NHẬN DIỆN CẢM XÚC KHUÔN MẶT

1. **Đặt vấn đề**

* Với bài toán nhận diện cảm xúc khuôn mặt, có 2 bước chính cần xử lý:
  + Bước một là xác định vị trí khuôn mặt trong webcam và xử lý hình ảnh khuôn mặt trước khi đưa vào nhận diện (xử lý input).
  + Bước 2 là cho ảnh (input) qua mạng neural đã được train để nhận được output là cảm xúc khuôn mặt.
* Với bài toán như vậy, các bước giải quyết của nhóm đặt ra là:
  + Xác định các loại cảm xúc trên khuôn mặt. (đã xong)
  + Thu thập bộ dữ liệu hình ảnh khuôn mặt đã được gắn nhãn (đã xong, tuy nhiên dữ liệu đa số là hình ảnh người châu Âu, đang xây dựng bộ dữ liệu thêm với hình ảnh của chính mình).
  + Xây dựng model CNN và train. (đã xây dựng xong model CNN, đang training cho model trên Google Collaboratory)
  + Nhận diện cảm xúc khuôn mặt trực tiếp trên webcam.

**II. Thực hiện**

1. Xác định các loại cảm xúc trên khuôn mặt

Theo một nghiên cứu thì có chủ yếu 7 loại cảm xúc khuôn mặt. (0=Tức giận, 1=Chán nản, 2=Lo sợ, 3=Vui vẻ, 4=Buồn, 5=Ngạc nhiên, 6=Bình thường).



Vì thế sẽ có 7 phân lớp cần gắn nhãn cho mỗi bức ảnh khuôn mặt.

2. Thu thập bộ dữ liệu hình ảnh khuôn mặt đã được gắn nhãn

Trên trang Kaggle đã có sẵn bộ dữ liệu cảm xúc khuôn mặt phù hợp với 7 cảm xúc nêu trên. Bộ dữ liệu gồm hơn 30000 bức ảnh khuôn mặt với độ phân giải 48x48 được lưu dưới định dạng csv. Mỗi record dữ liệu bao gồm 2 cột, cột “cảm xúc” với giá trị từ 0-6 tương ứng với từng loại cảm xúc. Cột “pixel” bao gồm giá trị pixel của từng điểm ảnh của bức ảnh.

Ngoài ra vì các bức ảnh hầu hết là hình ảnh người châu Âu, nên để thực hiện tốt hơn, nhóm đã xây dựng một công cụ để tự thu thập dữ liệu ảnh khuôn mặt. Công cụ xây dựng bằng OpenCV và bộ phân lớp HaarCasade như sau:

* Sử dụng OpenCV ghi hình trực tiếp từ camera của laptop.
* Sử dụng Haarcacasde xác định khuôn mặt trên video.
* Với mỗi khuôn mặt xác định được, chuyển sang dạng gray và resize bức ảnh xuống 48x48.
* Lưu lại dữ liệu bức ảnh đó.



Ví dụ để tạo ảnh với cảm xúc “Vui vẻ” thì yêu cầu mỗi người trong ảnh “cười”, sau đó tự động gắn nhãn các ảnh trên là “Vui vẻ”. Cách này có thể tạo ra dữ liệu nhanh, tuy nhiên trong một số trường hợp có thể bộ phân lớp Haarcacasde nhận diện khuôn mặt sai, vì vậy nhóm vẫn đang tối ưu thêm.

3. Nhận diện khuôn mặt trên hình ảnh.

Bước này thực hiện giống như đã nêu ở trên. Sử dụng OpenCV với bộ phân lớp Haarcascade đã được xây dựng sẵn để nhận diện khuôn mặt khá dễ dàng. Tuy nhiên vấn đề hiện tại là cần tối ưu các tham số để nhận diện chính xác hơn.

4. Xây dựng Mạng neural tích chập (Convolutional Neural Network).

* Mạng neural “myCNN” đã xây dựng:

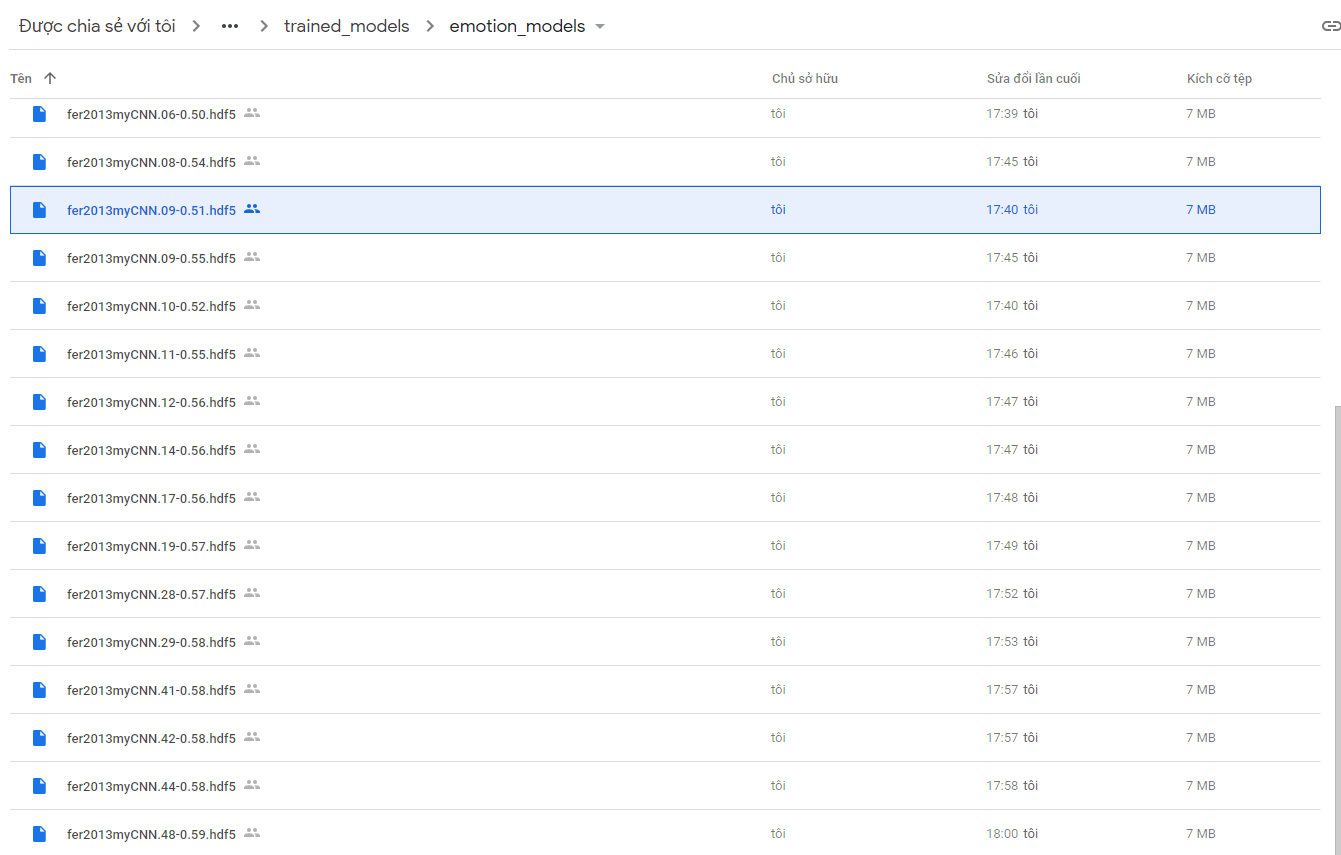
Layer (type) Output Shape Param #  
=================================================================  
image\_array (Conv2D) (None, 48, 48, 16) 800  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_1 (Batch (None, 48, 48, 16) 64  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_1 (Conv2D) (None, 48, 48, 16) 12560  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_2 (Batch (None, 48, 48, 16) 64  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
activation\_1 (Activation) (None, 48, 48, 16) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
average\_pooling2d\_1 (Average (None, 24, 24, 16) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dropout\_1 (Dropout) (None, 24, 24, 16) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_2 (Conv2D) (None, 24, 24, 32) 12832  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_3 (Batch (None, 24, 24, 32) 128  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_3 (Conv2D) (None, 24, 24, 32) 25632  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_4 (Batch (None, 24, 24, 32) 128  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
activation\_2 (Activation) (None, 24, 24, 32) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
average\_pooling2d\_2 (Average (None, 12, 12, 32) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dropout\_2 (Dropout) (None, 12, 12, 32) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_4 (Conv2D) (None, 12, 12, 64) 18496  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_5 (Batch (None, 12, 12, 64) 256  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_5 (Conv2D) (None, 12, 12, 64) 36928  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_6 (Batch (None, 12, 12, 64) 256  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
activation\_3 (Activation) (None, 12, 12, 64) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
average\_pooling2d\_3 (Average (None, 6, 6, 64) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dropout\_3 (Dropout) (None, 6, 6, 64) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_6 (Conv2D) (None, 6, 6, 128) 73856  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_7 (Batch (None, 6, 6, 128) 512  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_7 (Conv2D) (None, 6, 6, 128) 147584  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_8 (Batch (None, 6, 6, 128) 512  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
activation\_4 (Activation) (None, 6, 6, 128) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
average\_pooling2d\_4 (Average (None, 3, 3, 128) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dropout\_4 (Dropout) (None, 3, 3, 128) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_8 (Conv2D) (None, 3, 3, 256) 295168  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
batch\_normalization\_9 (Batch (None, 3, 3, 256) 1024  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
conv2d\_9 (Conv2D) (None, 3, 3, 7) 16135  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
global\_average\_pooling2d\_1 ( (None, 7) 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
predictions (Activation) (None, 7) 0  
=================================================================  
Tổng số tham số: 642,935  
Tổng số tham số có thể huấn luyện: 641,463  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

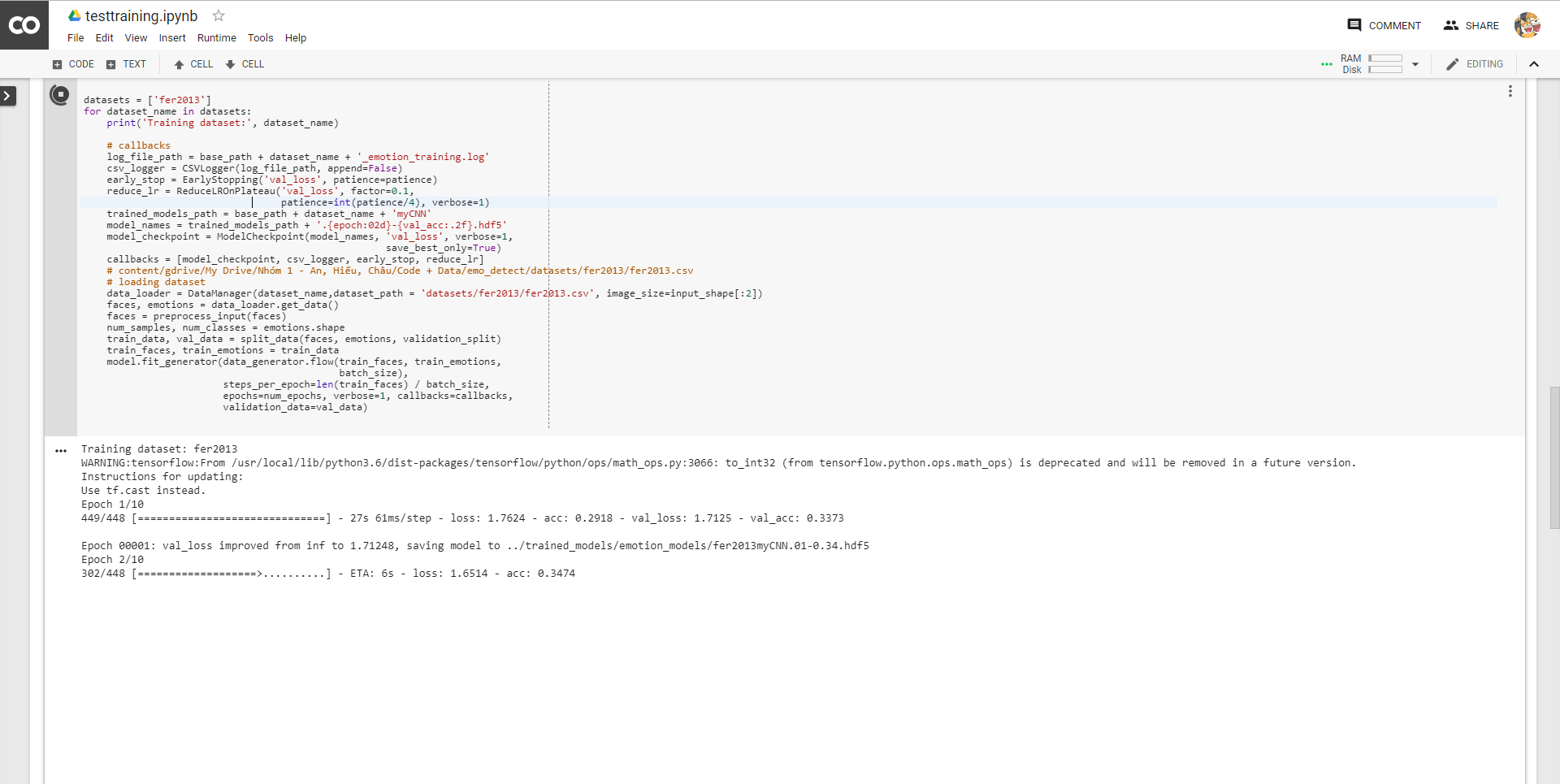
* Có thể tiếp theo sẽ sử dụng thử mạng LeNet hoặc mạng VGG\_s xem kết quả sau khi training có tốt hơn không.

5. Huấn luyện cho model đã xây dựng.

Nhóm đã thử tự cài đặt các công cụ cần thiết để Training trên máy của mình. Tuy nhiên GPU laptop khá yếu nên không thể training nhanh, đồng thời nhiệt độ máy lên >90 độ C nên đã dừng việc training và đang cố gắng chuyển dữ liệu lên Google Collaboration để training.

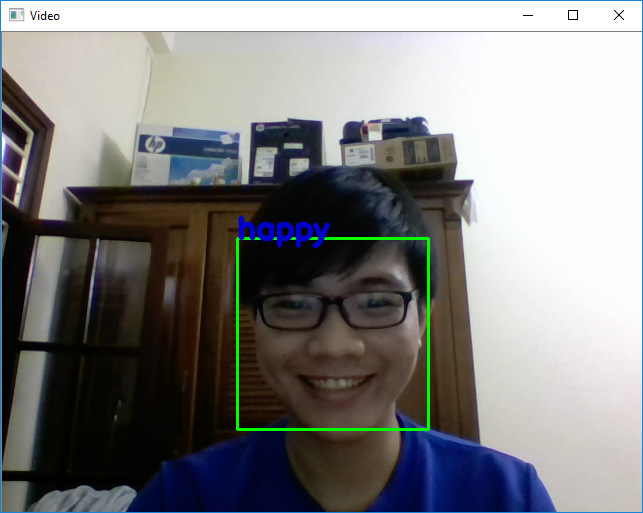
Mỗi lần training sẽ tự đồng bộ dữ liệu lên google drive khi accuracy được cải thiện so với epoch trước đó.. File lưu với dạng: fer2013<tên mạng>.<epoch>-<accuracy>.hdf5

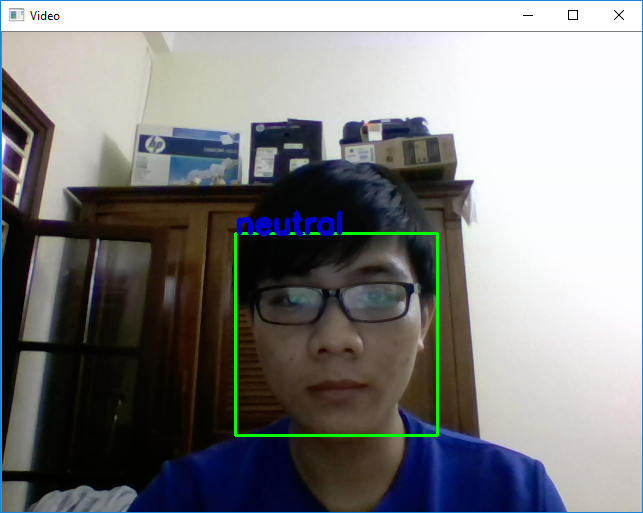




\*Kết quả chương trình tạm thời

* Max accuracy = 0.6016





III. Các công cụ & thư viện sử dụng:

* Anaconda (để xây dựng và quản lý environment python)
* Keras (với backend là Tensorflow).
* OpenCV 3.4.2
* Google Collaboratory.