TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ-VIỄN THÔNG

====o0o====

BÁO CÁO ĐỒ ÁN I

Đề tài:

NHẬN DIỆN CẢM XÚC KHUÔN MẶT QUA WEBCAM

GVHD: ThS Nguyễn Thị Kim Thoa

Nhóm SV thực hiện: **Hoàng Lê Diệu Hường (20151923) – Điện tử 2 K60**

Nguyễn Minh Hiếu (20151336) – Điện tử 3 K60

Phan Văn Hòa (20151599) – Điện tử 4 K60

Hà Nội, tháng 1 năm 2019

LỜI NÓI ĐẦU

Trong xu hướng xã hội ngày càng phát triển, trí tuệ nhân tạo đã góp phần không nhỏ trong việc giúp con người tiết kiệm sức lao động, đẩy nhanh quá trình tự động hóa và số hóa nền kinh tế của nhân loại. Điều này dẫn đến việc các máy móc tinh vi ra đời để hỗ trợ nhằm mang lại năng suất lao động cao nhất cho người lao động, nhà sản xuất và sau cùng là mang đến sản phẩm mang tính hiệu quả nhất đến tay người tiêu dùng, đáp ứng của nhu cầu hiện đại ngày càng cao.

Convolutional Neural Network (CNN – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay như hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động. Đi cùng với các nhu cầu đó, nhóm chúng em thực hiện đề tài "Nhận diện cảm xúc khuôn mặt qua webcam".

Cuối cùng, chúng em xin gửi lời cảm ơn chân thành tới cô giáo ThS Nguyễn Thị Kim Thoa đã hướng dẫn tận tình, tạo điều kiện tốt nhất giúp chúng em hoàn thành đề tài này!

MỤC LỤC

Danh sách hình vẽ

Danh sách bảng biểu

Chương 1. Xác định nhu cầu

Chương mở đầu sẽ giới thiệu được vấn đề mà đề tài cần giải quyết, mô tả được các phương pháp hiện có để giải quyết vấn để, trình bày mục đích của đề tài song song với việc giới hạn phạm vi của vấn đề mà đề tài sẽ tập trung giải quyết. Phần này cũng sẽ giới thiệu tóm tắt các nội dung sẽ đượcc trình bày trong các chương tiếp theo.

1.1. Nhu cầu về trí tuệ nhân tạo trong đời sống

Bài toàn nhận dạng cảm xúc đã bắt đầu được nghiên cứu từ những năm 1970 nhưng kết quả đạt được vẫn còn nhiều hạn chế. Hiện nay vấn đế này vẫn đang được rất nhiều người quan tâm bởi tính hấp dẫn cùng những vấn đề phức tạp của nó. Ngoài ra, việc kiểm thử bằng việc truyền dữ liệu đầu vào rất thủ công và không thực tế, cần phát triển hệ thống nhận diện thời gian thực như thông qua máy ảnh, webcam,...

1.2. Kết luận

Nhóm đã xác định được mục tiêu và lý do cần thực hiện đề tài "Nhận diện cảm xúc khuôn mặt qua webcam".

Chương 2. Xác định yêu cầu kỹ thuật

Chương 2 sẽ phân tích các yêu cầu về kỹ thuật của sản phẩm. Yêu cầu kỹ thuật được chia thành yêu cầu chức năng và yêu cầu phi chức năng. Yêu cầu chức năng là các yêu cầu về các chức năng chính của sản phẩm. Yêu cầu phi chức năng là các yêu cầu về môi trường hoạt động, nguồn điện, giao diện,… Để nêu các yêu cầu kỹ thuật trực quan hơn, nhóm 18 đã kẻ bảng yêu cầu chức năng và yêu cầu phi chức năng.

2.1. Yêu cầu chức năng

Bảng 2.1. Yêu cầu chức năng

Yêu cầu chức năng	
Dữ liệu vào	Ảnh số thu được thông qua webcam máy tính
Dữ liệu ra	Kết quả phát hiện khuôn mặt bằng khung và nhận diện cảm xúc bằng chữ

Nhóm chúng em đã thực hiện một hệ thống nhận diện cảm xúc khuôn mặt, bao gồm các trạng thái cảm xúc: Vui, Buồn, Cười, Ngáp ngủ,... Người dùng sẽ sử dụng webcam để Các yêu cầu chức năng này được tổng hợp trên Bảng 2.1 ở trên.

2.2. Yêu cầu phi chức năng

Bảng 2.2. Yêu cầu phi chức năng

Yêu cầu phi chức năng	
Chất lượng webcam	600x480
RAM	8GB
Tốc độ CPU	2.7 GHz

2.3. Kết luận

Kết thúc chương 2, nhóm đã có được bảng yêu cầu kỹ thuật đầy đủ và chi tiết, là cơ sở cho việc thiết kế và kiểm tra hệ thống.

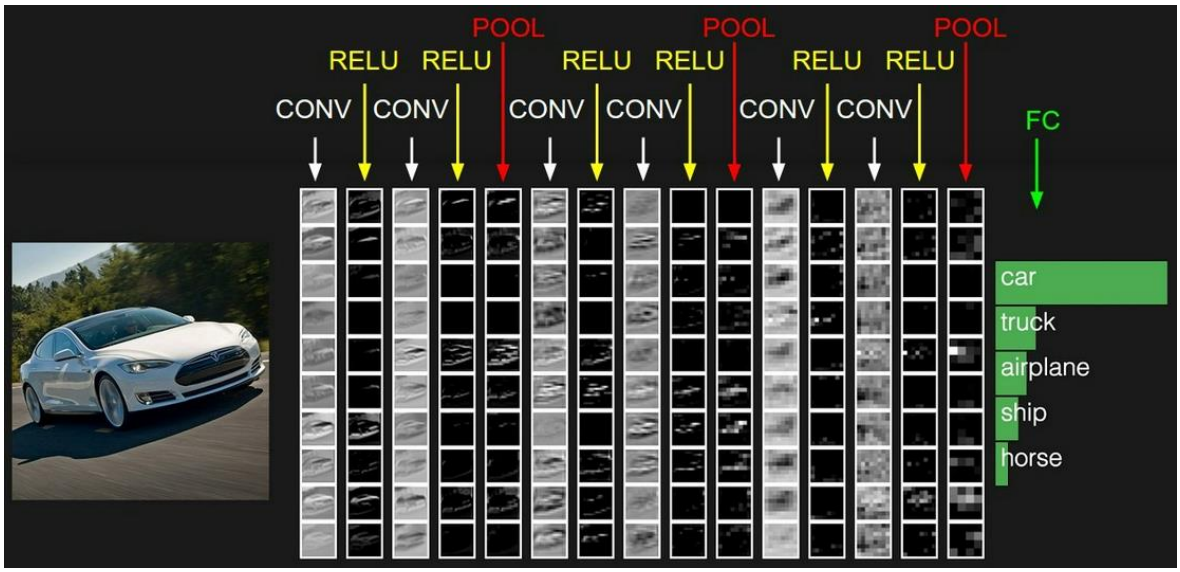
Chương 3. Cơ sở lý thuyết

Chương 3 sẽ phân tích lý thuyết mạng nơ-ron tích chập nói chung, mạng MobiNet nói riêng, Thư viện mã nguồn mở OpenCV, Tensorflow và các ứng dụng.

3.1. Mạng nơ-ron tích chập

3.1.1. Cấu trúc mạng nơ -ron tích chập

ConvNet có 02 phần chính: **Lớp trích lọc đặc trưng của ảnh** (Conv, Relu và Pool) và **Lớp phân loại** (FC và softmax).



Hình 3.1. Cấu trúc mạng Nơ-ron tích chập

Hình 3.1 chính là cấu trúc của mạng. Mạng Nơ-ron tích chập là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo. Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ

vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

CNNs so sánh hình ảnh theo từng mảnh. Các mảnh mà nó tìm được gọi là các feature. Bằng cách tìm ở mức thô các feature khớp nhau ở cùng vị trí trong hai hình ảnh, CNNs nhìn ra sự tương đồng tốt hơn nhiều so với việc khớp toàn bộ bức ảnh.

Mỗi feature giống như một hình ảnh mini - một mảng hai chiều nhỏ. Các feature khớp với các khía cạnh chung của các bức ảnh. Trong trường hợp các hình ảnh X, các feature bao gồm các đường chéo và hình chữ thập, sẽ nắm bắt tất cả những đặc điểm quan trọng của hầu hết các hình ảnh X. Những feature này có lẽ sẽ khớp với phần cánh và phần trung tâm của bất kỳ hình ảnh một X nào.

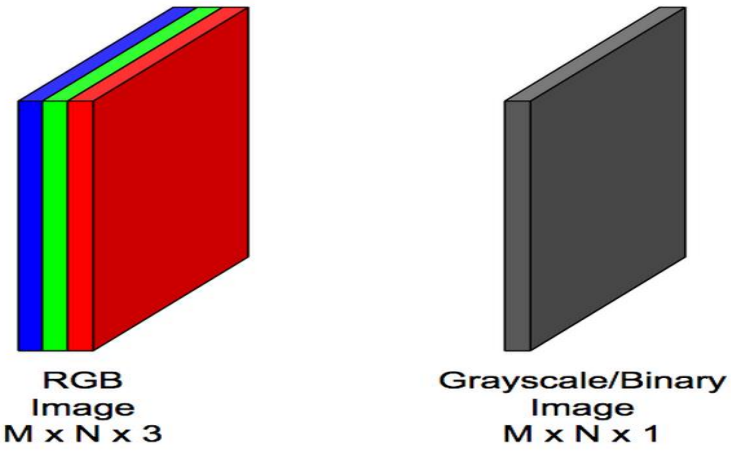
Đầu vào (dữ liệu training):

Input đầu vào là một bức ảnh được biểu diễn bởi ma trận pixel với kích thước: [W x H x D]

W: chiều rộng

H: chiều cao

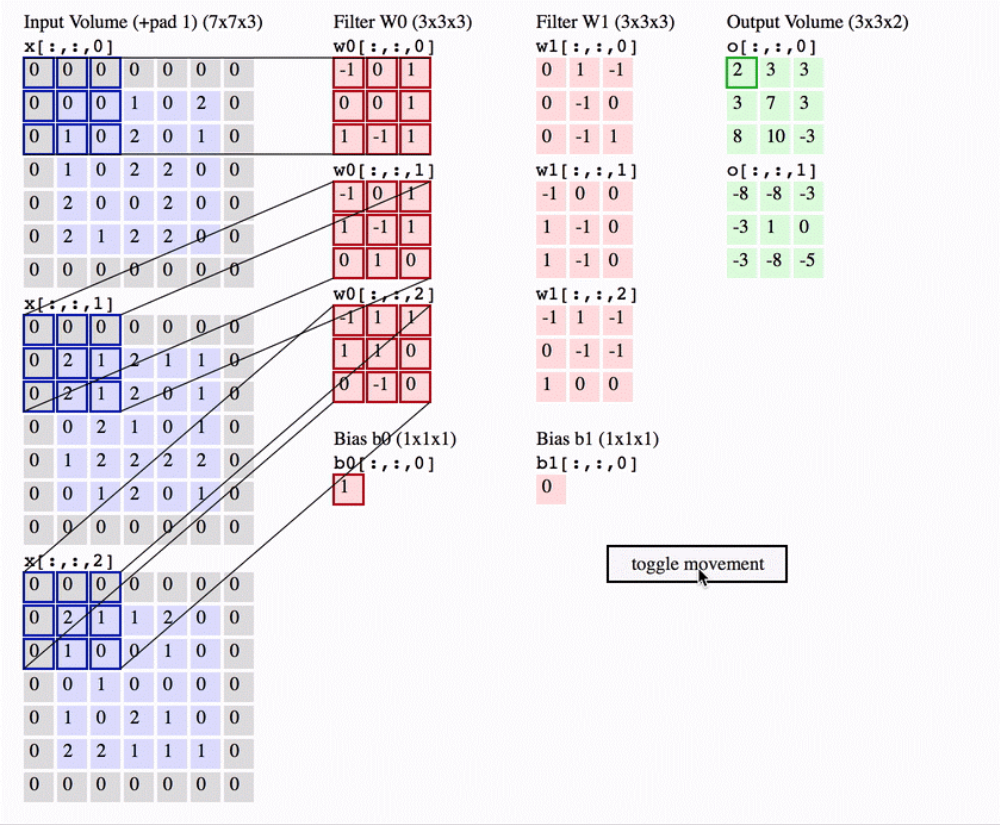
D: Là độ sâu, hay dễ hiểu là số lớp màu của ảnh. Ví dụ ảnh RBG sẽ là 3 lớp ảnh Đỏ, Xanh Dương, Xanh Lá (hình 3.2)



Hình 3.2. Kênh màu của ảnh RGB và ảnh xám

3.1.2. Lớp Convolution

Mục tiêu của các lớp tích chập là trích chọn các đặc trưng của ảnh đầu vào.



Hình 3.3. Ví dụ lớp tích chập

Hình 3.3 trình bày một ví dụ trực quan của lớp Convolution. Ảnh đầu vào được cho qua một bộ lọc chạy dọc bức ảnh. Bộ lọc có kích thước là (3x3 hoặc 5x5) và áp dụng phép tích vô hướng để tính toán, cho ra một giá trị duy nhất. Đầu ra của phép tích chập là một tập các giá trị ảnh được gọi là mạng đặc trưng (features map).

Dưới đây là một số các khái niệm cơ bản của phần này:

Filter, Kernel hay Feature Detector đều là cách gọi của ma trận lọc (như mình đã đề cập ở trên). Thông thường, ở các lớp đầu tiên của Conv Layer sẽ có kích thước là [5x5x3]

Convolved Feature, Activation Map hay Feature Map là đầu ra của ảnh khi cho bộ lọc chạy hết bức ảnh với phép tích vô hướng.

Receptive field là vùng ảnh được chọn để tính tích chập, hay bằng đúng cái kích thước của bộ lọc.

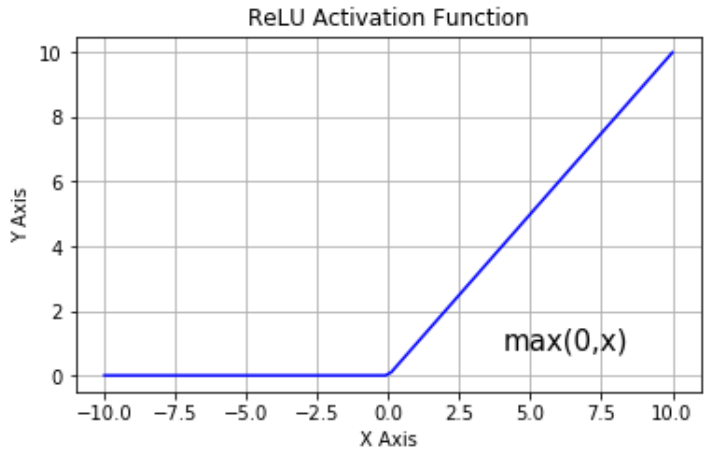
Depth là số lượng bộ lọc. Lưu ý: ở đây là số lượng bộ lọc (filter) chứ không phải số lượng kênh màu RBG như ở trên.

Stride được hiểu là khoảng cách dịch chuyển của bộ lọc sau mỗi lần tính. Ví dụ khi stride=2. Tức sau khi tính xong tại 1 vùng ảnh, nó sẽ dịch sang phải 2 pixel. Tương tự với việc dịch xuống dưới

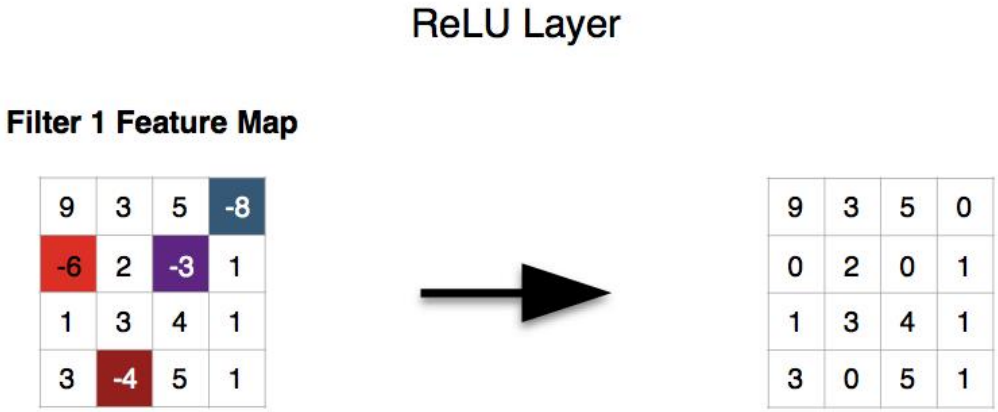
Zero-Padding là việc thêm các giá trị 0 ở xung quanh biên ảnh, để đảm bảo phép tích chập được thực hiện đủ trên toàn ảnh.

3.1.3. Lớp ReLU

ReLU layer áp dụng các hàm kích hoạt (activation function – hình 3.4) max(0,x) lên đầu ra của Conv Layer, có tác dụng đưa các giá trị âm về thành 0. Layer này không thay đổi kích thước của ảnh và không có thêm bất kì tham số nào.



Hình 3.4. Đồ thị hàm ReLU



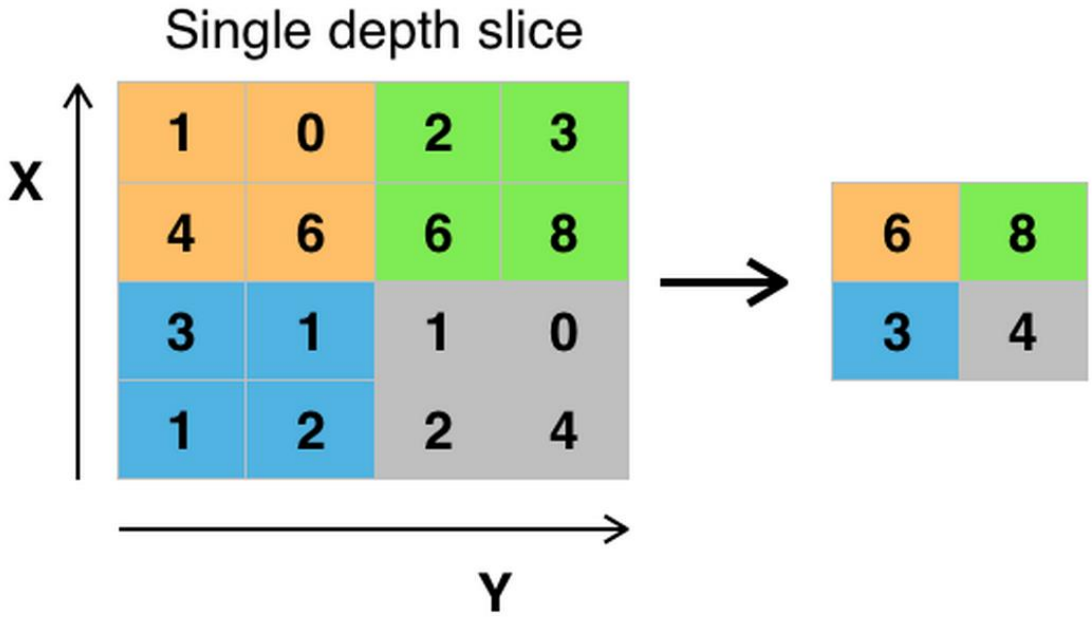
Hình 3.5. Ví dụ lớp ReLU

Mục đích của lớp ReLu là đưa ảnh một mức ngưỡng, ở đây là 0. Để loại bỏ các giá trị âm không cần thiết mà có thể sẽ ảnh hưởng cho việc tính toán ở các layer sau đó. Đầu ra của một layer ReLU có kích thước giống với đầu vào, chỉ là tất cả các giá trị âm được loại bỏ (hình 3.5).

3.1.4. Lớp pooling

Pooling Layer thực hiện chức năng làm giảm chiều không gian của đầu và giảm độ phức tạp tính toán của model ngoài ra Pool Layer còn giúp kiểm soát hiện tượng overffiting. Thông thường, Pool layer có nhiều hình thức khác nhau phù hợp cho nhiều bài toán, tuy nhiên Max Pooling là được sử dụng nhiều vào phổ biến hơn cả với ý tưởng cũng rất sát với thực tế con người đó là: Giữ lại chi tiết quan trọng hay hiểu ở trong bài toán này chính giữ lại pixel có giá trị lớn nhất.

Ví dụ: Max pooling với bộ lọc 2x2 và stride = 2. Bộ lọc sẽ chạy dọc ảnh. Và với mỗi vùng ảnh được chọn, sẽ chọn ra 1 giá trị lớn nhất và giữ lại (hình 3.6).

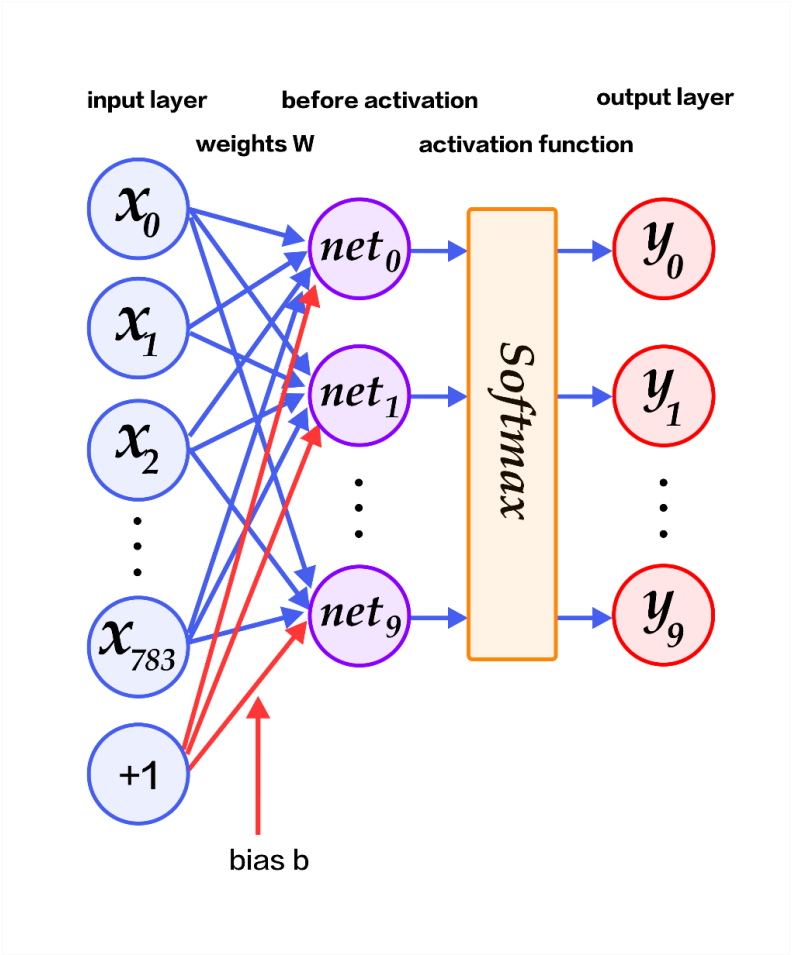


Hình 3.6. Ví dụ Max pooling kernel 2x2, stride = 2

Một layer pooling là hoạt động thực hiện pooling trên một hình ảnh hoặc một tập các hình ảnh. Đầu ra sẽ có cùng số lượng hình ảnh, nhưng mỗi cái sẽ có điểm ảnh ít hơn. Điều này cũng rất hữu ích trong việc quản lý tải trọng tính toán. Hạ một tấm ảnh 8 megapixel xuống còn 2 megapixel sẽ giúp mọi xử lý tải về trở nên dễ dàng.

3.1.5. Lớp Fully Connected

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



Hình 3.7. Mô hình lớp Fully Connected

Cũng như khi sử dụng mạng ANN truyền thống để xử lý những data có dạng matrix như image. Ta cần flatten data về dạng vector, sau đó đưa vào ANN như bình thường. Hay nói cách khác phần Fully-Connected Layer (FC Layer) chính là một mạng NN được gắn vào phần cuối của CNNs. Phần FC-Layer này chính là nơi từ các feature được extract bởi phần convolution và pooling tạo ra kết quả cuối cùng (Classification hoặc Regression).

3.2. Mạng MobiNets

3.2.1. Tổng quan về MobileNets

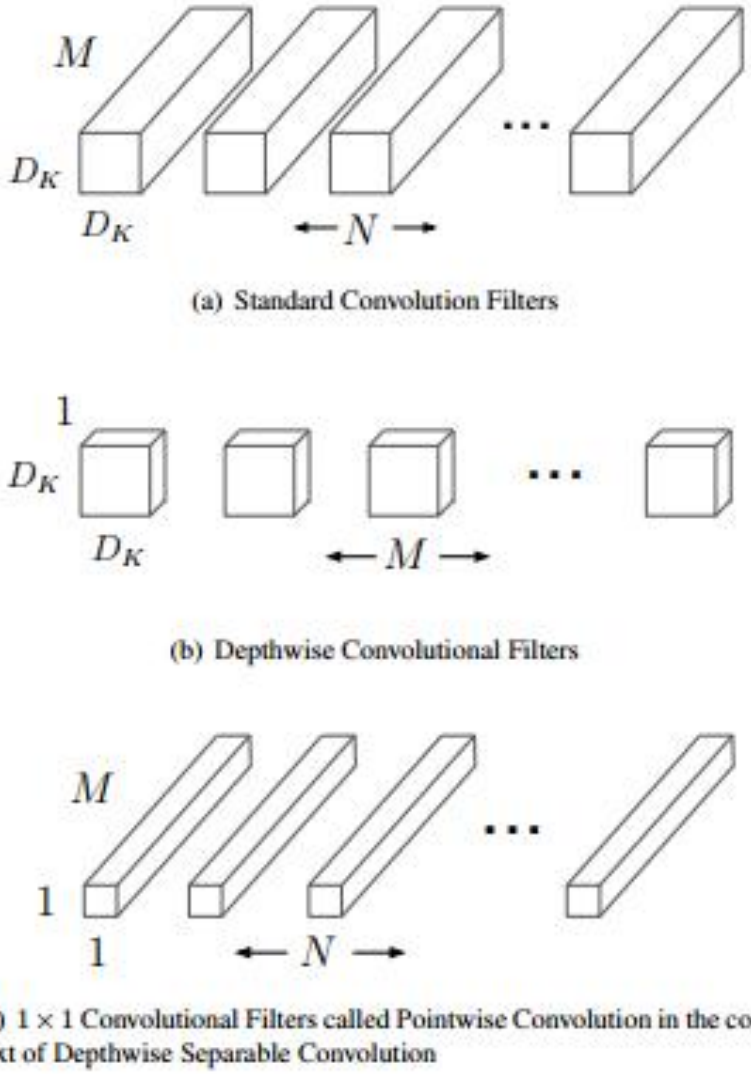
Trong quá trình phát triển mạng neural tích chập sâu (deep convolutional neural networks), Mobinets là một mô hình có hiệu quả được sử dụng trong các ứng dụng có tầm nhìn di động và nhúng bởi kích thước mạng nhỏ mà vẫn đảm bảo hiệu năng cao và giảm độ trễ. Mobinets dựa trên kiến trúc streamlined (thuật ngữ chỉ một dạng có tốc độ được gia tăng và dễ dàng di chuyển), sử dụng các tích chập có thể tách sâu (depthwise separable convolutions) để xây dựng các mạng neural sâu có trọng số nhỏ.

Các phần tiếp theo sẽ trình bày kiến trúc Mobilenets, 2 tham số tăng cường có thể cân bằng độ trễ và độ chính xác. Các tham số này cho phép các người xây dựng lựa chọn mô hình có kích thước phù hợp cho các ứng dụng cụ thể dựa trên các ràng buộc của bài toán.

3.2.2. Kiến trúc MobileNets

3.2.2.1 Depthwise separable convolutions

MobileNet được xây dựng dựa trên depthwise separable convolution, là một dạng của factorized convolutions. Factorized convolution là một covolution mà phân giải một convolution cơ bản thành một depthwise convolution và một 1x1 convolution (được gọi là pointwise convolution). Đối với MobileNets, depthwise convolution sử dụng một bộ lọc đơn cho mỗi kênh đầu vào. Sau đó pointwise convolution sử dụng 1x1 convolution để kết hợp các đầu ra của depthwise convolution. Một convolution cơ bản vừa lọc vừa kết hợp các đầu vào vào trong một tập các đầu ra mới chỉ trong một bước.. Depthwise separable convolution chia tập này thành 2 lớp, 1 lớp để lọc và 1 lớp để kết hợp. Quá trình phân giải này có ảnh hưởng mạnh trong việc giảm thiểu khối lượng tính toán và kích thước mô hình MobileNets. Hình 3.8 thể hiện cách mà một convolution cơ bản được phân giải thành một depthwise convolution và một 1x1 convolution



Hình 3.8. Convolution cơ bản (a) được phân giải thành depthwise convolution (b) và pointwise convolution (c)

Trên hình 3.8, M là số kênh đầu vào, N là số kênh đầu ra, D_K là số chiều không gian của hạt nhân convolution được giả định là khối vuông.

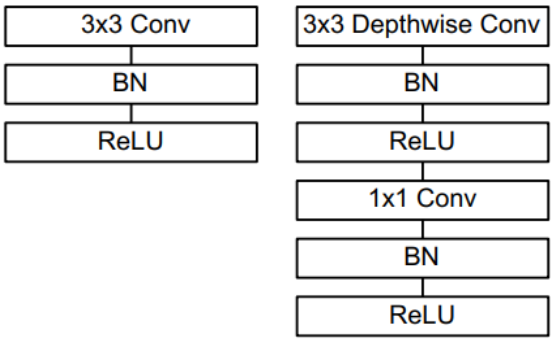
3.2.2.2 Cấu trúc mạng và huấn luyện mạng

Kiến trúc MobileNet được thể hiện trong bảng 3.1

Bảng 3.1. Kiến trúc một MobileNet

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Cột thứ nhất là dạng kiến trúc/bước nhảy của lớp, cột thứ hai là kích thước bộ lọc, cột thứ ba là kích thước đầu vào. Tất cả các lớp được theo sau bởi 1 batchnorm và tính phi tuyến ReLU với ngoại lệ là lớp kết nối đầy đủ, lớp đó không có tính phi tuyến và cấp dữ liệu vào lớp softmax để phân loại. Hình 3.9 so sánh một lớp có convolution thông thường, batchnorm và tính phi tuyến ReLU với lớp factorized có depthwise convolution, 1x1 convolution cũng như batchnorm và ReLU sau mỗi lớp chập. Quá trình down sampling được xử lí với strided convolution ở trong depthwise convolution như ở lớp đầu tiên. Quá trình pooling cuối cùng giảm độ phân giải không gian xuống 1 trước layer kết nối đầy đủ. MobileNet có 28 layers.



Hình 3.9. Convolution cơ bản với batchnorm và ReLu (trái) và depthwise separable convolution với các lớp depthwise và pointwise theo sau bởi batchnorm và ReLu (phải)

MobiNet dành 95% khối lượng tính toàn vào các 1x1 convolution, nơi có gần 75% tham số (bảng 3.2). Gần như toàn bộ các tham số khác nằm trong lớp kết nối đầy đủ

Bảng 3.2. Các tài nguyên mỗi lớp

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

MobiNet được luyện tập trong TensorFlow sử dụng RMSprop với gradient descent không đồng bộ. Tuy nhiên, trái với các mô hình luyện tập lớn, chúng ta sử dụng ít hơn các kĩ thuật chính quy và tăng cường dữ liệu vì các mô hình nhỏ gặp ít vấn đề hơn với overfitting. Khi luyện tập MobileNets chúng ta có thể giảm thiểu các hình ảnh bị méo bằng các giới hạn kích thước cắt ảnh được sử dụng. Ngoài ra, một điều rất quan trọng là đặt các trọng số nhỏ hoặc không có trọng số trên các bộ lọc depthwise vì ở đó có khá ít tham số

3.2.3. Hai tham số tăng cường

3.2.3.1 Width Multiplier: Mô hình nhỏ hơn

Mặc dù kiến trúc MobileNet đã nhỏ và ít trễ, song nhiều khi trong các trường hợp cụ thể có thể đòi hỏi mô hình nhỏ hơn và ít tính toán hơn, ở đây chúng ta dùng một tham số α được gọi là width multiplier, mục đính để giảm kích thước mạng đồng đều ở mỗi layer. Đối với một lớp, số lượng kênh đầu vào M sẽ trở thành αM, số lượng kênh đầu ra N sẽ trở thành αN, với $0<\alpha\leq 1$, thường α được chọn là 0.75, 0.5, 0.25. Tham số α có thể giảm thiểu khá nhiều khối lượng tính toán và số lượng tham số xuống xấp xỉ α^2 lần. Tham số α có thể được áp dụng với bất cứ cấu trúc mô hình nào để xác định mô hình nhỏ hơn với độ chính xác chấp nhận được, cân bằng giữa độ trễ và kích thước mạng

3.2.3.2 Resolution Multiplier: Giảm thiểu sự biểu diễn

Tham số thứ hai để giảm thiểu khối lượng tính toán của mạng neural là resolution multiplier ρ. Chúng ta áp dụng tham số này vào ảnh đầu vào và sự biểu diễn bên trong của mỗi lớp sẽ được giảm thiểu với hệ số ρ tương ứng. Trên thực tê chúng ta ngầm định thiết lập ρ bằng các thay đổi độ phân giải đầu vào. Với $0<\rho\leq 1$ thì khối lượng tính toán được giảm xuống ρ^2 lần

3.2.3.3 Hiệu quả khi sử dụng 2 tham số α và ρ

Bảng 3.3 thể hiện sự hiệu quả khi sử dụng 2 tham số α và ρ

Bảng 3.3. Sử dụng tài nguyên cho convolution cơ bản có chỉnh sửa

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

4 hàng thể hiện số phép nhân-cộng và tham số trong một lớp convolution đầy đủ, depthwise separable convolution, và khi áp dụng 2 tham số α, ρ.

3.2.4. Ứng dụng của MobileNet trong nhận diện các thuộc tính khuôn mặt

MobileNet có thể được sử dụng để nén một hệ thống lớn với các thủ tục luyện tập ẩn. Trong tác vụ phân loại thuộc tính khuôn mặt, chúng ta sẽ chứng minh mối quan hệ giữa MobileNet và distillation, một kĩ thuật chuyển đổi đối với mạng sâu. Sử dụng MobileNet sẽ giảm số lượng phân lớp thuộc tính khuôn mặt với 75 triệu tham số và 1.6 tỉ phép nhận-cộng. Kết hợp với khả năng mở rộng của luyện tập distillatinon và các tham số kĩ thuật của MobileNet, hệ thống sau cùng thể hiện được hiệu năng vượt trội (bảng 3.4) Có thể thấy từ bảng 4 tính ưu việt của MobileNet: nó đạt được độ chính xác trung bình (mean average precision – Mean AP) chấp nhận được trong khi chỉ tiêu thụ 1% số lượng phép nhân-cộng (khi sử dụng tham số α=0.5 và giảm độ phân giải ảnh đầu vào từ 224 xuống còn 128).

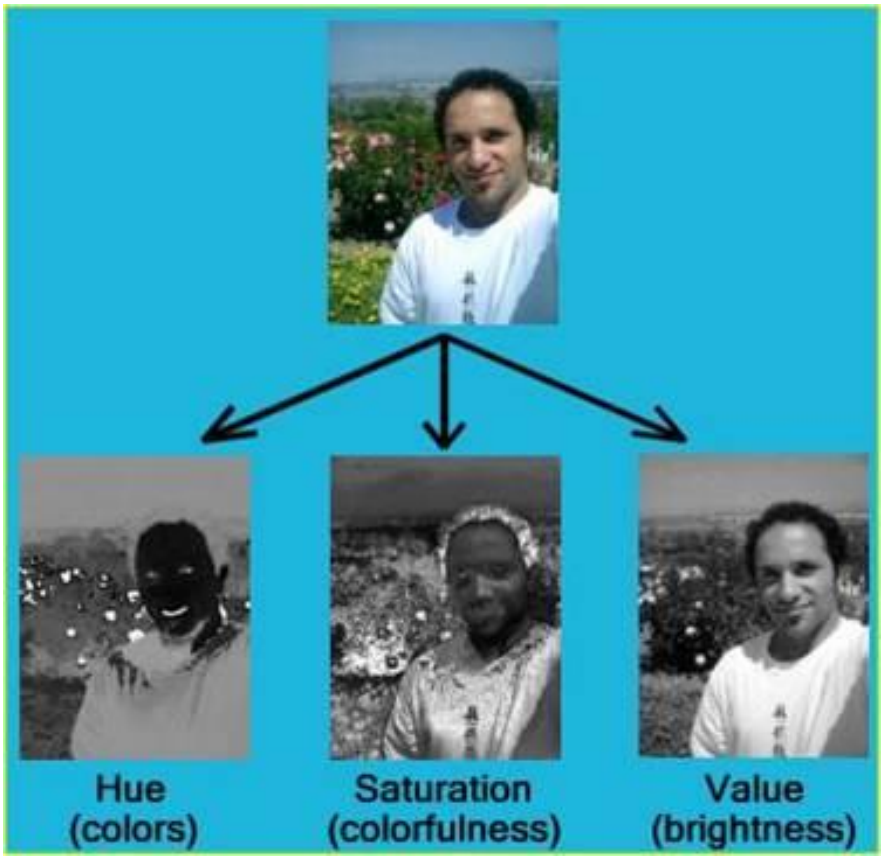
Bảng 3.4. Nhận diện các thuộc tính khuôn mặt sử dụng kiến trúc Mobinet

Width Multiplier / Resolution	Mean AP	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	88.7%	568	3.2
0.5 MobileNet-224	88.1%	149	0.8
0.25 MobileNet-224	87.2%	45	0.2
1.0 MobileNet-128	88.1%	185	3.2
0.5 MobileNet-128	87.7%	48	0.8
0.25 MobileNet-128	86.4%	15	0.2
Baseline	86.9%	1600	7.5

3.3. Thư viện mã nguồn mở OpenCV

OpenCV (hình 3.10) là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính (computer vision), xử lý ảnh và máy học, và các tính năng tăng tốc GPU trong hoạt động thời gian thực.

OpenCV được phát hành theo giấy phép BSD, do đó nó hoàn toàn miễn phí cho cả học thuật và thương mại. Nó có các interface C++, C, Python, Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để tính toán hiệu quả và với sự tập trung nhiều vào các ứng dụng thời gian thực. Được viết bằng tối ưu hóa C/C++, thư viện có thể tận dụng lợi thế của xử lý đa lõi. Được sử dụng trên khắp thế giới, OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần. Phạm vi sử dụng từ nghệ thuật tương tác, cho đến lĩnh vực khai thác mỏ, bản đồ trên web hoặc công nghệ robot.



Hình 3.10. Thư viện OpenCV

Trong thực tế OpenCV đang được sử dụng rộng rãi trong **các ứng dụng** bao gồm:

- Hình ảnh street view
- Kiểm tra và giám sát tự động
- Robot và xe hơi tự lái
- Phân tích hình ảnh y tế
- Tìm kiếm và phục hồi hình ảnh/video
- Phim - cấu trúc 3D từ chuyển động
- Nghệ thuật sắp đặt tương tác

Chức năng OpenCV

- Image/video I/O, xử lý, hiển thị (core, imgproc, highgui)
- Phát hiện các vật thể (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

3.4. Thư viện Tensorflow

Thư viện Tensorfow là thư viện mã nguồn mở dùng cho tính toán số học sử dụng đồ thị luồng dữ liệu.

Biểu đồ dưới đây cho thấy mức độ phổ biến của thư viện này.

Hình 3.11. Mức độ phổ biến của Tensorflow kể từ lúc opensource

Điểm nổi bật:

- Tích hợp sẵn rất nhiều các thư viện machine learning
- Có khả năng tương thích và mở rộng tốt. Được Google phát triển cho machine learning phục vụ cả nghiên cứu lẫn xây dựng các ứng dụng thực tế
- Phổ biến

Một số project nổi tiếng sử dụng thư viện Tensorflow

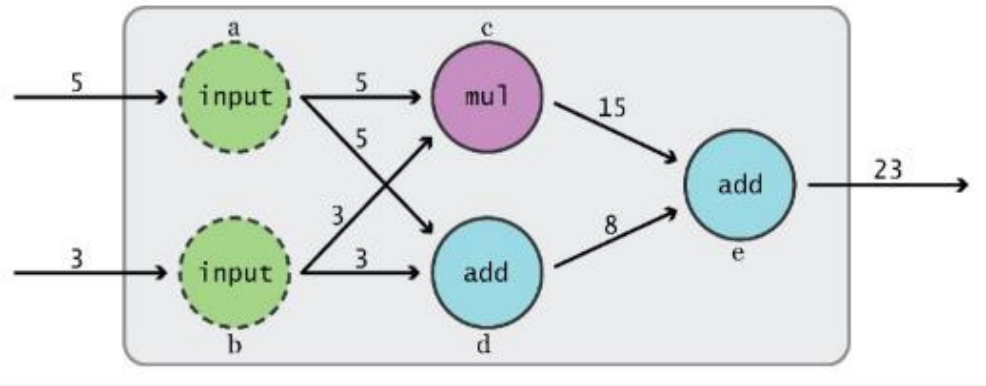
- Phân loại ung thư da – Dermatologist-level classification of skin cancer with deep neural networks (Esteva et al., Nature 2017)
- WaveNet: Text to speech – Wavenet: A generative model for raw audio (Oord et al., 2016)

- Vẽ hình – Draw Together with a Neural Network (Ha et al., 2017)
- Image Style Transfer Using Convolutional Neural Networks (Gatys et al., 2016) Tensorflow adaptation by Cameroon Smith (cysmith@github)

Data flow graphs:

Tensorflow phân biệt rạch ròi việc định nghĩa và tính toán trong quá trình thực thi (hình 3.12), bao gồm:

- Xây dựng, định nghĩa đồ thị(Graph)
- Sử dụng một Session để thực thi các tính toán trong đồ thị



Hình 3.12. Mô phỏng một đồ thị xây dựng bởi TF

Khái niệm Tensor

Tensor là một mảng có n chiều (n-dimensional array). Chẳng hạn như:

0-d tensor, còn được gọi là scalar hay chỉ là một số. Ví dụ: 1, 2, -5

1-d tensor, còn được gọi là vector. Ví dụ: [1 2 3 4], [5 8 7 9]

2-d tensor, còn được gọi là ma trận(matrix). Ví dụ: [1 2 3; 4 5 6; 7 8 9]

...

n-D tensor

3.5. Kết luận

Trên đây là tóm tắt cơ sở lý thuyết cơ bản, các đề tài thực tiễn sẽ dựa trên chúng để ghép nối, xây dựng nên các hệ thống có nghĩa, hiệu quả.

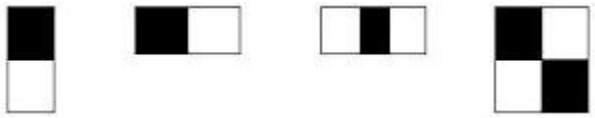
Chương 4. Thiết kế hệ thống nhận diện cảm xúc mặt người qua webcam

Chương 4 trình bày việc thiết kế chi tiết cho hệ thống nhận diện cảm xúc mặt người qua webcam.

4.1. Nhận dạng khuôn mặt

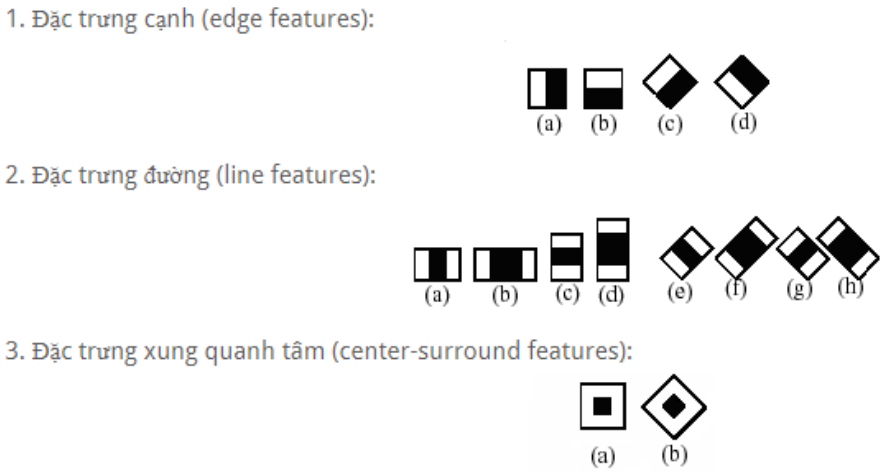
4.1.1. Đặc trưng Haar-like

Đặc trưng Haar-like (hình 4.1) do Viola và Jones công bố, gồm 4 đặc trưng cơ bản để xác định khuôn mặt người. Mỗi đặc trưng Haar–like là sự kết hợp của hai hay ba hình chữ nhật "trắng" hay "đen" như trong hình sau:



Hình 4.1. Các đặc trưng haar-like cơ bản

Để sử dụng các đặt trưng này vào việc xác định khuôn mặt người, 4 đặt trưng Haar-like cơ bản được mở rộng ra, và được chia làm 3 tập đặc trưng như sau (hình 4.2):



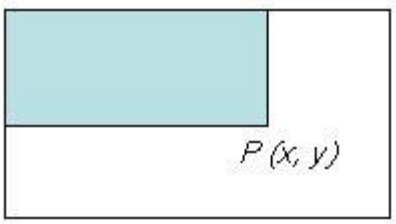
Hình 4.2. Các đặc trưng haar-like mở rộng

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

f(x) = Tổngvùng đen(các mức xám của pixel) - Tổngvùng trắng(các mức xám của pixel)

Sử dụng giá trị này, so sánh với các giá trị của các giá trị pixel thô, các đặc trưng Haar-like có thể tăng/giảm sự thay đổi in-class/out-of-class (bên trong hay bên ngoài lớp khuôn mặt người), do đó sẽ làm cho bộ phân loại dễ hơn.

Như vậy ta có thể thấy rằng, để tính các giá trị của đặc trưng Haar-like, ta phải tính tổng của các vùng pixel trên ảnh. Nhưng để tính toán các giá trị của các đặc trưng Haar-like cho tất cả các vị trí trên ảnh đòi hỏi chi phí tính toán khá lớn, không đáp ứng được cho các ứng dụng đòi hỏi tính run-time. Do đó Viola và Jones đưa ra một khái niệm gọi là Integral Image, là một mảng 2 chiều với kích thước bằng với kích của ảnh cần tính các đặc trưng Haar-like, với mỗi phần tử của mảng này được tính bằng cách tính tổng của điểm ảnh phía trên (dòng-1) và bên trái (cột-1) của nó. Bắt đầu từ vị trí trên, bên trái đến vị trí dưới, phải của ảnh, việc tính toán này đơn thuần chỉ đựa trên phép cộng số nguyên đơn giản, do đó tốc độ thực hiện rất nhanh (hình 4.3).



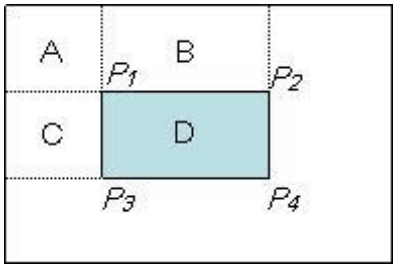
Hình 4.3. Cách tính Integral Image của ảnh

Sau khi đã tính được Integral Image, việc tính tổng các giá trị mức xám của một vùng bất kỳ nào đó trên ảnh thực hiện theo cách sau:

Giả sử ta cần tính tổng các giá trị mức xám của vùng D như trong hình 4, ta có thể tính như sau:

D = A + B + C + D – (A+B) – (A+C) + A

Với A + B + C + D chính là giá trị tại điểm P4 trên Integral Image, tương tự như vậy A+B là giá trị tại điểm P2, A+C là giá trị tại điểm P3, và A là giá trị tại điểm P1. Vậy ta có thể viết lại biểu thức tính D ở trên như sau (hình 4.4):



Hình 4.4. Ví dụ cách tính nhanh các giá trị mức xám của vùng D trên ảnh

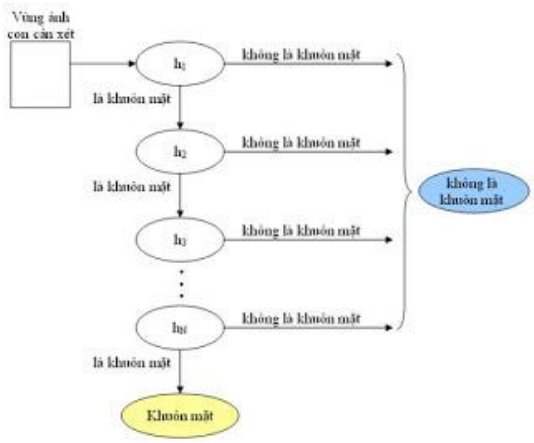
Tiếp theo, để chọn các đặc trưng Haar-like dùng cho việc thiết lập ngưỡng, Viola và Jones sử dụng một phương pháp máy học được gọi là AdaBoost. AdaBoost sẽ kết hợp các bộ phân loại yếu để tạo thành một bộ phân loại mạnh. Với bộ phân loại yếu chỉ cho ra câu trả lời chính xác chỉ hơn viện đoán một cách ngẫn nhiên một chút, còn bộ phân loại mạnh có thể đưa ra câu trả lời chính xác trên 60%.

4.1.2. AdaBoost

AdaBoost là một bộ phân loại 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.

Là một cải tiến của tiếp cận boosting, AdaBoost sử dụng thêm khái niệm trọng số (weight) để đánh dấu các mẫu khó nhận dạng. Trong quá trình huấn luyện, cứ mỗi weak classifiers 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.

Viola và Jones dùng AdaBoost kết hợp các bộ phân loại yếu sử dụng các đặc trưng Haar-like theo mô hình phân tầng (cascade) như sau (hình 4.5):



Hình 4.5. Mô hình phân tần kết hợp các bộ phân loại yếu để xác định khôn mặt

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

$$h_k = \begin{cases} 1 & \text{nếu } p_k f_k(x) < p_k \theta_k \\ 0 & \text{ngược lại} \end{cases}$$

x: cửa sổ con cần xét

Ok: ngưỡng (O = teta)

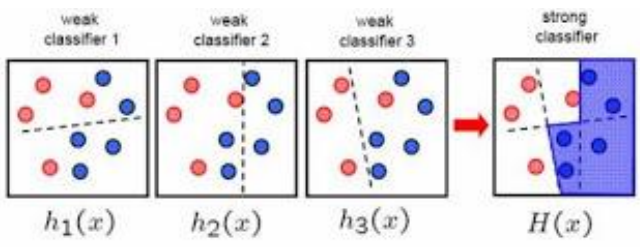
fk: giá trị của đặc trưng Haar-like

pk: hệ số quyết định chiều của phương trình

AdaBoost sẽ kết hợp các bộ phân loại yếu thành bộ phân loại mạnh như sau:

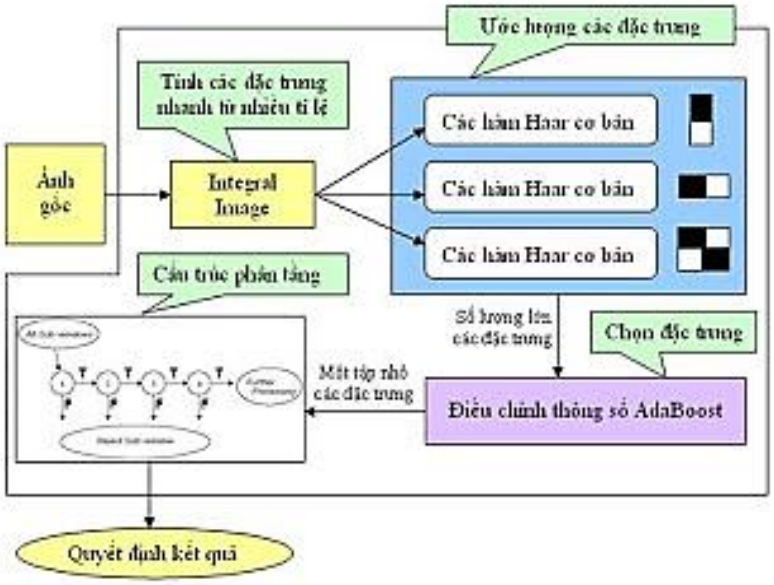
H(x) = sign(a1h1(x) +a2h2(x) + ... + anhn(x)) (a = alpha)

Với: at >= 0 là hệ số chuẩn hoá cho các bộ phân loại yếu



Hình 4.6. Kết hợp các bộ phân loại yếu thành bộ phân loại mạnh

4.1.3. Hệ thống xác định vị trí khuôn mặt người



Hình 4.7. Hệ thống xác định vị trí khuôn mặt người (Face detection system)

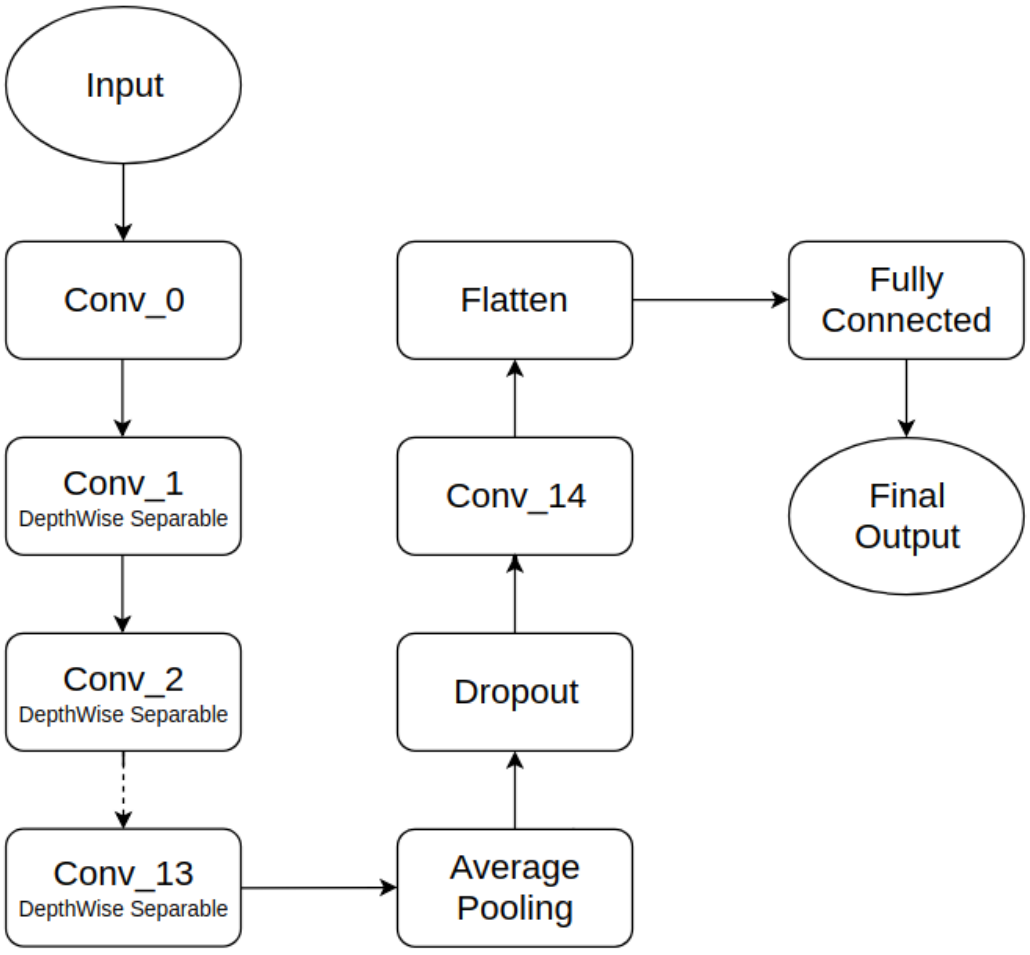
Như trên hình 4.7, từ ảnh gốc, ta tính Integral Image là mảng 2 chiều với phần tử (x, y) sẽ được tính bằng tổng của các phần tử (x', y') với x' < x và y' < y, mục đích để tính nhanh tổng của các giá trị mức xám của một vùng hình chữ nhật bất kỳ trên ảnh gốc. Các vùng ảnh con này được đưa qua các hàm Haar cơ bản để ước lượng đặc trưng, kết quả 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 được chuyển sang cho bộ quyết định kết quả (là tập các bộ phân loại yếu có cấu trúc như trong hình). Bộ quyết định sẽ tổng hợp kết quả là khuôn mặt người nếu kết quả của các bộ phân loại yếu trả về là khuôn mặt người.

Mỗi bộ phân loại yếu sẽ quyết định kết quả cho một đặc trưng Haar-like, được xác định ngưỡng đủ nhỏ để 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, 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 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.

4.2. Phân loại cảm xúc

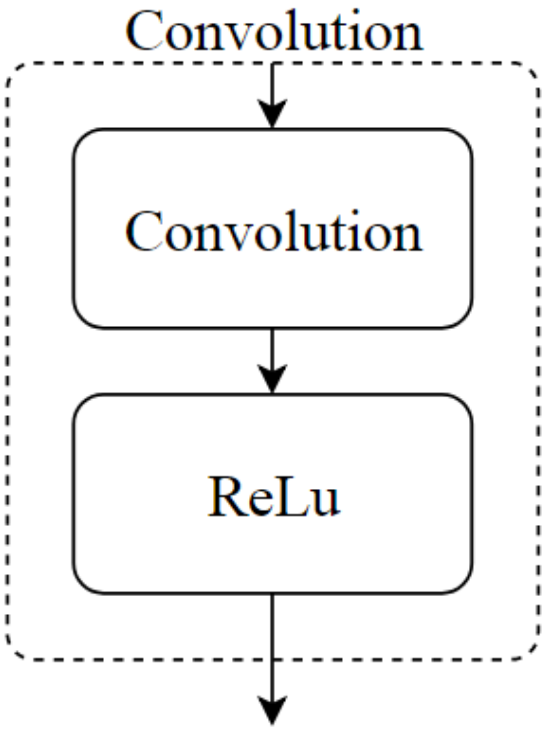
4.2.1. Kiến trúc mạng MobileNet

Trong đề tài này, chúng em xây dựng mạng MobileNet với kiến trúc bắt đầu là một layer Convolution bình thường, sau đó là 13 Layer DepthWise Separable Convolution rồi đến Layer Pooling sử dụng Average Pooling, thực hiện Dropout với Layer Dropout, tiếp theo là sử dụng tiếp một Layer Convolution. Cuối cùng sẽ là mạng Neural Network với Layer Fully Connected với 2 Layer là Input Layer và Output Layer.

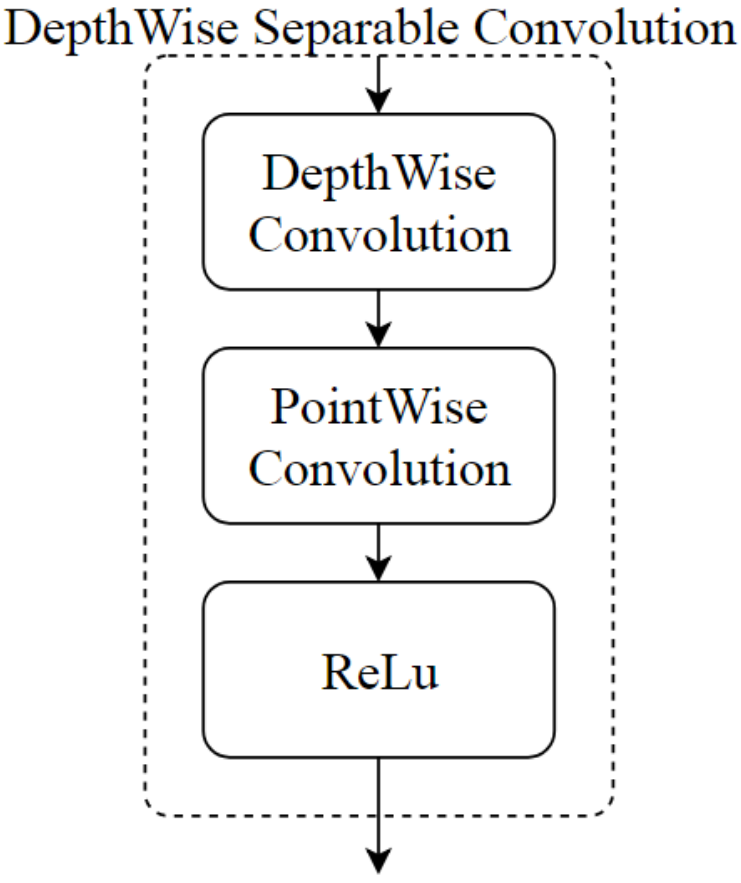


Hình 4.8. Kiến trúc mạng MobileNet sử dụng trong đề tài

Như đã nói ở trên, khác với Convolution, DepthWise Separable Convolution chứa hai Layer khác là DepthWise Convolution và PointWise Convolution. Ở đây, để thu gọn sơ đồ kiến trúc mạng MobileNet, trong mỗi layer Convolution và DepthWise Convolution đều đã có kèm them hàm ReLu.



Hình 4.9. Lớp Convolution



Hình 4.10. Lớp DepthWise Separable Convolution

Thông số các lớp trong kiến trúc mạng được trình bày trong bảng 4.1.

Bảng 4.1. Thông số kiến trúc của mạng MobileNet sử dụng trong đề tài

No	Layer	Input Size	Filter/Weight	Strides	Output Size
1	Conv 0	N x 224 x 224 x 3	3 x 3 x 3 x 32	2	N x 112 x 112 x 32
2	DepthWise Conv 1	N x 112 x 112 x 32	3 x 3 x 32 x 1	1	N x 112 x 112 x 32
	PointWise Conv 1	N x 112 x 112 x 32	1 x 1 x 32 x 64	1	N x 112 x 112 x 64
3	DepthWise Conv 2	N x 112 x 112 x 64	3 x 3 x 64 x 1	2	N x 56 x 56 x 64
	PointWise Conv 2	N x 56 x 56 x 64	1 x 1 x 64 x 128	1	N x 56 x 56 x 128
4	DepthWise Conv 3	N x 56 x 56 x 128	3 x 3 x 128 x 1	1	N x 56 x 56 x 128
	PointWise Conv 3	N x 56 x 56 x 128	1 x 1 x 128 x 128	1	N x 56 x 56 x 128
5	DepthWise Conv 4	N x 56 x 56 x 128	3 x 3 x 128 x 1	2	N x 28 x 28 x 128
	PointWise Conv 4	N x 28 x 28 x 128	1 x 1 x 128 x 256	1	N x 28 x 28 x 256
6	DepthWise Conv 5	N x 28 x 28 x 256	3 x 3 x 256 x 1	1	N x 28 x 28 x 256
	PointWise Conv 5	N x 28 x 28 x 256	1 x 1 x 256 x 256	1	N x 28 x 28 x 256
7	DepthWise Conv 6	N x 28 x 28 x 256	3 x 3 x 256 x 1	2	N x 14 x 14 x 256
	PointWise Conv 6	N x 14 x 14 x 256	1 x 1 x 256 x 512	1	N x 14 x 14 x 512
8	DepthWise Conv 7	N x 14 x 14 x 512	3 x 3 x 512 x 1	1	N x 14 x 14 x 512
	PointWise Conv 7	N x 14 x 14 x 512	1 x 1 x 512 x 512	1	N x 14 x 14 x 512
9	DepthWise Conv 8	N x 14 x 14 x 512	3 x 3 x 512 x 1	1	N x 14 x 14 x 512
	PointWise Conv 8	N x 14 x 14 x 512	1 x 1 x 512 x 512	1	N x 14 x 14 x 512
10	DepthWise Conv 9	N x 14 x 14 x 512	3 x 3 x 512 x 1	1	N x 14 x 14 x 512
	PointWise Conv 9	N x 14 x 14 x 512	1 x 1 x 512 x 512	1	N x 14 x 14 x 512
11	DepthWise Conv 10	N x 14 x 14 x 512	3 x 3 x 512 x 1	1	N x 14 x 14 x 512
	PointWise Conv 10	N x 14 x 14 x 512	1 x 1 x 512 x 512	1	N x 14 x 14 x 512
12	DepthWise Conv 11	N x 14 x 14 x 512	3 x 3 x 512 x 1	1	N x 14 x 14 x 512
	Point Wise Conv 11	N x 14 x 14 x 512	1 x 1 x 512 x 512	1	N x 14 x 14 x 512
13	DepthWise Conv 12	N x 14 x 14 x 512	3 x 3 x 512 x 1	2	N x 7 x 7 x 512
	PointWise Conv 12	N x 7 x 7 x 512	1 x 1 x 512 x 1024	1	N x 7 x 7 x 1024
14	DepthWise Conv 13	N x 7 x 7 x 1024	3 x 3 x 1024 x 1	1	N x 7 x 7 x 1024
	PointWise Conv 13	N x 7 x 7 x 1024	3 x 3 x 1024 x 2024	1	N x 7 x 7 x 1024
15	Average Pooling	N x 7 x 7 x 1024	7 x 7 x 1024 x 1	1	N x 1 x 1 x 1024

16	Dropout	N x 1 x 1 x 1024	NA	NA	N x 1 x 1 x 1024
17	Conv 14	N x 1 x 1 x 1024	1 x 1 x 1024 x 1001	1	N x 1 x 1 x 1001
18	Flatten	N x 1 x 1 x 1001	NA	NA	N x 1001
19	Fully Connected	N x 1001	1001 x 4	NA	N x 4

Trong đó, cột thứ nhất là dạng kiến trúc của lớp, cột thứ hai là kích thước đầu vàoc, cột thứ ba là kích thước bộ lọc, cột thứ tư là bước nhảy và cột cuối cùng là kích thước đầu ra.

4.3. Kết luận

Kết thúc chương 4, nhóm chúng em đã hoàn thành việc thiết kế chi tiết các module trong đề tài và sẵn sàng cho việc mô phỏng hoạt động của hệ thống.

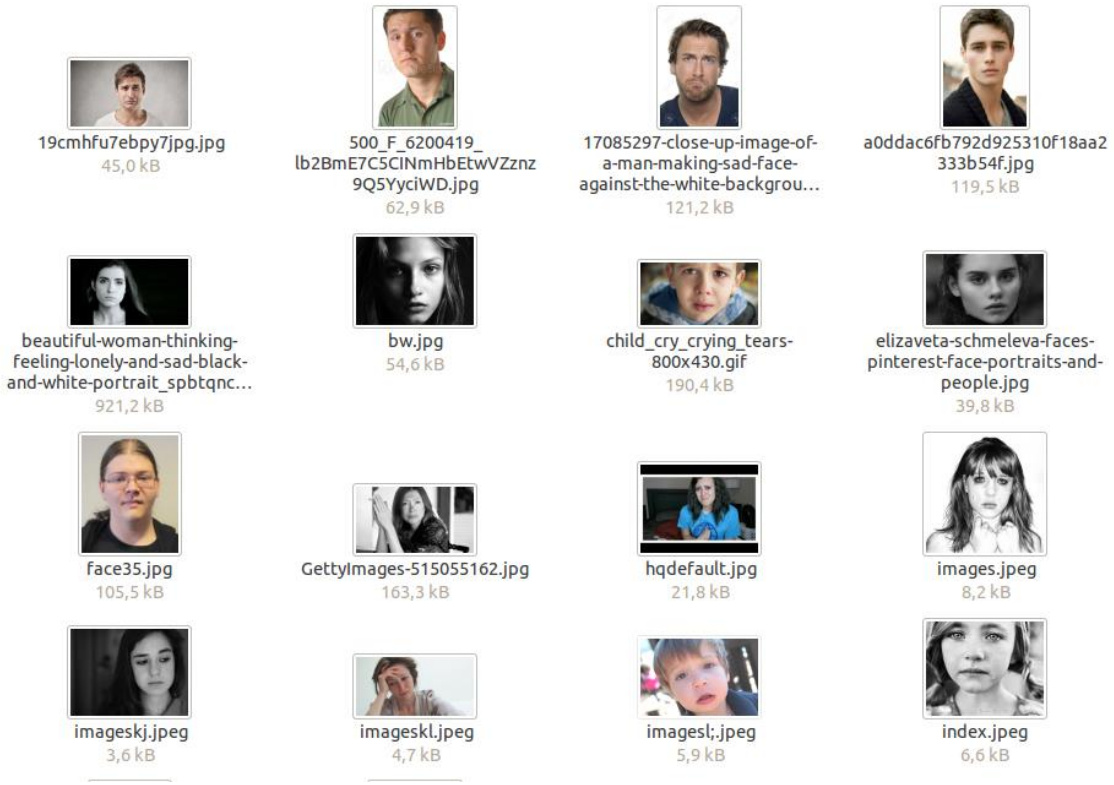
Chương 5. Mô phỏng và kiểm thử

Chương 5 là phần kiểm tra hoạt động của hệ thống đã chính xác hay chưa. Hệ thống sẽ được chạy trên hệ điều hành Ubuntu phiên bản 14.04.

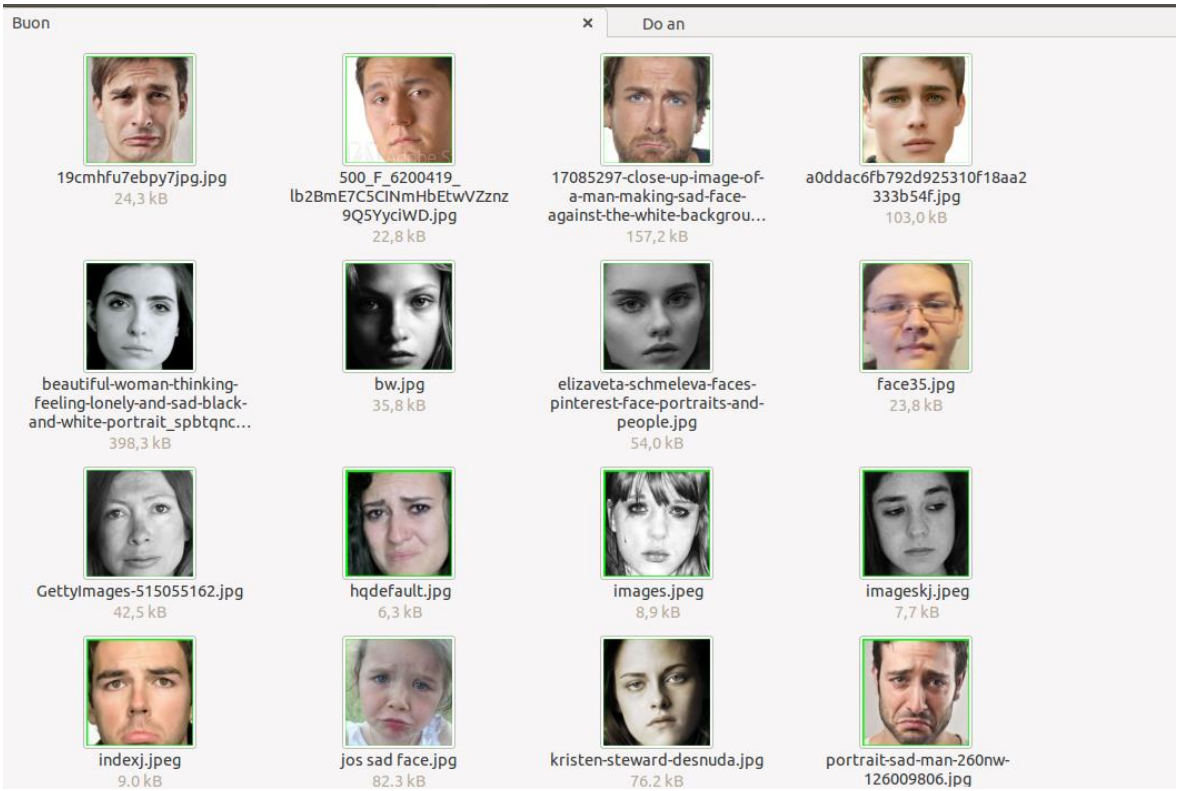
5.1. Nhận diện khuôn mặt

Dữ liệu phục vụ cho hệ thống là các hình ảnh được tải từ nhiều nguồn trên Internet. Trong đề tài này, hệ thống bao gồm 4 cảm xúc chính: Buồn, Vui vẻ, Cười, Ngáp ngủ. Sau khi sử dụng thuật toán Haar-like cascade, khuôn mặt sẽ được xác định và cắt thành hình mới, phục vụ cho công đoạn training dữ liệu.

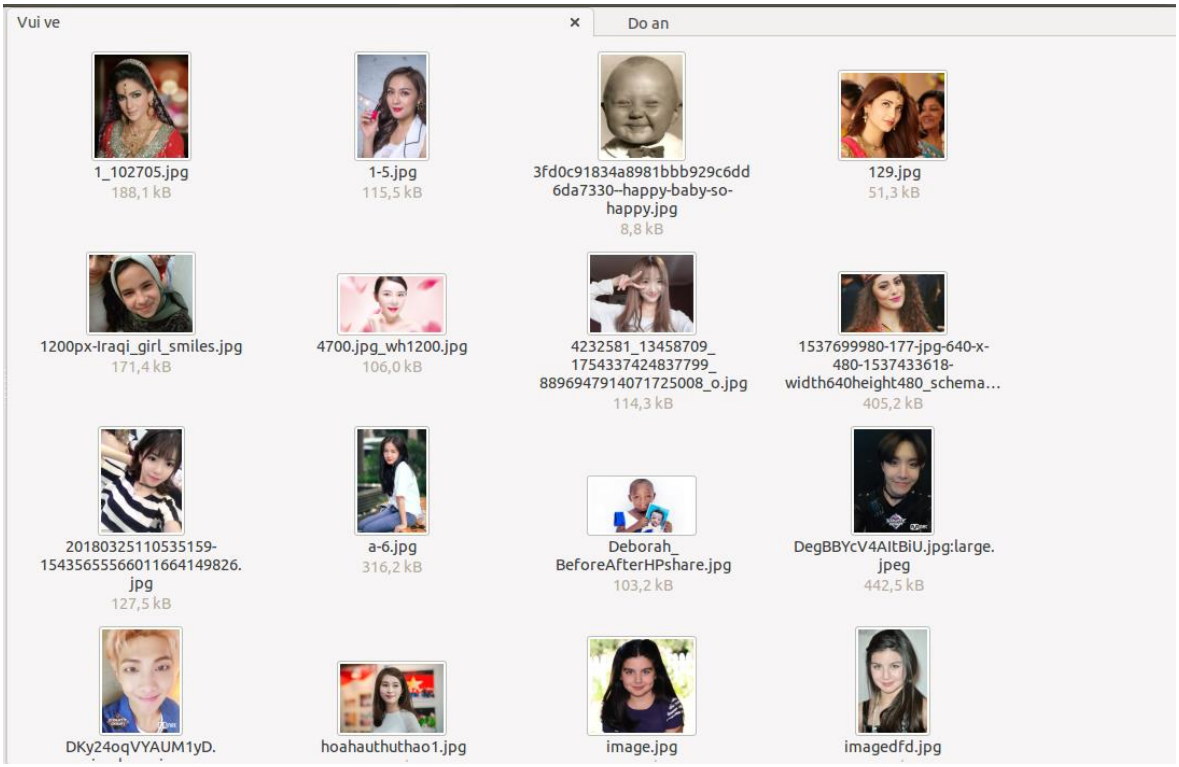
Dưới đây là một số hình ảnh trước và sau khi thực hiện thuật toán Haar-like cascade, sử dụng thư viện mã nguồn mở OpenCV.



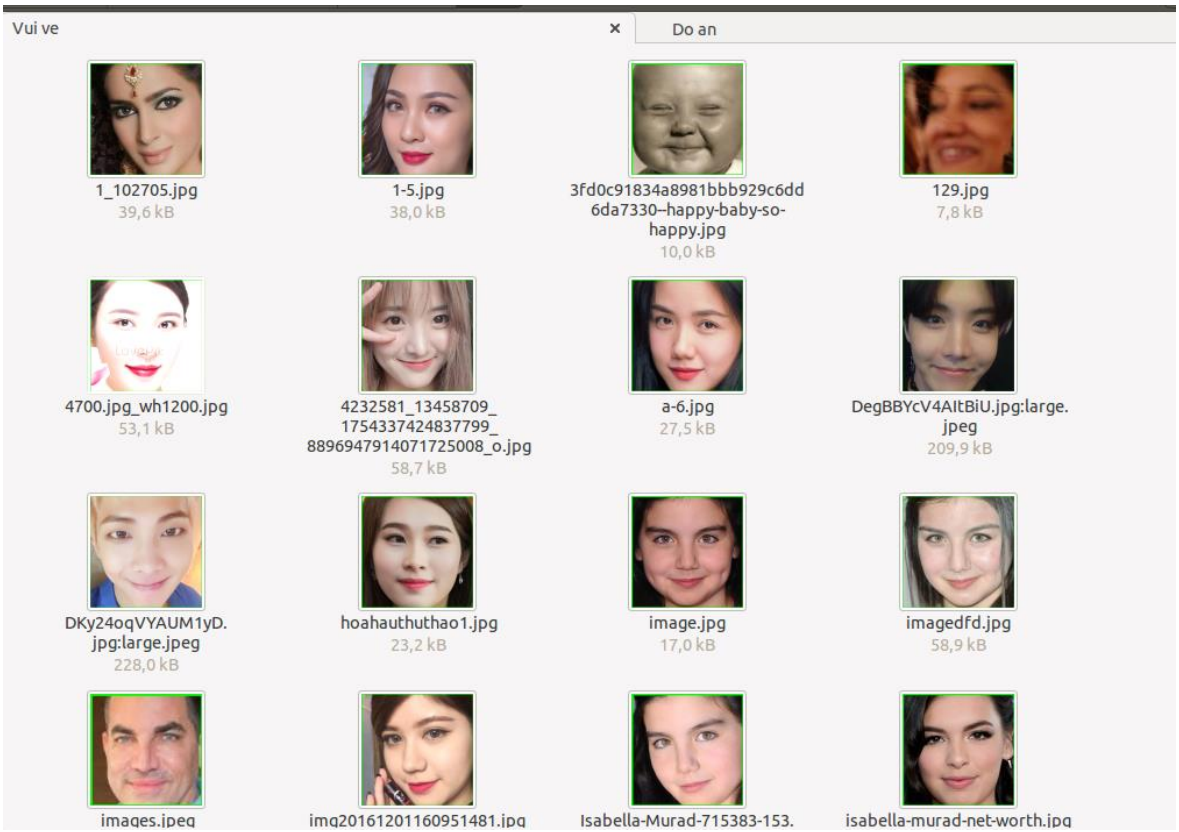
Hình 5.1. Một số hình ảnh biểu cảm "Buồn" trước HAAR-cascade



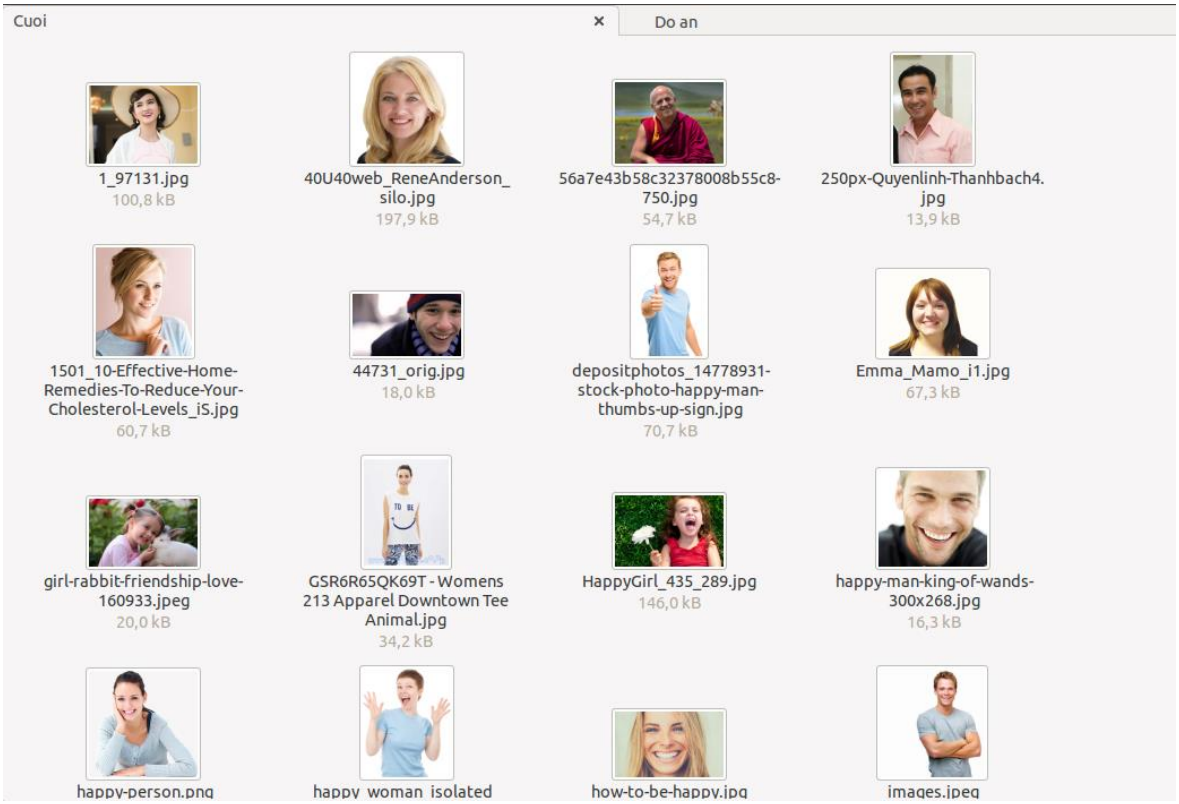
Hình 5.2. Một số hình ảnh biểu cảm "Buồn" sau HAAR cascade



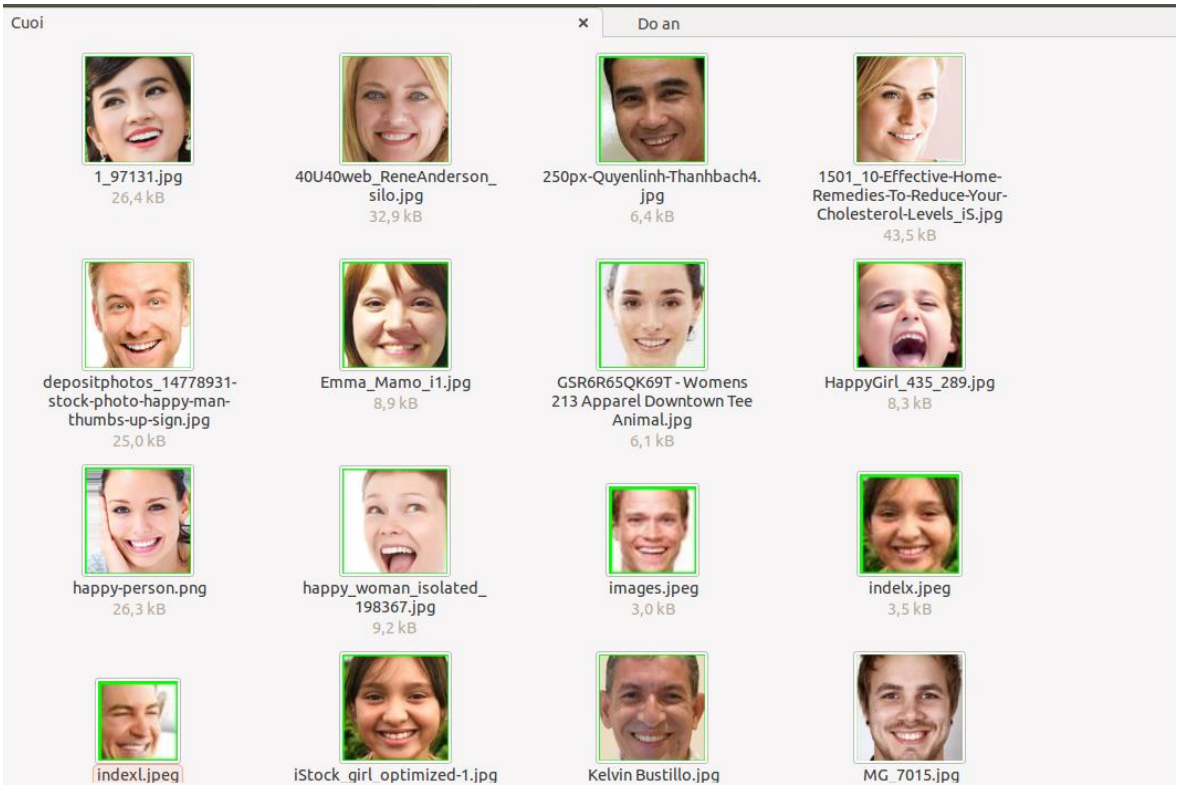
Hình 5.3. Một số hình ảnh biểu cảm "Vui vẻ" trước HAAR-cascade



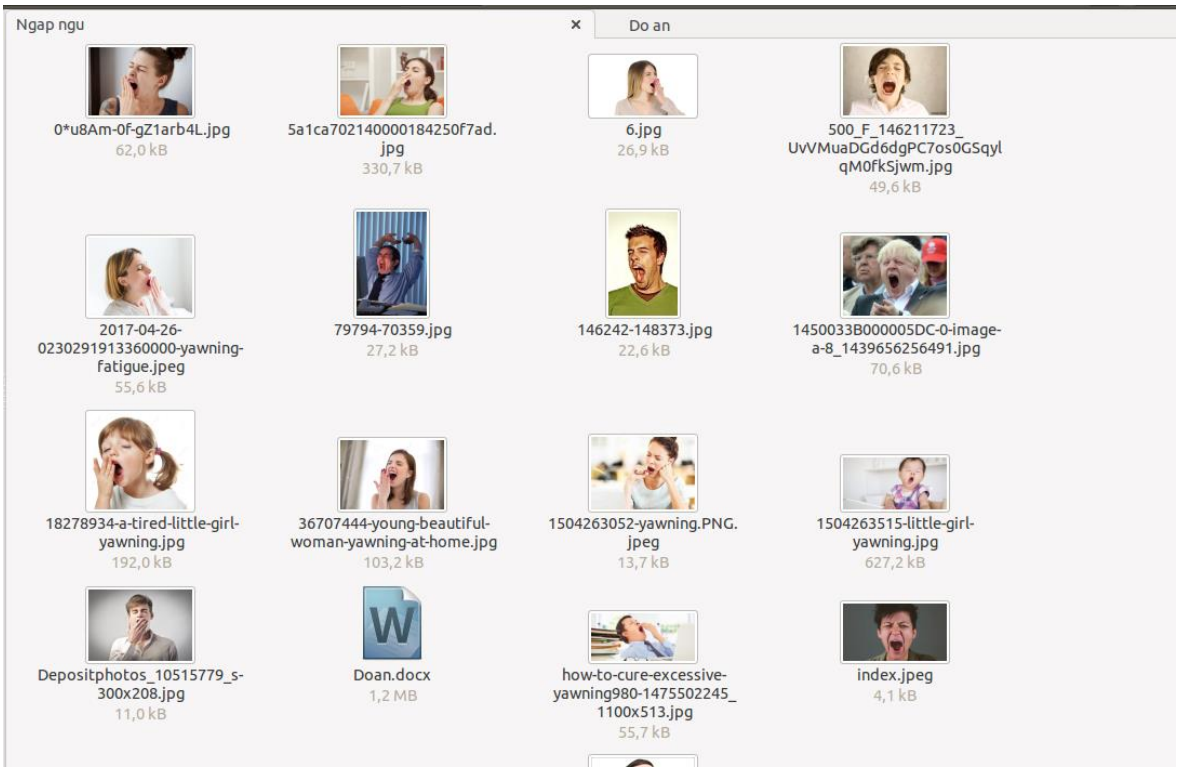
Hình 5.4. Một số hình ảnh biểu cảm "Vui vẻ" sau HAAR-cascade



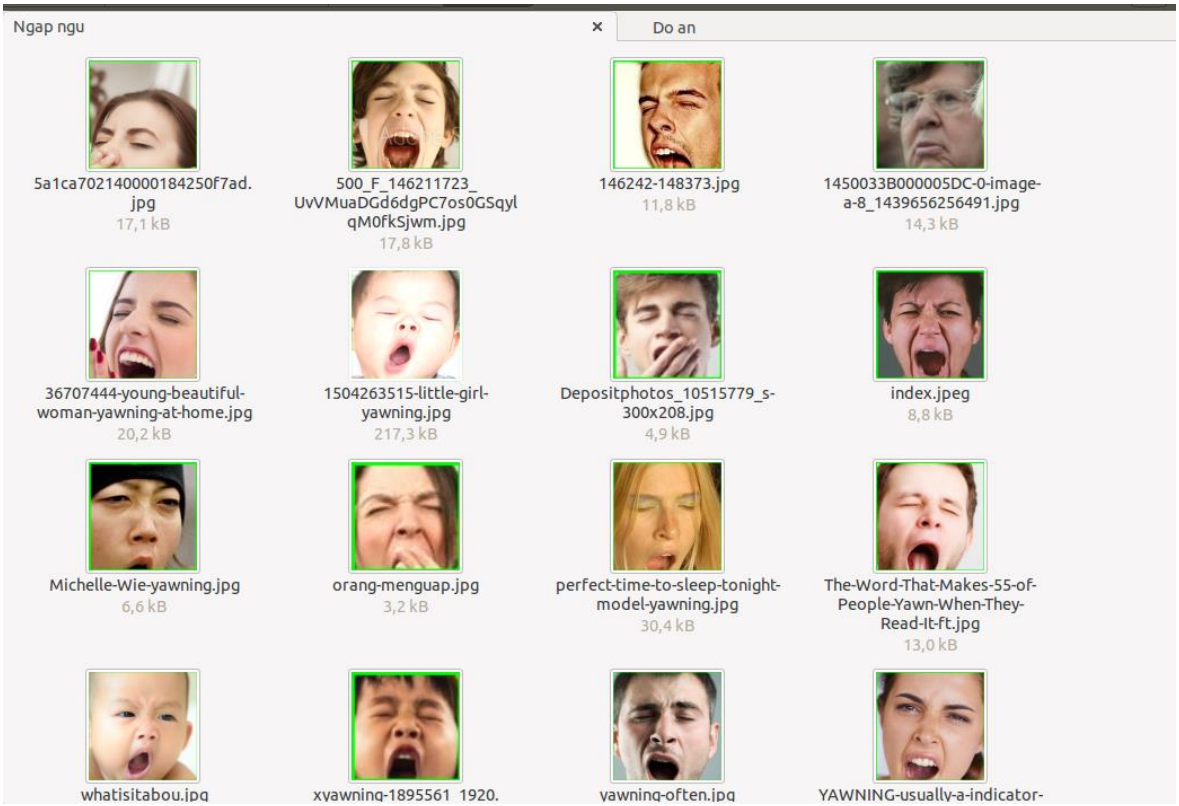
Hình 5.5. Một số hình ảnh biểu cảm "Cười" trước HAAR-cascade



Hình 5.6. Một số hình ảnh biểu cảm "Cười" sau HAAR-cascade



Hình 5.7. Một số hình ảnh biểu cảm "Ngáp ngủ" trước HAAR-cascade



Hình 5.8. Một số hình ảnh biểu cảm "Ngáp ngủ" sau HAAR-cascade

Sau khi hoàn thiện việc nhận diện khuôn mặt với thuật toán HAAR-like của OpenCV, những dữ liệu sẽ được sử dụng để đào tạo việc phân loại cảm xúc được trình bày trong mục dưới đây.

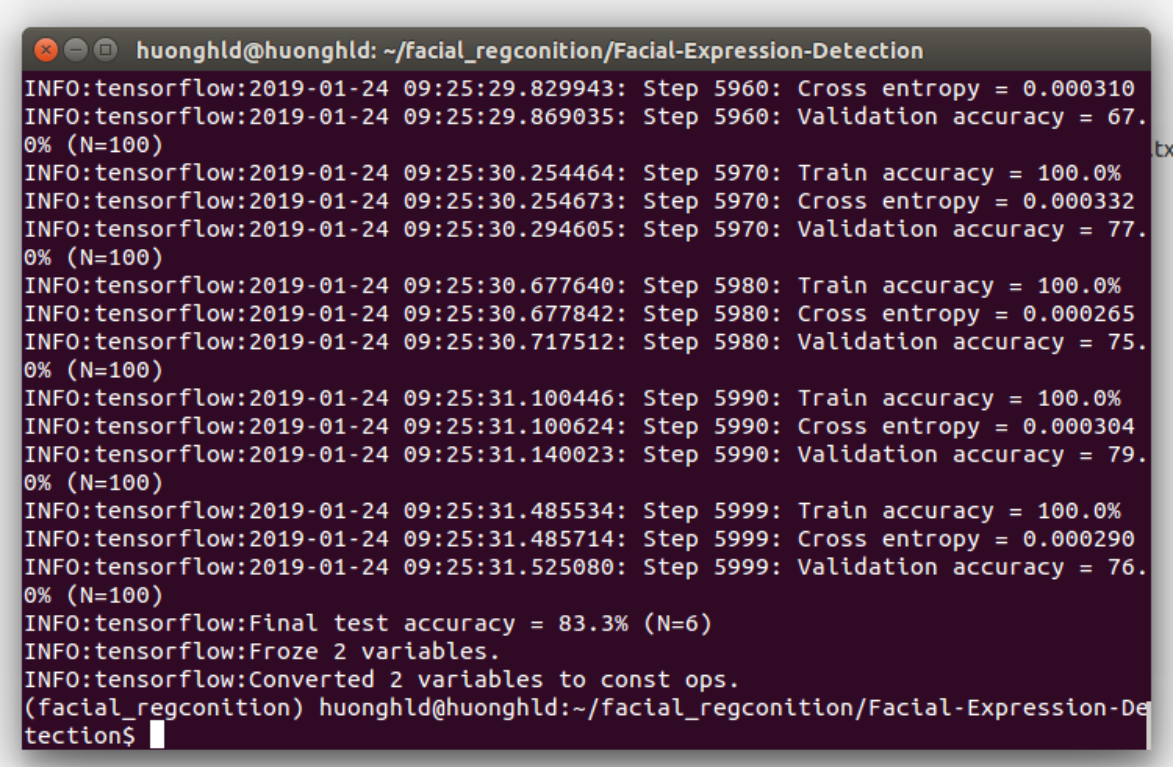
5.2. Phân loại cảm xúc

Dữ liệu sau HAAR-cascade sẽ được sử dụng làm dữ liệu đào tạo, với số lượng các mẫu đào tạo cụ thể như sau:

Bảng 5.1. Số lượng mẫu training

Biểu cảm	Số lượng mẫu
Buồn	31
Vui	22
Cười	27
Ngáp ngủ	30

Sau khi thực hiện training 6000 vòng với MobileNet, độ chính xác của hệ thống được tính toán xấp xỉ 83.3% (hình5.9).



Hình 5.9. Độ chính xác sau training

Dưới đây là một số hình ảnh kiểm thử chức năng của hệ thống. Được chạy dưới hệ điều hành Ubuntu 14.04, Kernel version 3.19.80, Camera chất lượng VGA.

Hình 5.10. Kiểm thử hoạt động của hệ thống

Ta thấy rằng hệ thống đã hoạt động đúng như yêu cầu đề ra.

5.3. Kết luận

Sau bước mô phỏng, chúng em nhận thấy hệ thống đã hoạt động đúng như chức năng yêu cầu.

Chương 6. Kết luận

Nhóm đã hoàn thành sản phẩm đạt các yêu cầu đề ra về chức năng và phi chức năng. Tuy quy mô đề tài còn nhỏ và đơn giản, nhưng chúng em đã nắm được các kiến cơ bản về mạng nơ-ron tích chập, mạng MobileNet, cũng như cách sử dụng các thư viện mã nguồn mở như OpenCV và Tensorflow. Để cải thiện hệ thống, chúng em sẽ mở rộng các loại biểu cảm cũng như số lượng mẫu đào tạo để hệ thống hoạt động chính xác hơn nữa.

Nhóm chúng em xin cảm ơn cô giáo ThS. Nguyễn Thị Kim Thoa đã tận tình hướng dẫn và giúp đỡ chúng em thực hiện đề tài này.

Tài liệu tham khảo

[1] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, Google Inc, 2017

[2] Vu H. Tiep, *Machine Learning Cơ bản*, NXB Khoa học kĩ thuật, 2018.