**1**



**Giáo viên hướng dẫn**: ThS. Trương Văn Thông

**Lớp**: DHKHMT13A

**Thành viên**:

1. Trần Ngọc Thương – 17037881
2. Đỗ Tùng Lâm – 17045111
3. Lê Minh Phú – 17023701

**TP.HỒ CHÍ MINH, ngày 18 tháng 12 năm 2020**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**

**BÁO CÁO**

**CÔNG NGHỆ MỚI TRONG PHÁT TRIỂN ỨNG DỤNG CNTT**

MỤC LỤC

[CHƯƠNG 1. Giới thiệu 3](#_Toc59452580)

[1. Giới thiệu đề tài 3](#_Toc59452581)

[2. Ứng dụng đề tài 4](#_Toc59452582)

[3. Hướng phát triển đề tài 4](#_Toc59452583)

[4. Bộ dữ liệu sử dụng 4](#_Toc59452584)

[CHƯƠNG 2. Giải pháp 6](#_Toc59452585)

[1. Sử dụng Mạng neural tích chập CNN (Convolutional Neural Networks) 6](#_Toc59452586)

[2. Framework hỗ trợ 11](#_Toc59452587)

[3. Mô hình sử dụng 14](#_Toc59452588)

[CHƯƠNG 3. Demo 15](#_Toc59452589)

[1. Giới thiệu 15](#_Toc59452590)

[CHƯƠNG 4. Tài liệu tham khảo 17](#_Toc59452591)

# Giới thiệu

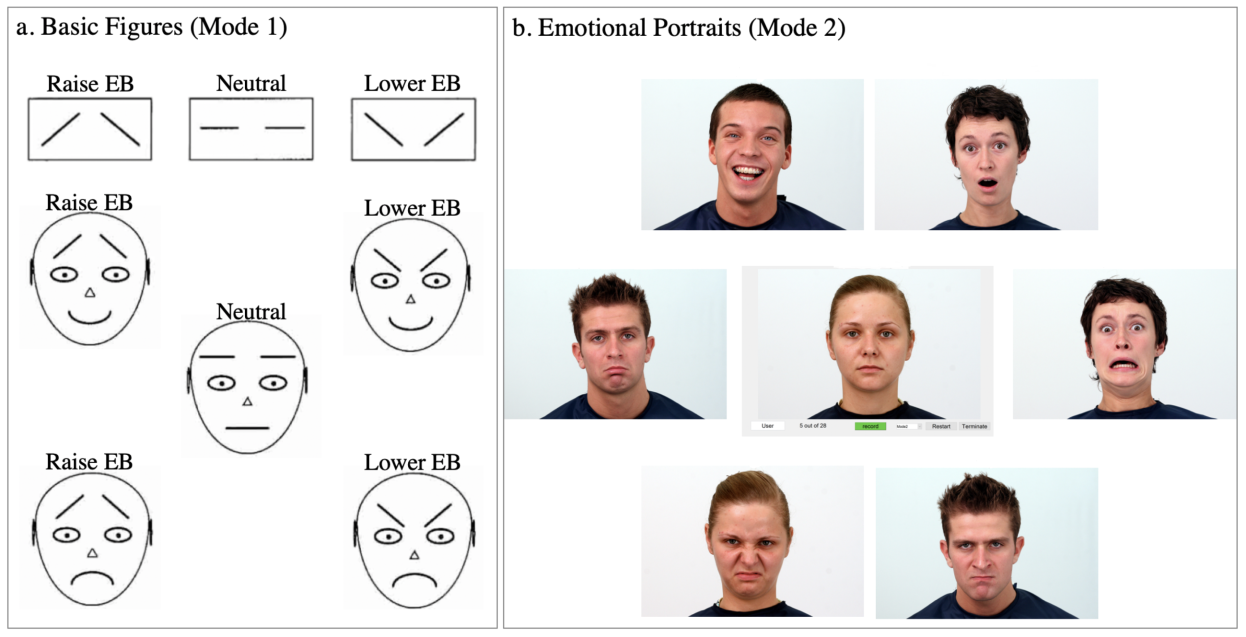
## Giới thiệu đề tài

Báo cáo đề tài môn Công nghệ mới trong phát triển ứng dụng CNTT với đề tài “Nhận diện trạng thái khuôn mặt”.

Đề tài nhận diện trạng thái khuôn mặt bao gồm 7 nhãn là ["angry", "disgust", "scared", "happy", "sad", "surprised", "neutral"].

Mỗi nhãn là một trạng thái ứng với biểu cảm trên khuôn mặt hiện tại người dùng:

* Nhãn là “angry” ứng với các biểu cảm tức giận.
* Nhãn “disgust” ứng với các biểu cảm ghê tởm.
* Nhãn “scared” ứng với các biểu cảm sợ hãi.
* Nhãn “happy” ứng với các biểu cảm vui mừng, hạnh phúc.
* Nhãn “sad” ứng với các biểu cảm buồn bả, âu sầu.
* Nhãn “surprised” ứng với các biểu cảm ngạc nhiên.
* Nhãn “neutral” ứng với các biểu cảm trung lập.



Happy

Surprised

Sad

Neutral

Scared

Disgust

Angry

Đánh giá được biểu cảm trên khuôn mặt một cách tự động thông qua các thiết bị số là việc cần thiết trong thời đại số công nghiệp 4.0 hiện nay vì dữ liệu người dùng ngày càng nhiều cần sự hỗ trợ của máy để giảm tiêu tốn sức lực con người.

## Ứng dụng đề tài

Việc đánh giá trạng thái biểu cảm của người dùng có thể tạo nên nhiều ứng dụng trong thực tế như :

* Tạo sản phẩm robot nhận dạng thái độ người đối diện bằng cách nhúng code vào mạch arduino (sản phẩm phần cứng).
* Phần mềm dạng web, app mobile, webapp để người dùng có thể dễ dàng tiếp cận công nghệ.
* Phần mềm dạng app mobile
* Quản lý thái độ của nhân viên khi tiếp xúc với khách hàng tránh được trường hợp nhân viên sử dụng thái độ tiêu cực khi làm việc, ảnh hưởng đến hình ảnh của doanh nghiệp.
* Công nghệ nhận diện biểu cảm, cảm xúc trên khuôn mặt con người sẽ giúp người sử dụng biết được mức độ quan tâm, phản ứng của người tiêu dùng với sản phẩm, dịch vụ của mình. Từ đó, giúp doanh nghiệp có cơ sở để đưa ra những chiến lược điều chỉnh hợp lý.
* Ứng dụng để người dùng luyện tập kỹ năng biểu cảm trên khuôn mặt hỗ trợ cho việc kiểm soát thái độ trên khuôn mặt.

## Hướng phát triển đề tài

Đề tài có thể được phát triển để nhận diện được nhiều thuộc tính trên khuôn mặt hơn (như nhận diện cảm xúc kết hợp xác định giới tính,..).

Nhận diện được nhiều loại cảm xúc hơn.

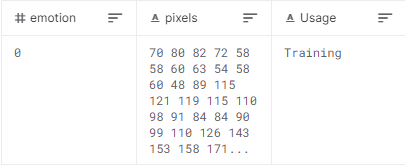
Nhận diện được nhiều khuôn mặt trong một khung ảnh.

## Bộ dữ liệu sử dụng

Bộ dữ liệu (dataset) sử dụng trong project: FER2013 (Facial Expression Detection) một bộ dataset nổi tiếng trên kaggle với 35,887 bức ảnh khuôn mặt có kích thước 48x48 pixel.

Bộ dataset sẽ được chia ra 80% train, 20% test.

Những hình ảnh của bộ dữ liệu này đã được lưu trữ dưới dạng file csv



Có 3 cột thuộc tính:

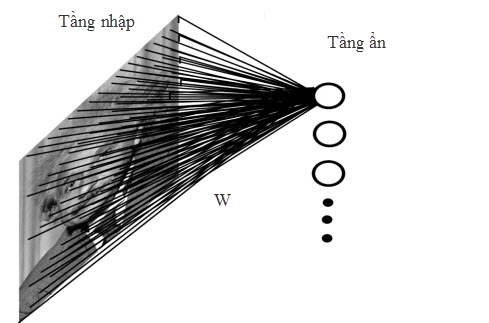
* emotion sẽ lưu thông tin của từng ảnh với các chỉ số sau: (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral ).
* pixels hình ảnh được lưu với dạng vector số
* usage thể hiện hình thức sử dụng của dữ liệu.

Link download: (<https://www.kaggle.com/deadskull7/fer2013>)

# Giải pháp

## Sử dụng Mạng neural tích chập CNN (Convolutional Neural Networks)

**Lý do lựa chọn mạng thuật toán CNN:**

****

Hình trên có kích thước 200\*200 nếu như giả sử số neural tầng ẩn là 40000 thì tổng trọng số phải lên tới 1,6 tỉ => điều này sẽ gây khó khăn cho việc tính toán.

Cụ thể khó khăn trong huấn luyện Mạng Nơ-ron thông thường: Chi phí xây dựng dữ liệu huấn luyện rất lớn và thời gian huấn luyện lâu.

Vì vậy, sử dụng mạng CNN trích xuất những đặc trưng quan trọng của hình ảnh làm giảm lượng dữ liệu đầu vào làm cho việc tính toán nhanh hơn.

**Cảm hứng của Mạng Nơ-ron Tích Chập:**

Tương tự như việc trẻ em học cách nhận diện đối tượng, chúng ta cần cho thuật toán học rất nhiều hình ảnh trước khi nó có thể đưa ra dự đoán cho hình ảnh đầu vào mà nó chưa từng thấy.

Máy tính “nhìn” theo cách khác con người. Trong thế giới máy tính chỉ có những con số. Mỗi hình ảnh có thể được biểu diễn dưới dạng mảng 2 chiều những con số được gọi là các pixel.

Để dạy thuật toán nhận diện đối tượng trong hình ảnh, ta sử dụng một loại Mạng Nơ-ron Nhân Tạo (Artificial Neural Network): Mạng Nơ-ron Tích Chập. Tên của nó được dựa trên phép tính quan trọng được sử dụng trong mạng đó là Tích Chập.

Mạng Nơ-ron Tích Chập lấy cảm hứng từ não người. Nghiên cứu trong những thập niên 1950 và 1960 của D.H Hubel và T.N Wiesel trên não của động vật đã đề xuất một mô hình mới cho việc cách mà động vật nhìn nhận thế giới. Trong báo cáo, hai ông đã diễn tả 2 loại tế bào nơ-ron trong não và cách hoạt động khác nhau: tế bào đơn giản (simple cell – S cell) và tế bào phức tạp (complex cell – C cell).

Các tế bào đơn giản được kích hoạt khi nhận diện các hình dáng đơn dản như đường nằm trong một khu vực cố định và một góc cạnh của nó. Các tế bào phức tạp có vùng tiếp nhận lơn hơn và đầu ra của nó không nhạy cảm với những vị trí cố định trong vùng.

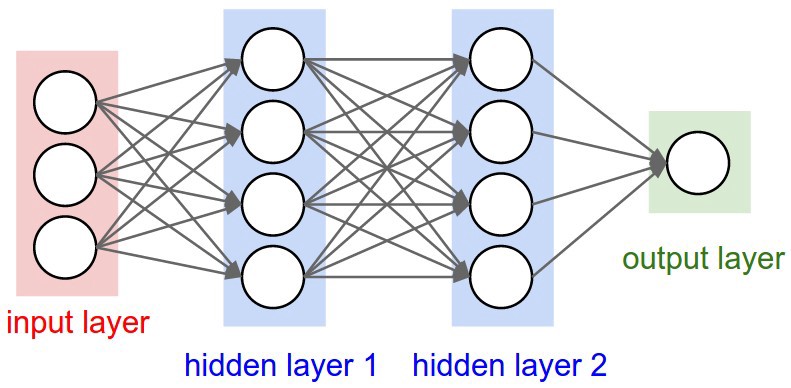
Trong thị giác, vùng tiếp nhận của một nơ-ron tương ứng với một vùng trên võng mạc nơi mà sẽ kích hoạt nơ-ron tương ứng.

Năm 1980, Fukushima đề xuất mô hình mạng nơ-ron có cấp bậc gọi là neocognitron. Mô hình này dựa trên khái niệm về S cell và C cell. Mạn neocognitron có thể nhận diện mẫu dựa trên việc học hình dáng của đối tượng.

Sau đó vào năm 1998, Mạng Nơ-ron Tích Chập được giới thiệu bởi Bengio, Le Cun, Bottou và Haffner. Mô hình đầu tiên của họ được gọi tên là LeNet-5. Mô hình này có thể nhận diện chữ số viết tay.

**Kiến trúc Mạng Nơ-ron Tích Chập:**

Mạng Nơ-ron Tích Chập có kiến trúc khác với Mạng Nơ-ron thông thường. Mạng Nơ-ron bình thường chuyển đổi đầu vào thông qua hàng loạt các tầng ẩn. Mỗi tầng là một tập các nơ-ron và các tầng được liên kết đầy đủ với các nơ-ron ở tầng trước đó. Và ở tầng cuối cùng sẽ là tầng kết quả đại diện cho dự đoán của mạng.



Mạng Nơ-ron Tích Chập được chia thành 3 chiều: rộng, cao, và sâu. Các nơ ron trong mạng không liên kết hoàn toàn với toàn bộ nơ-ron kế đến nhưng chỉ liên kết tới một vùng nhỏ. Cuối cùng, một tầng đầu ra được tối giản thành véc-tơ của giá trị xác suất.

CNNs gồm hai thành phần:

* Phần tầng ẩn hay phần rút trích đặc trưng: trong phần này, mạng sẽ tiến hành tính toán hàng loạt **phép tích chập** **(Convolution)** và **phép hợp nhất** **(pooling)** để phát hiện các đặc trưng.
* Phần phân lớp: tại phần này, một lớp với các liên kết đầy đủ (Fully-connected) sẽ đóng vai trò như một bộ phân lớp các đặc trưng đã rút trích được trước đó. Tầng này sẽ đưa ra xác suất của một đối tượng trong hình.

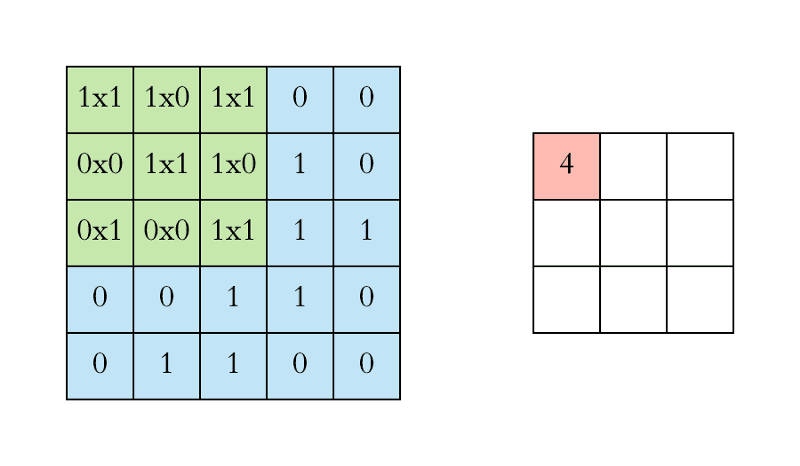
**Rút trích đặc trưng:**

Tích chập là một khối quan trọng trong CNN. Thuật ngữ tích chập được dựa trên một phép hợp nhất toán học của hai hàm tạo thành hàm thứ ba. Phép toán này kết hợp hai tập thông tin khác nhau.

Trong trường hợp CNN, tích chập được thực hiện trên giá trị đầu vào của dữ liệu và kernel/filter (thuật ngữ này được sử dụng khác nhau tùy tình huống) để tạo ra một bản đồ đặc trưng (feature map).

Ta thực hiện phép tích chập bằng cách trượt kernel/filter theo dữ liệu đầu vào. Tại mỗi vị trí, ta tiến hành phép nhân ma trận và tính tổng các giá trị để đưa vào bản đồ đặc trưng.

Trong hình dưới đây, thành phần kernel/filter (màu xanh lá) trượt trên đầu vào (màu xanh dương) và kết quả được trả về bản đồ đặc trưng (màu đỏ). Kernel/filter có kích thước là 3×3 trong ví dụ này.



Ta thực hiện phép tích chập trên đầu vào nhiều lần khác nhau. Mỗi lần sử dụng một kernel/filter khác nhau. Kết quả ta sẽ thu được những bản đồ đặc trưng khác nhau. Cuối cùng, ta kết hợp toàn bộ bản đồ đặc trưng này thành kết quả cuối cùng của tầng tích chập.

Tương tự như mạng nơ-ron thông thường, ta sử dụng một hàm kích hoạt (activate function) để có đầu ra dưới dạng phi tuyến. Trong trường hợp CNN, đầu ra của phép tích chập sẽ đi qua hàm kích hoạt nào đó ví dụ như hàm **ReLU** (rectified linear units).

Trong quá trình trượt kernel/filter trên dữ liệu đầu vào, ta sẽ quy định một **bước nhảy (stride)** với mỗi lần di chuyển. Thông thường ta lựa chọn thường chọn bước nhảy là 1. Nếu kích thước bước nhảy tăng, kernel/filter sẽ có ít ô trùng lắp.

Bởi vì kích thước đầu ra luôn nhỏ hơn đầu vào nên ta cần một phép xử lí đầu vào để đầu ra không bị co giãn. Đơn giản ta chỉ cần thêm một **lề nhỏ (Padding)** vào đầu vào. Một lề với giá trị **0 (Zero Padding)** sẽ được thêm vào xung quanh đầu vào trước khi thực hiện phép tích chập.

Thông thường, sau mỗi tầng tích chập, ta sẽ cho kết quả đi qua **một tầng hợp nhất (pooling layer)**. Mục đích của tầng này là để nhanh chóng giảm số chiều. Việc này giúp giảm thời gian học và hạn chế việc **overfitting**.

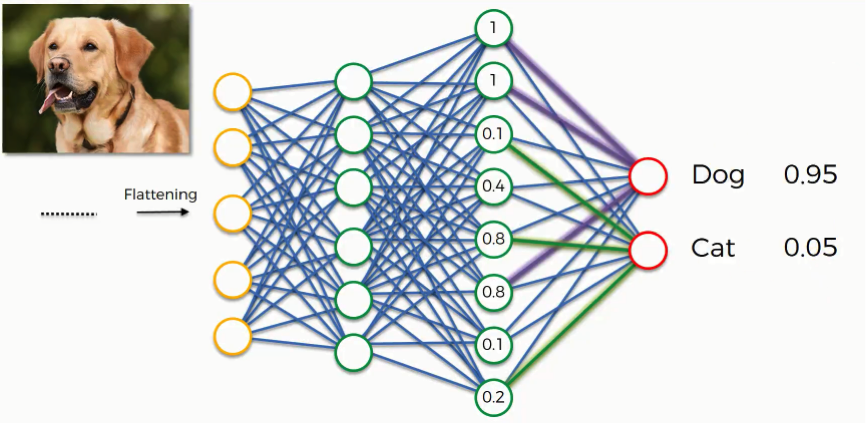
Một phép hợp nhất đơn giản thường được dùng đó là **max pooling**, phép này lấy giá trị lớn nhất của một vùng để đại diện cho vùng đó. Ngoài ra còn có **average pooling** và **global average pooling**. Kích thước của vùng sẽ được xác định trước để giảm kích thước của bản đồ đặc trưng nhanh chóng nhưng vẫn giữ được thông tin cần thiết.



**Phân lớp**

Trong phần phân lớp, ta sử dụng một vài tầng với kết nối đầy đủ (Fully-connected) để xử lí kết quả của phần tích chập. Vì đầu vào của mạng liên kết đầy đủ là 1 chiều, ta cần làm phẳng đầu vào trước khi phân lớp. Tầng cuối cùng trong mạng CNN là một tầng liên kết đầy đủ, phần này hoạt động tương tự như mạng nơ-ron thông thường.

Kết quả thu được cuối cùng cũng sẽ là một véc-tơ với các giá trị xác suất cho việc dự đoán như mạng nơ-ron thông thường.

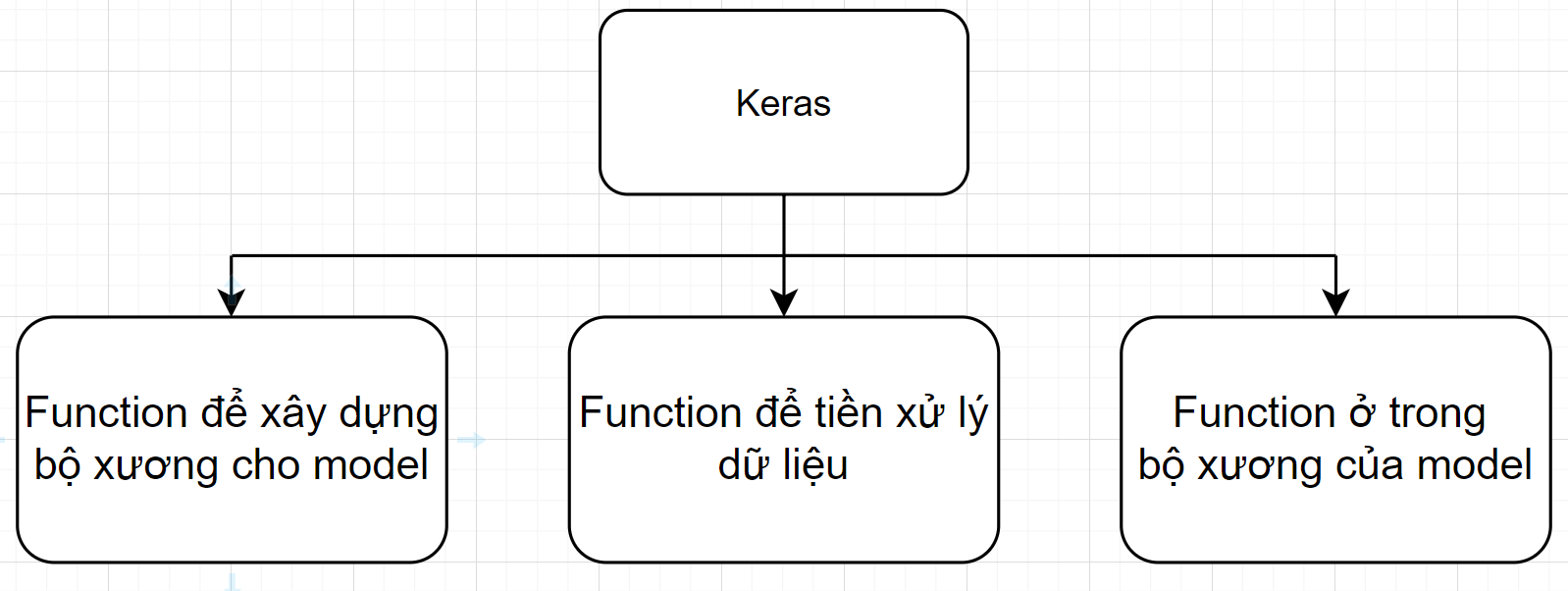


## Framework hỗ trợ

**Sklearn**: sử dụng cho việc phân chia dataset ngẫu nhiên theo phần trăm phục vụ cho quá trình train model. Thông qua *sklearn.train\_test-split*

**Keras**: sử dụng cho train model, tiền xử lý hình ảnh(dữ liệu).

Cấu trúc tổng quan của *keras*, và các api được sử dụng:



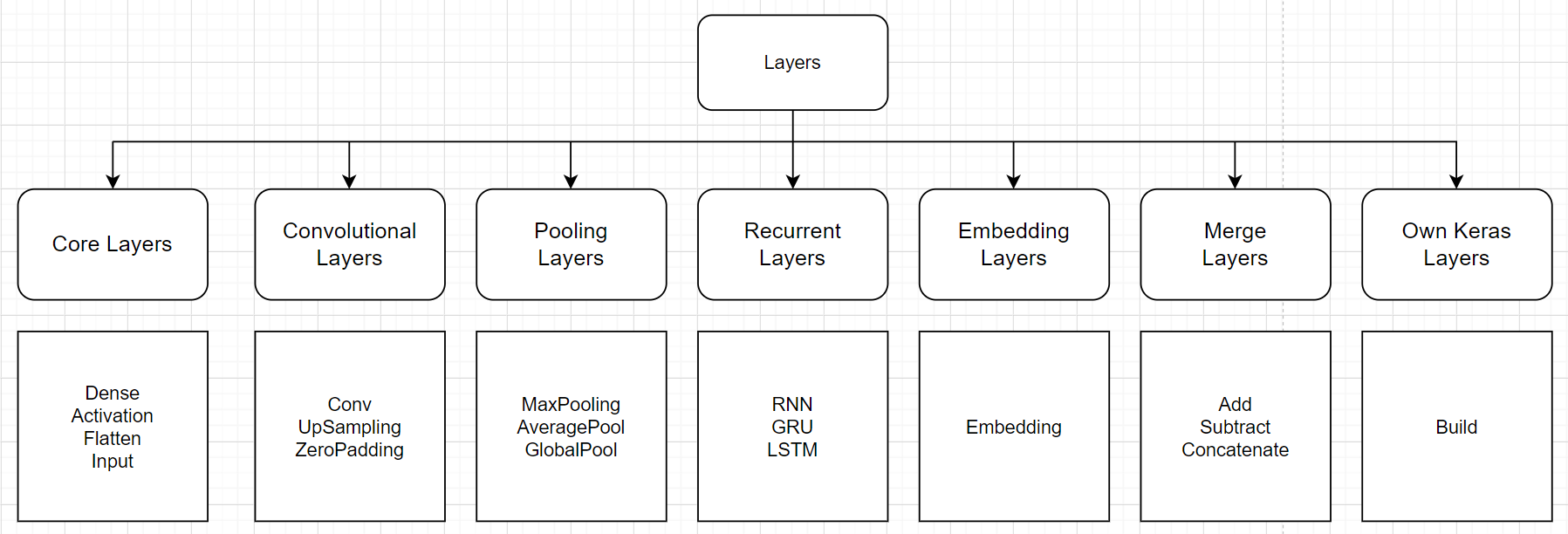
**Các module dùng cho xây dựng model trong phần sub-model của keras:**

Khởi tạo model trong keras: Thông qua api Sequential. Chúng ta khởi tạo model bằng funtion *Sequential()* sau đó dùng method add để thêm các layer.

Khi chúng ta khởi tạo một model thì có các method ta cần lưu ý là:

* *compile*: Sau khi build model xong dùng compile để biên tập lại toàn bộ model đã build. Ở đây có một số tham số để training model như: thuật toán training thông qua tham số *optimizer*, function loss của model chúng ta có thể sử dụng mặc định hoặc tự build thông qua tham số *loss*, chọn metrics hiện thị khi model được training.
* *summary* method này giúp chúng ta tổng hợp lại model xem model có bao nhiêu layer, tổng số tham số bao nhiêu,shape của mỗi layer.
* *fit* dùng để đưa data vào training để tìm tham số model(tương tự như sklearn).
* *predict* dùng để predict các new instance.
* *history* dùng để xem accuracy,loss qua từng epochs .

Tiếp theo chúng ta tìm hiểu đên sub-module Layers : Nó chứa các layers chuyên dụng để ta build các model như CNN,RNN,GANs..Có rất nhiều layers nên ta chỉ quan tâm đến một số layer thường sử dụng.



**Các API trong phần sub-model Layers của keras được sử dụng:**

* Core layer : chứa các layer mà hầu như model nào cũng sử dụng đến nó.
* *Activation* dùng để chọn activation trong layer(có thể dùng tham số activation thay thế).
  + - *linear* như trong linear regression
    - *softmax* dùng trong multi classifier
    - *relu* max(0,x) dùng trong các layer cnn,rnn để giảm chi phí tính toán
    - *tanh* range (-1,1)
    - *sigmoid* range (0,1) dùng nhiều trong binary class
* *Dropout* layer này dùng như regularization cho các layer hạn chế overfiting. Tham số cần chú ý :
  + - *rate* tỉ lệ dropout
    - *noise\_shape* cái này chưa tìm hiểu
    - *seed* random seed bình thường
* *Flatten* dùng để lát phằng layer để fully connection, vd : shape : 20x20 qua layer này sẽ là 400x1
* *Input* layer này sử dụng input như 1 layer như ví dụ trước ta đã nói.
* Convolutional Layers: chứa các layer trong mạng nơ ron tích chập
* *Conv2D* là convolution layer dùng để lấy feature từ image. tham số cần chú ý:
  + - *filters* số filter của convolution layer
    - *kernel\_size* size window search trên image
    - *strides* bước nhảy mỗi window search
    - *padding* same là dùng padding,valid là ko
    - *data\_format* format channel ở đầu hay cuối
* *UpSampling1D*,*UpSampling2D* Ngược lại với convolution layer
  + - *size* vd (2,2) có nghĩa mỗi pixel ban đầu sẽ thành 4 pixel
* *ZeroPadding1D*,*ZeroPadding2D* dùng để padding trên image.
  + - *padding* số pixel padding
* Pooling Layers : Chứa các layer dùng trong mạng CNN.
* *MaxPooling2D* dùng để lấy feature nổi bật (dùng max) và giúp giảm parameter khi training
* *AveragePooling2D* giống như maxpooling nhưng dùng Average
* *GlobalMaxPooling2D* chưa dùng bao giờ nên chưa hiểu nó làm gì

**Tiền xử lý dữ liệu:**

* Sequence Preprocessing
* Text Preprocessing
* Image Preprocessing
* ImageDataGenerator tạo thêm data bằng cách scale,rotation…

**Opencv:**

* Khởi tạo đối tượng: *cv2.CascadeClassifier():* dùng để phát hiện đặc điểm trên khuôn mặt thông qua bộ dữ liệu *haarcascade\_frontalface\_default.xml.*
* Bước 1: Hỗ trợ lấy hình ảnh realtime từ video thông qua API *cv2.Videocapture()*
* Bước 2: Resize hình ảnh lấy được từ camera thông qua bước 1.
* Bước 3:Chuyển ảnh lấy từ bước 2 qua ảnh xám thông qua API *cv2.cvtcolor()*
* Bước 4: trích xuất đặc điểm trên khuôn mặt trong hình xám lấy từ bước 3 dựa vào đối tượng chứa các đặc điệm trên khuôn mặt, nghĩa là bước này sẽ trích xuất ra khuôn mặt mà camera đang ghi hình.

**Numpy**: hỗ trợ tính toán trên ma trận.

**Imutils**:resize kích thước hình ảnh.

## Mô hình sử dụng

# Demo

## Giới thiệu

Công nghệ sử dụng trong web :

Programing Languages: python (version 3.6)

Frontend: html + css + javascript + jquery + boostrap

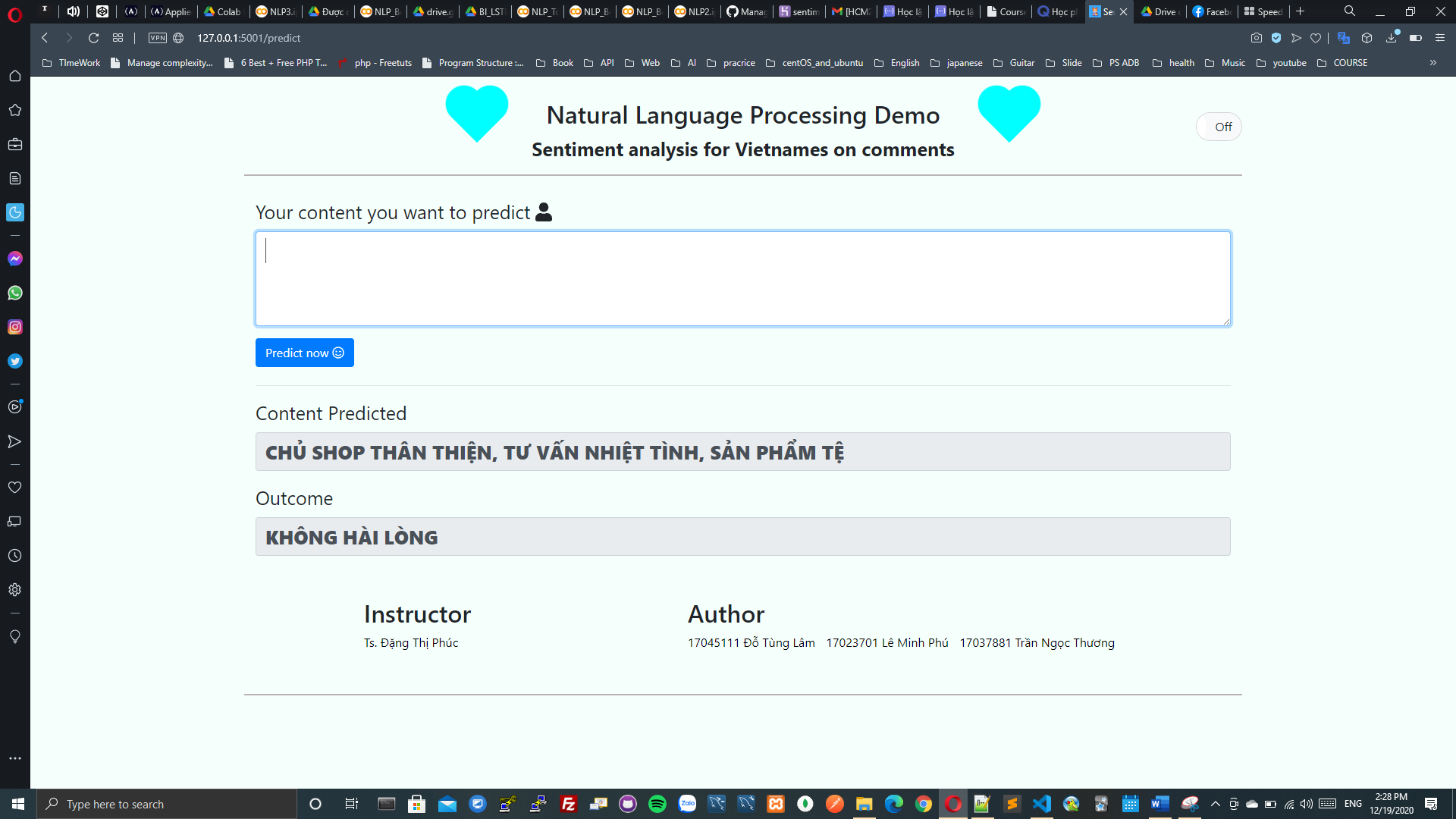
Backend: flask

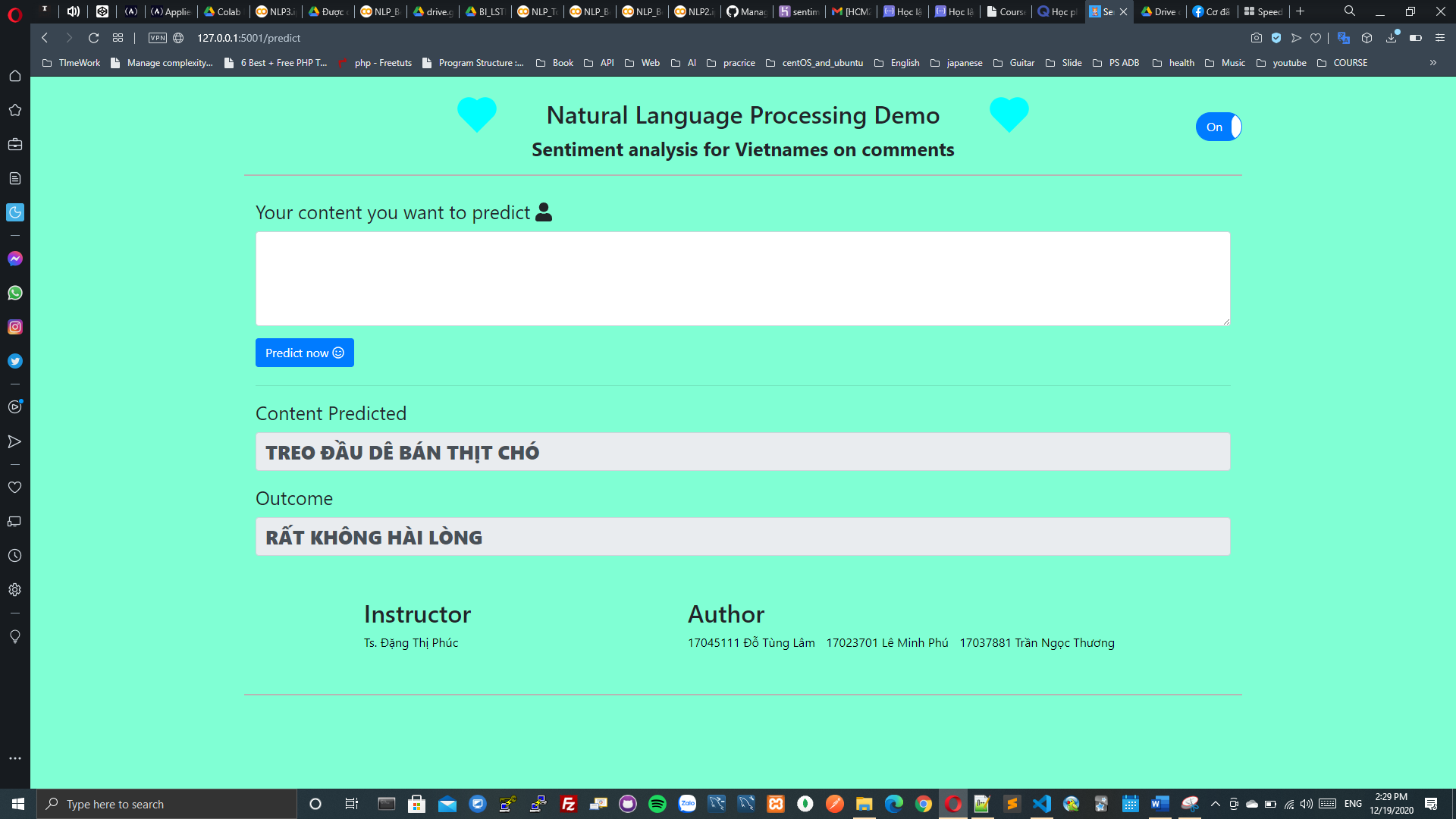
2: Sản phẩm

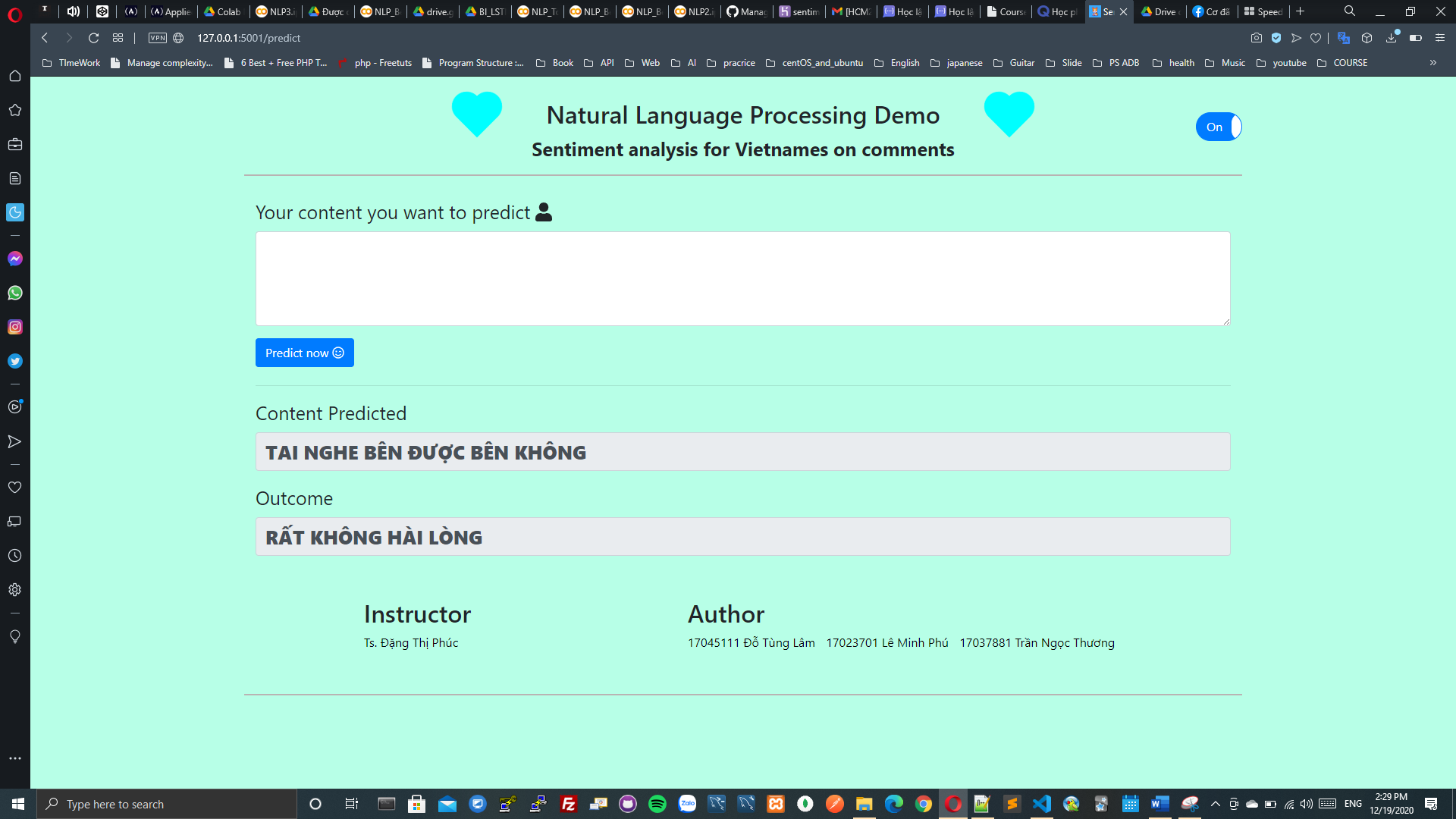
Name: Hero Predict

(để sử dụng được wen thì làm théo các bước trong file hướng dẫn readme.txt trong folder này)

3 một số hình ảnh dự thực tế của web







# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | S. Phuong, “Tổng quan đơn giản về Mạng nơ-ron tích chập (Convolutional Neural Networks),” 2018. [Trực tuyến]. Available: https://nhdp.net/blog/2018/11/tong-quan-don-gian-ve-mang-no-ron-tich-chap-convolutional-neural-networks/. [Đã truy cập 12 October 2020]. |
| [2] | Octavio Arriaga, Paul G. Ploger, Matias Valdenegro, "Real-time Convolutional Neural Networks for Emotion and Gender Classification," 2017. |
| [3] | O. Ayman, "Emotion-recognition," [Online]. Available: https://github.com/omar178/Emotion-recognition. [Accessed 12 10 2020]. |

Đã chỉnh sửa