình GoogLeNet được gọi là Inception. Khối Inception trích xuất thông tin một cách song song thông qua các tầng tích chập với kích thước cửa sổ khác nhau và các tầng gộp cực đại.



Hình 2.2. Hình 2.1 Khối Inception

Như mô tả ở hình trên, khối inception bao gồm bốn nhánh song song với nhau.

Nhánh đầu tiên sử dụng một tầng tích chập với cửa sổ trượt là 1x1

Nhánh thứ hai sử dụng hai tầng tích chập với cửa sổ trượt là 1x1, 3x3 đệm 1

Nhánh thứ ba sử dụng hai tầng tích chập với cửa sổ trượt là 1x1 và 5x5 đệm 2

Nhánh thứ ba sử dụng một tầng gộp tối đa với cửa sổ trượt là 3x3 đệm 1 và tầng tích chập 1x1

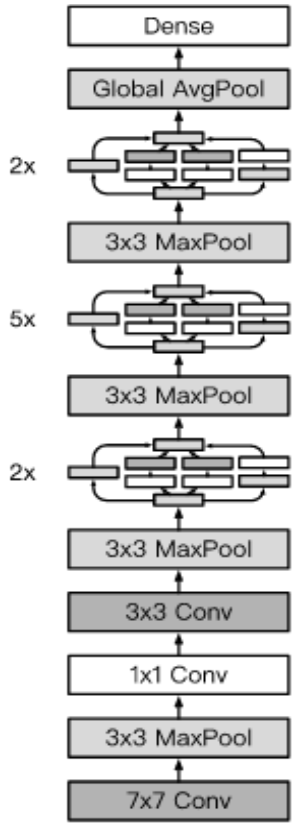
Ba nhánh đầu tiên sử dụng các cửa sổ 1x1, 3x3 và 5x5 để trích chọn đặc trưng tự từ các vùng không gian khác nhau. Tầng tích chập 1×1 giảm số kênh ở mức độ điểm ảnh. Phép gộp cực đại giảm độ phân giải

Sau khi giới thiệu mạng googlenet, các kỹ sư nghiên cứu của google đã giải thích vì sao họ lại sử dụng kích thước cửa sổ trượt là là 3x3 và 5x5. Đó là vì họ đã thực hiện các cửa sổ trượt từ 3x3, 5x5, 7x7 và 11x11 thì nhận thấy việc sử dụng kích thước cửa sổ lớn như 7x7 hay 11x11 sẽ đem lại hiệu quả cao hơn trong việc trích chọn đặc trưng. Tuy nhiên việc thay đổi sử dụng các cửa sổ với kích thước khác nhau đôi khi sẽ phải đánh đổi giữa hiệu năng, chi phí cho việc tính toán.

Mạng googlenet hiện nay đã trải qua rất nhiều bản nâng cấp, Google liên tục có nhiều cải tiến đáng kể trong các khối inception giúp cho việc tính toán diễn ra nhanh hơn. Ví dụ như 1 bộ lọc 5x5 sẽ được thay bằng 2 bộ lọc 3x3 nối nhau. Với trường hợp trên thì kết quả đem lại là tương đương nhưng việc tính toán diễn ra nhanh hơn.

b. Kiến trúc mạng

Sau đây là kiến trúc của mạng:



Hình 2.3. Kiến trúc mạng GoogletNet

GoogLeNet sử dụng tổng cộng 9 khối inception và tầng gộp trung bình toàn cục xếp chồng lên nhau. Phép gộp cực đại giữa các khối inception có tác dụng làm giảm kích thước chiều. Ở phần đầu trong kiến trúc có các khối xếp chồng lên nhau kế thừa từ thiết kế của VGG và phép gộp trung bình toàn cục giúp tránh phải sử dụng nhiều tầng kết nối đầy đủ liên tiếp ở cuối.

c. Đánh giá

Mạng GoogleNet đã cho thấy hướng giải quyết bài toán trích chọn đặc trưng rất mới. Thay vì sự kém hiệu quả trong việc xếp chồng các lớp tích chập thì GoogleNet đã đem lại hiệu quả rất tốt trong việc trích chọn đặc trưng. Nhưng cách làm như vậy vẫn còn hạn chế với mô hình mạng học quá sâu vì đặc trưng, thông tin dữ liệu vẫn có khả năng bị thay đổi lớn khi vào các tầng sâu và khi đó sẽ kém hiệu quả.

2.2.3 *Mạng ResNet*

a. Các lớp hàm số

Coi $\mathcal{F}$ là một lớp các hàm mà một kiến trúc mạng cụ thể (cùng với tốc độ học và các siêu tham số khác) có thể đạt được. Nói cách khác, với mọi hàm số $f \in \mathcal{F}$, luôn tồn tại một số tập tham số W có thể tìm được bằng việc huấn luyện trên một tập dữ liệu phù hợp. Giả sử f^* là hàm cần tìm. Sẽ rất thuận lợi nếu hàm này thuộc tập $\mathcal{F}$ nhưng thường không may mắn như vậy. Thay vào đó, ta sẽ cố gắng tìm các hàm số $f_{\mathcal{F}}^*$ tốt nhất có thể trong tập $\mathcal{F}$. Ví dụ, có thể thử tìm bằng cách giải bài toán tối ưu sau:

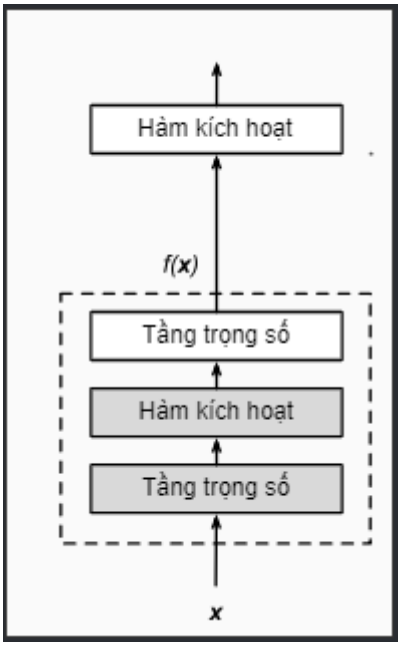
$$f_{\mathcal{F}}^* := \underset{f}{\operatorname{argmin}} L(X, Y, f) \tag{2.1}$$

Khá hợp lý khi giả sử rằng nếu thiết kế một kiến trúc khác $\mathcal{F}'$ mạnh mẽ hơn thì sẽ đạt được kết quả tốt hơn. Nói cách khác, ta kỳ vọng hàm số $f_{\mathcal{F}'}^*$. Chỉ khi các lớp hàm lớn hơn chứa các lớp nhỏ hơn, thì mới đảm bảo rằng việc tăng thêm các tầng sẽ tăng khả năng biểu diễn của mạng. Đây là câu hỏi mà He và các cộng sự đã suy nghĩ khi nghiên cứu các mô hình thị giác sâu năm 2016. Ý tưởng trọng tâm của ResNet là mỗi tầng được thêm vào nên có một thành phần là hàm số đồng nhất. Điều này có nghĩa rằng, nếu ta huấn luyện tầng mới được thêm vào thành một ánh xạ đồng nhất $f(x) = x$ thì mô hình mới sẽ hiệu quả ít nhất bằng mô hình ban đầu. Vì tầng được thêm vào có thể khớp dữ liệu huấn luyện tốt hơn, dẫn đến sai số huấn luyện cũng nhỏ hơn. Hàm số đồng nhất nên là hàm đơn giản nhất trong một tầng thay vì hàm null

$f(x) = 0$. Cách suy nghĩ này khá trừu tượng nhưng lại dẫn đến một lời giải đơn giản đáng ngạc nhiên. Đó là khối phần dư [11].

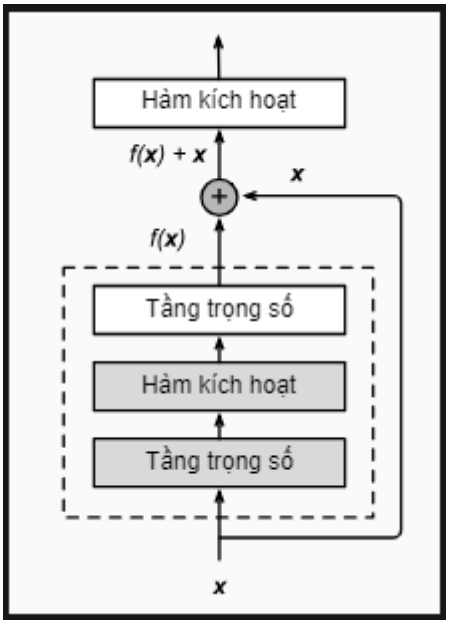
b. Khối phần dư

Một mạng nơ-ron bình thường sẽ có kiến trúc như sau:



Hình 2.4. Kiến trúc mạng nơ-ron

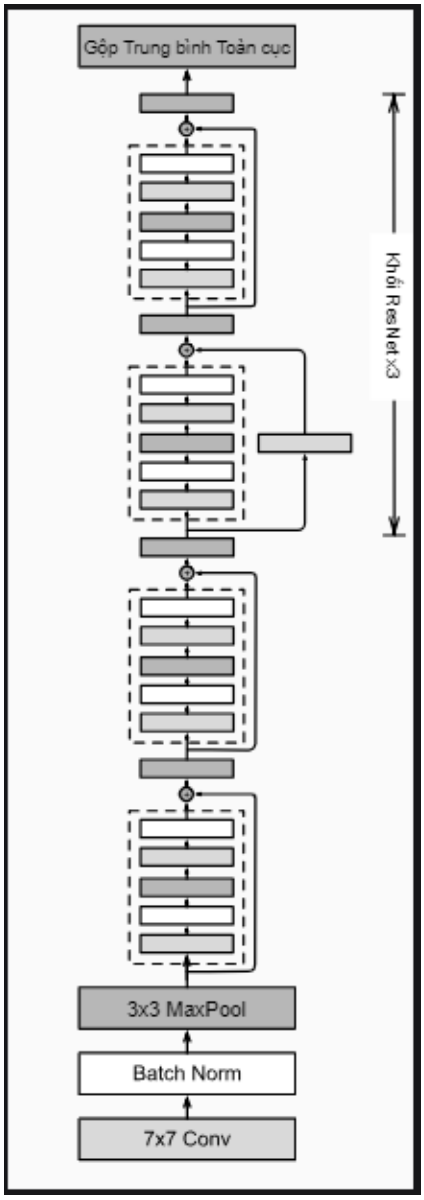
Mạng nhận giá trị đầu vào kí hiệu là x. Giả sử ánh xạ lý tưởng muốn học được là f(x)f(x), và được dùng làm đầu vào của hàm kích hoạt. Nhưng đổi với việc sử dụng khối phần dư thì phần nằm trong viền nét đứt bên phải chỉ cần tham số hoá độ lệch khỏi giá trị x bởi vì ta đã trả về $x + f(x)$ và ở đây sẽ dễ tối ưu hơn vì chỉ cần đặt $f(x) = 0$. Sau đây là kiến trúc khối phần dư:



Hình 2.5. Kiến trúc khối phần dư

c. Kiến trúc mạng

ResNet có thiết kế tầng tích chập 3×3 giống VGG. Khối phần dư có hai tầng tích chập 3×3 với cùng số kênh đầu ra. Mỗi tầng tích chập được theo sau bởi một tầng chuẩn hóa theo batch và một hàm kích hoạt ReLU. Ta đưa đầu vào qua khối phần dư rồi cộng với chính nó trước hàm kích hoạt ReLU cuối cùng. Thiết kế này đòi hỏi đầu ra của hai tầng tích chập phải có cùng kích thước với đầu vào, để có thể cộng lại với nhau.



Hình 2.6. Kiến trúc mạng Resnet

Hai tầng đầu tiên của ResNet giống hai tầng đầu tiên của GoogLeNet: tầng tích chập 7x7 với 64 kênh đầu ra với sải bước 2 và được gộp cực đại bằng với cửa sổ là 3x3 với sải bước 2.

GoogLeNet sử dụng bốn mô-đun được tạo thành từ các khối Inception. ResNet sử dụng bốn mô-đun được tạo thành từ các khối phần dư có cùng số kênh đầu ra. Mô-đun đầu tiên có số kênh bằng số kênh đầu vào. Vì trước đó đã sử dụng tầng gộp cực đại với sải bước 2, nên không cần phải giảm chiều cao và chiều rộng ở mô-đun này.

Trong các mô-đun sau, khối phần dư đầu tiên nhân đôi số kênh, đồng thời giảm một nửa chiều cao và chiều rộng.

Có 4 tầng tích chập trong mỗi mô-đun (không tính tầng tích chập 1×1). Cộng thêm tầng tích chập đầu tiên và tầng kết nối đầy đủ cuối cùng, mô hình có tổng cộng 18 tầng. Do đó, mô hình này thường được gọi là ResNet-18. Có thể thay đổi số kênh và các khối phần dư trong mô-đun để tạo ra các mô hình ResNet khác nhau, ví dụ mô hình 152 tầng của ResNet-152. Mặc dù có kiến trúc lõi tương tự như GoogLeNet, cấu trúc của ResNet đơn giản và dễ sửa đổi hơn. Tất cả các yếu tố này dẫn đến sự phổ cập nhanh chóng và rộng rãi của ResNet.

d. Kết luận

Mạng Resnet sinh ra đã khắc phục được điểm yếu rất lớn trong hầu hết các mô hình mạng. Ý tưởng về khối phần dư (hay còn gọi là kết nối tắt) đã giúp cho việc hạn chế mất mát các đặc trưng quan trọng khi đi qua một mô hình mạng rất sâu.

2.2.4 *Mạng Inception-ResNet*

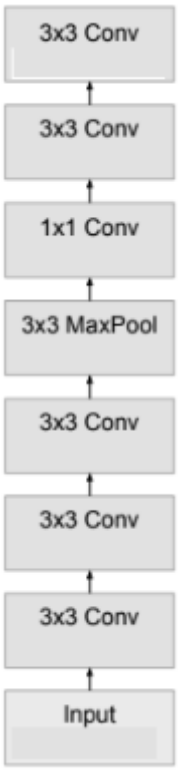
Inception-ResNet là một mạng học rất sâu. Cấu tạo của mạng cũng rất đặc biệt. Kiến trúc mạng inception-ResNet sử dụng nhiều khối mạng con như Inception-A, Inception-B, Inception-C, Reduction-A, Reduction-B, Stem. Chính nhữ đặc điểm trên cho một mạng inception phần dư với hiệu năng rất tốt.

Sau đây là kiến trúc mạng chi tiết:

Hình 2.7. Kiến trúc mạng Inception-ResNet

a. Khối STEM

Dữ liệu sẽ lần lượt đi qua các mạng tích chập để trích chọn ra đặc trưng. Về kiến trúc thì các mạng xếp chồng tại khối này không mới nhưng đem lại hiệu quả khá tốt.

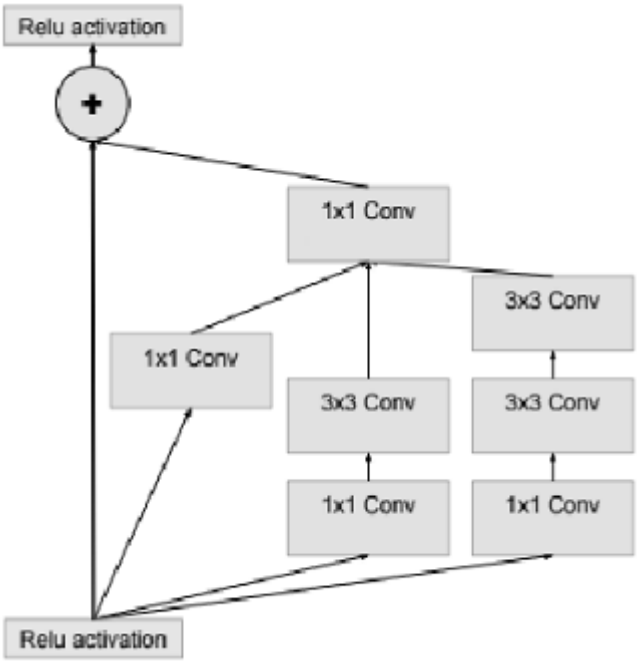


Hình 2.8. Khối STEM

b. Khối iception phần dư(IRB)

Sự kết hợp giữa khối inception và khối phần dư được thể hiện ở việc dữ liệu đầu vào sau khi đi qua các khối inception sẽ được cộng với chính nó. Điều này sẽ giúp cho quá trình học khi vào các lớp sâu cho đến rất sâu thì ta không bị mất đi những thuộc tính quan trọng. Cụ thể ta có khối inception phần dư là Inception-A, Inception-B, Inception-C.

Sau đây là kiến trúc của mạng Inception-A:

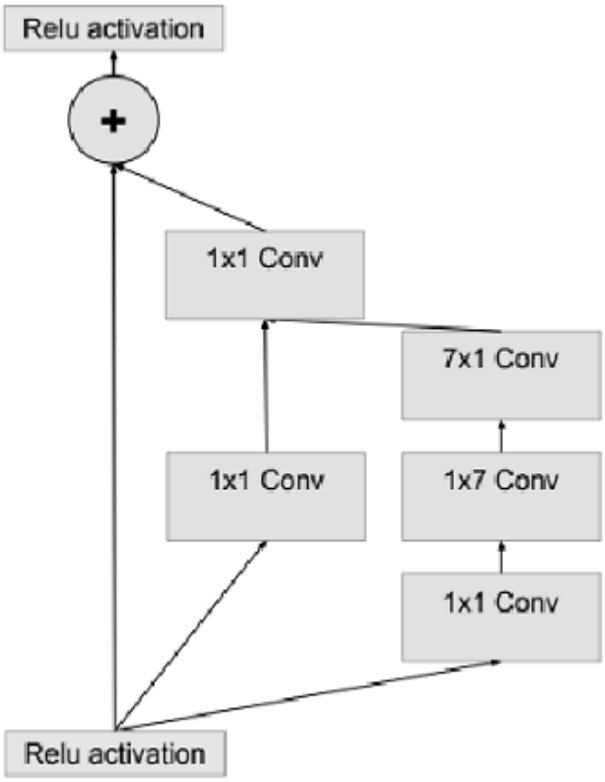


Hình 2.9. Khối Inception-A

Khối Inception này bao gồm 3 nhánh song song. Ba nhánh đều sử dụng các tầng tích chập với kích thước cửa sổ trượt khác nhau để trích chọn đặc trưng. Cụ thể nhánh thứ nhất gồm một tầng tích chập 1x1. Nhánh thứ hai gồm một tầng tích chập 1x1, 3x3. Nhánh thứ ba gồm ba tầng tích chập là 1x1, 3x3, 3x3

Ba nhánh đều sử dụng các tầng tích chập với kích thước cửa sổ trượt khác nhau để trích chọn đặc trưng. Cả ba nhánh sử dụng phần đệm phù hợp để đầu vào và đầu ra của khối có cùng chiều cao và chiều rộng. Cuối cùng là dữ liệu đầu vào sẽ được cộng với đầu ra của tầng tích chập 1x1 cuối cùng để làm input cho hàm kích hoạt tiếp theo

Sau đây là kiến trúc của mạng Inception –B:

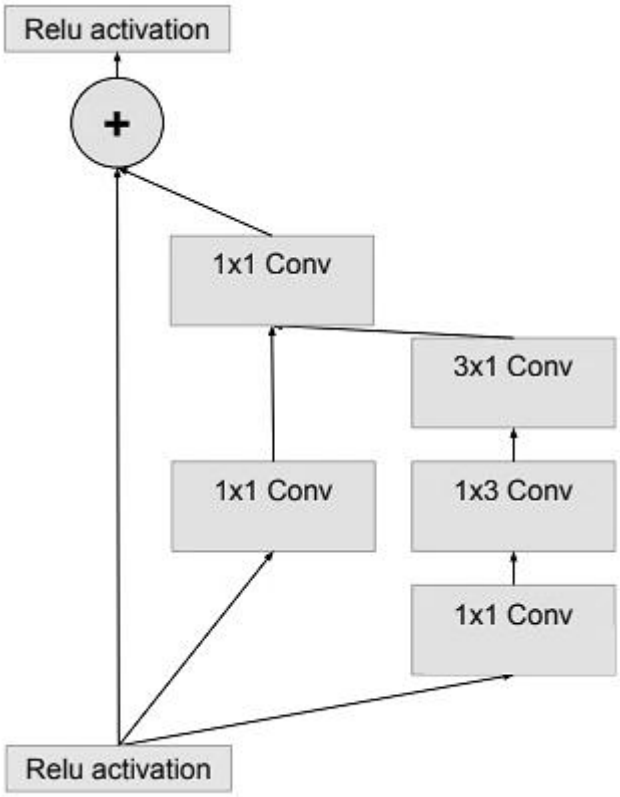


Hình 2.10. Khối Inception-B

Khối Inception B bao gồm hai nhánh song song và thiết kế đơn giản hơn khối inception A. Nhánh thứ nhất chỉ gồm 1 mạng tích chập 1x1. Nhánh thứ hai bao gồm 3 mạng 1x1, 1x7, 7x7.

Cả hai nhánh sử dụng phần đệm phù hợp để đầu vào và đầu ra của khối có cùng chiều cao và chiều rộng. Cuối cùng là 1 tầng tích chập 1x1 để thay đổi số lượng kênh. Cuối cùng là dữ liệu đầu vào sẽ được cộng với đầu ra của tầng tích chập 1x1 cuối cùng để làm input cho hàm kích hoạt tiếp theo

Sau đây là kiến trúc của mạng khối Inception –C:



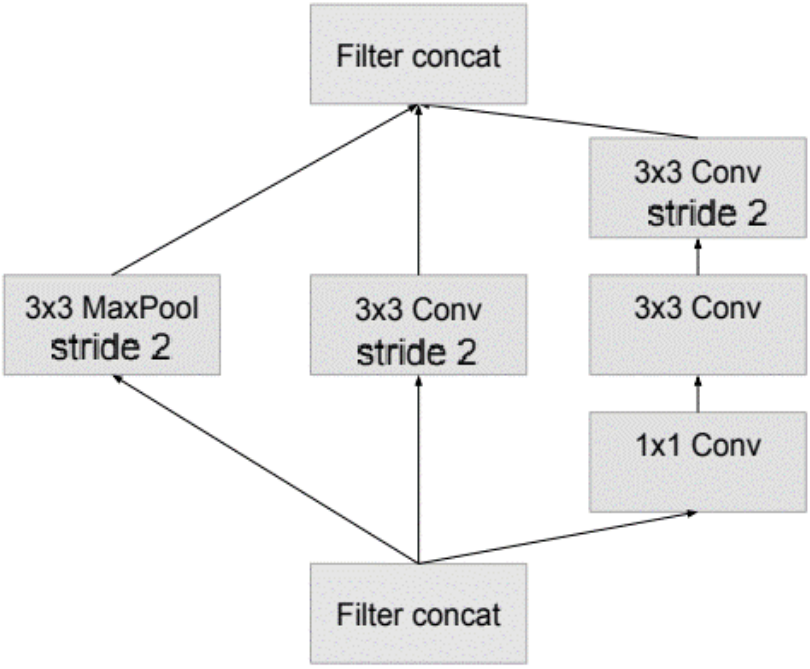
Hình 2.11. Khối Inception-C

Đây là khối inception phần dư cuối cùng được sử dụng trong mạng. Khối inception C có cấu tạo giống như khối inception B chỉ khác về thích thước. Khối inception này cũng sở hữu hai nhánh song song với nhau. Nhánh thứ nhất chỉ có một tầng tích chập 1x1. Nhánh thứ hai gồm 3 tầng tích chập là 1x1, 1x3, 3x1

Cả hai nhánh sử dụng phần đệm phù hợp để đầu vào và đầu ra của khối có cùng chiều cao và chiều rộng. Cuối cùng là 1 tầng tích chập 1x1 để thay đổi số lượng kênh. Cuối cùng là dữ liệu đầu vào sẽ được cộng với đầu ra của tầng tích chập 1x1 cuối cùng để làm input cho hàm kích hoạt tiếp theo

c. Khối reduction

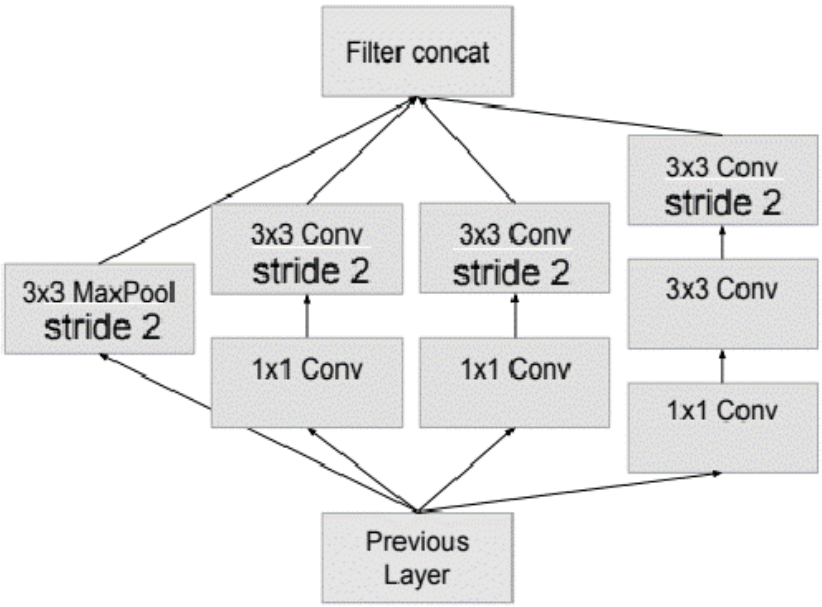
Ta có hai khối reduction là Reduction-A và Reduction-B. Sau đây là kiến trúc của mạng Reduction-A:



Hình 2.12. Khối Reduction A

Đây là khối giảm chiều thứ nhất. Ở khối reduction A bao gồm 3 nhánh chính. Cụ thể nhánh thứ nhất gồm một tầng gộp tối đa 3x3. Nhánh thứ hai gồm một tầng tích chập 3x3. Nhánh thứ ba gồm ba tầng tích chập là 1x1, 3x3, 3x3. Ở tầng cuối trong cả ba nhánh đều sử dụng stride 2 để giảm depth trước khi kết hợp cả ba tầng.

Khối Reduction-B sẽ có kiến trúc như sau:



Hình 2.13. Khối Reduction B

Khối reduction B gồm 4 nhánh xử lý song song bao gồm nhánh thứ nhất sở hữu một tầng gộp cực đại duy nhất. Nhánh thứ hai gồm hai tầng tích chập 1x1 và 3x3. Nhánh thứ ba gồm ba tương tự như nhánh thứ hai nhưng khác số lượng bộ lọc vào tầng 3x3.Nhánh thứ tư bao gồm 3 tầng tích chập xếp chồng đó là 1x1, 3x3, 3x3.. Tại tầng cuối cùng mỗi nhánh đều sử dụng stride là 2 để giảm depth

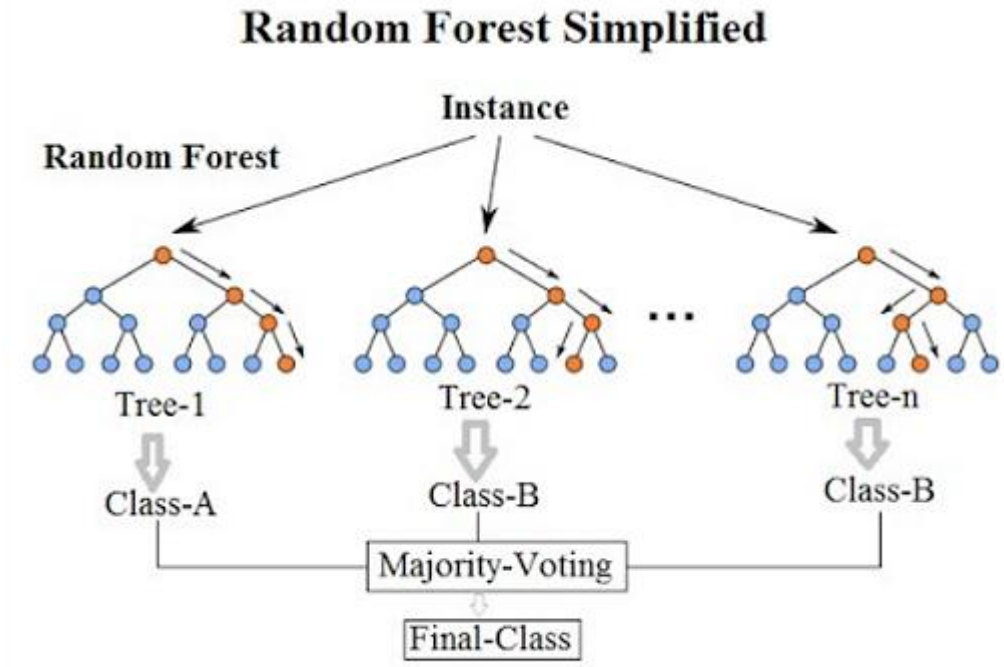
2.3 Rừng ngẫu nhiên

2.3.1 *Giới thiệu*

Rừng ngẫu nhiên (Random Forest) hay còn được gọi là rừng quyết định ngẫu nhiên là một kỹ thuật học tập được sử dụng để giải quyết các nhiệm vụ học tập có giám sát như phân loại và hồi quy. Một tính năng ưu việt của rừng ngẫu nhiên là nó có thể khắc phục được sự quá khớp (overfitting) trên tập dữ liệu đào tạo của mình.

2.3.2 *Kiến trúc*

Kiến trúc mô hình của rừng ngẫu nhiên là một tập hợp của nhiều cây quyết định. Mỗi một cây quyết định sẽ trả ra một kết quả dự báo. Ngoài ra mô hình cũng được chạy trên rất nhiều các sub-sample.



Hình 2.14. Kiến trúc của rừng ngẫu nhiên

Ý tưởng của Random Forest khá đơn giản. Thuật toán này sinh một số cây quyết định (thường là vài trăm) và sử dụng chúng. [12] đã chỉ ra các câu hỏi của cây quyết định sẽ là câu hỏi về các thuộc tính. Ví dụ: "Cánh hoa có dài hơn 1.7cm hay không?". Câu giá trị ở nút lá sẽ là các lớp (class). Sử dụng hàng trăm cây quyết định là bất khả thi với con người, nhưng máy tính có thể làm việc này tương đối dễ dàng. Những cây này thực sự được đào tạo trên các phần khác nhau của cùng một tập huấn luyện. Về mặt kỹ thuật, một cây riêng lẻ được trồng rất sâu có xu hướng học hỏi từ các mẫu rất khó đoán. Loại cây này tạo ra các vấn đề quá mức trên các bộ huấn luyện. Hơn nữa, độ lệch thấp làm cho trình phân loại có hiệu suất thấp ngay cả khi chất lượng dữ liệu của bạn tốt về mặt tính năng.

2.3.3 *Quá trình bootstrapping*

Rừng ngẫu nhiên sử dụng hàng trăm hàng ngàn cây quyết định nhưng nếu như tất cả các cây được dựng theo cùng một cách, chúng sẽ cho những câu trả lời giống nhau. Như vậy chẳng khác gì chúng ta chỉ sử dụng một cây quyết định duy nhất cả. Từ đó, Rừng ngẫu nhiên đã sử dụng bootstrapping để giải quyết vấn đề này. Quá trình bootstrapping được rừng ngẫu nhiên sử dụng là thuật toán sẽ chọn ra ngẫu nhiên các quan sát (observations) để đảm bảo rằng không phải tất cả các cây quyết định cho cùng câu trả lời. Cụ thể rừng ngẫu nhiên thực hiện sẽ xoá một số quan sát và lặp lại một số khác một cách ngẫu nhiên khiến mỗi cây quyết định sẽ đều có những thay đổi riêng.

2.3.4 *Quá trình attribute sampling*

Để chắc chắn các cây quyết định được tạo ra sẽ hoàn toàn khác nhau, rừng ngẫu nhiên sử dụng thêm kỹ thuật lấy mẫu thuộc tính (attribute sampling). Rừng ngẫu nhiên thực hiện quá trình attribute sampling bằng cách loại bỏ ngẫu nhiên một số câu hỏi khi xây dựng cây quyết định. Điều này sẽ giúp tạo nên tính ngẫu nhiên cho thuật toán. Trường hợp câu hỏi tốt nhất bị loại bỏ thì câu hỏi khác sẽ được thay thế và từ đó một cây quyết định hoàn toàn mới sẽ được xây dựng.

2.3.5 *Kết quả dự đoán*

Random Forest là thuật toán thuộc lớp mô hình kết hợp (ensemble model). Kết quả của thuật toán dựa trên bầu cư đa số từ nhiều cây quyết định. Do đó mô hình có độ tin cậy cao hơn và độ chính xác tốt hơn so với những mô hình phân loại tuyến tính đơn giản như logistic hoặc linear regression.

2.3.6 *Tham số của Random Forest*

Các tham số thường được sử dụng khi huấn luyện random forest là n_estimators, max_depth, min_samples_split, max_features, max_features, class_weight, min_impurity_split.

Trong đó n_estimators là số lượng các trees trên một cây quyết định. Tham số max_depth là độ sâu lớn nhất của một cây quyết định. Tham số min_samples_split là số lượng mẫu tối thiểu cần thiết để phân chia một internal node. Nếu kích thước mẫu ở một internal node nhỏ hơn ngưỡng thì ta sẽ không rẽ nhánh internal node. Tham số max_features là số lượng các đặc trưng được xem xét khi tìm kiếm phương án phân chia tốt nhất. Mặc định là toàn bộ các đặc trưng đầu vào. Tham số class_weight là trọng số tương ứng với mỗi lớp. Mặc định là None, khi đó các lớp sẽ có mức độ quan trọng như nhau. Tham số min_impurity_split là ngưỡng để dừng sớm (early stopping) quá trình phát triển của cây quyết định. Nó sẽ tiếp tục phân chia nếu độ vẩn đục (impurity) lớn hơn ngưỡng, trái lại thì nó là node leaf.

2.3.7 *Sử dụng random forest để phân loại, định danh cho khuôn mặt*

Đối với bài toán nhận diện khuôn mặt. nhiều nghiên cứu cho thấy Random forest đem lại kết quả tốt trong cả hai việc là trích chọn đặc trưng và phân loại khuôn mặt [13]. Trong khuôn khổ bài luận này, random forest sẽ chỉ được áp dụng cho việc phân loại khuôn mặt. Dữ liệu ảnh sau khi qua mô hình tích chập Inception-ResNet sẽ trả ra cho chúng ta kết quả là một vectơ đặc trưng. Ta sẽ sử dụng random forest để huấn luyện các vectơ đặc trưng đó với nhãn tương ứng. Sau khi kết thúc huấn luyện ta sẽ thu được mô hình phân loại. Mô hình phân loại này sẽ được sử dụng trong thực thế với dữ liệu đầu vào là một vectơ đặc trưng và nó sẽ trả lại nhãn với xác suất tương

ứng với vectơ đặc trưng đầu vào. Từ đó ta có thể xác định khuôn mặt đầu vào giống với ai nhất.

2.4 Kết luận

Chương này trình bày về sơ đồ thiết kế hệ thống nhận diện khuôn mặt, mạng trích chọn đặc trưng và kỹ thuật phân loại rừng ngẫu nhiên. Trong đó, nội dung chương có đi sâu về mạng học sâu googlenet, resnet từ đó đưa ra ưu nhược điểm khi kết hợp hai mạng thành mạng inception resnet. Chương 3 sẽ trình bày về kết quả thu được khi sử dụng mạng inception resnet kết hợp với rừng ngẫu nhiên cho bài toán nhận diện khuôn mặt.

CHƯƠNG 3. THỬ NGHIỆM VÀ ĐÁNH GIÁ

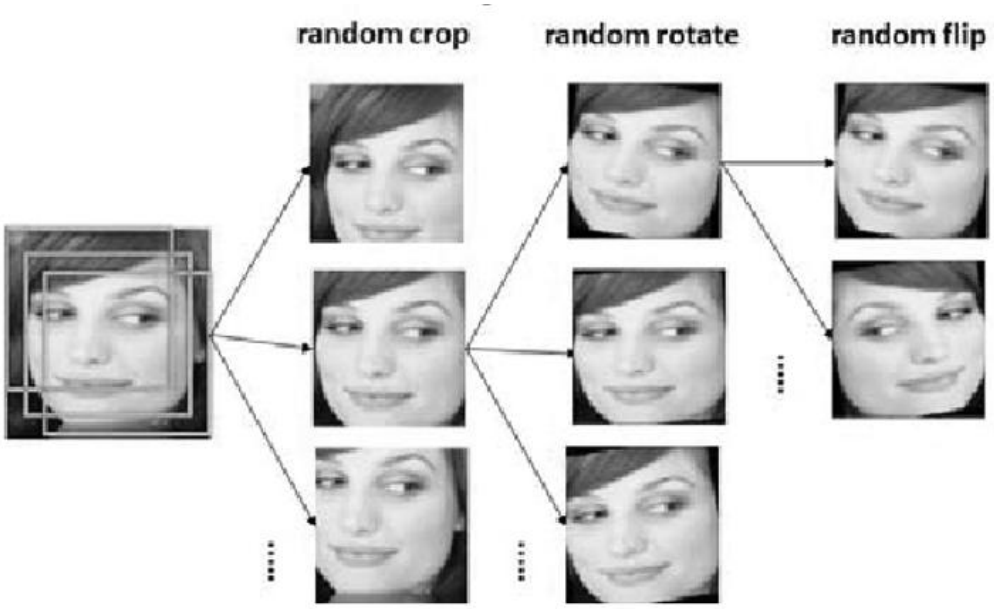
3.1 Bộ dữ liệu đầu vào

Bộ dữ liệu được sử dụng trong luận văn là hai bộ dữ liệu faces94 [10] và CASIA-WebFace [20]. Trong đó bộ dữ liệu faces94 được thu thập từ trung tâm nghiên cứu Center for Machine Perception thuộc đại học kỹ thuật Séc và bộ dữ liệu CASIA-WebFace được sưu tầm từ viện tự động hóa tại học viện khoa học Trung Quốc(CASIA). Luận văn sẽ sử dụng hai bộ dữ liệu vào các mục đích khác nhau. Bộ dữ liệu faces94 chứa khoảng 1300 ảnh của 153 người. Đây là bộ dữ liệu nhỏ được dùng để thử nghiệm và đánh giá hiệu quả của kiến trúc mạng trước khi sử dụng bộ dữ liệu chính để huấn luyện. Bộ dữ liệu CASIA-WebFace chứa khoảng 500.000 ảnh được thu thập từ 10.000 người. Trước khi đưa vào huấn luyện cho mô hình trích chọn đặc trưng thì những bức ảnh này sẽ được duyệt qua mô hình mtcnn để xác định vị trí tọa độ khuôn mặt của mỗi người trong tập dữ liệu.

3.2 Quá trình huấn luyện

Sau khi đã có dữ liệu đầu vào là các khuôn mặt. Ta tiến hành huấn luyện mạng Inception-ResNet với hàm tối ưu được sử dụng là adagrad. Vì adagrad là một hàm tối ưu tự điều chỉnh tốc độ học. Với những người ít kinh nghiệm trong bài toán này việc lựa chọn một mô hình kèm với một phương pháp tối ưu cần truyền vào là tốc độ học sẽ rất khó khăn vì nó tốn thời gian, công sức để đánh giá nên em quyết định sử dụng adagrad. Adagrad coi tốc độ học cũng là một tham số và điều chỉnh tốc độ học sao cho learning nhỏ khi mà dữ liệu ít khác biệt và tốc độ học khi dữ liệu nhiều khác biệt. Việc tự điều chỉnh tốc độ học trong hàm tối ưu này sẽ giúp mô hình của ta dễ dàng sử dụng hơn. Hàm mục tiêu được sử dụng trong nghiên cứu này là cross-entropy-loss.

Để cho mô hình có thể học tốt ta sẽ tiến hành sử dụng các phương pháp tăng cường dữ liệu qua các tham số là random_rotate, random_flip, random_crop.



Hình 3.1. Một số phương pháp tăng cường dữ liệu

Trong đó random_crop là cắt ngẫu nhiên một phần của bức ảnh giúp mô hình tránh được việc học quá khớp. Tham số random_flip là lật ảnh. Bức ảnh sẽ được lật ngẫu nhiên sang trái và phải. Tham số random_rotate sẽ tăng cường dữ liệu ảnh được xoay từ ảnh gốc với thiết lập là một góc 10 độ sang trái và phải. Một mô hình trích chọn đặc trưng tốt khi dữ liệu đủ lớn và đa dạng để việc học hạn chế rơi vào tình trạng quá khớp. Các phương pháp tăng cường dữ liệu trên sẽ được áp dụng trên hai bộ dữ liệu faces94 và CASIA-WebFace trong quá trình huấn luyện.

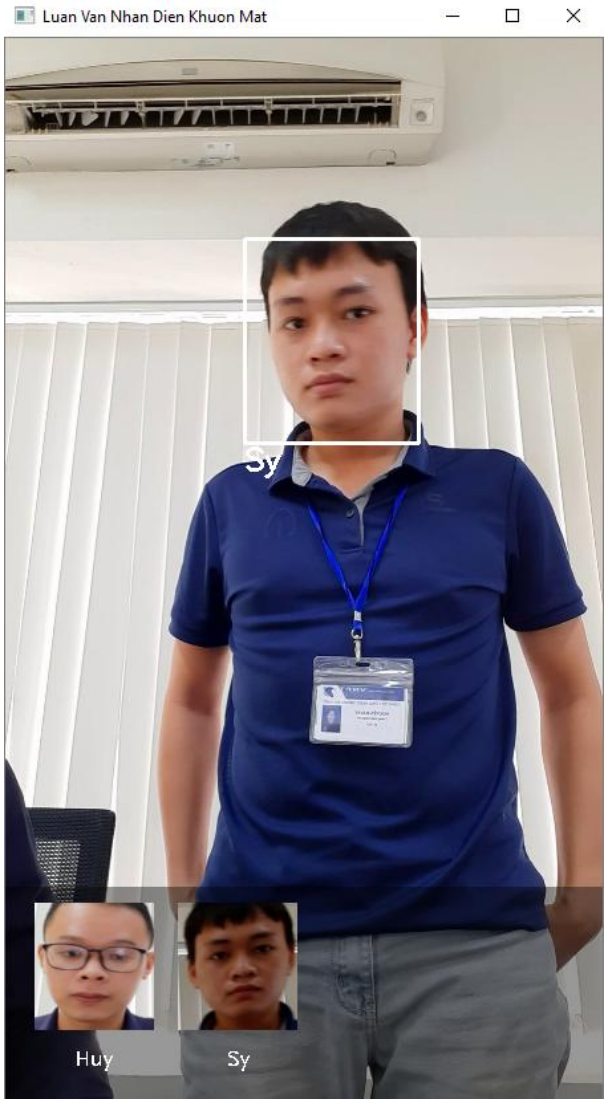
Quá trình huấn luyện sẽ được áp dụng trên một số kiến trúc mạng với tập dữ liệu faces94 để đánh giá về hiệu quả trước khi huấn luyện thật trên bộ dữ liệu chính là CASIA-WebFace. Các bộ dữ liệu lần lượt được đưa qua mô hình mtcnn để xác định vị trí khuôn mặt trước khi đưa vào huấn luyện. Kiến trúc được áp dụng để huấn luyện với faces94 là Inception-Resnet nguyên bản và Inception-Resnet sau khi được cắt bỏ hai tầng Inception-C. Dữ liệu faces94 sẽ được tách làm hai phần huấn luyện và kiểm thử với tỉ lệ 7:3. Thời gian huấn luyện cho một lần duyệt(epoch) khoảng 15 phút với dữ liệu faces94 đã được tăng và sau khoảng 35 epoch thì hai mạng đều hội tụ và độ chính xác không còn tăng và lỗi không giảm khi tiếp tục huấn luyện. Kết quả đánh giá trên tập test cho thấy với mạng Inception-Resnet nguyên bản thì độ chính xác là 99.87% và mạng Inception-Resnet sau khi được cắt bỏ hai tầng Inception-C là

97.68%. Thử nghiệm trên cho thấy kết mạng Inception-Resnet nguyên bản hiệu quả hơn. Ta thấy mạng không đủ độ sâu như mạng Inception-Resnet sau khi được điều chỉnh ở trên khả năng học sẽ bị kém đi rõ rệt và độ chính xác chỉ có 97,68% trên một tập dữ liệu nhỏ, đơn giản thì sẽ rất khó để huấn luyện ra được một mô hình tốt với bộ dữ liệu lớn. Từ đánh giá trên, kiến trúc mạng Inception-Resnet nguyên bản sẽ được sử dụng để huấn luyện mô hình trích chọn đặc trưng này. Dữ liệu được sử dụng cho việc huấn luyện này là CASIA-WebFace. Đây là một tập dữ liệu về khuôn mặt rất lớn được thu thập từ học viện khoa học Trung Quốc. Để có một mô hình trích chọn đặc trưng thật tốt thì số lượng dữ liệu về khuôn mặt phải đủ lớn, đủ độ đa dạng vè hình dáng, kích thước cũng như màu sắc. Thời gian diễn ra khoảng 350 giờ với 90 lần duyệt. Sau khi đã có được mô hình trích chọn đặc trưng tùy vào từng bài toán ta sẽ tiến hành trích chọn đặc trưng của các nhãn đầu vào để mô hình phân loại bằng thuật toán random forest học các đặc tính đó.

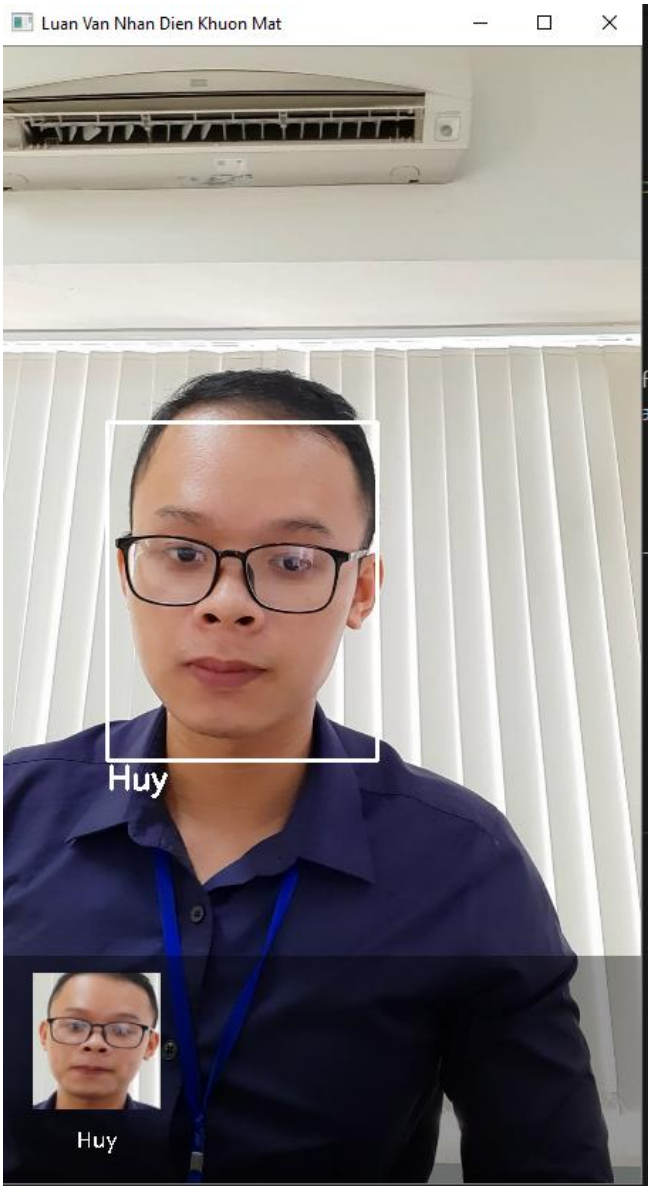
3.3 Thử nghiệm chạy hệ thống nhận diện khuôn mặt nhận diện khách hàng VIP của khách sạn

Máy chủ được sử dụng cho việc thực nghiệm sử dụng vi xử lý Intel Core i5-8400 với xung nhịp cơ bản là 2.8GHz. Dung lượng bộ nhớ trong là 16 GB và bộ xử lý đồ họa được sử dụng là GTX 1070 Ti 8G GDDR5. Hệ thống nhận diện khách hàng VIP sẽ được phát triển trên nền tảng Ubuntu và ngôn ngữ lập trình được sử dụng là python version 3.6 cùng với các bộ thư viện như opencv, keras, tensorflow để phục vụ cho bài toán xử lý ảnh.

Mô hình thử nghiệm được huấn luyện với tập dữ liệu khoảng 200 khách hàng, mỗi vị khách có khoảng 10-30 ảnh khuôn mặt trong cơ sở dữ liệu. Trong quá trình phân loại em đã chọn ra được ngưỡng là 40% để xác định khuôn mặt. Ngưỡng này sử dụng được sử dụng để tránh việc xác suất khuôn mặt trả ra quá thấp gây ra việc nhận nhầm thông tin. Sau khi thiết lập địa chỉ luồng phát video trực tiếp thì ta có thể khởi động hệ thống. Sau đây là một số hình ảnh hệ thống hoạt động :



Hình 3.2. Hệ thống nhận diện khuôn bình thường



Hình 3.3. Hệ thống nhận diện khuôn mặt có đeo kính

Với mỗi khách hàng có trong danh sách sau khi xuất hiện sẽ được lưu lại 10 giây để dễ dàng quan sát. Hệ thống có thể nhận diện khuôn mặt một cách bình thường cả khi người đó đang đeo kính. Từ đó có thể thấy mô hình trích rút đặc trưng và mô hình phân loại đang hoạt động rất tốt trong thực tế. Hệ thống nhận diện khách hàng mỗi giây có thể xử lý 5 khung hình với độ phân giải fullhd.

3.4 Đánh giá

Để đánh phân loại ta sẽ thực hiện lấy vectơ đặc trưng từ mô hình trích chọn đặc trưng và đưa vào randomforest để huấn luyện. Thiết lập trong random forest của

em hiện tại đang là 100 cây. Các tập dữ liệu được sử dụng để đánh giá trong luận văn này là Faces94 [14], Faces95[15], Faces96[16], Grimace[17]. Tập dữ liệu sẽ được dùng 75% cho việc huấn luyện và 25% còn lại cho việc kiểm thử mô hình phân loại. Sau đó ta sẽ so sánh kết quả của mô hình vừa được huấn luyện với kết quả của một số phương pháp được sử dụng tron nghiên cứu [18] [19] như là LDA, PCA, LBP, SVM based on LDA và MLP kết hợp với PCA và DCT.

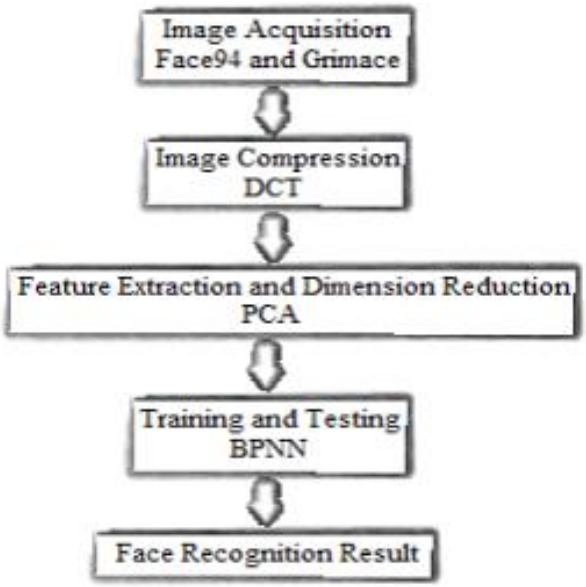
Bảng 2.1. Bảng đánh giá độ chính xác giữa các mô hình

	Faces94 dataset	**Faces95 dataset**	**Faces96 dataset**	**Grimace dataset**
Inception Resnet V1+ Random forest	99.1%	99.5%	98.2%	99.95
Multilayer perceptron + PCA + DCT	100%	-	-	100%
SVM based on LDA (RBF kernel)	97.4%	-	95.1%	100 %
LBP	85.93%	80.47%	84.145	86.45%
PCA	72.1%	69.87%	70.95%	74.79%
LDA	79.39%	76.61%	78.34%	81.93%

Sự kết hợp giữa hai mô hình Inception Resnet V1 và Random forest đã cho lại kết quả khá cao với các tập dữ liệu trên. Sự khác biệt lớn nhất giữa các phương pháp là việc trích chọn đặc trưng. Mạng Inception Resnet V1 có thể học được các đặc trưng trên khuôn mặt tốt hơn hẳn so với các phương pháp học máy truyền thống. Kiến trúc mạng tích chập này có khả năng học đặc trưng cực kì tốt vì nó vẫn có thể trích rút các đặc trưng chính của khuôn mặt trong các trường hợp thiếu sáng, khuôn mặt không đầy đủ hay như người đang đeo kính.

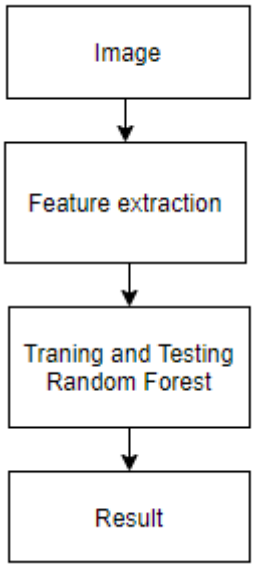
Dựa trên kết quả ta có thể thấy phương pháp học máy như SVM cũng đem lại kết quả khá ấn tượng nhưng kết quả cao nhất lại là Inception Resnet V1 + Random forest và MLP + PCA + DCT. Chúng ta sẽ đánh giá chi tiết về ưu nhược điểm của

hai phương pháp này. Dựa vào kết quả trên ta có thể thấy việc kết hợp của nhiều phương pháp MLP, PCA và DCT đem đến hiệu quả khá tốt với mức độ chính xác luôn hơn Inception Resnet V1 và Random forest từ 0.1-1%.



Hình 3.4. Luồng xử lý của hệ thống sử dụng phương pháp PCA và DCT

Đánh giá qua về mô hình trên ta có thể thấy cả hai phương pháp này đều kết hợp giữa học máy và học sâu. Trong hai thiết kế, một thiết kế sử dụng học máy để trích chọn đặc trưng và dùng phương pháp học sâu để phân loại, một thiết kế sử dụng học sâu để trích chọn đặc trưng và dùng phương pháp học máy để phân loại. Cả hai thiết kế dù đem lại kết quả nhận diện tốt nhưng trong thiết kế sử dụng MLP, PCA, DCT thì ta có thể thấy thiết kế rất phức tạp và mô hình phân loại đang được sử dụng là một mạng lan truyền ngược lớn. Thời gian tối thiểu để nhận diện một khuôn mặt rơi vào khoảng 20 giây. Đó là một khoảng thời gian quá lớn vì phương pháp này sử dụng quá nhiều kỹ thuật.



Hình 3.5. Luồng xử lý của hệ thống Inception Resnet và Random forest

Còn lại với Inception Resnet V1 kết hợp với Random forest cho ta độ chính xác thấp hơn khoảng 1% so với phương pháp trên nhưng thời gian để nhận diện khuôn mặt chỉ rơi vào khoảng 0.2 giây. Từ đó ta thấy phương pháp sử dụng mạng Inception Resnet V1 kết hợp với Random forest dễ dàng triển khai thực tế hơn dù độ chính xác từ phương pháp này đem lại thấp hơn PCA kết hợp với DCT một chút.

3.5 Kết luận

Chương này trình bày về quá trình huấn luyện, kiểm thử, đánh giá chất lượng của mô hình trích chọn đặc trưng inception resnet và thử nghiệm chạy hệ thống nhận diện khách hàng VIP tại khách sạn. Kết quả của hệ thống khá tốt với thời gian xử lý khoảng 0.2 giây một khung hình cùng với khả năng nhận diện được khuôn mặt từ nhiều góc độ, sắc thái, điều kiện khác nhau, điều mà các mô hình học máy truyền thống chưa xử lý tốt.

KẾT LUẬN

Bài toán nhận diện khuôn mặt không còn là một vấn đề mới nhưng nhận diện khuôn mặt dựa trên các mạng học sâu đang rất được quan tâm.

Trên cơ sở tìm hiểu và nghiên cứu các phương pháp nhận diện khuôn mặt áp dụng vào hệ thống nhận diện khuôn mặt qua camera, luận văn đã đạt được những kết quả sau. Đó là tìm hiểu, thực nghiệm các mô hình mạng học sâu, học máy như nơ-ron tích chập và rừng ngẫu nhiên để xử lý bài toán nhận diện. Phân tích kết quả thu được và tìm ra mô hình mạng học sâu thích hợp cho bài toán nhận diện khuôn mặt. Xây dựng thành công hệ thống nhận diện khuôn mặt qua camera có chức năng phát hiện khuôn mặt trực tiếp qua video.

Hệ thống sau khi được phát triển cho thấy rằng việc mô hình mạng trích chọn đặc trưng đang đem tới kết quả rất là tốt tuy nhiên cũng xuất hiện một số mặt hạn chế. Đó là mô hình phân loại này không phù hợp với tập dữ liệu quá lớn. Điều này không phải là do kết quả phân loại kém mà do thời gian huấn luyện lại mô hình phân loại khá lâu.

Qua những kết quả và hạn chế của ứng dụng đã cho thấy việc xây dựng hệ thống nhận diện khuôn mặt còn đòi hỏi phải thực hiện, nghiên cứu và áp dụng thêm nhiều phương pháp mới. Về hướng pháp triển tương lại, luận văn sẽ được đề xuất nghiên cứu thêm một số hướng mới. Hướng thứ nhất đó là tiến hành nghiên cứu, áp dụng và đánh giá các mô hình phân loại mới khi áp dụng vào bài toán thay rừng ngẫu nhiên như là Sparse Representation-based classification, Support Vector Machine, Linear Classifier. Hướng tiếp theo là nghiên cứu một số phương pháp sử dụng khoảng cách khi huấn luyện với triplet loss. Phương pháp này sẽ cần dung lượng bộ nhớ lớn nhưng không cần huấn luyện và sử dụng mô hình phân loại.

Do thời gian thực hiện luận văn không nhiều nên chắc chắn luận văn không thể tránh khỏi những hạn chế và thiếu sót. Em rất mong nhận được các ý kiến đóng góp.

DANH MỤC CÁC TÀI LIỆU THAM KHẢO

[1] Abdulrahman Alkandari, Soha Jaber Aljaber (2015), "*Principle Component Analysis algorithm (PCA) for image recognition*", ICCTIM, pp. 76-80.

[2] Liton Chandra Paul, Abdulla Al Sumam (2012), "*Face Recognition Using Principal Component Analysis Method*" International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 1, pp. 135-139.

[3] LDA Juwei Lu, Kostantinos N. Plataniotis, and Anastasios N. Venetsanopoulos (2003), "Face Recognition Using LDA-Based Algorithms", IEEE TRANSACTIONS ON NEURAL NETWORKS, pp. 195-200.

[4] Alaa Eleyan, Hasan Demirel (2006), "*PCA and LDA Based Face Recognition Using Feedforward Neural Network Classifier*" Conference: Multimedia Content Representation, pp. 200-206.

[5] Philip H. Swain, Hans Hauska (1977), "*The decision tree classifier: Design and potential*" IEEE Transactions on Geoscience Electronics 15, pp. 142-147.

[6] Georgios Karalis (2020), "*Decision Trees and Applications"* Advances in Experimental Medicine and Biology 1194, pp. 239-242.

[7] Manish Mishra, Monika Srivastava (2014), "*A view of Artificial Neural Network*", IEEE ICAETR - 2014, pp. 1-3.

[8] Saad Albawi, Tareq Abed Mohammed (2017), "*Understanding of a Convolutional Neural Network*", International Conference on Engineering and Technology (ICET).

[9] Yushi Chen, Hanlu Jiang, Chunyang Li, Xiuping Jia, Pedram Ghamisi (2016), "*Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks*" IEEE Transactions on Geoscience and Remote Sensing 54, pp. 6232 – 6251.

[10] Musab Coşkun, Ayşegül Uçar, Özal Yildirim, Yakup Demir (2017), "*Face recognition based on convolutional neural network*" 2017 International Conference on Modern Electrical and Energy Systems (MEES), pp. 376-379.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2016), "*Deep Residual Learning for Image Recognition*" 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778.

[12] Jehad Ali, Rehanullah Khan, Nasir Ahmad (2012), "Random Forests and Decision Trees" JCSI International Journal of Computer Science Issues 9, pp. 272-276.

[13] Haiyan Guan, Jonathan Li (2012), "*RANDOM FORESTS-BASED FEATURE SELECTION FOR LAND-USE CLASSIFICATION USING LIDAR DATA AND ORTHOIMAGERY*" ISPRS - International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences, pp. 203-208.

[14] Faces94, http://cmp.felk.cvut.cz/~spacelib/faces/faces94.html. Truy cập ngày 01/11/2020.

[15] Faces95, http://cmp.felk.cvut.cz/~spacelib/faces/faces95.html. Truy cập ngày 01/11/2020.

[16] Faces96, http://cmp.felk.cvut.cz/~spacelib/faces/faces96.html. Truy cập ngày 01/11/2020.

[17] Grimace, http://cmp.felk.cvut.cz/~spacelib/faces/grimace.html. Truy cập ngày 01/11/2020.

[18] A Vinay, Abhijay Gupta, Aprameya Bharadwaj, Arvind Srinivasan, K N Balasubramanya Murthy, S Natarajan (2018*),"Deep Learning on Binary Patterns for Face Recognition*", International Conference on Computational Intelligence and Data Science, pp. 77-83.

[19] Nawaf Hazim (2016), "*Face Recognition using PCA-BPNN with DCT Implemented on Face94 and Grimace Databases*", International Journal of Computer Applications, pp. 8-13.

[20] CASIA-WebFace, https://pgram.com/dataset/casia-webface/. Truy cập ngày 01/11/2020.