

HỌC VIỆN NGÂN HÀNG
KHOA CÔNG NGHỆ THÔNG TIN & KINH TẾ SỐ



BÁO CÁO TRÍ TUỆ NHÂN TẠO
Đề tài: NHẬN DIỆN BIỂU CẢM KHUÔN MẶT

Giảng viên hướng dẫn: TS. Vũ Trọng Sinh

Nhóm lớp học phần: 232IS54A01

Nhóm sinh viên thực hiện: Nhóm 11

HÀ NỘI – 06/2024

HỌC VIỆN NGÂN HÀNG
KHOA CÔNG NGHỆ THÔNG TIN & KINH TẾ SỐ



BÁO CÁO TRÍ TUỆ NHÂN TẠO
Đề tài: NHẬN DIỆN BIỂU CẢM KHUÔN MẶT

Giảng viên hướng dẫn: TS. Vũ Trọng Sinh

Nhóm sinh viên thực hiện: Nhóm 11

STT	Họ tên	Mã sinh viên	Đóng góp (%)
1	Nguyễn Thế Nghĩa (NT)	24A4042602	25%
2	Phan Tiến Dũng	24A4042432	18%
3	Hà Thị Ngân	24A4042601	21%
4	Ngô Mạnh Thắng	24A4042611	18%
5	Trần Tiến Thịnh	24A4042613	18%

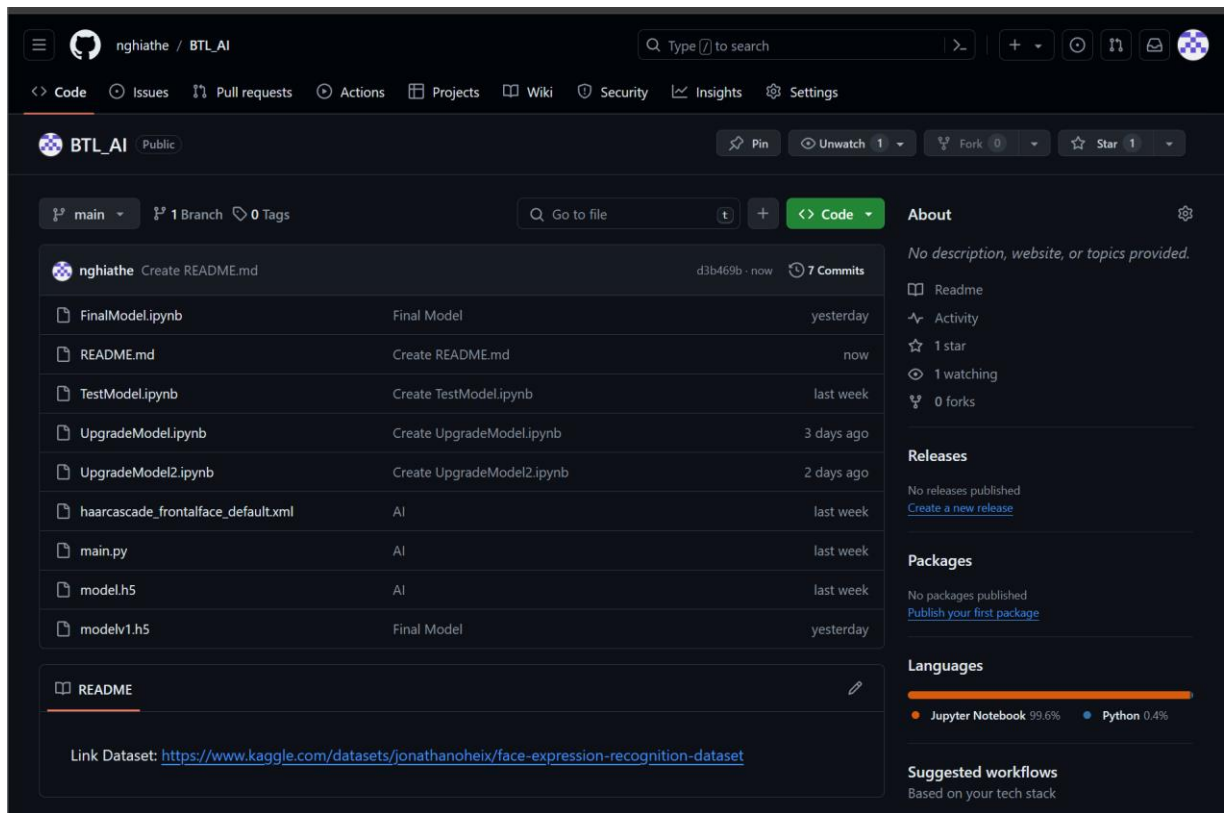
HÀ NỘI – 06/2024

PHÂN CÔNG NHIỆM VỤ

STT	Họ tên	Mã sinh viên	Phần đóng góp	Đóng góp (%)
1	Nguyễn Thế Nghĩa (NT)	24A4042602	Tổng hợp Code, Báo cáo, xây dựng, triển khai mô hình	25%
2	Phan Tiến Dũng	24A4042432	Tiền xử lý dữ liệu	18%
3	Hà Thị Ngân	24A4042601	Đánh giá mô hình	21%
4	Ngô Mạnh Thắng	24A4042611	Thu thập dữ liệu	18%
5	Trần Tiến Thịnh	24A4042613	Trích chọn đặc trưng	18%

HOẠT ĐỘNG NHÓM

1. Theo dõi tiến độ trên Github



2. GitHub của nhóm: https://github.com/nghiathe/BTL_AI

3. Dataset: <https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset>

MỤC LỤC

LỜI CAM ĐOAN	vi
LỜI CẢM ƠN	vii
DANH MỤC HÌNH ẢNH	viii
DANH MỤC TỪ VIẾT TẮT	ix
LỜI MỞ ĐẦU	1
NỘI DUNG	2
Chương 1. Giới thiệu đề tài.....	2
1.1 Đặt vấn đề.....	2
1.2 Cơ sở hình thành đề tài.....	3
1.3 Mục tiêu đề tài.....	3
1.4 Đối tượng và phương pháp nghiên cứu.....	4
1.5 Đối tượng sử dụng và ý nghĩa đề tài	5
1.6 Phạm vi dự án.....	5
1.7 Kết cấu bài báo cáo	6
Chương 2. Trí tuệ nhân tạo và tiền xử lý dữ liệu	7
2.1 Trí tuệ nhân tạo	7
2.1.1 Khái niệm.....	7
2.1.2 Các ứng dụng của AI.....	7
2.2 Nhập các thư viện và thiết lập môi trường.....	8
2.3 Tiền xử lý dữ liệu	9
2.3.1 Đọc và gán nhãn cho hình ảnh.....	10
2.3.2 Chia dữ liệu thành các tập huấn luyện, kiểm tra và xác thực	11
2.3.3 Tiền xử lý hình ảnh	11
2.3.4 Tăng cường dữ liệu	12
2.3.5 Mã hóa dữ liệu	13
2.4 Kỹ thuật trích chọn đặc trưng	15
Chương 3. Xây dựng và huấn luyện mô hình	17

3.1. Khái niệm CNN	17
3.1.1 Lớp tích chập - Convolution Layer.....	17
3.1.2 Lớp gộp - Pooling layer	18
3.1.3 Lớp được kết nối đầy đủ - Full Connected	19
3.2. Xây dựng mô hình.....	19
3.2.1. Lớp thứ nhất (1st Layer)	19
3.2.2. Lớp thứ hai (2nd Layer).....	20
3.2.3. Lớp thứ ba (3rd Layer).....	20
3.2.4. Lớp thứ tư (4th Layer)	21
3.2.5. Lớp làm phẳng (Flatten Layer)	21
3.2.6. Lớp kết nối đầy đủ thứ nhất (Fully Connected Layer 1)	22
3.2.7. Lớp kết nối đầy đủ thứ hai (Fully Connected Layer 2)	22
3.2.8. Lớp đầu ra (Output Layer)	22
3.2.9. Biên dịch mô hình (Compilation)	23
3.2.10. Checkpoint	26
3.2.11. EarlyStopping.....	26
3.2.12. ReduceLROnPlateau.....	26
3.2.13. Callback List	27
3.2.14. Cấu hình huấn luyện	27
3.3. Huấn luyện mô hình.....	27
3.4. Lưu mô hình.....	28
Chương 4. Đánh giá mô hình.....	29
4.1. Biểu đồ độ chính xác.....	29
4.2. Báo cáo phân loại.....	32
4.3. Ma trận nhầm lẫn	33
Chương 5. Triển khai mô hình	35
5.1 Classifier Cascade	35
5.2 Nhận diện khuôn mặt	35

5.3 Tiền xử lý ảnh khuôn mặt	36
5.4 Chuẩn hóa dữ liệu	37
5.5 Chuyển đổi dữ liệu	38
5.6 Dự đoán cảm xúc	39
KẾT LUẬN	44
TÀI LIỆU THAM KHẢO	45

LỜI CAM ĐOAN

Nhóm chúng em xin cam đoan kết quả đạt được trong báo cáo là sản phẩm nghiên cứu, tìm hiểu của chúng em. Trong toàn bộ nội dung của báo cáo, những điều được trình bày hoặc là của nhóm chúng em tìm hiểu hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Chúng em xin hoàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Hà Nội, ngày 10 tháng 06 năm 2024

NHÓM SINH VIÊN THỰC HIỆN

Đại diện nhóm 11

Nguyễn Thế Nghĩa

LỜI CẢM ƠN

Nhóm 11 chúng em gửi lời cảm ơn chân thành tới Học viện Ngân hàng và Khoa Công nghệ thông tin & Kinh tế số đã tạo cơ hội cho sinh viên chúng em được học tập trong điều kiện thuận lợi, chất lượng đào tạo cùng nghiệp vụ chuyên môn hoàn toàn đáp ứng được nhu cầu của người học và nhu cầu thực tế. Môn Trí tuệ nhân tạo đã giúp chúng em nắm bắt không chỉ kiến thức nền tảng của trí tuệ nhân tạo – một nhánh đang rất phát triển của ngành công nghệ thông tin- mà hơn thế nữa, chúng em còn được học sâu hơn về học máy (Machine Learning), học sâu (Deep Learning). Ở học phần này, chúng em được học về tiền xử lý dữ liệu, các phương pháp trích chọn đặc trưng, xây dựng mô hình, đánh giá, triển khai mô hình. Về học sâu, chúng em được học cách xử lý ngôn ngữ tự nhiên, thị giác máy tính. Từ những kiến thức nền tảng và chuyên sâu đó đã giúp chúng em có kiến thức để hoàn thành bài tập này.

Đồng thời, nhóm chúng em trân trọng cảm ơn sự giảng dạy, chỉ dẫn nhiệt tình, tâm huyết của thầy Vũ Trọng Sinh đối với môn Trí tuệ nhân tạo và bài báo cáo này. Do nhiều yếu tố khác nhau mà trong bài không thể tránh khỏi những sai sót, kính mong thầy nhận xét, góp ý để chúng em kịp thời khắc phục, rút kinh nghiệm cho các dự án sau này.

Lời sau cùng, em xin kính chúc tất các quý thầy, cô trong Khoa Công nghệ thông tin & Kinh tế số luôn mạnh khỏe, đào tạo thật nhiều thế hệ sinh viên có đủ tâm, đủ tầm, đủ tài giúp ích cho xã hội.

Chúng em xin chân thành cảm ơn!

Nhóm sinh viên thực hiện

DANH MỤC HÌNH ẢNH

Hình 1. Luồng CNN xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.	17
Hình 2. Hình ảnh ma trận hình ảnh và ma trận bộ lọc.	18
Hình 3. Kiến trúc mô hình	25
Hình 4. Train Model.....	28
Hình 5. Biểu đồ tương quan giữa Epoch và độ chính xác.	30
Hình 6. Kết quả hiển thị.	32
Hình 7. Biểu đồ ma trận nhầm lẫn.	33
Hình 8. Kết quả chương trình 1	41
Hình 9. Kết quả chương trình 2	41
Hình 10. Kết quả chương trình 3	42
Hình 11. Kết quả chương trình 4.....	42
Hình 12. Kết quả chương trình 5	43

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa	Tiếng Việt
1	AI	Artificial Intelligence	Trí tuệ nhân tạo
2	CIVAMS	CMC Intelligent Video Analytics and Management System	Hệ thống quản lý và phân tích video thông minh CMC
3	NIST	National Institute of Standards and Technology	Viện Tiêu chuẩn và Công nghệ
4	EDR	Emotion Detection Recognition	Nhận dạng phát hiện cảm xúc
5	CAGR	Compound Annual Growth Rate	Tỷ lệ tăng trưởng kép hàng năm
6	AGI	Artificial General Intelligence	Trí tuệ nhân tạo tổng hợp
7	ASI	Artificial Super Intelligence	Siêu trí tuệ nhân tạo
8	ANI	Artificial Narrow Intelligence	Trí tuệ nhân tạo hẹp
9	ASR	Automatic Speech Recognition	Nhận dạng giọng nói tự động
10	FAQ	Frequently Asked Questions	Các câu hỏi thường gặp
11	NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
12	CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
13	BGR	Blue Green Red	
14	ReLU	Rectified Linear Unit	Đơn vị tuyến tính chỉnh lưu

LỜI MỞ ĐẦU

Với xu thế phát triển của thế giới hiện đại, con người không chỉ quan tâm đến các yếu tố vật chất mà còn chú trọng hơn đến yếu tố tinh thần. Vì vậy, nhận diện biểu cảm khuôn mặt trở thành lĩnh vực nghiên cứu tiềm năng trong trí tuệ nhân tạo (AI) và học máy (Machine Learning). Nhận biết được biểu cảm khuôn mặt mở ra khả năng lớn cho xã hội, nhất là trong các ngành dịch vụ chăm sóc khách hàng, hỗ trợ chăm sóc sức khỏe tâm lý đến các công việc như nâng cao hiệu quả làm việc của nhân viên, gia tăng hiệu quả giáo dục, đào tạo.

Nghiên cứu và ứng dụng nhận diện biểu cảm khuôn mặt trở thành xu hướng mới. Hệ thống phát triển và ứng dụng công nghệ này có khả năng phân tích, giải mã các cảm xúc khác nhau như vui vẻ, buồn bã, ngạc nhiên, sợ hãi, tức giận, trung tính dựa trên những đặc điểm khuôn mặt. Những tiến bộ trong lĩnh vực này đã và đang mang lại hiệu quả, lợi ích to lớn trong các hệ thống thông minh, giúp hệ thống trở nên nhạy bén, hoàn thiện hơn.

Song song với những lợi ích hữu hình đó là những mối đe dọa mà con người cần đối mặt, giải quyết khi xây dựng nên công nghệ này. Đầu tiên là về tính bảo mật, quyền riêng tư, an toàn khi sử dụng những hệ thống có sử dụng hình ảnh khuôn mặt. Bởi khuôn mặt mỗi người lại chứa những yếu tố sinh trắc học riêng biệt, kẻ xấu có thể lợi dụng công nghệ để đánh cắp những dữ liệu này và sử dụng vào mục đích bất hợp pháp, trái phép. Thứ hai, khó khăn do sự đa dạng văn hóa, ngôn ngữ, trình độ con người của từng quốc gia, dân tộc cũng là một cản trở cần phải cân nhắc thận trọng. Ngoài ra, công nghệ còn đối mặt với nhiều khía cạnh khác như độ chính xác của hệ thống nhận diện đối với biểu cảm phức tạp của con người, độ chính xác của hệ thống, điều kiện nhận diện biểu cảm, vấn đề đạo đức khác. Vì những lý do chính trên mà việc nghiên cứu, phát triển các giải pháp cho công nghệ này yêu cầu sự sâu rộng về kiến thức, kỹ thuật, đồng thời có sự nhạy bén trong hiểu biết và ứng dụng vào thực tế.

Đề tài “***Nhận diện biểu cảm khuôn mặt***” của nhóm chủ yếu khám phá các phương pháp, kỹ thuật mới trong lĩnh vực này, phân tích khó khăn hiện tại, các giải pháp phát triển trong tương lai. Nhóm nghiên cứu đề tài này với mong muốn thông qua đề tài, nhóm có cái nhìn toàn diện, hiểu rõ hơn về tiềm năng và ứng dụng của nhận diện biểu cảm khuôn mặt cũng như tầm quan trọng của lĩnh vực AI trong nâng cao chất lượng cuộc sống con người, tạo ra những đóng góp, giá trị thiết thực giúp ích cho xã hội.

NỘI DUNG

Chương 1. Giới thiệu đề tài

1.1 Đặt vấn đề

Nhận diện biểu cảm khuôn mặt đang dần trở thành công nghệ có ý nghĩa quan trọng trong nhiều lĩnh vực của cuộc sống, khoa học, bao gồm các ngành khác nhau như tâm lý học, giáo dục, y tế đến các ngành công nghiệp giải trí, an ninh an toàn. Công nghệ hiện đại này giúp máy tính học và hiểu được các biểu cảm trên khuôn mặt người. Việc này đóng góp vào sự phát triển của lĩnh vực trí tuệ nhân tạo bởi vì máy tính hiểu được biểu cảm của con người, từ đó có thể phản hồi lại con người, cải thiện giao tiếp giữa con người và máy móc.

Công nghệ nhận diện biểu cảm khuôn mặt ngày càng có bước phát triển vượt bậc qua các năm gần đây, tuy nhiên nhiều công ty chưa hiểu đầy đủ về tầm quan trọng của việc đầu tư vào dữ liệu liên quan đến biểu cảm khuôn mặt. Các nhà quản lý chưa chú trọng vào đầu tư các dữ liệu này để nâng cao trải nghiệm người dùng, cải thiện dịch vụ khách hàng, nâng cao hiệu quả làm việc của nhân viên, đưa ra các quyết định kinh doanh chiến lược.

Theo một báo cáo “Tổng quan về thị trường phát hiện và nhận dạng cảm xúc (EDR)” thì thị trường EDR toàn cầu được định giá 1.7 tỷ USD vào năm 2012⁹ và dự kiến đạt 4.8 tỷ USD vào năm 2028, đạt tốc độ tỷ lệ tăng trưởng kép hàng năm CAGR lên đến 12% trong giai đoạn 2020 – 2028 (Exactitude Consultancy, 2022).

Vào ngày 15/12/2023, giải pháp nhận diện khuôn mặt CIVAMS của Tập đoàn công nghệ CMC đã được Viện Tiêu chuẩn và Công nghệ quốc gia Mỹ (NIST) công bố lọt top 12 thế giới, đồng thời đạt top 1 tại Việt Nam (Thành Nguyễn, 2023). Gần đây có thêm sự kiện Viettel AI cũng lọt top 4 thế giới về công nghệ nhận diện khuôn mặt do NIST công bố (Lan Nhi, 2024). Đây quả là những thành tích đáng vui mừng, tự hào đối với sự phát triển của nền công nghệ nước nhà. Hiện nay, công nghệ nhận diện biểu cảm khuôn mặt được ứng dụng rộng rãi trong các lĩnh vực như bán lẻ, ngân hàng, y tế...

Tuy nhiên, trong quá trình triển khai công nghệ mới vẫn còn nhiều thách thức như vấn đề bảo mật, quyền riêng tư, độ chính xác, khả năng tích hợp với các hệ thống hiện tại. Để khắc phục những vấn đề trên, các công ty cần phải thường xuyên tự đánh giá, cập nhật công nghệ mới nhất, tuân thủ các luật, quy định do nhà nước ban hành.

Phân tích dữ liệu và công nghệ nhận diện biểu cảm khuôn mặt là một lĩnh vực tiềm năng, có cơ hội lớn đem lại nhiều lợi ích cho doanh nghiệp và xã hội trên các ngành

nghe khác như cải thiện dịch vụ chăm sóc khách hàng, nâng cao hiệu quả làm việc, tăng cường an ninh an toàn, chăm sóc sức khỏe.

1.2 Cơ sở hình thành đề tài

Nhận diện biểu cảm khuôn mặt không phải là đề tài quá mới mẻ, song vẫn là đề tài được quan tâm, đầu tư, khai thác bởi những lợi ích mà nó mang lại cho xã hội, doanh nghiệp. Lời nói có thể chân thực hoặc sai lệch nhưng những biểu cảm, cảm xúc trên khuôn mặt sẽ phản ánh chính xác thái độ, cảm xúc của người đó đối với những trải nghiệm của họ. Đối với các nhà kinh doanh, nắm bắt được cảm xúc khách hàng là vô cùng cần thiết. Đây là yếu tố then chốt để đưa ra những chiến lược phù hợp, đặc biệt là trong ngành dịch vụ chăm sóc khách hàng. Các nhà quản lý nhân lực khi biết được cảm xúc, tâm trạng của nhân viên có thể cải thiện môi trường làm việc, từ đó thúc đẩy, làm tăng hiệu quả công việc.

Tận dụng những thành tựu sẵn có trong lĩnh vực nhận diện khuôn mặt cả về công nghệ và dữ liệu, đề tài tìm hiểu về “Nhận diện biểu cảm khuôn mặt” sử dụng các mô hình, thuật toán để phân tích cảm xúc, biểu cảm của người sử dụng khi ở trước camera. Bảy biểu cảm cơ bản mà dự án có thể nhận diện được bao gồm: Angry – Tức giận, cáu kỉnh; Disgust – Ghê tởm, ghê sợ; Fear – Sợ hãi, lo sợ; Happy – Vui vẻ, hạnh phúc; Neutral – Tự nhiên, trung tính, trung lập; Sad – Buồn bã, đau khổ; Surprise – Ngạc nhiên, bất ngờ.

Chỉ bằng các biểu cảm cơ bản, đơn giản trên cũng đã có thể giúp ích rất nhiều cho những người sử dụng công nghệ nhận diện biểu cảm khuôn mặt. Vậy nên, việc xây dựng mô hình nhận diện biểu cảm khuôn mặt là cần thiết và có thể giải quyết nhu cầu của nhiều nhà lãnh đạo khi đề tài được phát triển hơn trong tương lai.

1.3 Mục tiêu đề tài

Nhận thấy nhu cầu thực tiễn và tính cấp thiết, đề tài “**Nhận diện biểu cảm khuôn mặt**” là đề tài nhóm lựa chọn để tìm hiểu, phân tích, nghiên cứu dựa trên bộ dữ liệu có sẵn là tập các ảnh lưu trữ biểu cảm khuôn mặt khác nhau như vui, buồn, giận dữ, ngạc nhiên, sợ hãi, trung lập trên Kaggle. Bộ dữ liệu gồm hai tệp dữ liệu là tập “Train” với 28821 ảnh và tập “Validation” với 7066 ảnh được sử dụng trong mô hình. Hệ thống được nhóm xây dựng có khả năng nhận diện cảm xúc trên khuôn mặt người dùng từ video thời gian thực, sử dụng webcam để thu thập hình ảnh khuôn mặt, đưa ra dự đoán cảm xúc người dùng dựa trên đặc điểm, đặc trưng của khuôn mặt.

Kết quả của phân tích dữ liệu thu thập bao gồm:

- Xác định các yếu tố ảnh hưởng đến biểu cảm khuôn mặt: Phân tích dữ liệu có thể giúp xác định những yếu tố như tình huống, ngữ cảnh, ánh sáng, góc chụp có ảnh hưởng đến độ chính xác của việc nhận diện đúng biểu cảm, cảm xúc trên khuôn mặt.
- Nhận diện biểu cảm khuôn mặt trong thời gian thực: Dựa trên các yếu tố đã được xác định, phân tích dữ liệu giúp xây dựng một mô hình nhận diện biểu cảm khuôn mặt với độ chính xác cao theo thời gian thực.
- Tìm hiểu và phân tích tâm lý học thông qua biểu cảm: Phân tích dữ liệu có thể giúp hiểu rõ hơn về trạng thái tâm lý con người thông qua biểu cảm khuôn mặt, từ đó đưa ra những biện pháp phù hợp giúp cải thiện tâm lý hoặc trạng thái cảm xúc.

Đối với các doanh nghiệp, dữ liệu phân tích có thể dùng để:

- Xây dựng các ứng dụng cải thiện trải nghiệm khách hàng: Phân tích dữ liệu có thể giúp xây dựng nên các ứng dụng nhận diện biểu cảm khuôn mặt để cải thiện trải nghiệm khách hàng như đối với các ngành dịch vụ khách hàng, bán lẻ, chăm sóc sức khỏe.
- Theo dõi và đánh giá cảm xúc của người dùng trong các tình huống khác nhau, kịp thời điều chỉnh chiến lược tương tác, tiếp cận với khách hàng theo hướng phù hợp hơn.
- Phát triển các chương trình đào tạo, giáo dục nhằm mục tiêu nâng cao khả năng nhận diện và phản ứng với các biểu cảm khuôn mặt, cải thiện kỹ năng giao tiếp và tương tác xã hội.

Mục tiêu của đề tài là xây dựng một công cụ đơn giản, hữu ích cho những người có nhu cầu hiểu rõ hơn về cảm xúc, phản ứng của con người. Tùy vào mục đích sử dụng của doanh nghiệp trong tìm ra chiến lược, giải pháp phù hợp, cải thiện tương tác, đáp ứng nhu cầu người dùng một cách hiệu quả.

1.4 Đối tượng và phương pháp nghiên cứu

- Đối tượng nghiên cứu:
Bộ dữ liệu công khai, có sẵn trên Kaggle: **Face expression recognition dataset**
 - Đa dạng về chủng tộc, giới tính, độ tuổi: phản ánh sự khác biệt trong cách biểu lộ cảm xúc con người trên các nền văn hóa, nhân khẩu học khác nhau.
 - Số lượng mẫu đủ lớn: 35 887 ảnh đen trắng về khuôn mặt người – đảm bảo tính chính xác và tin cậy của kết quả nghiên cứu.
 - Chất lượng hình ảnh: đảm bảo ảnh chụp rõ ràng, đủ ánh sáng, không bị nhiễu để có thể trích xuất chính xác các đặc điểm trên khuôn mặt.

- Phân loại rõ ràng: phân loại cảm xúc cụ thể thành 7 tập khác nhau, có gán nhãn rõ ràng: Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise (Giận, Ghê tởm, Sợ, Vui, Trung tính, Buồn, Ngạc nhiên) để đánh giá hiệu quả thuật toán.

Tập dữ liệu nhận dạng biểu cảm khuôn mặt với số lượng đủ lớn, nguồn hình ảnh đa dạng, phong phú, liên quan đến đề tài “**Nhận diện biểu cảm khuôn mặt**” của nhóm.

- Phương pháp nghiên cứu:
 - Học có giám sát: sử dụng tập dữ liệu có chú thích biểu cảm để huấn luyện mô hình nhận diện.
 - Học không giám sát: tự động phân nhóm các biểu cảm dựa trên đặc điểm khuôn mặt được trích xuất.
 - Học tăng cường: sử dụng phản hồi liên tục để cải thiện hiệu suất mô hình theo thời gian.

1.5 Đối tượng sử dụng và ý nghĩa đề tài

- Đối tượng sử dụng: Nhận diện biểu cảm khuôn mặt có tiềm năng ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như các nhà quản lý doanh nghiệp, tổ chức, các nhà phát triển phần mềm, an ninh, giáo dục, y tế, dịch vụ chăm sóc khách hàng, giải trí.
- Ý nghĩa đề tài: Mô hình sử dụng giải thuật nhận diện và phân loại giúp các đối tượng sử dụng xác định, hiểu rõ hơn về cảm xúc của con người trong thời gian thực.

1.6 Phạm vi dự án

- Phát hiện và xử lý khuôn mặt: Sử dụng kỹ thuật phát hiện khuôn mặt Haar Cascade (Thuật toán xếp tầng Haar – Haar Cascade Algorithm). Đây là thuật toán cho phép phát hiện các đối tượng trong hình ảnh, bất kể tỷ lệ chúng trong khung hình ảnh và vị trí (Abhishek Jaiswal, 2023).
- Tiền xử lý dữ liệu: Chuyển đổi ảnh màu sang ảnh xám, chuẩn hóa kích thước và giá trị pixel để đảm bảo dữ liệu đầu vào đồng nhất.
- Trích chọn đặc trưng và nhận diện cảm xúc: Áp dụng mô hình học sâu mạng nơ-ron tích chập (CNN) để trích xuất các đặc trưng khuôn mặt và dự đoán cảm xúc.
- Hiển thị kết quả: Hiển thị nhãn cảm xúc trực tiếp trên khung hình video, kèm theo hình chữ nhật bao quanh khuôn mặt được nhận diện.

1.7 Kết cấu bài báo cáo

Đề tài bao gồm các chương chính sau:

- Chương 1: Giới thiệu đề tài
- Chương 2: Trí tuệ nhân tạo và tiền xử lý dữ liệu
- Chương 3: Xây dựng và triển khai mô hình
- Chương 4: Đánh giá mô hình và kết luận

Chương 2. Trí tuệ nhân tạo và tiền xử lý dữ liệu

2.1 Trí tuệ nhân tạo

2.1.1 Khái niệm

Trí tuệ nhân tạo (Artificial intelligence) là công nghệ cho phép máy tính và máy móc mô phỏng trí thông minh của con người và khả năng giải quyết vấn đề (IBM, 2024).

AI được chia thành 2 loại:

- AI mạnh: được tạo thành từ trí tuệ nhân tạo tổng hợp (AGI) và trí tuệ siêu nhân tạo (ASI). Tuy nhiên, AI mạnh chưa có ví dụ thực tế mà vẫn hoàn toàn là lý thuyết.
- AI yếu hay còn gọi là AI hẹp hoặc trí tuệ nhân tạo hẹp (ANI): được đào tạo để thực hiện nhiệm vụ cụ thể. AI yếu ứng dụng trong Siri của Apple, Alexa của Amazon, xe tự lái...

2.1.2 Các ứng dụng của AI

AI có rất nhiều ứng dụng trong thế giới thực. Một số trường hợp sau được sử dụng phổ biến nhất:

- Nhận dạng giọng nói hay nhận dạng giọng nói tự động (ASR): sử dụng NLP để xử lý lời nói con người thành văn bản.
- Dịch vụ khách hàng: các đại lý ảo và chatbot trực tuyến để trả lời các câu hỏi thường gặp (FAQ).
- Thị giác máy tính: cho phép máy tính và hệ thống lấy thông tin có ý nghĩa từ hình ảnh, video, ... rồi thực hiện các nhiệm vụ cần thiết. Ví dụ chụp ảnh X quang, ô tô tự lái.
- Chuỗi cung ứng: Robot thích ứng hoạt động dựa trên thông tin thiết bị để đưa ra các quyết định tự chủ. Các công cụ NLP có thể hiểu lời nói con người và biết phản ứng lại với những lời nói đó.
- Dự báo thời tiết: ứng dụng của học máy nâng cao vào các mô hình dự đoán thời tiết để dự báo phù hợp hơn, chính xác hơn.
- Phát hiện bất thường: mô hình AI cho phép lướt qua lượng lớn dữ liệu và khám phá ra các dữ liệu bất thường trong tập dữ liệu ví dụ như lỗi thiết bị, lỗi do người dùng...

Nhân diện biểu cảm khuôn mặt là một ứng dụng của thị giác máy tính.

2.2 Nhập các thư viện và thiết lập môi trường

Một số thư viện cần thiết được sử dụng trong bài để xây dựng và huấn luyện mô hình mạng nơ-ron tích chập (CNN) để nhận biết cảm xúc trên khuôn mặt người là:

- ``os``: Cung cấp các hàm để tương tác với hệ điều hành, hữu ích cho các thao tác với tệp.
- ``cv2``: Thư viện OpenCV cho các tác vụ thị giác máy tính, được sử dụng ở đây để xử lý hình ảnh.
- ``numpy``: Thư viện tính toán số học, được sử dụng cho các thao tác mảng.
- ``tensorflow``: Thư viện TensorFlow cho học sâu.
- ``train_test_split`` từ `sklearn.model_selection`: Chia dữ liệu thành các tập huấn luyện và kiểm tra.
- ``ImageDataGenerator`` từ `tensorflow.keras.preprocessing.image`: Tạo ra các lô dữ liệu tăng cường.
- ``LabelEncoder`` từ `sklearn.preprocessing`: Mã hóa các nhãn mục tiêu với các giá trị giữa 0 và `n_classes-1`.
- ``to_categorical`` từ `keras.utils`: Chuyển đổi vector lớp (số nguyên) thành ma trận lớp nhị phân.
- ``Sequential`` từ `keras.models`: Ngăn xếp tuyến tính của các lớp để xây dựng mô hình học sâu.
- ``Dense, Conv2D, Dropout, BatchNormalization, MaxPooling2D, Flatten`` từ `keras.layers`: Các loại lớp khác nhau được sử dụng trong kiến trúc CNN.
- ``Optimizers`` (Adam, RMSprop, SGD) từ `keras.optimizers`: Các thuật toán được sử dụng để cập nhật trọng số mô hình trong quá trình huấn luyện.
- ``plt`` từ `matplotlib.pyplot`: Thư viện để tạo các hình ảnh minh họa, được sử dụng ở đây để vẽ các đường cong huấn luyện/kiểm tra.
- Callbacks (`ModelCheckpoint`, `EarlyStopping`, `ReduceLROnPlateau`) từ `keras.callbacks`: Các hàm được gọi trong quá trình huấn luyện tại một số điểm nhất định để cải thiện hiệu suất mô hình hoặc xử lý các gián đoạn huấn luyện.

```
import os
import cv2
import numpy as np
import tensorflow
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
```

```

from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, BatchNormalization,
MaxPooling2D, Flatten
from keras.optimizers import Adam, RMSprop, SGD
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau

```

- Biến ‘data_dir’ chỉ định thư mục nơi bộ dữ liệu được lưu trữ. Thư mục ‘images/train’ là nơi chứa các thư mục con với hình ảnh được tổ chức thành các lớp hoặc danh mục khác nhau. Các hình ảnh trong thư mục con được sử dụng để huấn luyện mô hình nhận diện cảm xúc trên khuôn mặt.

```

# Dataset Directory
data_dir = "images/train"

```

- Biến ‘sub_folders’ được gán danh sách các thư mục con trong thư mục của bộ dữ liệu ‘data_dir’. Mỗi thư mục đại diện cho một lớp dữ liệu. Vậy nên tên thư mục con được đặt tên tương ứng với lớp mà nó đại diện.

```

# Classes
sub_folders = os.listdir(data_dir)

```

- Hai danh sách ‘images’ và ‘labels’ là nơi lưu trữ dữ liệu hình ảnh cùng các nhãn tương ứng. Các danh sách này được điền vào bởi quá trình tải dữ liệu. Mỗi hình ảnh từ các lớp được thêm vào mảng ‘images’ thì đồng thời các nhãn tương ứng cũng được thêm vào mảng ‘labels’.

```

# Declaring the lists for images and labels
images = []
labels = []

```

2.3 Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là một bước quan trọng trong mọi dự án học máy hay trí tuệ nhân tạo, bao gồm cả lĩnh vực thị giác máy tính. Quá trình này là các kỹ thuật để chuẩn

bị dữ liệu, chuyển đổi dữ liệu thành dạng phù hợp giúp mô hình có thể sử dụng hiệu quả, dễ dàng cho việc phân tích.

2.3.1 Đọc và gán nhãn cho hình ảnh

Duyệt qua các thư mục con và đọc tất cả hình ảnh từ các thư mục con. Mỗi thư mục con đều được gán nhãn tương ứng với một biểu cảm khuôn mặt.

- Đoạn mã lặp qua từng thư mục con trình danh sách ‘sub_folders’, đại diện cho các lớp biểu cảm được gán nhãn tương ứng bên trên. Sau khi truy cập vào từng hình ảnh trong thư mục con thì dùng thư viện OpenCV để đọc và thêm vào danh sách ‘images’. Ngoài ra, OpenCV còn thêm nhãn tương ứng – là tên của thư mục con – vào danh sách ‘labels’.

Trong đoạn mã, có bước chuyển đổi ảnh màu sang ảnh xám nhằm làm giảm thông tin dư thừa, tập trung tốt hơn vào các đặc trưng trên khuôn mặt.

```
# Accessing the labels
for sub_folder in sub_folders:
    label = sub_folder

    # Constructing the path to the current sub-folder
    path = os.path.join(data_dir, sub_folder)

    # Listing all images in the current sub-folder
    sub_folder_images = os.listdir(path)

    # Accessing the Images
    for image_name in sub_folder_images:
        # Constructing the path to the current image
        image_path = os.path.join(path, image_name)
        # Loading the image using OpenCV
        img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE) # Read as
        grayscale
        # Appending the image to the list of images
        images.append(img)
        # Appending the label corresponding to the current sub-folder to the
        List of labels
        labels.append(label)
```

- Sử dụng hàm ‘np.array()’ để danh sách các hình ảnh và nhãn tương ứng được chuyển đổi thành các mảng NumPy, in ra số lượng ảnh. Việc sử dụng mảng

Numpy giúp tăng hiệu suất khi thực hiện các phép toán trên toàn bộ mảng, tiện dụng khi cần sử dụng thao tác ma trận.

```
# Converting the lists of images and labels to NumPy arrays
images = np.array(images)
labels = np.array(labels)
print(len(images))
```

2.3.2 Chia dữ liệu thành các tập huấn luyện, kiểm tra và xác thực

Bộ dữ liệu được chia thành các tập huấn luyện – train, validation. Sử dụng hàm ‘train_test_split’ từ thư viện ‘scikit-learn’ để thực hiện việc kiểm tra. Cụ thể:

- ‘X_train’, ‘y_train’: Hình ảnh và nhãn của tập huấn luyện.
- ‘X_val’, ‘y_val’: Hình ảnh và nhãn của tập validation.
- ‘X_test’, ‘y_test’: Hình ảnh và nhãn của tập kiểm tra.

Bộ dữ liệu được chia như sau: kích thước tập kiểm tra là 20%, tập validation là 10% từ bộ dữ liệu gốc. Tham số ‘random_state’ để đảm bảo tính tái tạo của việc chia tập.

```
# Splitting Dataset into training, validation, and test sets
X_train, X_test, y_train, y_test = train_test_split(
    images, labels, test_size=0.2, random_state=42)

X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.1, random_state=42)
```

2.3.3 Tiền xử lý hình ảnh

Xây dựng hàm tiền xử lý để xử lý hình ảnh đầu vào trước khi đưa chúng vào mô hình mạng nơ-ron.

- Chuẩn hóa các giá trị pixel bằng cách chia mỗi pixel cho 255.0, điều này đưa các giá trị pixel về phạm vi [0, 1].
- Thay đổi kích thước của tất cả hình ảnh thành kích thước cố định là 48x48 pixel bằng cách sử dụng hàm **resize** của OpenCV.
- Định hình mảng hình ảnh để phù hợp với hình dạng đầu vào được mong đợi bởi mô hình mạng nơ-ron. Hình dạng này là ‘(batch_size, chiều cao, chiều rộng,

kênh)', trong đó 'batch_size' được đặt là -1 để chỉ định kích thước batch size và 'channels' được đặt là 1 cho các hình ảnh xám.

```
# Preprocess the image
def preprocessing(img):
    img = img / 255.0
    img = cv2.resize(img, (48, 48))

    return img.reshape(-1, 48, 48, 1) # Reshape to match input shape
```

- Áp dụng hàm tiền xử lý lên từng phần tử trong các tập dữ liệu và kiểm tra bằng hàm **map**, sau đó chuyển đổi danh sách kết quả thành các mảng NumPy.
- 'map(preprocessing, X_train)': Áp dụng hàm 'preprocessing' cho mỗi hình ảnh trong 'X_train'.
- 'list(map(...))': Chuyển đổi đối tượng map thành một danh sách.
- 'np.array(...)': Chuyển đổi danh sách các hình ảnh đã tiền xử lý thành một mảng NumPy.
- Thay đổi kích thước các mảng NumPy để loại bỏ đi các chiều không cần thiết, biến đầu vào thành dạng phù hợp cho mạng nơ-ron tích chập – CNN:
- 'reshape(-1, 48, 48, 1)': Điều chỉnh hình dạng của các mảng dữ liệu đầu vào để có hình dạng '(batch_size, 48, 48, 1)', trong đó -1 chỉ ra rằng kích thước lô được xác định động dựa trên số lượng mẫu.

```
# Apply preprocessing to training, validation, and test sets
X_train = np.array(list(map(preprocessing, X_train)))
X_val = np.array(list(map(preprocessing, X_val)))
X_test = np.array(list(map(preprocessing, X_test)))

# Reshape input data to remove unnecessary dimension
X_train = X_train.reshape(-1, 48, 48, 1)
X_val = X_val.reshape(-1, 48, 48, 1)
X_test = X_test.reshape(-1, 48, 48, 1)
```

2.3.4 Tăng cường dữ liệu

Từ thư viện Keras, khởi tạo đối tượng 'ImageDataGenerator' với các tham số tăng cường dữ liệu khác nhau để tăng cường các hình ảnh huấn luyện. Tăng cường dữ liệu là một kỹ thuật được sử dụng để tăng kích thước của tập dữ liệu huấn luyện bằng cách áp

dụng các biến đổi ngẫu nhiên cho các hình ảnh, giúp cải thiện khả năng tổng quát hóa và tính đáng tin cậy của mô hình.

- `'width_shift_range'`: Di chuyển ngẫu nhiên chiều rộng của hình ảnh một phần nhỏ của chiều rộng của nó.
- `'height_shift_range'`: Di chuyển ngẫu nhiên chiều cao của hình ảnh một phần nhỏ của chiều cao của nó.
- `'zoom_range'`: Phóng to hoặc thu nhỏ ngẫu nhiên hình ảnh.
- `'shear_range'`: Áp dụng biến đổi cắt ngẫu nhiên cho hình ảnh.
- `'rotation_range'`: Xoay ngẫu nhiên hình ảnh theo một phạm vi góc nhất định.
- Sau đó, phương thức `'fit()'` được gọi trên đối tượng `'ImageDataGenerator'` để tính toán các thống kê cần thiết cho việc tăng cường dữ liệu dựa trên dữ liệu huấn luyện.

```
# Initialize ImageDataGenerator for data augmentation
data_gen = ImageDataGenerator(
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    shear_range=0.1,
    rotation_range=10
)

# Compute necessary statistics for data augmentation
data_gen.fit(X_train)
```

2.3.5 Mã hóa dữ liệu

Từ thư viện 'scikit-learn', sử dụng công cụ 'LabelEncoder' để mã hóa các nhãn từ giá trị chuỗi thành các giá trị số do yêu cầu bắt buộc của hầu hết các mô hình học máy là đầu vào phải ở dạng số học. Phương thức 'fit()' được gọi đến để phù hợp bộ mã hóa với mảng các nhãn lớp, cho phép nó ánh xạ các nhãn lớp thành các giá trị số để mô hình có thể hiểu và thực hiện.

```
# Encode the class labels
label_encoder = LabelEncoder()
label_encoder.fit(labels)

# Encode the class labels for training, validation, and test sets
y_train = label_encoder.transform(y_train)
```



```
y_val = label_encoder.transform(y_val)
y_test = label_encoder.transform(y_test)
```

Các nhãn lớp của các tập dữ liệu đã phân chia được mã hóa bằng phương thức 'transform()' của đối tượng 'LabelEncoder' để chuyển đổi, chuẩn bị dữ liệu sử dụng trong việc huấn luyện, đánh giá mô hình.

Chuyển đổi nhãn mã hóa số thành định dạng one-hot để sử dụng trong mô hình học sâu. Sử dụng one-hot encoded categorical arrays, biến đổi từng giá trị thành các đặc trưng nhị phân chỉ chứa giá trị 1 hoặc 0.

Xây dựng biến 'num_classes' được gán bằng đúng số lượng các lớp trong tập dữ liệu. Danh sách các lớp đã được mã hóa từ chuỗi sang số nguyên được chứa trong 'label_encoder.classes_', hàm 'len()' để trả về số lượng phần tử trong danh sách này.

```
# Get the number of classes
num_classes = len(label_encoder.classes_)

# Convert encoded class labels to one-hot encoded categorical arrays
y_train_categorical = to_categorical(y_train, num_classes=num_classes)
y_val_categorical = to_categorical(y_val, num_classes=num_classes)
y_test_categorical = to_categorical(y_test, num_classes=num_classes)
```

Sau đó, hàm 'to_categorical()' được sử dụng để chuyển đổi các nhãn lớp được mã hóa thành các mảng danh mục được mã hóa one-hot. Việc này là cần thiết cho các nhiệm vụ phân loại đa lớp, trong đó mỗi nhãn lớp được biểu diễn dưới dạng một vector nhị phân với 1 ở vị trí tương ứng với chỉ số lớp và các giá trị 0 ở tất cả vị trí còn lại trong vector.

Tóm lại, phần tiền xử lý dữ liệu gồm các kỹ thuật chính sau đây:

- Đọc và gán nhãn hình ảnh: Đọc hình ảnh từ các thư mục và gán nhãn dựa trên tên thư mục.
- Chia dữ liệu: Chia dữ liệu thành các tập huấn luyện, kiểm tra và xác thực.
- Chuẩn hóa và chuyển đổi kích thước hình ảnh: Chuẩn hóa giá trị pixel, chuyển đổi kích thước hình ảnh, giảm chiều dữ liệu, chuyển ảnh màu sang ảnh xám.
- Tăng cường dữ liệu bằng cách sử dụng kỹ thuật tăng cường dữ liệu để tạo ra các biến thể của hình ảnh ban đầu.
- Mã hóa dữ liệu và one-hot encoding: là mã hóa nhãn của các lớp từ dạng chuỗi sang dạng số nguyên và chuyển đổi thành định dạng one-hot.

2.4 Kỹ thuật trích chọn đặc trưng

Quá trình trích chọn đặc trưng là một bước quan trọng, quyết định sự thành công của mô hình trong việc phân loại biểu cảm khuôn mặt. Dưới đây là tóm tắt về quá trình trích chọn đặc trưng trong dự án này:

1. Chuẩn bị và Tiền xử lý Dữ liệu:

Các hình ảnh khuôn mặt được chuyển đổi thành dạng thang độ xám để giảm bớt độ phức tạp.

Hình ảnh được chuẩn hóa (normalization) về khoảng $[0, 1]$ và thay đổi kích thước về 48×48 để đảm bảo tính đồng nhất.

Quá trình tiền xử lý giúp chuẩn bị dữ liệu đầu vào cho các lớp tích chập của mô hình, đảm bảo các đặc trưng không gian được bảo toàn.

2. Kiến trúc Mạng Nơ-ron Tích chập (CNN):

Mô hình bao gồm nhiều lớp tích chập (Conv2D) với số lượng bộ lọc tăng dần (64, 128, 512) và các kích thước bộ lọc khác nhau (5×5 , 3×3).

Các lớp tích chập giúp trích xuất các đặc trưng không gian từ hình ảnh, bao gồm các cạnh, góc, và các cấu trúc phức tạp hơn.

Mỗi lớp tích chập được theo sau bởi lớp chuẩn hóa lô (BatchNormalization) giúp ổn định và tăng tốc độ huấn luyện.

3. Các Kỹ thuật Chính quy hóa:

Lớp lấy mẫu cực đại (MaxPooling2D) giúp giảm kích thước không gian của đặc trưng, giữ lại các đặc trưng quan trọng và giảm thiểu overfitting.

Lớp dropout (Dropout) với tỷ lệ 0.3 được thêm vào sau mỗi lớp tích chập và lớp kết nối đầy đủ để ngăn chặn overfitting bằng cách bỏ ngẫu nhiên một số đơn vị trong quá trình huấn luyện.

4. Lớp Kết nối Đầy đủ (Fully Connected Layers):

Sau khi các đặc trưng không gian được trích xuất và làm phẳng bởi lớp Flatten, chúng được đưa vào các lớp kết nối đầy đủ (Dense) với 256 và 512 đơn vị.

Các lớp này giúp học các đặc trưng phức tạp hơn và kết hợp chúng lại để thực hiện phân loại biểu cảm.

5. Lớp Đầu ra (Output Layer):

Lớp đầu ra (Dense) sử dụng hàm kích hoạt softmax để phân loại các biểu cảm khuôn mặt thành các lớp khác nhau.

Chương 3. Xây dựng và huấn luyện mô hình

Mô hình CNN – mạng nơ-ron tích chập là mô hình nhóm sử dụng trong bài, huấn luyện mô hình để phân loại cảm xúc.

3.1. Khái niệm CNN

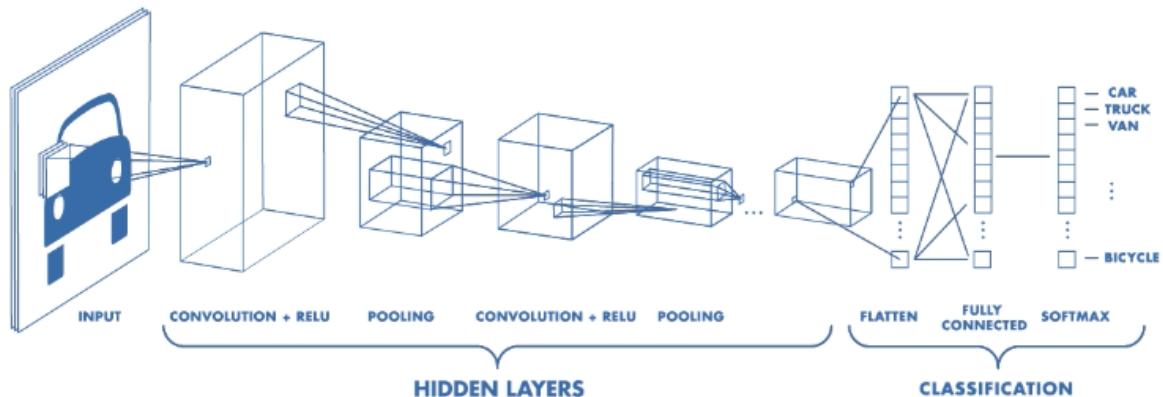
Mạng nơ-ron tích chập (CNN) là một tập hợp con của học máy và là trung tâm của các thuật toán học sâu. Mô hình CNN bao gồm các lớp nút với hai đầu chứa lớp đầu vào và đầu ra, ở giữa hai lớp này có thể chứa một hoặc nhiều lớp ẩn. Các nút liên kết với nhau theo trọng số cùng các ngưỡng liên quan. Nếu đầu ra của một nút riêng lẻ bất kỳ có giá trị cao hơn ngưỡng chỉ định thì nút đó được kích hoạt, gửi dữ liệu tới lớp kế tiếp trong mạng. Ngược lại, nếu không có giá trị cao hơn ngưỡng chỉ định thì không có dữ liệu nào được truyền đến lớp kế tiếp trong mạng.

CNN được phân biệt với các mạng nơ-ron khác bởi hiệu suất vượt trội của mô hình này với tín hiệu đầu vào là hình ảnh, giọng nói hoặc âm thanh. CNN có ba lớp chính:

Lớp tích chập - Convolution Layer

Lớp gộp - Pooling Layer

Lớp được kết nối đầy đủ (Full Connected)



Hình 1. Luồng CNN xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

3.1.1 Lớp tích chập - Convolution Layer

Đây là lớp đầu tiên – lớp cốt lõi của CNN để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì liên kết giữa các pixel bằng cách tìm hiểu các tính năng hình

ảnh, sử dụng ô vuông nhỏ của dữ liệu đầu vào. Đó là phép toán với hai đầu vào như ma trận hình ảnh và một bộ lọc.

- An image matrix (volume) of dimension **($h \times w \times d$)**
- A filter (**$f_h \times f_w \times d$**)
- Outputs a volume dimension **($h - f_h + 1$) \times ($w - f_w + 1$) \times 1**



Hình 2. Hình ảnh ma trận hình ảnh và ma trận bộ lọc.

- Số lượng bộ lọc: ảnh hưởng đến độ sâu của đầu ra.
- Stride là khoảng cách hoặc số pixel mà hạt nhân di chuyển trên ma trận đầu vào.
- Zero-padding thường được sử dụng khi các bộ lọc không vừa với hình ảnh đầu vào. Điều này dẫn đến tất cả các phần tử nằm ngoài ma trận đầu vào về 0, tạo ra đầu ra lớn hơn hoặc có kích thước bằng nhau. Có ba loại đệm:
 - Phần đệm hợp lệ: Điều này còn được gọi là không có phần đệm. Trong trường hợp này, tích chập cuối cùng sẽ bị loại bỏ nếu kích thước không bằng nhau.
 - Phần đệm giống nhau: Phần đệm này đảm bảo rằng lớp đầu ra có cùng kích thước với lớp đầu vào.
 - Phần đệm đầy đủ: Loại phần đệm này làm tăng kích thước của đầu ra bằng cách thêm các số 0 vào đường viền của đầu vào.

Sau mỗi thao tác tích chập, CNN áp dụng phép biến đổi Đơn vị tuyến tính chỉnh lưu (ReLU) cho bản đồ đặc trưng, đưa tính phi tuyến vào mô hình.

Hàm phi tuyến ReLU có đầu ra là: $f(x) = \max(0, x)$. Hàm ReLU quan trọng bởi ReLU giới thiệu tính phi tuyến trong ConvNet và dữ liệu trong thế giới chúng ta tìm hiểu là các giá trị tuyến tính không âm. Một số hàm phi tuyến tính khác như 'tanh', 'sigmoid' có thể thay thế ReLU nhưng có hiệu suất thấp hơn ReLU.

3.1.2 Lớp gộp - Pooling layer

Lớp gộp làm giảm kích thước, số lượng tham số đầu vào bằng cách quét toàn bộ tham số đầu vào và bộ lọc này không có trọng số. Tuy nhiên, kernel áp dụng hàm tổng

hợp cho các giá trị trong trường tiếp nhận, điền vào mảng đầu ra. Có hai loại gộp nhóm chính:

- Nhóm tối đa: Bộ lọc đến đầu vào, chọn pixel có giá trị tối đa để gửi đến mảng đầu ra. Cách tiếp cận này được sử dụng nhiều hơn so với cách gộp trung bình.
- Nhóm trung bình: Bộ lọc tới đầu vào, tính toán giá trị trung bình trong trường tiếp nhận để gửi đến mảng đầu ra.

Tuy nhiên thông tin bị mất trong lớp tổng hợp nhưng nó mang lại lợi ích về việc nâng cao hiệu quả, hạn chế nguy cơ trang bị quá mức, giảm độ phức tạp.

3.1.3 Lớp được kết nối đầy đủ - Full Connected

Tên lớp đã mô tả chính xác bản thân lớp được kết nối đầy đủ. Mỗi nút lớp đầu ra trong lớp kết nối đầy đủ đều kết nối trực tiếp với nút ở lớp trước đó.

Lớp thực hiện phân loại dựa trên đặc điểm được trích xuất qua các lớp trước và các bộ lọc khác nhau của chúng. Lớp kết nối đầy đủ thường kích hoạt softmax để phân loại đầu vào một cách thích hợp, tạo ra xác suất từ 0 đến 1.

CNN tăng cường khả năng nhận dạng hình ảnh và nhiệm vụ cho thị giác máy tính.

3.2. Xây dựng mô hình

Hàm `build_model` định nghĩa kiến trúc của mạng nơ-ron tích chập (CNN) cho việc nhận dạng biểu cảm khuôn mặt.

3.2.1. Lớp thứ nhất (1st Layer)

Lớp tích chập với 64 bộ lọc kích thước (5, 5), kích hoạt ReLU, và chuẩn hóa lô. Các lớp MaxPooling và Dropout được thêm vào để chính quy hóa:

- **Lớp tích chập (Conv2D):**
 - Số bộ lọc: 64
 - Kích thước bộ lọc: (5, 5)
 - Hàm kích hoạt: ReLU
 - Đệm (padding): 'same' (giữ nguyên kích thước không gian sau tích chập)
 - Kích thước đầu vào: (48, 48, 1) (ảnh thang độ xám 48x48)
- **Chuẩn hóa lô (BatchNormalization):**
 - Chuẩn hóa đầu ra từ lớp tích chập để tăng tốc độ huấn luyện và ổn định mô hình.

- **Lớp lấy mẫu cực đại (MaxPooling2D):**
 - Kích thước cửa sổ: (2, 2)
 - Giảm kích thước không gian đầu ra từ lớp tích chập.
- **Lớp dropout (Dropout):**
 - Tỷ lệ dropout: 0.3
 - Giảm thiểu overfitting bằng cách bỏ ngẫu nhiên một số đơn vị trong quá trình huấn luyện.

3.2.2. Lớp thứ hai (2nd Layer)

Lớp tích chập với 128 bộ lọc kích thước (3, 3), kích hoạt ReLU, và chuẩn hóa lô. Các lớp MaxPooling và Dropout được thêm vào để chính quy hóa.

- **Lớp tích chập (Conv2D):**
 - Số bộ lọc: 128
 - Kích thước bộ lọc: (3, 3)
 - Hàm kích hoạt: ReLU
 - Đệm (padding): 'same'
- **Chuẩn hóa lô (BatchNormalization):**
 - Chuẩn hóa đầu ra từ lớp tích chập.
- **Lớp lấy mẫu cực đại (MaxPooling2D):**
 - Kích thước cửa sổ: (2, 2)
- **Lớp dropout (Dropout):**
 - Tỷ lệ dropout: 0.3

3.2.3. Lớp thứ ba (3rd Layer)

- Lớp tích chập với 512 bộ lọc kích thước (3, 3), kích hoạt ReLU, và chuẩn hóa lô. Các lớp MaxPooling và Dropout được thêm vào để chính quy hóa.
- **Lớp tích chập (Conv2D):**
 - Số bộ lọc: 512
 - Kích thước bộ lọc: (3, 3)
 - Hàm kích hoạt: ReLU

- Đệm (padding): 'same'
- **Chuẩn hóa lô (BatchNormalization):**
 - Chuẩn hóa đầu ra từ lớp tích chập.
- **Lớp lấy mẫu cực đại (MaxPooling2D):**
 - Kích thước cửa sổ: (2, 2)
- **Lớp dropout (Dropout):**
 - Tỷ lệ dropout: 0.3

3.2.4. Lớp thứ tư (4th Layer)

Lớp tích chập với 512 bộ lọc kích thước (3, 3), kích hoạt ReLU, và chuẩn hóa lô. Các lớp MaxPooling và Dropout được thêm vào để chính quy hóa.

- **Lớp tích chập (Conv2D):**
 - Số bộ lọc: 512
 - Kích thước bộ lọc: (3, 3)
 - Hàm kích hoạt: ReLU
 - Đệm (padding): 'same'
- **Chuẩn hóa lô (BatchNormalization):**
 - Chuẩn hóa đầu ra từ lớp tích chập.
- **Lớp lấy mẫu cực đại (MaxPooling2D):**
 - Kích thước cửa sổ: (2, 2)
- **Lớp dropout (Dropout):**
 - Tỷ lệ dropout: 0.3

3.2.5. Lớp làm phẳng (Flatten Layer)

Làm phẳng đầu ra từ các lớp tích chập để đưa vào các lớp kết nối đầy đủ.

- **Lớp làm phẳng (Flatten):**
 - Chuyển đổi đầu ra từ các lớp tích chập (3D) thành một vector (1D) để đưa vào các lớp kết nối đầy đủ.

3.2.6. Lớp kết nối đầy đủ thứ nhất (Fully Connected Layer 1)

Lớp Dense với 256 đơn vị và kích hoạt ReLU, tiếp theo là chuẩn hóa lô và dropout để chính quy hóa.

- **Lớp Dense:**
 - Số đơn vị: 256
 - Hàm kích hoạt: ReLU
- **Chuẩn hóa lô (BatchNormalization):**
 - Chuẩn hóa đầu ra từ lớp Dense.
- **Lớp dropout (Dropout):**
 - Tỷ lệ dropout: 0.3

3.2.7. Lớp kết nối đầy đủ thứ hai (Fully Connected Layer 2)

Lớp Dense với 512 đơn vị và kích hoạt ReLU, tiếp theo là chuẩn hóa lô và dropout để chính quy hóa.

- **Lớp Dense:**
 - Số đơn vị: 512
 - Hàm kích hoạt: ReLU
- **Chuẩn hóa lô (BatchNormalization):**
 - Chuẩn hóa đầu ra từ lớp Dense.
- **Lớp dropout (Dropout):**
 - Tỷ lệ dropout: 0.3

3.2.8. Lớp đầu ra (Output Layer)

Lớp Dense với kích hoạt softmax cho phân loại đa lớp, với số đơn vị bằng số lớp trong tập dữ liệu.

- **Lớp Dense:**
 - Số đơn vị: số lớp trong tập dữ liệu (num_classes)
 - Hàm kích hoạt: Softmax (cho phân loại đa lớp)

3.2.9. Biên dịch mô hình (Compilation)

Mô hình được biên soạn với tối ưu hóa Adam, hàm mất mát cross-entropy phân loại danh mục và độ chính xác là độ đo.

- **Tối ưu hóa (Optimizer):**
 - Sử dụng Adam với tốc độ học: 0.001
- **Hàm mất mát (Loss):**
 - Cross-entropy phân loại danh mục (categorical_crossentropy)
- **Độ đo (Metric):**
 - Độ chính xác (accuracy)

```
# Building Model
def build_model():
    model = Sequential()
    # 1st Layer
    model.add(Conv2D(64, (5, 5), strides=(1, 1), padding='same',
activation='relu', input_shape=(48, 48, 1)))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(2, 2))
    model.add(Dropout(0.3))

    # 2nd Layer
    model.add(Conv2D(128, (3, 3), strides=(1, 1), padding='same',
activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(2, 2))
    model.add(Dropout(0.3))

    # 3rd layer
    model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same',
activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(2, 2))
    model.add(Dropout(0.3))

    # 4th layer
    model.add(Conv2D(512, (3, 3), strides=(1, 1), padding='same',
activation='relu'))
    model.add(BatchNormalization())
```

```

model.add(MaxPooling2D(2, 2))
model.add(Dropout(0.3))

# Flatten Layer
model.add(Flatten())

# Fully connected layer 1
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

# Fully connected layer 2
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

# Output layer
model.add(Dense(num_classes, activation='softmax'))

# Compiling the model
model.compile(optimizer=Adam(learning_rate=0.001),
loss='categorical_crossentropy', metrics=['accuracy'])

return model

```

```

# Build the model
model = build_model()

# Print model summary
print(model.summary())

```

Phương thức `summary()` được gọi trên mô hình đã xây dựng để hiển thị một tóm tắt về kiến trúc của nó, bao gồm các lớp, hình dạng đầu ra và số lượng tham số có thể huấn luyện.

...

Model: "sequential"

...

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	1,664
batch_normalization (BatchNormalization)	(None, 48, 48, 64)	256
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	73,856
batch_normalization_1 (BatchNormalization)	(None, 24, 24, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590,336
batch_normalization_2 (BatchNormalization)	(None, 12, 12, 512)	2,048
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 512)	0
dropout_2 (Dropout)	(None, 6, 6, 512)	0
conv2d_3 (Conv2D)	(None, 6, 6, 512)	2,359,808
batch_normalization_3 (BatchNormalization)	(None, 6, 6, 512)	2,048
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1,179,904
batch_normalization_4 (BatchNormalization)	(None, 256)	1,024
dropout_4 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131,584
batch_normalization_5 (BatchNormalization)	(None, 512)	2,048
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 7)	3,591

...

Total params: 4,348,679 (16.59 MB)

...

Trainable params: 4,344,711 (16.57 MB)

...

Non-trainable params: 3,968 (15.50 KB)

Hình 3. Kiến trúc mô hình

3.2.10. Checkpoint

Đối tượng gọi lại **ModelCheckpoint** được khởi tạo để lưu trọng số của mô hình trong quá trình huấn luyện. Nó theo dõi độ chính xác của validation (**val_acc**) và chỉ lưu mô hình tốt nhất (được xác định bởi độ chính xác validation tối đa) vào tệp được chỉ định **"model.keras"**.

```
checkpoint = ModelCheckpoint("model.keras", monitor="val_acc", verbose=1,
save_best_only=True)
```

3.2.11. EarlyStopping

Đối tượng gọi lại **EarlyStopping** được khởi tạo để theo dõi sự mất mát trên tập validation (**val_loss**). Nó dừng quá trình huấn luyện nếu sự mất mát trên tập validation không cải thiện trong một số epoch nhất định (**patience**), được chỉ định bởi người dùng. Quá trình huấn luyện sẽ được dừng sớm để ngăn chặn hiện tượng quá mức và các trọng số của mô hình hoạt động tốt nhất sẽ được khôi phục (**restore_best_weights=True**).

```
early_stopping = EarlyStopping(
    monitor='val_loss',
    min_delta=0,
    patience=3,
    verbose=1,
    restore_best_weights=True
)
```

3.2.12. ReduceLROnPlateau

Đối tượng gọi lại **ReduceLROnPlateau** được khởi tạo để điều chỉnh động tốc độ học trong quá trình huấn luyện dựa trên sự mất mát trên tập validation (**val_loss**). Nếu sự mất mát trên tập validation không cải thiện trong một số epoch nhất định (**patience**), tốc độ học sẽ được giảm đi một hệ số được chỉ định bởi người dùng (**factor**). Điều này giúp cải thiện quá trình huấn luyện và ngăn mô hình bị kẹt trong các cực tiểu cục bộ.

```
reduce_learningrate = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=3,
    verbose=1,
    min_delta=0.0001
)
```

3.2.13. Callback List

Danh sách `callbacks_list` là một danh sách chứa các gọi lại được sử dụng trong quá trình huấn luyện của mô hình. Nó bao gồm các gọi lại `EarlyStopping`, `ModelCheckpoint`, và `ReduceLROnPlateau`, được sử dụng để theo dõi sự mất mát trên tập validation, lưu mô hình tốt nhất, và điều chỉnh tốc độ học, tương ứng.

```
callbacks_list = [early_stopping, checkpoint, reduce_learningrate]
```

3.2.14. Cấu hình huấn luyện

Phương thức `compile()` được gọi trên mô hình để cấu hình quá trình huấn luyện. Nó xác định hàm mất mát, bộ tối ưu hóa và các đánh giá độc lập sẽ được sử dụng trong quá trình huấn luyện.

- **Hàm Mất Mát:** Cross-entropy phân loại danh mục được sử dụng làm hàm mất mát cho các vấn đề phân loại đa lớp.
- **Bộ Tối Ưu Hóa:** Bộ tối ưu hóa Adam được chọn với tốc độ học là 0.001.
- **Đánh Giá:** Độ chính xác được sử dụng làm độc lập để theo dõi hiệu suất của mô hình trong quá trình huấn luyện.

```
model.compile(  
    loss='categorical_crossentropy',  
    optimizer=Adam(learning_rate=0.001),  
    metrics=['accuracy']  
)
```

3.3. Huấn luyện mô hình

Phương thức `fit()` được gọi trên mô hình để huấn luyện nó trên dữ liệu huấn luyện. Nó nhận các tham số sau:

- `data_gen.flow(X_train, y_train_categorical, batch_size=128)`: Một bộ sinh ra mẫu mà cung cấp các lô dữ liệu huấn luyện được tăng cường. Tăng cường dữ liệu được áp dụng ngay lập tức bằng cách sử dụng đối tượng `ImageDataGenerator` được xác định trước.
- `validation_data=(X_val, y_val_categorical)`: Dữ liệu validation để đánh giá hiệu suất của mô hình sau mỗi epoch.
- `epochs=30`: Số lượng epoch để huấn luyện mô hình.
- `verbose=1`: Xác định chế độ verbose. Ở đây, nó in thanh tiến trình trong quá trình huấn luyện.

```
history = model.fit(  

```

```
data_gen.flow(X_train, y_train_categorical, batch_size=128),
validation_data=(X_val, y_val_categorical),
epochs=30,
verbose=1
)
```

```
# Training the model
history = model.fit(
    data_gen.flow(X_train, y_train_categorical, batch_size=128),
    validation_data=(X_val, y_val_categorical),
    epochs=30,
    verbose=1
)
```

[22] Python

```
... Epoch 1/30
C:\Users\theng\anaconda3\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call
self._warn_if_super_not_called()
163/163 ----- 146s 871ms/step - accuracy: 0.1884 - loss: 2.3977 - val_accuracy: 0.2515 - val_loss: 2.4737
Epoch 2/30
163/163 ----- 138s 844ms/step - accuracy: 0.2501 - loss: 1.9346 - val_accuracy: 0.2398 - val_loss: 2.1800
Epoch 3/30
163/163 ----- 139s 850ms/step - accuracy: 0.3142 - loss: 1.7632 - val_accuracy: 0.3066 - val_loss: 2.0319
Epoch 4/30
163/163 ----- 137s 842ms/step - accuracy: 0.3801 - loss: 1.6045 - val_accuracy: 0.3938 - val_loss: 1.5525
Epoch 5/30
163/163 ----- 146s 897ms/step - accuracy: 0.3987 - loss: 1.5664 - val_accuracy: 0.3643 - val_loss: 1.6734
Epoch 6/30
163/163 ----- 137s 842ms/step - accuracy: 0.4363 - loss: 1.4670 - val_accuracy: 0.4341 - val_loss: 1.4743
Epoch 7/30
163/163 ----- 139s 852ms/step - accuracy: 0.4558 - loss: 1.4065 - val_accuracy: 0.4705 - val_loss: 1.3765
Epoch 8/30
163/163 ----- 145s 891ms/step - accuracy: 0.4847 - loss: 1.3375 - val_accuracy: 0.5165 - val_loss: 1.3030
Epoch 9/30
163/163 ----- 148s 907ms/step - accuracy: 0.4988 - loss: 1.2995 - val_accuracy: 0.5286 - val_loss: 1.2411
Epoch 10/30
163/163 ----- 150s 920ms/step - accuracy: 0.5147 - loss: 1.2605 - val_accuracy: 0.5551 - val_loss: 1.1819
Epoch 11/30
163/163 ----- 149s 912ms/step - accuracy: 0.5335 - loss: 1.2363 - val_accuracy: 0.5473 - val_loss: 1.1740
Epoch 12/30
163/163 ----- 146s 891ms/step - accuracy: 0.5365 - loss: 1.2117 - val_accuracy: 0.5499 - val_loss: 1.1888
Epoch 13/30
163/163 ----- 139s 850ms/step - accuracy: 0.5403 - loss: 1.2017 - val_accuracy: 0.5308 - val_loss: 1.2538
...
Epoch 29/30
163/163 ----- 148s 905ms/step - accuracy: 0.6345 - loss: 0.9684 - val_accuracy: 0.6136 - val_loss: 1.0206
Epoch 30/30
163/163 ----- 144s 880ms/step - accuracy: 0.6379 - loss: 0.9521 - val_accuracy: 0.5876 - val_loss: 1.0641
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Hình 4. Train Model

3.4. Lưu mô hình

Mô hình đã được huấn luyện được lưu vào một tệp có tên là "modelv1.h5" bằng cách sử dụng phương thức `save()`. Tệp này chứa kiến trúc của mô hình, trọng số, và cấu hình huấn luyện, cho phép bạn tải lại mô hình sau này để sử dụng cho việc dự đoán hoặc huấn luyện tiếp theo.

```
model.save('modelv1.h5')
```

Chương 4. Đánh giá mô hình

4.1. Biểu đồ độ chính xác

Sử dụng thư viện matplotlib để vẽ biểu đồ, giúp hình dung hiệu suất của mô hình học máy trong quá trình huấn luyện và kiểm tra. Đặc biệt tập trung vào việc hiển thị độ chính xác (accuracy) của mô hình trong suốt các epoch (lượt huấn luyện).

```
plt.plot(history.history['accuracy'],label='train_accuracy', marker='o')
plt.plot(history.history['val_accuracy'],label='val_accuracy',marker='o')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

plt.plot(history.history['accuracy'], label='train_accuracy', marker='o'):

- `plt.plot()`: Hàm này dùng để vẽ một đường trên biểu đồ.
- `history.history['accuracy']`: `history` là đối tượng chứa kết quả huấn luyện của mô hình (thường được trả về bởi phương thức `fit` của một mô hình trong Keras). `history.history` là từ điển chứa các giá trị của các chỉ số được theo dõi trong quá trình huấn luyện. `history.history['accuracy']` là danh sách các giá trị độ chính xác trên tập huấn luyện qua từng epoch.
- `label='train_accuracy'`: Gán nhãn cho đường này là "train_accuracy" (độ chính xác trên tập huấn luyện).
- `marker='o'`: Mỗi điểm dữ liệu trên đường này sẽ được đánh dấu bằng ký hiệu 'o' (hình tròn).

plt.plot(history.history['val_accuracy'], label='val_accuracy', marker='o'):

- Tương tự như trên, nhưng lần này `history.history['val_accuracy']` là danh sách các giá trị độ chính xác trên tập kiểm tra (validation) qua từng epoch.
- `label='val_accuracy'`: Gán nhãn cho đường này là "val_accuracy" (độ chính xác trên tập kiểm tra).
- `marker='o'`: Mỗi điểm dữ liệu trên đường này cũng sẽ được đánh dấu bằng ký hiệu 'o' (hình tròn).

plt.title('Model Accuracy'):

- `plt.title()`: Đặt tiêu đề cho biểu đồ.

- 'Model Accuracy': Tiêu đề của biểu đồ là "Model Accuracy" (Độ chính xác của mô hình).

plt.xlabel('Epochs'):

- plt.xlabel(): Đặt nhãn cho trục x.
- 'Epochs': Nhãn trục x là "Epochs" (các lượt huấn luyện).

plt.ylabel('Accuracy'):

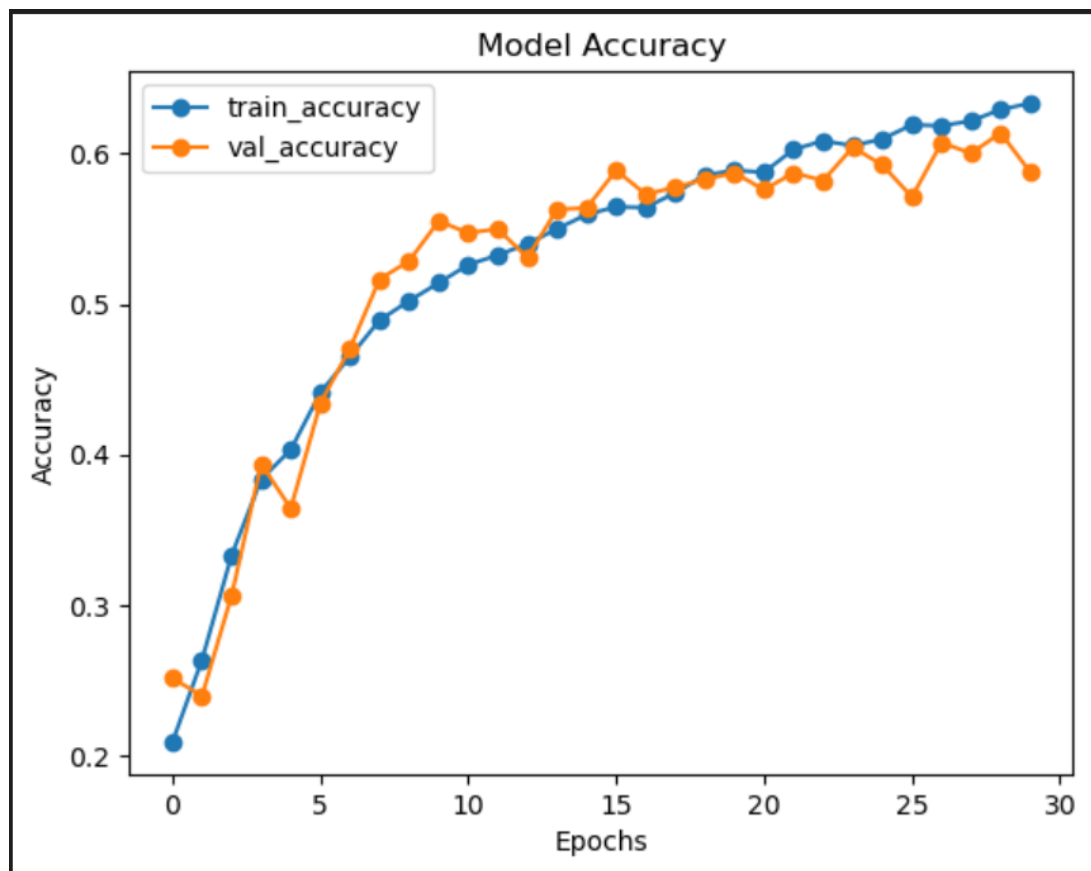
- plt.ylabel(): Đặt nhãn cho trục y.
- 'Accuracy': Nhãn trục y là "Accuracy" (Độ chính xác).

plt.legend():

- plt.legend(): Hiện thị chú thích (legend) trên biểu đồ. Chú thích sẽ sử dụng các nhãn được gán bởi các tham số label trong các lệnh plt.plot() ở trên.

plt.show():

- plt.show(): Hiện thị biểu đồ đã được vẽ lên màn hình.



Hình 5. Biểu đồ tương quan giữa Epoch và độ chính xác.

Kết luận:

1. Đường "train_accuracy" (độ chính xác trên tập huấn luyện)

Đường này cho biết mức độ chính xác của mô hình khi áp dụng trên dữ liệu huấn luyện. Mỗi điểm trên đường này tương ứng với độ chính xác đạt được sau mỗi epoch.

- **Đường đi lên:** Độ chính xác trên tập huấn luyện tăng dần theo từng epoch, điều này cho thấy mô hình đang học tốt từ dữ liệu huấn luyện.
- **Độ chính xác cao:** Nếu đường đạt độ chính xác rất cao (gần 1.0), điều này cho thấy mô hình có thể đã học rất kỹ từ dữ liệu huấn luyện.

2. Đường "val_accuracy" (độ chính xác trên tập kiểm tra)

Đường này cho biết mức độ chính xác của mô hình khi áp dụng trên dữ liệu kiểm tra (validation). Tương tự như trên, mỗi điểm trên đường này tương ứng với độ chính xác đạt được sau mỗi epoch khi áp dụng trên dữ liệu chưa được sử dụng trong quá trình huấn luyện.

- **Đường đi lên:** Độ chính xác trên tập kiểm tra tăng dần theo từng epoch, điều này cho thấy mô hình đang tổng quát hóa tốt hơn với dữ liệu chưa từng thấy.
- **Độ chính xác cao:** Nếu đường đạt độ chính xác cao nhưng không quá gần với giá trị của tập huấn luyện, điều này cho thấy mô hình đang tổng quát hóa tốt.

3. So sánh giữa "train_accuracy" và "val_accuracy"

Cả hai đường đều tăng và đạt giá trị cao:

Điều này cho thấy mô hình đang học tốt từ dữ liệu huấn luyện và tổng quát hóa tốt với dữ liệu kiểm tra. Đây là dấu hiệu của một mô hình mạnh mẽ và cân bằng.

4. Các giai đoạn chính trong quá trình huấn luyện

- **Giai đoạn ban đầu:**
 - Trong các epoch đầu tiên, cả hai đường thường tăng nhanh. Điều này là do mô hình đang học các mẫu cơ bản từ dữ liệu.
- **Giai đoạn ổn định:**
 - Sau một số epoch, tốc độ tăng của các đường có thể chậm lại và đạt đến điểm bão hòa. Điều này cho thấy mô hình đang tiếp cận giới hạn khả năng học từ dữ liệu hiện tại.

4.2. Báo cáo phân loại

Để đánh giá hiệu suất của mô hình, nhóm đã sử dụng một số chỉ số phổ biến trong học máy. Các chỉ số này bao gồm độ chính xác (precision), độ nhạy (recall), và điểm F1. Báo cáo phân loại cho thấy sự đánh giá chi tiết về các chỉ số này cho từng lớp của dữ liệu.

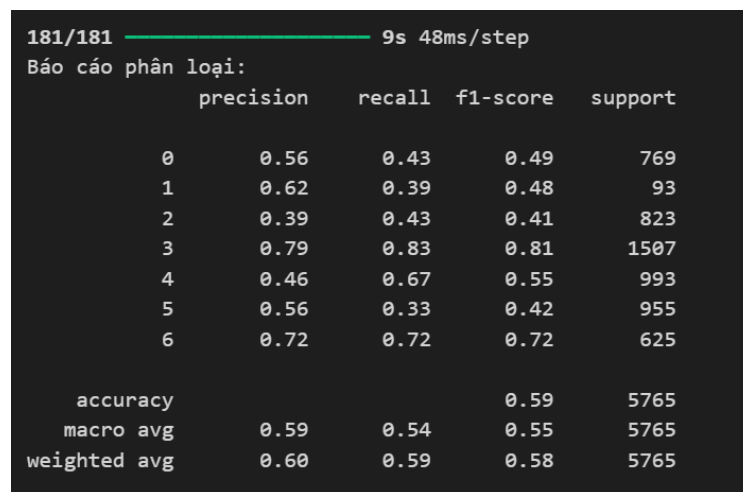
- **Precision:** Độ chính xác của mô hình trong việc dự đoán đúng các mẫu thuộc một lớp nhất định. Precision cao nghĩa là ít các trường hợp dự đoán sai thuộc lớp đó.
- **Recall:** Khả năng của mô hình phát hiện đúng các mẫu thực sự thuộc lớp đó. Recall cao nghĩa là mô hình ít bỏ sót các mẫu thuộc lớp đó.
- **F1-score:** Trung bình điều hòa giữa precision và recall, cung cấp một chỉ số cân bằng hơn khi có sự chênh lệch giữa hai chỉ số này.

Dưới đây là đoạn mã để tạo báo cáo phân loại:

```
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score, roc_curve, log_loss, accuracy_score
import seaborn as sns

y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = y_test

print("Báo cáo phân loại:")
print(classification_report(y_true, y_pred_classes))
```



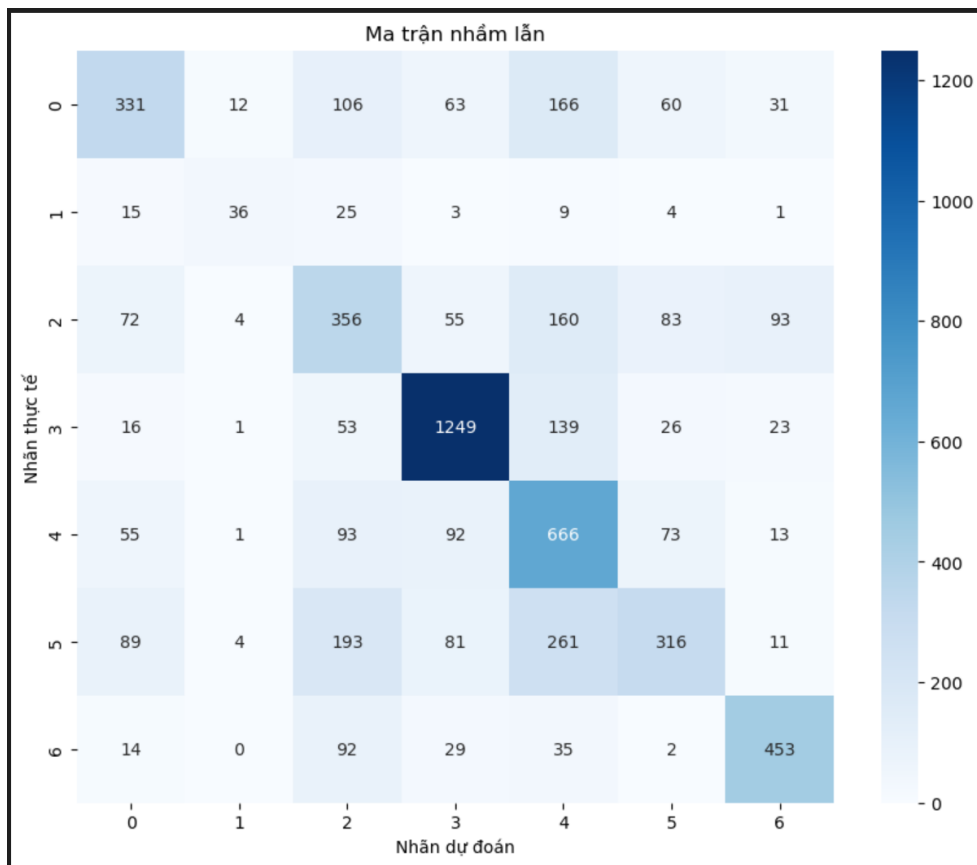
	precision	recall	f1-score	support
0	0.56	0.43	0.49	769
1	0.62	0.39	0.48	93
2	0.39	0.43	0.41	823
3	0.79	0.83	0.81	1507
4	0.46	0.67	0.55	993
5	0.56	0.33	0.42	955
6	0.72	0.72	0.72	625
accuracy			0.59	5765
macro avg	0.59	0.54	0.55	5765
weighted avg	0.60	0.59	0.58	5765

Hình 6. Kết quả hiển thị.

4.3. Ma trận nhầm lẫn

Ma trận nhầm lẫn là công cụ hữu ích để trực quan hóa hiệu suất của mô hình. Nó cho thấy số lượng dự đoán đúng và sai của mô hình cho mỗi lớp, giúp chúng ta nhận ra các lỗi phổ biến mà mô hình mắc phải. Thông qua ma trận nhầm lẫn, chúng ta có thể hiểu rõ hơn về việc mô hình đã hoạt động như thế nào, cụ thể là mô hình đã nhận diện chính xác bao nhiêu mẫu cho mỗi lớp và đã nhầm lẫn bao nhiêu mẫu giữa các lớp khác nhau.

```
conf_matrix = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Ma trận nhầm lẫn')
plt.xlabel('Nhãn dự đoán')
plt.ylabel('Nhãn thực tế')
plt.show()
```



Hình 7. Biểu đồ ma trận nhầm lẫn.

Cấu trúc của Ma trận:

- Các hàng đại diện cho các lớp thực tế (Nhãn thực tế).
- Các cột đại diện cho các lớp được dự đoán (Nhãn dự đoán).
- Các phần tử trên đường chéo (từ trên trái đến dưới phải) đại diện cho số lượng dự đoán đúng cho mỗi lớp.
- Các phần tử ngoài đường chéo đại diện cho các dự đoán sai.

Kết luận:

- Mô hình có độ chính xác cao nhất khi dự đoán lớp 3, với 1249 dự đoán đúng.
- Các lớp 1 và 6 có số lượng dự đoán đúng thấp nhất, chỉ ra những khu vực cần cải thiện.
- Các dự đoán sai lan rộng qua nhiều lớp, với sự nhầm lẫn đáng kể giữa các lớp cụ thể (ví dụ: lớp 2 thường bị nhầm với lớp 0, 4 và 6).

Chương 5. Triển khai mô hình

Phần này của báo cáo sẽ trình bày chi tiết về việc triển khai mô hình phát hiện cảm xúc khuôn mặt từ camera thời gian thực, sử dụng thư viện Keras và OpenCV. Đoạn mã chính thực hiện việc này được lưu trong tệp main.py.

5.1 Classifier Cascade

Cascade Classifier là một thuật toán học máy thường ứng dụng trong phát hiện, nhận dạng đối tượng, đặc biệt là trong các nhiệm vụ nhận diện khuôn mặt.

Vào năm 2001, Paul Viola và Michael Jones đề xuất Cascade Classifier trong bài báo của họ về một phương pháp phát hiện đối tượng (Object Detection) hiệu quả bằng cách sử dụng bộ phận phân loại xếp tầng dựa trên tính năng Haar: “Rapid Object Detection using a Boosted Cascade of Simple Features” (OpenCV, 2024). Thuật toán được thiết kế để có hiệu quả tính toán và có khả năng phát hiện theo thời gian thực.

Haar-like features: Cascade Classifier dựa trên các đặc trưng giống Haar, là tính năng hình chữ nhật đơn giản mã hóa các mẫu cường độ cục bộ trong ảnh. Các đặc trưng này nắm bắt các biến thể về độ sáng và tối ở các vùng khác nhau của ảnh (Soulpage, 2024).

5.2 Nhận diện khuôn mặt

Phương pháp sử dụng *Cascade Classifier*— một lớp của OpenCV - được nhóm lựa chọn để phát hiện khuôn mặt xuất hiện trong khung hình hay xác định vị trí khuôn mặt trong ảnh. Đây không phải một kỹ thuật trích chọn đặc trưng nhưng là một bước trích chọn đặc trưng trước khi thực hiện phân loại cảm xúc bằng mô hình học sâu đã huấn luyện.

```
# Load pre-trained model và cascade classifier cho việc nhận dạng khuôn mặt
face_classifier = cv2.CascadeClassifier(r'D:\HocKy6\AI02\BTL_AI-
main_final\BTL_AI-main\haarcascade_frontalface_default.xml')
```

'haarcascade_frontalface_default.xml' là tệp XML chứa các thông tin về mô hình cascade classifier dùng để phát hiện khuôn mặt. Tệp XML này chứa các giá trị trọng số đã được huấn luyện trên nhiều hình ảnh để có thể nhận diện khuôn mặt một cách hiệu quả.

Khi thực thi dòng lệnh, OpenCV sẽ tải cascade classifier từ tệp XML. Sau đó, classifier để phát hiện khuôn mặt trong các hình ảnh hoặc video frames mà người dùng cung cấp.

Cách Cascade Classifier hoạt động:

- Cascade classifier hoạt động bằng cách áp dụng một chuỗi các bộ phân loại đơn giản (weak classifiers) tuần tự trên một vùng ảnh cụ thể.
- Mỗi bộ phân loại kiểm tra xem vùng ảnh đó có khả năng chứa khuôn mặt hay không.
- Nếu vùng ảnh vượt qua tất cả các bộ phân loại trong chuỗi, nó được coi là một khuôn mặt.
- Quá trình này rất nhanh và hiệu quả, vì mỗi bộ phân loại chỉ kiểm tra một số đặc trưng đơn giản, giúp nhận diện khuôn mặt trong thời gian thực.

Một mô hình học máy được huấn luyện sẵn và lưu trữ trong tệp. Sử dụng hàm 'load_model' trong thư viện Keras chạy trên nền tảng TensorFlow. Hàm này tải và khôi phục toàn bộ mô hình, bao gồm cả cấu trúc và trọng số của nó.

Tệp 'modelv1.h5' là tệp lưu trữ mô hình học sâu được huấn luyện. Tệp '.h5' là định dạng tệp được Keras sử dụng để lưu trữ mô hình bao gồm cả cấu trúc mạng nơ-ron và các trọng số đã học.

```
classifier = load_model(r'D:\HocKy6\AI02\BTL_AI-main_final\BTL_AI-main\modelv1.h5')
```

Tạo một danh sách chứa các nhãn cảm xúc mà mô hình cần dự đoán. Các nhãn bao gồm 7 cảm xúc cơ bản của con người là Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise. Sau khi mô hình dự đoán, kết quả dự đoán sẽ được ánh xạ tới một trong các nhãn này để xác định cảm xúc khuôn mặt trong hình ảnh.

```
# Danh sách nhãn cảm xúc
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
```

5.3 Tiền xử lý ảnh khuôn mặt

Chuyển đổi ảnh sang định dạng xám (grayscale) để tăng tốc độ xử lý, tăng hiệu suất của mô hình. Chuyển ảnh màu sang ảnh xám bởi ảnh xám chỉ có một kênh màu thay vì ba kênh (BGR), kích thước dữ liệu giảm đi một phần ba.

Tham số 'cv2.cvtColor' là hàm có sẵn trong OpenCV dùng để chuyển đổi không gian màu của hình ảnh. 'cv2.COLOR_BGR2GRAY' là tham số xác định loại màu

chuyển đổi sang mà ở đây chuyển từ ảnh màu (BGR) sang không gian màu xám (grayscale).

```
# Chuyển đổi ảnh sang định dạng xám để tăng tốc độ xử lý
gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

# Phát hiện khuôn mặt trong ảnh
faces = face_classifier.detectMultiScale(gray)
```

Dùng bộ phân loại Haar Cascade với đối tượng ‘face_classifier’ đã khởi tạo trước đó với mục đích phát hiện khuôn mặt. Phương thức ‘detectMultiScale’ dùng để phát hiện khuôn mặt trong khung hình. Phương thức này trả về một danh sách các hình chữ nhật mà mỗi hình chữ nhật đại diện cho một khuôn mặt được phát hiện trong khung hình.

Trước khi ảnh khuôn mặt được đưa vào mô hình dự đoán cần phải cắt và thay đổi kích thước ảnh. Cần trích xuất khuôn mặt từ ảnh xám, nghĩa là lấy ra khuôn mặt từ toàn bộ khung hình, chỉ giữ lại vùng chứa khuôn mặt. Sau đó thực hiện điều chỉnh kích thước khuôn mặt về kích thước chuẩn (48x48 pixel) để phù hợp với yêu cầu đầu vào của mô hình dự đoán cảm xúc.

```
# Trích xuất khu vực chứa khuôn mặt từ ảnh xám
roi_gray = gray[y:y+h,x:x+w]
# Chuyển đổi kích thước khu vực khuôn mặt để phù hợp với mô hình
roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)
```

- Biến ‘roi_gray’ (Region of Interest in Gray) lưu trữ vùng ảnh xám chứa khuôn mặt được trích xuất.
- Tham số (48,48) là kích thước mới của hình ảnh (chiều rộng và chiều cao), ảnh ở đây sẽ có kích thước cố định đều bằng 48x48 pixel.
- ‘interpolation=cv2.INTER_AREA’ là phương pháp nội suy (interpolation) được sử dụng khi thay đổi kích thước hình ảnh.

Sau khi thực hiện hai lệnh này, kết quả thu được ‘roi_gray’ là ảnh xám của khuôn mặt đã được thay đổi kích thước về 48x48 pixel, sẵn sàng cho bước tiếp theo là chuẩn bị đầu vào cho mô hình dự đoán cảm xúc.

5.4 Chuẩn hóa dữ liệu

Chuẩn hóa giá trị pixel về khoảng [0, 1] để mô hình hoạt động tốt hơn khi đầu vào được chuẩn hóa, các thuật toán hội tụ nhanh hơn, ổn định hơn bởi các giá trị đầu vào

trong khoảng cố định và nhỏ hơn giá trị ban đầu. Ngoài tác dụng tăng cường hiệu quả của mô hình học sâu, chuẩn hóa còn giúp tránh tràn số.

```
# Chuẩn hóa khu vực khuôn mặt và chuyển đổi thành numpy array
roi = roi_gray.astype('float')/255.0
```

- Giá trị pixel của vùng khuôn mặt đã được trích xuất và thay đổi kích thước về khoảng [0, 1].
- ‘astype('float')’ là phương thức chuyển đổi kiểu dữ liệu của mảng ‘roi_gray’ sang kiểu ‘float’. Ảnh lưu mặc định dưới dạng số nguyên 8-bit không có dấu, giá trị nằm trong khoảng từ 0 đến 255.
- Thực hiện phép chia ‘/ 255.0’ để đưa giá trị pixel về khoảng từ 0.0 đến 1.0

5.5 Chuyển đổi dữ liệu

Thực hiện quá trình chuyển đổi ảnh khuôn mặt đã chuẩn hóa thành mảng NumPy. Đồng thời thêm một chiều mới để tạo thành một batch có kích thước 1, chuẩn bị cho việc đưa vào mô hình học sâu.

```
roi = img_to_array(roi)
roi = np.expand_dims(roi,axis=0)
```

- Hàm ‘img_to_array’ từ thư viện Keras giúp chuyển đổi ảnh từ định dạng PIL hoặc định dạng tương tự thành mảng NumPy.
- ‘roi’ là ảnh khuôn mặt đã chuẩn hóa có kích thước 48x48 pixel với các giá trị pixel nằm trong khoảng [0, 1].
- Hàm ‘expand_dims’ từ thư viện NumPy thêm một chiều mới vào mảng ‘roi’ tại vị trí được chỉ định bởi ‘axis’.
- ‘axis=0’ chỉ định rằng chiều mới sẽ được thêm vào đầu mảng, tức là trước chiều hiện tại.

Nguyên nhân phải thêm chiều dữ liệu là do các mô hình học sâu (như CNN) trong Keras thường yêu cầu đầu vào có dạng batch, tức là một mảng 4D với thứ tự ‘(batch_size, height, width, channels)’. Tuy nhiên, ‘roi’ hiện tại có dạng ‘(48, 48, 1)’ (1 kênh do ảnh xám), nhưng mô hình yêu cầu đầu vào phải có dạng ‘(1, 48, 48, 1)’ (1 batch với 1 ảnh). Vậy nên sau khi thực thi dòng lệnh thì ‘roi’ sẽ có định dạng phù hợp với yêu cầu đầu vào của mô hình học sâu.

5.6 Dự đoán cảm xúc

Mô hình học sâu được huấn luyện để dự đoán cảm xúc và sau đó ánh xạ dự đoán đó thành nhãn cảm xúc tương ứng.

```
# Dự đoán cảm xúc sử dụng mô hình đã load
prediction = classifier.predict(roi)[0]

# Chọn nhãn có xác suất cao nhất
label = emotion_labels[prediction.argmax()]
```

Hàm ‘predict’ của mô hình này được sử dụng để dự đoán xác suất các cảm xúc khác nhau từ ảnh đầu vào ‘roi’.

Kết quả của ‘classifier.predict(roi)’ là một mảng 2D với kích thước ‘(1,num_classes)’, trong bài của nhóm thì ‘num_classes’ là số lượng các nhãn cảm xúc và bằng 7.

‘[0]’ để lấy mảng con đầu tiên, tức là lấy dự đoán cho ảnh duy nhất trong batch. Kết quả này là một mảng 1D với kích thước ‘(num_classes,)’, chứa xác suất cho mỗi nhãn cảm xúc.

```
from keras.models import load_model
from time import sleep
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np

face_classifier =
cv2.CascadeClassifier(r'C:\Users\theng\Documents\CODE\C#\BTL\Emotion_Detection_CNN\haarcascade_frontalface_default.xml')
classifier
=load_model(r'C:\Users\theng\Documents\CODE\C#\BTL\Emotion_Detection_CNN\model_v1.h5')

emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sad',
'Surprise']

cap = cv2.VideoCapture(0)
```

```

while True:
    _, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,255), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_gray = cv2.resize(roi_gray, (48,48), interpolation=cv2.INTER_AREA)

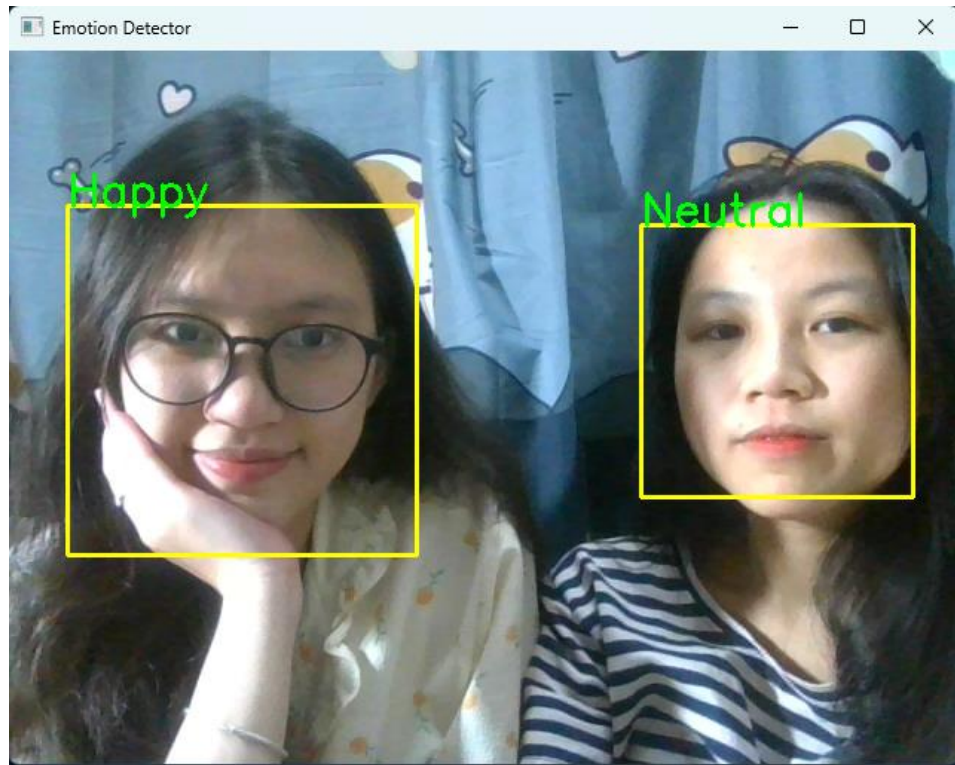
        if np.sum([roi_gray])!=0:
            roi = roi_gray.astype('float')/255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi, axis=0)

            prediction = classifier.predict(roi)[0]
            label=emotion_labels[prediction.argmax()]
            label_position = (x,y)
            cv2.putText(frame, label, label_position, cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,255,0), 2)
        else:
            cv2.putText(frame, 'No
Faces', (30,80), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
        cv2.imshow('Emotion Detector', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

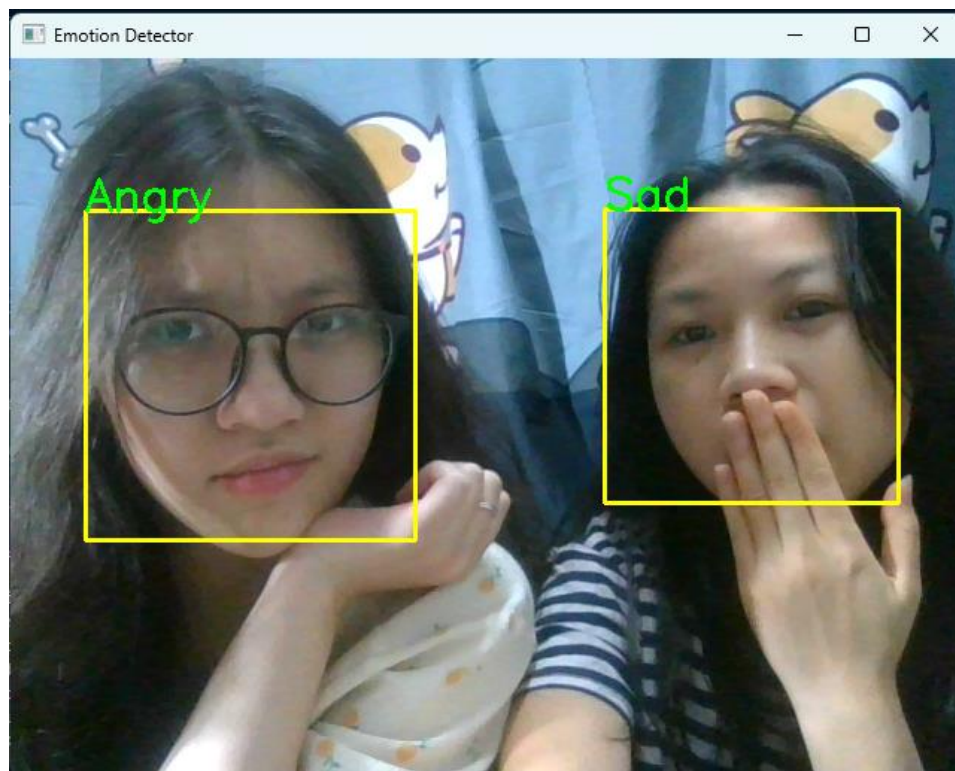
cap.release()
cv2.destroyAllWindows()

```

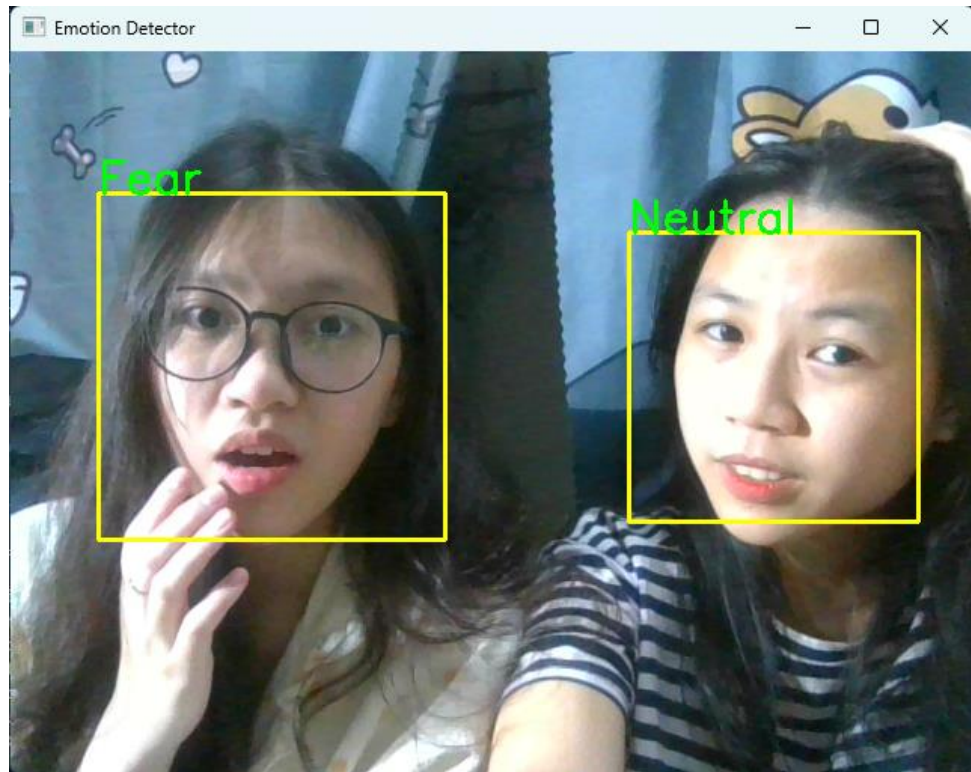
Kết quả:



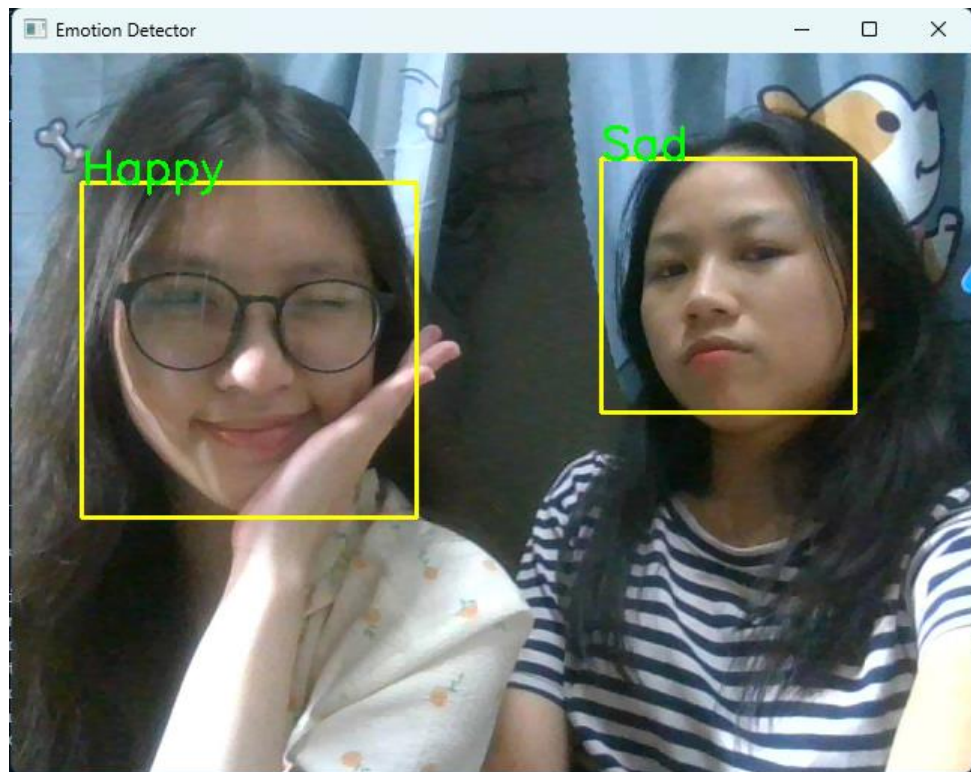
Hình 8. Kết quả chương trình 1



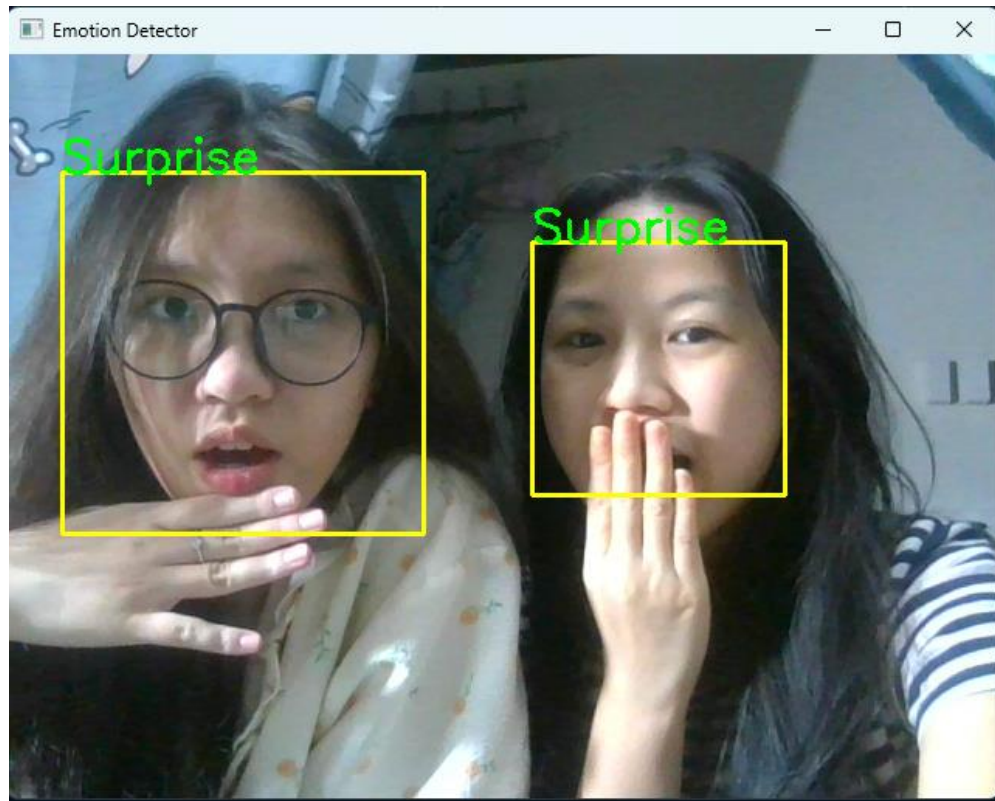
Hình 9. Kết quả chương trình 2



Hình 10. Kết quả chương trình 3



Hình 11. Kết quả chương trình 4



Hình 12. Kết quả chương trình 5

KẾT LUẬN

Thông qua bài tập nhóm thực hiện đề tài “Nhận diện biểu cảm khuôn mặt”, chúng em đã có cơ hội áp dụng kiến thức môn học Trí tuệ nhân tạo vào thực tiễn. Việc tham gia dự án này không chỉ giúp chúng em củng cố những kiến thức đã học mà còn mở rộng thêm hiểu biết về lĩnh vực trí tuệ nhân tạo. Quá trình tự tìm hiểu, nghiên cứu và thực hiện dự án đã mang lại cho chúng em nhiều kiến thức mới cùng những kinh nghiệm và kỹ năng hữu ích cho định hướng nghề nghiệp trong tương lai.

Dự án của nhóm 11 tập trung vào việc xây dựng một mô hình nhận diện biểu cảm khuôn mặt thông qua webcam. Mô hình này có khả năng nhận diện khuôn mặt người xuất hiện trong khung hình và gán nhãn cảm xúc dựa trên các đặc trưng khuôn mặt. Qua đó, chúng em đã áp dụng các kỹ thuật học máy và xử lý hình ảnh để tạo nên một sản phẩm có thể nhận biết và phân loại các trạng thái cảm xúc khác nhau từ dữ liệu hình ảnh.

Dù còn những hạn chế, chúng em tin rằng dự án này đã mang lại nhiều ý nghĩa và giá trị thực tế cho các thành viên trong nhóm. Nó không chỉ là một bài tập thực hành mà còn là một bước chuẩn bị quan trọng cho việc lựa chọn và theo đuổi nghề nghiệp trong lĩnh vực Trí tuệ nhân tạo. Nhóm 11 hy vọng rằng những kinh nghiệm và kiến thức thu được từ dự án sẽ là hành trang quý báu cho tương lai của mỗi thành viên.

TÀI LIỆU THAM KHẢO

Lan Nhi (28/02/2024). *Viettel AI đạt top 4 thế giới, top 1 Việt Nam về công nghệ nhận diện khuôn mặt*. Ngày truy cập: 11/06/2024.

Đường dẫn: <https://viettelfamily.com/news/chuyen-dich-so/viettel-ai-dat-top-4-the-gioi-top-1-viet-nam-ve-cong-nghe-nhan-dien-khuon-mat>

Thành Nguyễn (21/12/2023). *Giải pháp CIVAMS của CMC lọt top 12 thế giới và top 1 Việt Nam*. Ngày truy cập: 11/06/2024.

Đường dẫn: <https://thitruongtaichinhhtiente.vn/giai-phap-civams-cua-cmc-lot-top-12-the-gioi-va-top-1-viet-nam-55133.html>

Exactitude Consultancy (03/2022). *Thị trường phát hiện và nhận dạng cảm xúc (EDR) theo Công cụ phần mềm (Nhận dạng khuôn mặt, Nhận dạng giọng nói và giọng nói, Cảm biến sinh học), Công nghệ (Học máy, Trí tuệ nhân tạo), Ngành sử dụng cuối (Chính phủ, Chăm sóc sức khỏe, Giải trí, Giao thông vận tải, Khác) và theo Khu vực (Bắc Mỹ, Châu Âu, Châu Á Thái Bình Dương, Nam Mỹ, Trung Đông và Châu Phi), Dự báo và Xu hướng Toàn cầu từ 2019 đến 2028*. Ngày truy cập: 11/06/2024.

Đường dẫn: <https://exactitudeconsultancy.com/vi/reports/1697/emotion-detection-and-recognition-edr-market/>

Abhishek Jaiswal (10/08/2023). *Guide to Haar Cascade Algorithm with Object Detection Example*. Ngày truy cập: 12/06/2024.

Đường dẫn: <https://www.analyticsvidhya.com/blog/2022/04/object-detection-using-haar-cascade-opencv/>

IBM. *What is AI?* Ngày truy cập: 12/06/2024.

Đường dẫn: <https://www.ibm.com/topics/artificial-intelligence>

IBM. *What are convolutional neural networks?* Ngày truy cập: 12/06/2024

Đường dẫn: <https://www.ibm.com/topics/convolutional-neural-networks>

OpenCV. *Cascade Classifier*. Ngày truy cập 12/06/2024.

Đường dẫn: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

Soulpage. *What is Cascade Classifier? Cascade Classifier Explained*. Ngày truy cập 12/06/2024.

Đường dẫn: <https://soulpageit.com/ai-glossary/cascade-classifier-explained/#:~:text=A%20Cascade%20Classifier%2C%20also%20known,popular%20for%20face%2Ddetection%20tasks>.

Chung Pham Van (12/10/2020). *[Deep Learning] Tìm hiểu về mạng tích chập (CNN)*. Ngày truy cập: 12/06/2024.

Đường dẫn: <https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2>