TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

****

**MÔN HỌC: CƠ SỞ VÀ ỨNG DỤNG AI**

BÁO CÁO CUỐI KỲ

**NHẬN DIỆN CẢM XÚC KHUÔN MẶT**

**NGÀNH CÔNG NGHỆ KỸ THUẬT MÁY TÍNH**

Sinh viên: **PHAN CÔNG DANH**

MSSV: 19119160

**VÕ MINH HẬU**

MSSV: 19119174

TP. HỒ CHÍ MINH – 05/2022

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

BÁO CÁO CUỐI KỲ

**NHẬN DIỆN CẢM XÚC KHUÔN MẶT**

**NGÀNH CÔNG NGHỆ KỸ THUẬT MÁY TÍNH**

Sinh viên: **PHAN CÔNG DANH**

MSSV: 19119160

**VÕ MINH HẬU**

MSSV: 19119174

Hướng dẫn**: TS. TRƯƠNG NGỌC SƠN**

TP. HỒ CHÍ MINH – 5/2022

MỤC LỤC

[DANH MỤC HÌNH iv](#_Toc104814164)

[DANG MỤC BẢNG v](#_Toc104814165)

[CÁC TỪ VIẾT TẮT vi](#_Toc104814166)

[BẢN PHÂN CÔNG NỘI DUNG THỰC HIỆN ĐỀ TÀI 1](#_Toc104814167)

[CHƯƠNG 1 GIỚI THIỆU CHI TIẾT MÔ HÌNH 2](#_Toc104814168)

[1.1 Mạng nơ-ron tích chập (Convolutional Neural Network) 2](#_Toc104814169)

[1.2 Cấu trúc mạng nơ-ron tích chập 2](#_Toc104814170)

[CHƯƠNG 2 THIẾT KẾ VÀ THỰC THI ỨNG DỤNG NHẬN DIỆN CẢM XÚC KHUÔN MẶT 6](#_Toc104814171)

[2.1 Giới thiệu và đặc điểm tập dữ liệu 6](#_Toc104814172)

[2.2 Quá trình tạo tập dữ liệu 7](#_Toc104814173)

[2.3 Lựa chọn thông số cho quá trình huấn luyện 8](#_Toc104814174)

[CHƯƠNG 3 KẾT QUẢ VÀ THẢO LUẬN 11](#_Toc104814175)

[3.1 Kết quả huấn luyện mạng nơ-ron 11](#_Toc104814176)

[3.2 Kết quả thực nghiệm 14](#_Toc104814177)

[TÀI LIỆU THAM KHẢO 17](#_Toc104814178)

[PHỤ LỤC 18](#_Toc104814179)

# DANH MỤC HÌNH

[Hình 1.1: Kiến trúc cơ bản của CNN. 2](#_Toc104814295)

[Hình 1.2: Phép tính tích chập. 3](#_Toc104814296)

[Hình 1.3: Feature map từ các kernel khác nhau. 3](#_Toc104814297)

[Hình 1.4: Đồ thị hàm ReLU. 4](#_Toc104814298)

[Hình 1.5: Hàm Max Pooling với bộ lọc 2x2 và bước trượt là 2. 4](#_Toc104814299)

[Hình 1.6: Hàm Average Pooling với bộ lọc 2x2 và bước trượt là 2. 4](#_Toc104814300)

[Hình 1.7: Chuyển ma trận 2 chiều sang dạng vec-tơ. 5](#_Toc104814301)

[Hình 2.1: Dữ liệu tập huấn luyện. 6](#_Toc104814232)

[Hình 2.2: Dữ liệu hình ảnh cảm xúc ngạc nhiên. 6](#_Toc104814233)

[Hình 3.1: Kết quả huấn luyện trong 10 epoch đầu tiên. 12](#_Toc104814202)

[Hình 3.2: Kết quả huấn luyện trong 5 epoch cuối cùng. 12](#_Toc104814203)

[Hình 3.3: Biểu đồ độ chính xác và hàm lỗi của tập huấn luyện và kiểm tra. 13](#_Toc104814204)

[Hình 3.4: Kết quả thực nghiệm mô hình nhận dạng cảm xúc. 16](#_Toc104814205)

# DANG MỤC BẢNG

[Bảng 1.1: Tập dữ liệu FER2013 7](#_Toc104813974)

# CÁC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **TỪ VIẾT TẮT** | **TÊN ĐẦY ĐỦ** |
| 1 | CNN | Convolutional Neural Network |
| 2 | FER | Facial Expression Recognition |
| 3 | ReLU | Rectified Linear Unit |

# BẢNG PHÂN CÔNG NỘI DUNG THỰC HIỆN ĐỀ TÀI

|  |  |  |
| --- | --- | --- |
| **STT** | **Nội dung** | **Người thực hiện** |
| 1 | Viết báo cáo chương 1 | Võ Minh hậu |
| 2 | Viết báo cáo chương 2 | Phan Công Danh |
| 3 | Viết báo cáo chương 3 | Võ Minh hậu |
| 4 | Xây dựng và huấn luyện mô hình | Võ Minh hậu  Phan Công Danh |
| 5 | Kiểm thử mô hình | Phan Công Danh |
| 6 | Thiết kế Power Point | Võ Minh Hậu  Phan Công Danh |

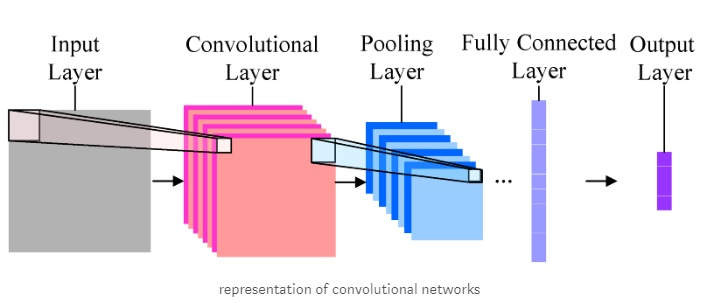
# CHƯƠNG 1 GIỚI THIỆU CHI TIẾT MÔ HÌNH

## 1.1 Mạng nơ-ron tích chập (Convolutional Neural Network)

Mạng nơ-ron tích chập (CNN) là một mạng nơ-ron được sử dụng để xử lý các dữ liệu 2 chiều. CNN được sử dụng nhiều trong lĩnh vực thị giác máy tính như nhận diện đối tượng. Nó là mạng nơ-ron sử dụng phép tính tích chập thay cho phép nhân thông thường, phải có ít nhất 1 lớp sử dụng phép tính tích chập trong CNN.

## 1.2 Cấu trúc mạng nơ-ron tích chập

Mạng nơ-ron tích chập là sự kết hợp của mạng nơ-ron với các lớp tích chập, lớp pooling và các lớp kết nối đầy đủ. Một mạng nơ-ron tích chập sẽ có cấu trúc cơ bản như hình 1.1.



Hình 1.1: Kiến trúc cơ bản của CNN.

- Lớp đầu vào (Input layer): lớp đầu vào của CNN thường là một ảnh 2 chiều, có kích thước nhỏ. Ví dụ: một ảnh có kích thước là 48x48 pixels với 3 kênh màu xanh lục, xanh dương, đỏ sẽ có kích thước cho lớp ngõ vào là 48x48x3.

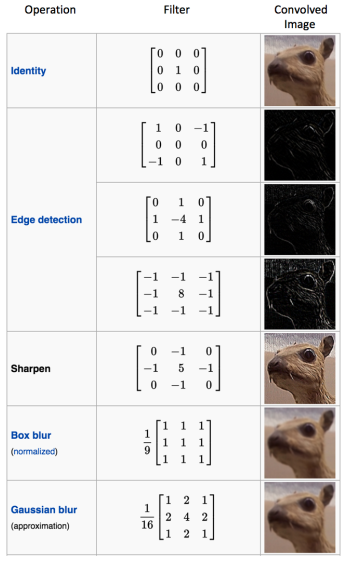
- Lớp tích chập (Convolution layer): lớp này sẽ thực hiện phép nhân tích chập giữa ảnh đầu vào (ở dạng ma trận) với các bộ lọc (filter) hoặc kernel cũng là một dạng ma trận. Ứng với mỗi bộ lọc (hoặc kernel) sẽ có tương ứng 1 ngõ ra được gọi là biểu đồ đặt điểm ảnh (feature map). Hình 1.2 là ví dụ cụ thể cho chức năng của lớp tích chập.

Ảnh có chứa văn bản, đồng hồ

Mô tả được tạo tự động

Hình 1.2: Phép tính tích chập.

Ví dụ trên có ảnh đầu là 5x5x1 và lớp filter là 3x3x1, nhân tích chập chúng với nhau sẽ tạo ra một feature map có kích thước 3x3x1. Tùy vào loại bộ lọc (hoặc kernel) sẽ tạo ra các feature map có đặc điểm khác nhau như hình 1.3.



Hình 1.3: Feature map từ các kernel khác nhau.

Các bộ lọc có các thông số như sau: depth (độ sâu, số lượng bộ lọc), filter size (kích thước bộ lọc), stride (bước trượt, quyết định số điểm ảnh thay đổi sau mỗi lần trượt của bộ lọc), zero-padding (kích thước lớp viền ngoài của ảnh, ban đầu là bằng 0).

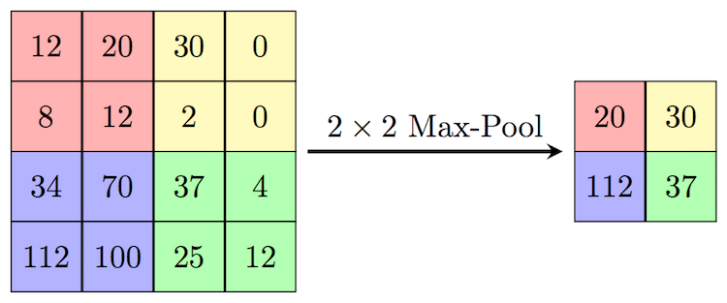
- Lớp hàm kích hoạt (Activation layer): lớp hàm kích hoạt trong CNN thường sử dụng hàm ReLU để chuẩn hóa các Feature map, vì hàm ReLU có khả năng lọc đi các giá trị âm nên nó cho ra kết quả huấn luyện nhanh hơn các hàm khác như Sigmoid, Tanh. Đồ thị hàm ReLU được mô tả trong hình 1.4.



Hình 1.4: Đồ thị hàm ReLU.

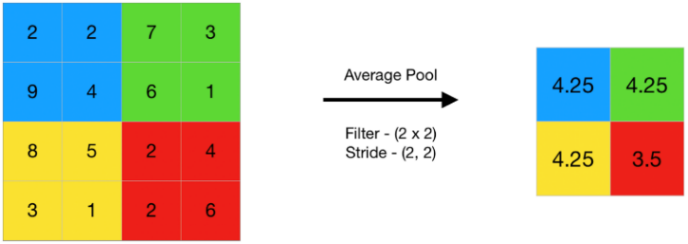
- Lớp gộp (Pooling layer): Lớp gộp sẽ giảm bớt số lượng tham số của các feature map nhưng vẫn giữ được các thông tin quan trọng của nó. Các hàm Pooling thường được sử dụng trong mạng nơ-ron tích chập là: Max Pooling, Average pooling.

Hàm Max Pooling sẽ lấy giá trị phần tử lớn nhất trong một bộ lọc mà không quan tâm đến vị trí của giá trị cao nhất đó nằm ở đâu. Vì thế khi ảnh dịch chuyển nhẹ ngõ ra cũng sẽ không thay đổi, ví dụ như hình 1.5 bên dưới.



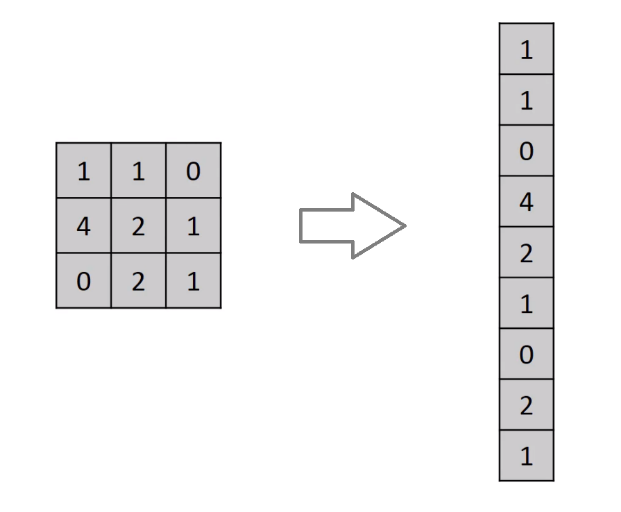
Hình 1.5: Hàm Max Pooling với bộ lọc 2x2 và bước trượt là 2.

Hàm Average Pooling sẽ lấy giá trị trung bình của các phần tử trong một bộ lọc, ví dụ như hình 1.6 bên dưới.



Hình 1.6: Hàm Average Pooling với bộ lọc 2x2 và bước trượt là 2.

- Lớp kết nối đầy đủ (Fully connected layer): lớp kết nối đầy đủ thường được sử dụng để làm phổ biến các tổ hợp phi tuyến từ ma trận tích chập đầu ra. Ngõ vào của nó là các vec-tơ một chiều. Dó đó, ngõ vào của lớp kết nối đầy đủ phải được chuyển thành vec-tơ.



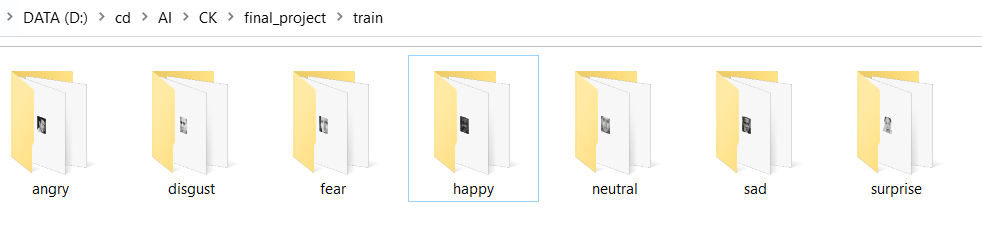
Hình 1.7: Chuyển ma trận 2 chiều sang dạng vec-tơ.

# CHƯƠNG 2 THIẾT KẾ VÀ THỰC THI ỨNG DỤNG NHẬN DIỆN CẢM XÚC KHUÔN MẶT

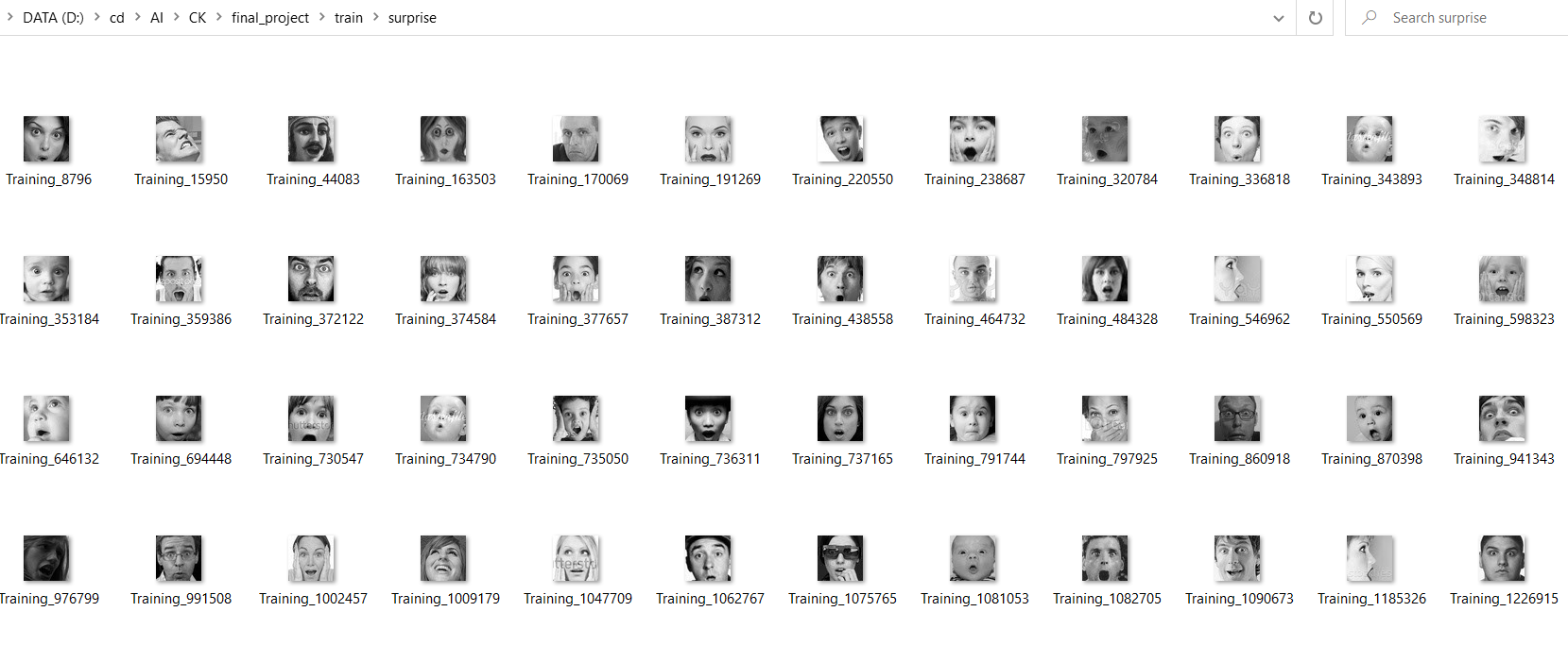
## 2.1 Giới thiệu và đặc điểm tập dữ liệu

Trong đề tài này, nhóm chúng tôi sử dụng tập dữ liệu FER2013 (Facial Expression Recognition 2013 Dataset), được giới thiệu trong hội thảo ICML 2013 về những thách thức trong Representation Learning. Bộ dữ liệu này rất phổ biến với 35,887 ảnh khuôn mặt đều có kích thước là 48x48 pixels và 1 kênh màu (ảnh xám), dữ liệu được thu thập bởi API tìm kiếm hình ảnh của Google. Cụ thể bộ dữ liệu bao gồm 28,709 ảnh trong tập huấn luyện và 7,178 ảnh trong tập test và mỗi hình ảnh được dán nhãn thuộc 1 trong 7 loại cảm xúc khác nhau: Angry (tức giận), Disgust (chán ghét), Fear (sợ hãi), Happy (vui), Sad (buồn), Surprise (ngạc nhiên) và Neutral (bình thường).

Hình 2.1 bên dưới là thư mục train bao gồm các thư mục con chứa dữ liệu hình ảnh của 7 cảm xúc đã liệt kê ở trên và tương tự như vậy với thư mục test. Cụ thể ví dụ ở hình 2.2 là dữ liệu hình ảnh được dán nhãn cảm xúc ngạc nhiên trong tập huấn luyện.



Hình 2.1: Dữ liệu tập huấn luyện.



Hình 2.2: Dữ liệu hình ảnh cảm xúc ngạc nhiên.

Để nắm rõ chi tiết hơn về tập dữ liệu FER2013, số lượng dữ liệu hình ảnh sẽ được thống kê qua bảng dưới đây:

Bảng 1.1: Tập dữ liệu FER2013

|  |  |  |
| --- | --- | --- |
|  | Train | Test |
| Angry | 3995 | 958 |
| Disgust | 436 | 111 |
| Fear | 4097 | 1024 |
| Happy | 7215 | 1774 |
| Neutral | 4965 | 1233 |
| Sad | 4830 | 1247 |
| Surprise | 3171 | 831 |
| Total | 28709 | 7178 |

## 2.2 Quá trình tạo tập dữ liệu

Trước khi tạo lập dữ liệu cho mạng CNN cần phải sử dụng một số thư viện để đơn giản hóa việc xây dựng mô hình và quá trình tạo lập dữ liệu. Ở đề tài này, nhóm sử dụng thư viện Keras giúp rút ngắn thời gian xây dựng cũng như kiểm thử các mạng học sâu. Thư viện Keras được sử dụng như sau:

|  |
| --- |
| from keras.models import Sequential  from keras.layers import Dense, Dropout, Flatten  from keras.layers import Conv2D  from keras.optimizers import Adam  from keras.layers import MaxPooling2D  from keras.preprocessing.image import ImageDataGenerator  import matplotlib.pyplot as plt |

Như đã trình bày ở phần trên, tập dữ liệu FER2013 được phân ra làm 2 tập con ở trong 2 thư mục train và test. Mỗi điểm ảnh trong ảnh được biểu diễn bằng một giá trị nhị phân 8 bit (0- 255), giá trị này quá lớn đối với các trọng số được khởi tạo trong mạng CNN. Vì vậy cần phải chuẩn hóa dữ liệu ảnh về một giá trị nhỏ hơn trong khoảng [0,1]. Đồng thời bằng cách lật hình ảnh theo chiều ngang có thể tăng gấp đôi số lượng ảnh huấn luyện, giúp tránh hiện tượng quá khớp.

|  |
| --- |
| train\_datagen = ImageDataGenerator(rescale=1./255, horizontal\_flip=True)  val\_datagen = ImageDataGenerator(rescale=1./255, horizontal\_flip=True) |

Quá trình thực hiện tải dữ liệu vào mô hình mạng CNN như sau:

|  |
| --- |
| train\_dir = 'train'  val\_dir = 'test'  train\_generator = train\_datagen.flow\_from\_directory(  train\_dir,  target\_size=(48,48),  batch\_size=64,  color\_mode="grayscale",  class\_mode='categorical')  validation\_generator = val\_datagen.flow\_from\_directory(  val\_dir,  target\_size=(48,48),  batch\_size=64,  color\_mode="grayscale",  class\_mode='categorical') |

Ở đây, train\_generator là giá trị của tập train với mỗi ảnh train có kích thước là 48x48x1 (ảnh xám), batch\_size = 64, giá trị ngõ ra mong muốn của hình ảnh đầu vào sẽ là tên thư mục chứa hình ảnh đó (happy, angry…). Tương tự validation\_generator là giá trị của tập test cũng với kích thước là 48x48x1 (ảnh xám), batch\_size = 64.

## 2.3 Lựa chọn thông số cho quá trình huấn luyện

Mô hình sử dụng kiến trúc mạng nơ-ron tích chập bao gồm 2 lớp chập (Conv2D) + 1 lớp gộp (MaxPooling2D) + 1 lớp chập + 1 lớp gộp + 1 lớp chập + 1 lớp gộp + 1 lớp chuyển từ ma trận sang vec-tơ (Flatten) và 2 lớp kết nối đầy đủ (Dense). Ở mỗi lớp, cần khai báo các thông số như số lượng, kích thước, hàm kích hoạt. Kỹ thuật Dropout được thực hiện giữa các lớp trong mạng nơ-ron nhằm tránh hiện tượng quá khớp.

|  |
| --- |
| emotion\_model = Sequential()  emotion\_model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(48,48,1)))  emotion\_model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu'))  emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))  emotion\_model.add(Dropout(0.25))  emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))  emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))  emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))  emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))  emotion\_model.add(Dropout(0.25))  emotion\_model.add(Flatten())  emotion\_model.add(Dense(1024, activation='relu'))  emotion\_model.add(Dropout(0.5))  emotion\_model.add(Dense(7, activation='softmax'))  emotion\_model.summary() |

Các lớp chập ở mô hình trên đều có kích thước kernel là 3x3 và hàm kích hoạt là ReLU, nhưng có số lượng kernel khác nhau (32, 64, 128). Sau mỗi lớp chập như thế, kích thước của ảnh lại giảm đi 2 pixels chiều dọc và chiều ngang. Các lớp Maxpooling có kích thước 2x2 và bước trượt là 2 sẽ tạo ra những feature map, đồng thời kích thước của ảnh ngõ vào lại bị giảm thêm một nửa sau mỗi lớp pooling, dữ liệu sau đó được chuyển từ ma trận ảnh sang vec-tơ để đưa vào ngõ ra có hàm kích hoạt là Softmax thông qua các lớp kết nối đầy đủ. Mỗi lớp mạng được tạo ra bằng phương thức add() với các thông số tùy theo mỗi lớp. Cấu trúc mạng sau khi khởi tạo được mô tả như sau:

|  |
| --- |
| Model: "sequential"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  ========================================================  conv2d (Conv2D) (None, 46, 46, 32) 320  conv2d\_1 (Conv2D) (None, 44, 44, 64) 18496  max\_pooling2d (MaxPooling2D) (None, 22, 22, 64) 0  dropout (Dropout) (None, 22, 22, 64) 0  conv2d\_2 (Conv2D) (None, 20, 20, 128) 73856  max\_pooling2d\_1 (MaxPooling 2D) (None, 10, 10, 128) 0  conv2d\_3 (Conv2D) (None, 8, 8, 128) 147584  max\_pooling2d\_2 (MaxPooling 2D) (None, 4, 4, 128) 0  dropout\_1 (Dropout) (None, 4, 4, 128) 0  flatten (Flatten) (None, 2048) 0  dense (Dense) (None, 1024) 2098176  dropout\_2 (Dropout) (None, 1024) 0  dense\_1 (Dense) (None, 7) 7175  ========================================================  Total params: 2,345,607  Trainable params: 2,345,607  Non-trainable params: 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

# CHƯƠNG 3 KẾT QUẢ VÀ THẢO LUẬN

## 3.1 Kết quả huấn luyện mạng nơ-ron

Ở mô hình này, hàm lỗi có sẵn trong thư viện Keras được sử dụng là categorical\_crossentropy, dùng để tính toán cross-entropy loss giữa các giá trị dự đoán và giá trị mong muốn. Lớp optimizer cho phép lựa chọn giải thuật cập nhật trọng số để tối ưu hàm mất mát, trong bài này nhóm sử dụng giải thuật Adam - một phương pháp giảm độ dốc ngẫu nhiên, tốc độ học được lựa chọn là 0.0001. Metrics dùng để thống kê danh sách các thông số mà mô hình đánh giá trong quá trình huấn luyện và kiểm tra.

|  |
| --- |
| emotion\_model.compile(loss='categorical\_crossentropy',optimizer= Adam(learning\_rate=0.0001, decay=1e-6),metrics=['accuracy']) |

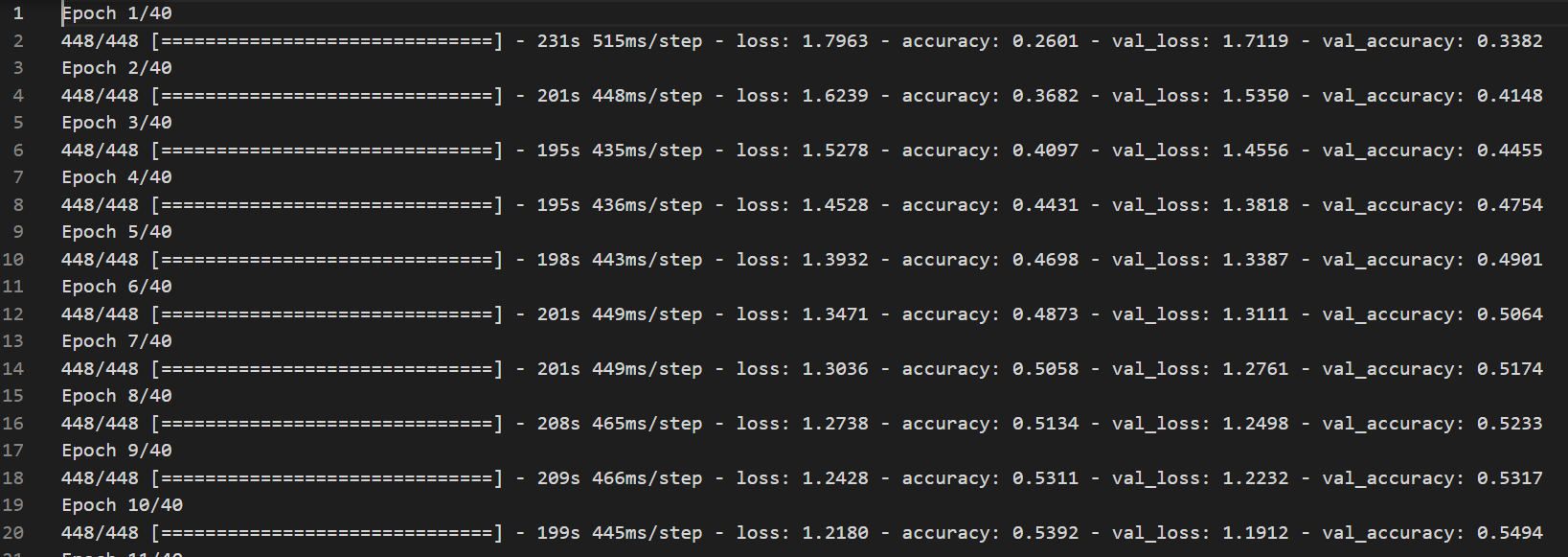
Sau đó tiến hành huấn huyện model với dữ liệu đầu vào là train\_generator (giá trị tập train), 40 epochs với số bước của mỗi epoch là 448. Dữ liệu dùng để tính toán các thông số đánh giá mô hình ở cuối mỗi epoch như độ chính xác và tỉ lệ lỗi là validation\_generator (giá trị tập test), số bước là 112. Ngoài ra, nhóm tiến hành xáo trộn dữ liệu huấn luyện một cách ngẫu nhiên nhằm cải thiện độ chính xác.

|  |
| --- |
| history = emotion\_model.fit(  train\_generator,  steps\_per\_epoch=28709 // 64,  epochs=40,  validation\_data=validation\_generator,  validation\_steps=7178 // 64,  shuffle=True,  ) |

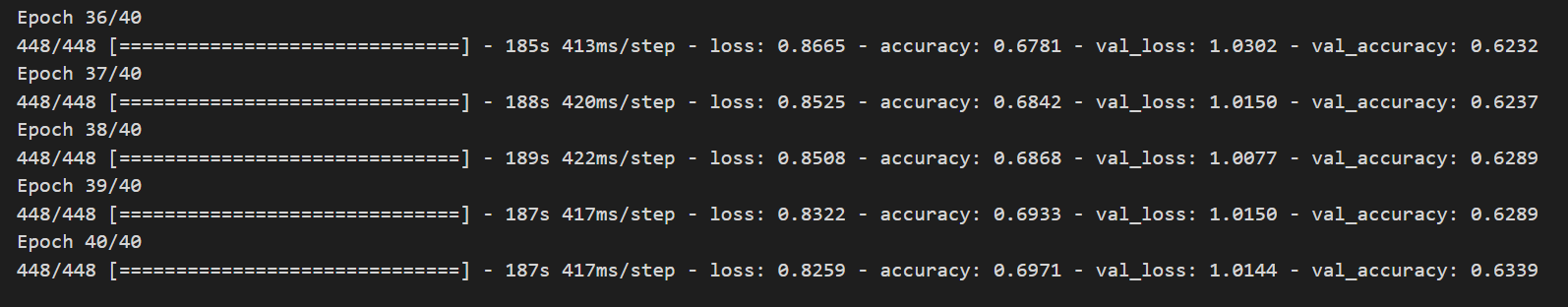
Sau khi huấn luyện xong, mô hình sẽ được lưu dưới dạng file .h5.

|  |
| --- |
| emotion\_model.save\_weights('model.h5') |

Kết quả huấn luyện được thể hiện thông qua hình 3.1 và hình 3.2 bên dưới:



Hình 3.1: Kết quả huấn luyện trong 10 epoch đầu tiên.

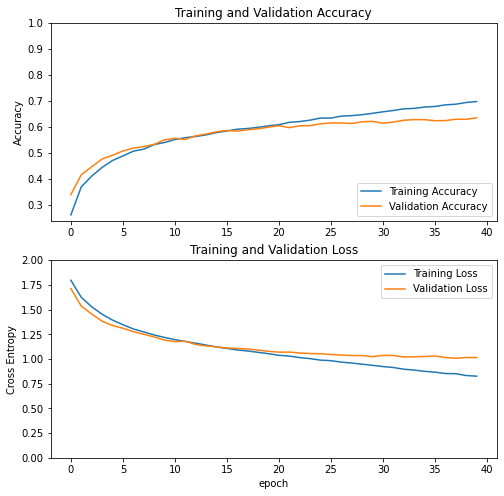


Hình 3.2: Kết quả huấn luyện trong 5 epoch cuối cùng.

Dùng thư viện Matplotlib để tạo các biểu đồ thể hiện kết quả quá trình huấn luyện:

|  |
| --- |
| acc = history.history['accuracy']  val\_acc = history.history['val\_accuracy']  loss = history.history['loss']  val\_loss = history.history['val\_loss']  plt.figure(figsize=(8, 8))  plt.subplot(2, 1, 1)  plt.plot(acc, label='Training Accuracy')  plt.plot(val\_acc, label='Validation Accuracy')  plt.legend(loc='lower right')  plt.ylabel('Accuracy')  plt.ylim([min(plt.ylim()),1])  plt.title('Training and Validation Accuracy')  plt.subplot(2, 1, 2)  plt.plot(loss, label='Training Loss')  plt.plot(val\_loss, label='Validation Loss')  plt.legend(loc='upper right')  plt.ylabel('Cross Entropy')  plt.ylim([0,2.0])  plt.title('Training and Validation Loss')  plt.xlabel('epoch')  plt.show() |

Kết quả được thể hiện ở dạng biểu đồ trong hình 3.3 bên dưới.



Hình 3.3: Biểu đồ độ chính xác và hàm lỗi của tập huấn luyện và kiểm tra.

Nhận xét: Độ chính xác ở những epoch đầu tăng khá nhanh, các epoch tiếp theo có tốc độ tăng chậm dần. Tương tự hàm lỗi giảm rất nhanh ở những epoch đầu và chậm dần về sau. Giá trị độ chính xác và hàm lỗi của tập kiểm tra có độ sai lệch không lớn so với tập huấn luyện (63% với 69%). Từ đó cho thấy mô hình hoạt động tốt ở cả hai quá trình huấn luyện và kiểm tra, không bị các hiện tượng không khớp và quá khớp.

## 3.2 Kết quả thực nghiệm

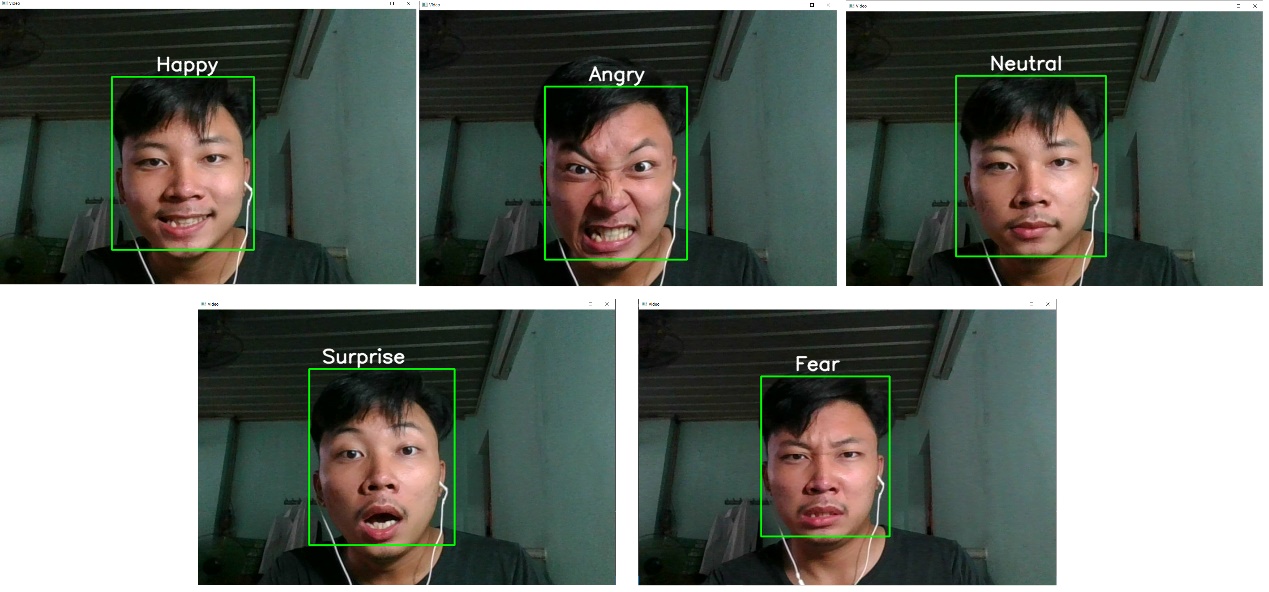
Sau khi huấn luyện xong mô hình, nhóm báo cáo thực hiện kiểm tra mô hình bằng việc sử dụng camera để nhận diện cảm xúc khuôn mặt. trước tiên, cần khai báo các thư viện, và tải các trọng số đã được huấn luyện vào mô hình để kiểm tra

|  |
| --- |
| from tkinter import \*  import cv2  import numpy as np  from keras.models import Sequential  from keras.layers import Dense, Dropout, Flatten  from keras.layers import Conv2D  from keras.layers import MaxPooling2D  emotion\_model = Sequential()  emotion\_model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(48, 48, 1)))  emotion\_model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu'))  emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))  emotion\_model.add(Dropout(0.25))  emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))  emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))  emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))  emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))  emotion\_model.add(Dropout(0.25))  emotion\_model.add(Flatten())  emotion\_model.add(Dense(1024, activation='relu'))  emotion\_model.add(Dropout(0.5))  emotion\_model.add(Dense(7, activation='softmax'))  emotion\_model.load\_weights('model.h5') |

Với sự hỗ trợ của mô-đun phát hiện đối tượng Haar Cascade của OpenCV, một mô-đun phát hiện khuôn mặt, giúp cho việc nhận dạng cảm xúc trở nên trực quan và chính xác hơn. Haar Cascade nhận biết được khuôn mặt người bằng cách dùng các bộ lọc để tìm các đặt điểm trên khôn mặt con người như lông mi, mũi, miệng… Vùng khoanh này sẽ được chụp lại thành hình ảnh và chuẩn hóa về kích thước 48x48x1 (ảnh xám) để có thể đưa vào mô hình nhận dạng cảm xúc, tránh lấy những hình ảnh thừa làm giảm độ chính xác của mô hình.

|  |
| --- |
| cv2.ocl.setUseOpenCL(False)  emotion\_dict = {0: " Angry ", 1: "Disgust", 2: " Fear ", 3: " Happy ", 4: " Neutral ",5: " Sad ", 6: "Surprise"}  cap = cv2.VideoCapture(0)  while True:  ret, frame = cap.read()  frame = cv2.flip(frame, 1)  if not ret:  break  bounding\_box = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')  gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  num\_faces=bounding\_box.detectMultiScale(gray\_frame,scaleFactor=1.3, minNeighbors=5)  for (x, y, w, h) in num\_faces:  cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (0, 255, 0), 2)  roi\_gray\_frame = gray\_frame[y:y + h, x:x + w]  cropped\_img = np.expand\_dims(np.expand\_dims(cv2.resize(roi\_gray\_frame, (48, 48)), -1), 0)  emotion\_prediction = emotion\_model.predict(cropped\_img)  maxindex = int(np.argmax(emotion\_prediction))  cv2.putText(frame, emotion\_dict[maxindex],  (x+20, y-60), cv2.FONT\_HERSHEY\_SIMPLEX, 1,  (255, 255, 255), 2, cv2.LINE\_AA)  cv2.imshow('Video', cv2.resize(frame,(1200,860),  interpolation = cv2.INTER\_CUBIC))  if cv2.waitKey(1) & 0xFF == ord('q'):  exit(0)  cap.release()  cv2.destroyAllWindows() |

Và đây là kết quả của quá trình thực nghiệm mô hình nhận cảm xúc khuôn mặt:



Hình 3.4: Kết quả thực nghiệm mô hình nhận dạng cảm xúc.

# TÀI LIỆU THAM KHẢO

1. TS. Trương Ngọc Sơn, 2020, *TRÍ TUỆ NHÂN TẠO CƠ SỞ VÀ ỨNG DỤNG,* Nxb. Đại học Quốc gia Thành phố Hồ Chí Minh, TP. HCM.
2. To Duc Thang, *Làm quen với Keras,* VIBLO, <https://viblo.asia/p/lam-quen-voi-keras-gGJ59mxJ5X2>, 30/5/2022.
3. Manas Sambare, *FER-2013,* kaggle, <https://www.kaggle.com/datasets/msambare/fer2013>, 24/5/2022.
4. Ducmanhkhtd, *NGHIÊN CỨU VÀ ỨNG DỤNG CÁC KỸ THUẬT NHẬN DẠNG CẢM XÚC QUA KHUÔN MẶT,* 2KVN, <https://2kvn.com/nghien-cuu-va-ung-dung-cac-ky-thuat-nhan-dang-cam-xuc-qua-khuon-mat-p5f31373133#:~:text=FER2013%20%C4%91%C6%B0%E1%BB%A3c%20gi%E1%BB%9Bi%20thi%E1%BB%87u%20trong,%C4%91%E1%BB%83%20h%E1%BB%8Dc%20khi%20hu%E1%BA%A5n%20luy%E1%BB%87n>, 26/5/2022.

# PHỤ LỤC

Link Video báo cáo: <https://youtu.be/17RDmzVhH2M>