**BÁO CÁO MÔN HỌC**

ĐỀ TÀI: NHẬN DIỆN CẢM XÚC SỬ DỤNG MẠNG NƠ RON TÍCH CHẬP

MỤC LỤC

[I/ TỔNG QUAN 3](#_Toc36628023)

[II/ CƠ SỞ LÝ THUYẾT 3](#_Toc36628024)

[**2.1. Mạng CNN** 3](#_Toc36628025)

[**2.2. Vecto đặc trưng** 6](#_Toc36628026)

[**2.3. HOG (Histogram of Oriented Gradients) – Trích xuất đặt trưng của ảnh** 6](#_Toc36628027)

[**2.4. Facial landmark (các điểm quan trọng trên gương mặt)** 10](#_Toc36628028)

[**2.5. Dropout trong CNN** 10](#_Toc36628029)

[II/ THỰC HÀNH 11](#_Toc36628030)

[**1. Dữ liệu** 12](#_Toc36628031)

[**2. Môi trường thực hiện** 12](#_Toc36628032)

[**3. Quy trình thực hiện** 13](#_Toc36628033)

[**3.1 Trainning** 13](#_Toc36628034)

[**3.2. Testing and Using** 15](#_Toc36628035)

[**4. Một số hình ảnh** 16](#_Toc36628036)

[IV/ ĐÁNH GIÁ VÀ PHÁT TRIỂN 16](#_Toc36628037)

[**4.1 Đánh giá** 17](#_Toc36628038)

[**4.2 Nhận xét** 17](#_Toc36628039)

[V/ TÀI LIỆU THAM KHẢO 18](#_Toc36628040)

# I/ TỔNG QUAN

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. Mục tiêu của khóa luận này là nghiên cứu và đánh giá về các phương pháp nhận dạng mặt người trong việc nhận dạng ra các cảm xúc cơ bản.

Dữ liệu cho các mô hình huấn luận có thể sử dụng **Fer2013.** Dữ liệu bao gồm hình ảnh xám 48x48 pixel của khuôn mặt. Các khuôn mặt đã được đăng ký tự động để khuôn mặt ít nhiều được căn giữa và chiếm khoảng cùng một khoảng trống trong mỗi hình ảnh. Nhiệm vụ là phân loại từng khuôn mặt dựa trên cảm xúc thể hiện trong biểu cảm khuôn mặt theo một trong bảy loại (0 = Giận dữ, 1 = Ghê tởm, 2 = Sợ hãi, 3 = Hạnh phúc, 4 = Buồn, 5 = Bất ngờ, 6 = Trung lập ). Train gồm 28.709 sample. Test được sử dụng bao gồm 3.589 samples

Đối với các bài toán nhận dạng trước đều phải trải qua các quá trình sử dụng các bộ lọc (Normalized Box Filter, Gauss, Bilateral,  Sobel, Scharr, …) để phát hiện các đặt trưng cũng như xử lý ảnh). Nhưng đối với Deep learning thì những việc đó đã có sẵn

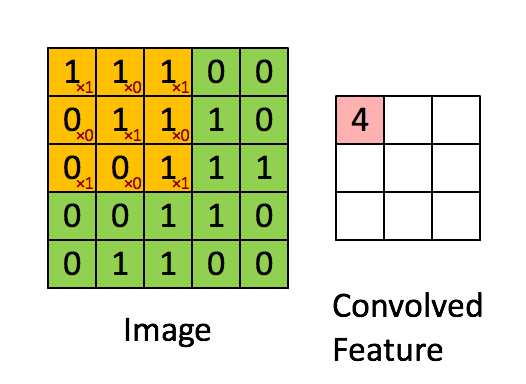
# II/ CƠ SỞ LÝ THUYẾT

## **2.1. Mạng CNN**

* Convolutional Neural Network (CNN) là một mô hình kiến trúc deep learning nổi tiếng. Hầu hết các bài toán liên quan đến nhận dạng và phân loại ảnh hiện nay đều sử dụng phương pháp này trong việc xây dựng model từ cơ bản cho đến phức tạp. Các kiến trúc như Alex, LeNet, Inception, VGG… đều có sử dụng CNN là thành phần trong mô hình của chún
* Đặc trưng của mạng CNN là không cần phải trích xuất đặt trưng của ảnh. Do vậy không yêu cầu người lập trình có sự hiểu biết về lĩnh vực của đối tượng cần phân lại
* Tuy nhiên do số lượng parameter nhiều nên yêu cầu về dữ liệu là lớn. Nếu không có đủ dữ liệu, thì kết quả thu được sẽ có độ chính xác không cao

**2.1.1 Phép tích chập với filter (kernel)**

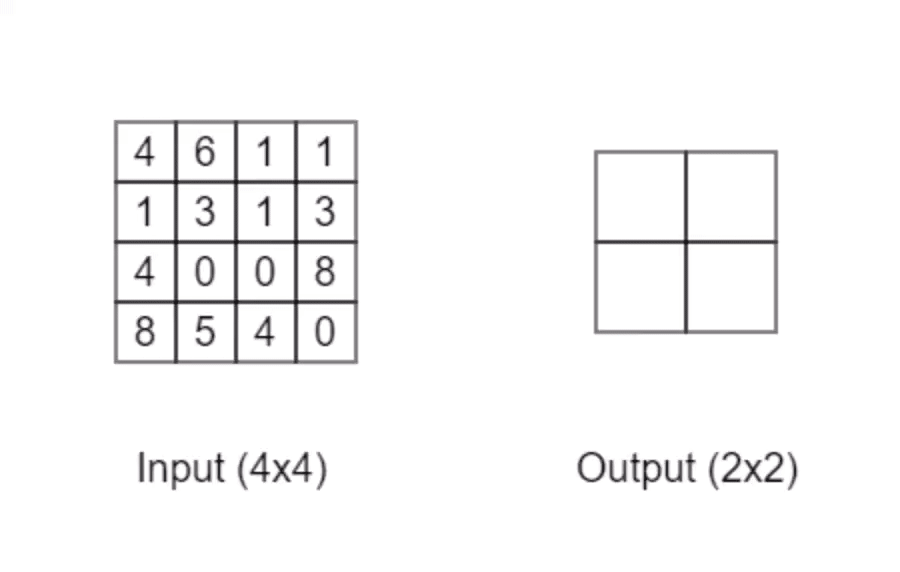
Convolutional Neural Network là một kiến trúc lấy ý tưởng từ việc áp dụng các filters để tìm ra các features cho ảnh đầu vào, sau đó các sử dụng các features đó cho việc phân loại ảnh. **Trong kiến trúc này có nhiều thành phần, nhưng core của nó là một thành phần được gọi là lớp Convolution (CONV). Lớp CONV này sẽ chứa các filters cho kiến trúc CNN**. **Tuy nhiên, thay vì phải tự đưa ra các filters thủ công như CV, các filters sẽ được tìm ra một cách tự động thông qua quá trình trainning**. Có nghĩa là các giá trị trong mỗi một filter lúc này (chính là các weights) ban đầu sẽ được khởi tạo ngẫu nhiên, sau quá trình học mà chúng sẽ được cập nhật lại, và kết quả cố định cuối cùng được ổn định khi kết thúc training. Khi xây dựng CNN, chúng ta chỉ cần chỉ ra là ở mỗi lớp CONV cần số lượng bao nhiêu filter, size mỗi filter, giá trị stride, giá trị padding…



**2.1.2 Lớp tổng hợp (pooling layer)**

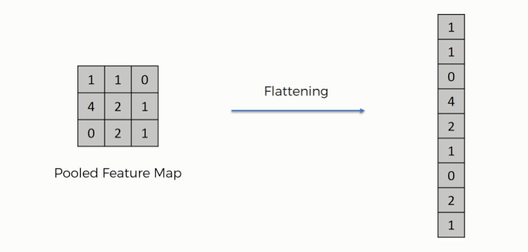
Lớp pooling thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron.

Có 2 phương pháp chủ yếu đó là **Max pooling** và **Average pooling**

****

**2.1.3 Fully connected layer**

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,…) thì tensor của output của layer cuối cùng, kích thước H\*W\*D, sẽ được chuyển về 1 vector kích thước (H\*W\*D)



Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model.

Fully connected layer là một mạng MLP với đầu vào là các ma trận cuối của quá trình Convolution và Pooling và đầu ra các một mảng các phần tử tương ứng với số lượng đối tượng được phân chia.

## **2.2. Vecto đặc trưng**

Trích xuất thuộc tính(feature extraction) là một quá trình nhằm biến dữ liệu phức tạp đầu vào thành một cách biểu diễn dữ liệu đơn giản hơn, phù hợp hơn cho các thuật toán học máy. Dữ liệu sau khi xử lý đã được lược bỏ phần dữ liệu dư thừa, giữ lại những dữ liệu có ích cho bài toán cần xử lý.

Đối với bài xử lý ảnh mặt người thông thường người ta hay xử dụng 2 phương pháp trích xuất cơ bản là **Facial Landmark** và **Histogram of Oriented Gradients**

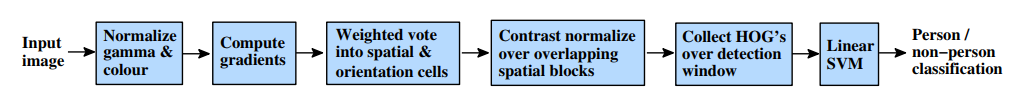
## **2.3. HOG (Histogram of Oriented Gradients) – Trích xuất đặt trưng của ảnh**

Phương pháp rút trích đặc trưng hình ảnh HOG xuất bản ở hội nghị CVPR 2005 được đề xuất bởi tác giả là Dalal và Triggs. Bài báo gốc HOG đề xuất phương pháp rút trích đặc trưng sử dụng các thống kê histogram về hướng trên ảnh gradient cho bài toán phát hiện người (human detection).

Rút trích đặc trưng hình ảnh nằm ở bước 3 để giải bài toán phát hiện đối tượng.

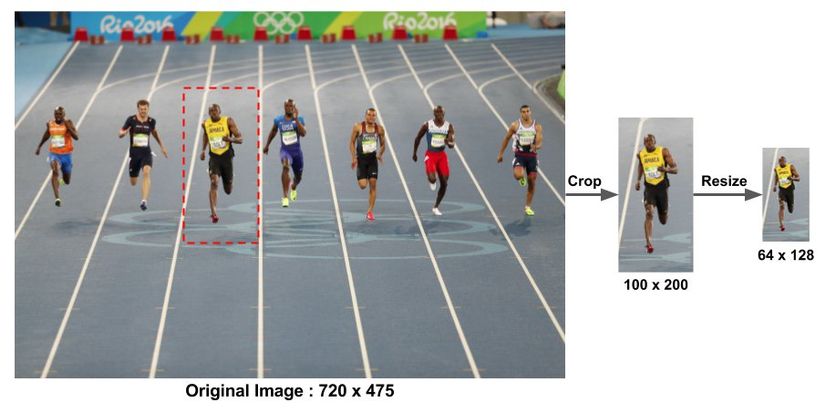
1. Đọc ảnh
2. Tiền xử lý (preprocessing): hình ảnh được đưa qua bước tiền xử lý để thực hiện các thao tác như cân bằng sáng, làm mờ, …
3. Trích đặc trưng ảnh (feature extraction): bằng cách sử dụng các phương pháp rút trích đặc trưng ảnh ta sẽ thu được **vector đặc trưng** của ảnh. Nói một cách nôm na thân quen đó chính là bạn mã hóa hình ảnh thành một vector, và vector này mang những đặc trưng (các số thực) đại diện cho ảnh đó.
4. Huấn luyện mô hình học máy (training): với phương pháp truyền thống, ta thường sử dụng mô hình SVM trong **machine learning** để phân tách các vector đặc trưng thành các lớp cần phân loại.
5. Kiểm thử (validation): sau khi huấn luyện xong mô hình học máy bạn cần phải đánh giá mô hình mình đã huấn luyện đạt độ chính xác là bao nhiêu phần trăm trên tập kiểm thử này. Khi bạn đã hài lòng với kết quả kiểm thử, ta có thể dừng quá trình huấn luyện.

**HIỆN THỰC GIẢI THUẬT HOG**



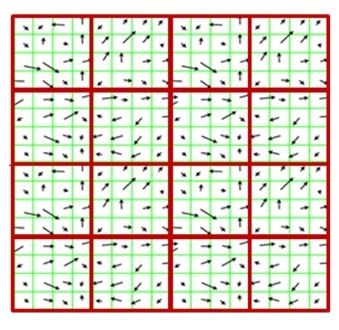
1. Chuẩn hóa mức sáng và màu sắc của ảnh cũng như kích thước

Trong bài toán này, để thuận tiện cho việc chia đều hình ảnh thành các khối, ô và tính toán đặc trưng ở các bước tiếp theo, chúng ta cần resize kích thước tất cả các hình ảnh trong tập dữ liệu về một kích thước chung.

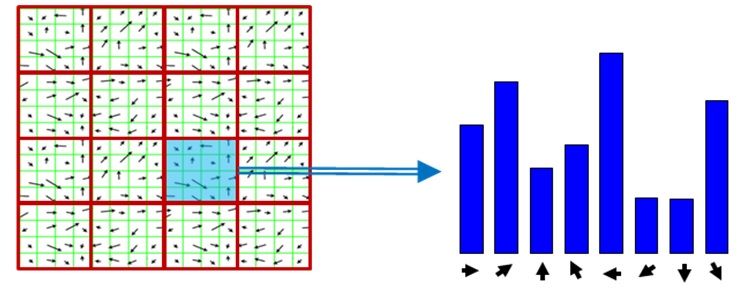


1. Tính gradient của ảnh

Đây là bước đầu tiên, được thực hiện bằng hai phép nhân chập ảnh gốc với 2 chiều, tương ứng với các toán tử lấy đạo hàm theo hai hướng Ox và Oy.



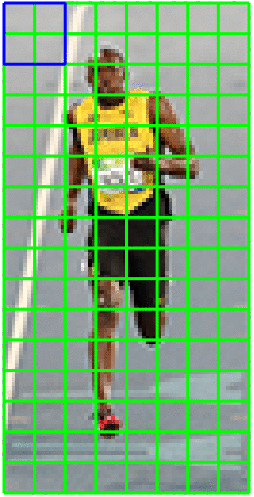
1. Vote hướng vào cell (histogram)



1. Chuẩn hóa theo block

Ta cần chia các block giống như chia cell ở phía trên. Đơn vị của ta không còn là các điểm ảnh nữa mà là các cell. Người ta thường sử dụng hoặc 2 x 2 hoặc 3 x 3 cell\_per\_block có được độ chính xác hợp lý trong hầu hết các trường hợp.Các block này sẽ chồng lên nhau.

1. Trích vector đặc trưng của các block đã chuẩn hóa cho ảnh



* Với mỗi hình ảnh kích thước 64x128, chia thành các block 16x16 chồng nhau, sẽ có 7 block ngang và 15 block dọc, nên sẽ có 7x15 = 105 blocks.
* Mỗi block gồm 4 cell. Khi áp dụng biểu đồ 9-bin cho mỗi cell, mỗi block sẽ được đại diện bởi một vector có kích thước 36x1.
* Vì vậy, khi nối tất cả các vector trong một block lại với nhau, ta sẽ thu được vector đặc trưng HOG của ảnh có kích thước 105x36x1 = 3780x1.

## **2.4. Facial landmark (các điểm quan trọng trên gương mặt)**

Facial landmark là xác định các vị trí như eye(mắt), nose(mũi),mounth(miệng) trên khuôn mặt(face). Nó có rất nhiều ứng dụng vui mà ta thường thây trên các app điện thoại chẳng hạn ( swap face,draw in face or tạo hiệu ứng trên khuôn mặt).



Thư viện Dlib được viết bằng ngôn ngữ C++ do **Davis King** tạo ra vào năm 2012. Được sử dụng nhiều trong lĩnh vực computer vision đặc biệt là nhận dạng object và face.

## **2.5. Dropout trong CNN**

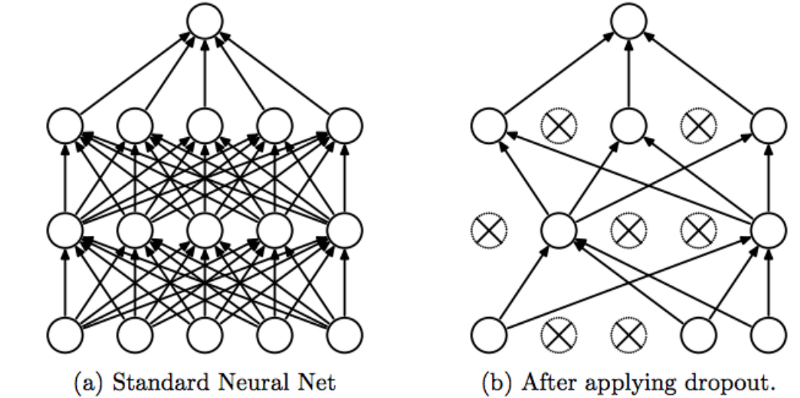
Trước đây, trong machine learning, người ta thường sử dụng regularization để ngăng chặn over-fititng. Regularization làm giảm over-fitting bằng cách thêm yếu tố “phạt” vào hàm độ lỗi (loss function). Bằng việc thêm vào điểm phạt này, mô hình được huấn luyện sẽ giúp các features weights giảm đi sự phụ thuộc lẫn nhau. Đối với những ai đã sử dụng Logistic Regression rồi thì sẽ không xa lạ với thuật ngữ phạt L1(Laplacian) và L2 (Gaussian).

Dropout là một kỹ thuật khác, một cách tiếp cận khác để regularization trong mạng neural netwoks

Kỹ thuật dropout được thực hiện như sau:

**Trong pha train:** với mỗi hidden layer, với mỗi trainning sample, với mỗi lần lặp, chọn ngẫu nhiên p phần trăm số node và bỏ qua nó (bỏ qua luôn hàm kích hoạt cho các node bị bỏ qua).

**Trong pha test:** Sử dụng toàn bộ activations, nhưng giảm chúng với tỷ lệ p (do chúng ta bị miss p% hàm activation trong quá trình train).



# II/ THỰC HÀNH

## **1. Dữ liệu**

Để huấn luyện model, sử dụng datset Fer2013 chứa 30.000 hình ảnh biểu cảm khuôn mặt được nhóm thành bảy loại: Tức giận, Ghê tởm, Sợ hãi, Vui, Buồn, Bất ngờ và Trung lập.

Kích thước mỗi ảnh sẽ là 48x48

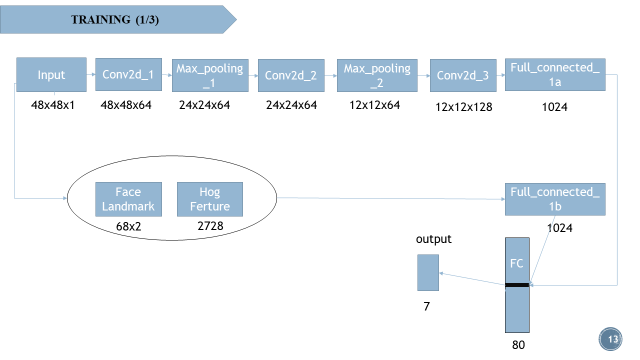
Ta cũng trích xuất các đặt trưng HOG và nhập dữ liệu hình ảnh thô với các mốc khuôn mặt + hog vào mạng CNN

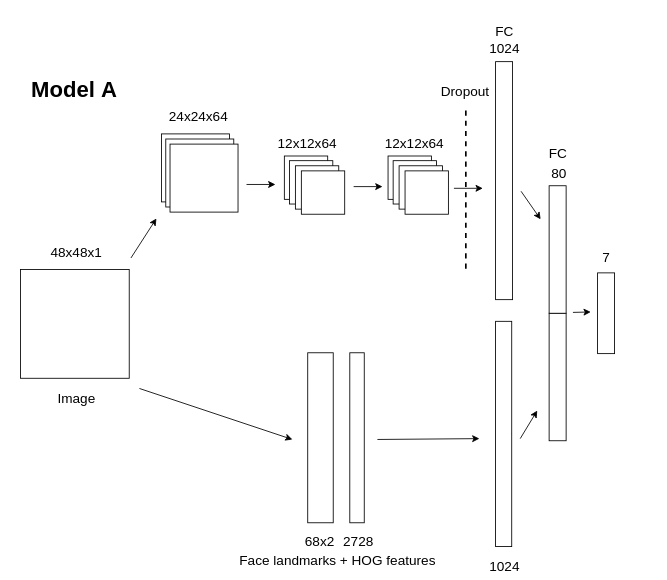
## **2. Môi trường thực hiện**

* Tensorflow
* Python 3.6
* Tflearn
* Numpy
* Argparse
* [optional] Hyperopt + pymongo + networkx
* [optional] dlib, imutils, opencv 3
* [optional] scipy, pandas, skimage

## **3. Quy trình thực hiện**

### **3.1 Trainning**





**Code của model**

def build\_modelA(optimizer=HYPERPARAMS.optimizer, optimizer\_param=HYPERPARAMS.optimizer\_param,

    learning\_rate=HYPERPARAMS.learning\_rate, keep\_prob=HYPERPARAMS.keep\_prob,

    learning\_rate\_decay=HYPERPARAMS.learning\_rate\_decay, decay\_step=HYPERPARAMS.decay\_step):

    images\_network = input\_data(shape=[None, NETWORK.input\_size, NETWORK.input\_size, 1], name='input1')

    images\_network = conv\_2d(images\_network, 64, 5, activation=NETWORK.activation)

    #images\_network = local\_response\_normalization(images\_network)

    if NETWORK.use\_batchnorm\_after\_conv\_layers:

        images\_network = batch\_normalization(images\_network)

    images\_network = max\_pool\_2d(images\_network, 3, strides = 2)

    images\_network = conv\_2d(images\_network, 64, 5, activation=NETWORK.activation)

    if NETWORK.use\_batchnorm\_after\_conv\_layers:

        images\_network = batch\_normalization(images\_network)

    images\_network = max\_pool\_2d(images\_network, 3, strides = 2)

    images\_network = conv\_2d(images\_network, 128, 4, activation=NETWORK.activation)

    if NETWORK.use\_batchnorm\_after\_conv\_layers:

        images\_network = batch\_normalization(images\_network)

    images\_network = dropout(images\_network, keep\_prob=keep\_prob)

    images\_network = fully\_connected(images\_network, 1024, activation=NETWORK.activation)

    if NETWORK.use\_batchnorm\_after\_fully\_connected\_layers:

        images\_network = batch\_normalization(images\_network)

    if NETWORK.use\_landmarks or NETWORK.use\_hog\_and\_landmarks:

        if NETWORK.use\_hog\_sliding\_window\_and\_landmarks:

            landmarks\_network = input\_data(shape=[None, 2728], name='input2')

        elif NETWORK.use\_hog\_and\_landmarks:

            landmarks\_network = input\_data(shape=[None, 208], name='input2')

        else:

            landmarks\_network = input\_data(shape=[None, 68, 2], name='input2')

        landmarks\_network = fully\_connected(landmarks\_network, 1024, activation=NETWORK.activation)

        if NETWORK.use\_batchnorm\_after\_fully\_connected\_layers:

            landmarks\_network = batch\_normalization(landmarks\_network)

        landmarks\_network = fully\_connected(landmarks\_network, 40, activation=NETWORK.activation)

        if NETWORK.use\_batchnorm\_after\_fully\_connected\_layers:

            landmarks\_network = batch\_normalization(landmarks\_network)

        images\_network = fully\_connected(images\_network, 40, activation=NETWORK.activation)

        network = merge([images\_network, landmarks\_network], 'concat', axis=1)

    else:

        network = images\_network

    network = fully\_connected(network, NETWORK.output\_size, activation='softmax')

    if optimizer == 'momentum':

        optimizer = Momentum(learning\_rate=learning\_rate, momentum=optimizer\_param,

                    lr\_decay=learning\_rate\_decay, decay\_step=decay\_step)

    elif optimizer == 'adam':

        optimizer = Adam(learning\_rate=learning\_rate, beta1=optimizer\_param, beta2=learning\_rate\_decay)

    else:

        print( "Unknown optimizer: {}".format(optimizer))

    network = regression(network, optimizer=optimizer, loss=NETWORK.loss, learning\_rate=learning\_rate, name='output')

    return network

1. Các ảnh đầu vào ở dạng 48x48x1 (kích thước 48x48 và là ảnh gray)

2. Trích xuất Face landmark và các đặt trưng HOG của ảnh

Cho ảnh qua mạng cnn với các tham số đã cài đặt

3. Ta dùng thêm kỹ thuật dropout để hạn chế model train được rơi vào tình trạng over-fitting

4. Tại tầng Fully connected ta kết hợp với các đặt trưng từ Face landmark và HOG để tăng độ chính xác của model

### **3.2. Testing and Using**

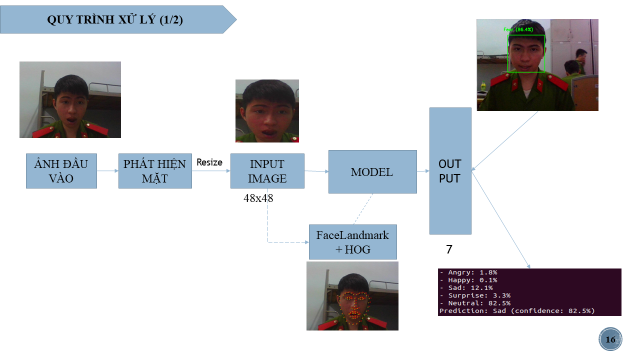
**3.2.1 Quy trình xử lý khi sử dụng**

e. Sử dụng model đã train được tiếng hành xử lý với hình ảnh từ camera thu được.

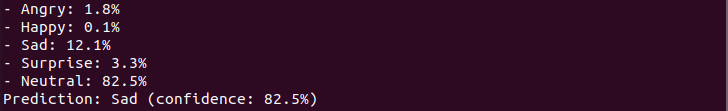
f. Để tiến hành demo ta còn xử dụng thêm các modal khác như là Face-detection, để xác định được các gương mặt để đưa vào modal vừa huấn luyện

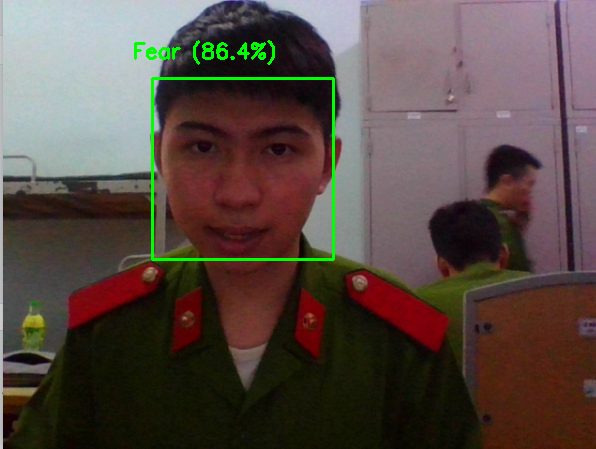
## **4. Một số hình ảnh**

Đầu vào



Đầu ra





# IV/ ĐÁNH GIÁ VÀ PHÁT TRIỂN

## **4.1 Đánh giá**

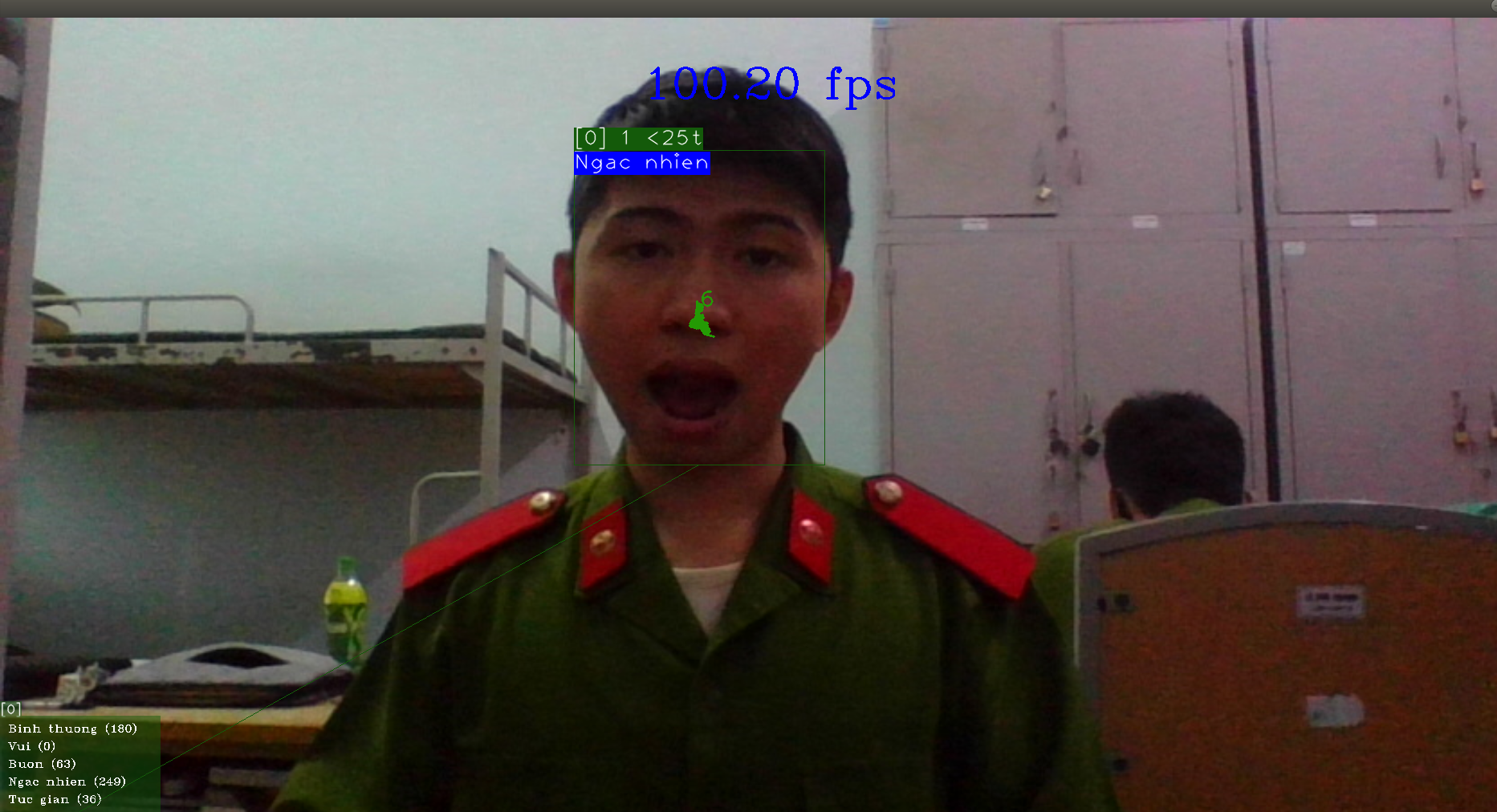
|  |  |
| --- | --- |
| Tình huống | Độ chính xác |
| CNN (on raw pixels) | 62,4 |
| CNN + Face landmarks | 63,5 |
| CNN + Face landmarks + HOG | 65,7 |

## **4.2 Nhận xét**

* Kết quả thu được còn chưa được thật sự chính xác (độ chính xác rơi vào ~62%)
* Khi thực hiện demo bằng camera với hình ảnh liên tục thì kết quả không tốt.
* Kết quả đầu ra thay đổi liên tục dù người mẫu không thay đổi nhiều, chứng tỏ mô hình chưa tốt
* Trong thực tế việc thay đổi cảm xúc có thể diễn ra trong một khoảng thời gian dài, kéo dài trong nhiều frame. Thế nên cần có những cơ chế để thực hiện lưu giữ.
* Các cảm xúc nên cần có trọng số để tính toán độ quan trọng của các cảm xúc. Những cảm xúc tiêu cực cần có trọng số lớn hơn. Thể hiện sự quan tâm của người sử dụng sản phẩm.
* Có thể kết hợp với các model khác như nhận diện gương mặt, nhận diện tuổi, giới tính, …

**4.3 So sánh với mô hình trên thế giới và phát triển demo**

Một số pre-train model về nhận diện cảm xúc như **emotions-recognition** (của **OpenVINO™ Toolkit Pre-Trained Models)** có độ chính xác 70,2% đối với 5 cảm xúc ('neutral', 'happy', 'sad', 'surprise', 'anger').



Sử dụng các pre-train model của Openvino ( nhận dạng cảm xúc, nhận dạng tuổi, nhận dạng gương mặt)

# V/ TÀI LIỆU THAM KHẢO

1. <https://minhng.info/tutorials/histograms-of-oriented-gradients.html>
2. <https://thorpham.github.io/blog/2018/04/20/face-landmark/>
3. <https://github.com/amineHorseman/facial-expression-recognition-using-cnn>