BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CMC

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: TRÍ TUỆ NHÂN TẠO

Tên đề tài: Xây dựng hệ thống nhận diện cảm xúc khuôn mặt

Nhóm sinh viên: Nguyễn Minh Đức - BIT230109
Nguyễn Quỳnh Như - BIT230303
Nguyễn Trí Như - BIT230304

Giảng viên hướng dẫn: Phạm Thị Anh Lê

Hà Nội, tháng 8 - 2024

MỤC LỤC

MỞ ĐẦU

Lời đầu tiên, chúng em xin cảm ơn thầy Vũ Việt Vũ và cô Phạm Thị Anh Lê - giảng viên của môn Trí Tuệ Nhân Tạo đã tận tình giảng dạy những kiến thức và hướng dẫn chúng em trong quá trình thực hiện bài tập lớn này! Nhờ có sự giảng dạy tận tình và kiến thức sâu rộng của thầy cô, chúng em đã được trau dồi kiến thức và áp dụng vào đề tài của mình.

Với sự phát triển của công nghệ hiện đại, kèm theo là sự xuất hiện của những yêu cầu về xây dựng hệ thống và ứng dụng có khả năng giống con người như nhận diện và xác định sự vật, hiện tượng. Để làm được điều đó, một thuật ngữ mới đã xuất hiện: "Trí tuệ nhân tạo" (Artificial Intelligence).

Trí tuệ nhân tạo là một lĩnh vực khoa học máy tính, trong đó các hệ thống máy tính được lập trình để thực hiện các nhiệm vụ mà thông thường cần đến trí thông minh của con người. Điều này bao gồm việc áp dụng các kỹ thuật học máy (machine learning) và học sâu (deep learning). Cụ thể, chúng ta cho máy học từ những thông tin xung quanh và huấn luyện nó như một con người thật sự. Sau đó, máy sẽ được thử nghiệm với những mẫu chưa học và kết quả sẽ được đánh giá. Trong quá trình này, chúng ta cần thay đổi và chỉnh sửa nội dung huấn luyện để giảm thiểu sai sót đến mức tối đa. Điều này tương tự như việc thay đổi phương pháp dạy học cho học sinh khi kết quả không đạt như mong muốn.

Trong đề tài này, chúng em thực hành huấn luyện một hệ thống trí tuệ nhân tạo để nhận diện cảm xúc của con người. Chúng em bắt đầu bằng việc tìm hiểu về nguồn dữ liệu cần có và xác định. Trong bài báo này, chúng em thử nghiệm một kiến trúc học sâu dựa trên nhiều lớp tích chập (ConvNet) để phát hiện cảm xúc khuôn mặt. Dữ liệu thu được từ webcam sẽ được định vị khuôn mặt bằng phương pháp haar cascade từ thư viện OpenCV, sau đó dữ liệu được chuyển vào mạng học sâu với đầu ra xác suất (softmax), trả về xác suất của 7 loại cảm xúc do hệ thống tính toán được. Kết quả thử nghiệm trên bộ dữ liệu FER-2013 đạt 66.3%, nằm trong TOP5 mô hình học máy tốt nhất của dataset này

Bằng việc áp dụng những kiến thức học được trong học phần, chúng em bắt đầu việc xây dựng và triển khai hệ thống nhân diện cảm xúc khuôn mặt dựa trên tập dữ liệu có sẵn là FER-2013. Đây là một thử thách lớn cho sinh viên chúng em nhưng nó mang lại nhiều điều - bao gồm về kiến thức môn học, cách trình bày nội dung và kĩ năng làm việc nhóm. Chúng em xin chân thành cảm ơn thầy/ cô của trường Đại học CMC đã hướng dẫn chúng em với sự nhiệt huyết, truyền cảm hứng tuyệt vời!

PHẦN I. TỔNG QUAN

1. Giới thiệu bài toán

1.1. Tổng quan

- Khuôn mặt con người là một bức tranh sống động thể hiện nhiều cảm xúc mà không cần lời nói. Đây là một trong những phương tiện giao tiếp tự nhiên và mạnh mẽ nhất, cho phép chúng ta truyền đạt cảm xúc một cách trực tiếp. Khác với các hình thức giao tiếp phi ngôn ngữ khác, cảm xúc trên khuôn mặt mang tính phổ quát, có thể được nhận diện và hiểu bởi mọi người, bất kể văn hóa hay ngôn ngữ.

- Hiện nay, lĩnh vực nhận dạng và phân tích cảm xúc qua khuôn mặt đang thu hút sự quan tâm lớn trong cộng đồng nghiên cứu và ứng dụng công nghệ. Cảm xúc mà chúng ta thể hiện qua khuôn mặt chiếm đến 55% trong tổng thể giao tiếp phi ngôn ngữ, điều này cho thấy sự quan trọng của nó trong các mối quan hệ xã hội. Theo các nghiên cứu, có bảy cảm xúc cơ bản: trung tính, giận dữ, ghê tởm, sợ hãi, hạnh phúc, buồn và bất ngờ, mà tất cả đều có thể nhận diện được thông qua các hình ảnh khuôn mặt.

- Nhận dạng cảm xúc qua khuôn mặt có ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như là:

- **Giáo dục**: Phân tích phản ứng của học sinh trong thời gian thực giúp đo lường hiệu quả giảng dạy.
- **Tiếp thị**: Các công ty có thể nắm bắt phản hồi của khách hàng đối với sản phẩm và quảng cáo, từ đó điều chỉnh chiến lược kinh doanh.
- **Chơi game**: Công nghệ thực tế ảo sử dụng nhận dạng cảm xúc để nâng cao trải nghiệm người chơi.
- **Bảo mật**: Phát hiện hành vi đáng ngờ trong đám đông, hỗ trợ trong việc ngăn chặn tội phạm.
- **Chăm sóc sức khỏe**: Phân tích tình trạng sức khỏe của bệnh nhân qua biểu hiện cảm xúc, hỗ trợ trong việc chẩn đoán và điều trị.
- **Dịch vụ khách hàng**: Cải thiện trải nghiệm khách hàng thông qua phân tích cảm xúc, giúp tăng cường sự hài lòng và hiệu quả giao tiếp.

- Hệ thống nhận dạng cảm xúc khuôn mặt đang ngày càng trở nên phổ biến, từ việc theo dõi tình trạng sức khỏe đến phát hiện hành vi không an toàn, cho thấy tiềm năng to lớn của công nghệ này trong việc nâng cao chất lượng cuộc sống.

2. Một số thông tin liên quan.

- Lĩnh vực nhận dạng cảm xúc qua khuôn mặt đang đối mặt với các thách thức như độ chính xác của mô hình, sự đa dạng của dữ liệu và khả năng xử lý trong thời gian thực. Các hệ thống cần cải thiện để nhận diện chính xác trong mọi tình huống và phản hồi nhanh chóng.

- Công nghệ này sẽ phát triển theo hướng tích hợp với trí tuệ nhân tạo tổng hợp (AGI) và Internet vạn vật (IoT), giúp hệ thống trở nên thông minh hơn và tương tác tốt hơn với các thiết bị khác. Nghiên cứu cũng tập trung vào việc nâng cao độ chính xác và mở rộng ứng dụng trong nhiều lĩnh vực.

3. Kết luận

- Việc nhận dạng cảm xúc qua khuôn mặt và những ứng dụng tiềm năng của nó trong các lĩnh vực khác nhau có tầm quan trọng vô cùng lớn.

- **Đóng góp của nghiên cứu**: Nêu rõ những đóng góp của nghiên cứu này đối với lĩnh vực nhận dạng cảm xúc và công nghệ trí tuệ nhân tạo.
- **Hướng phát triển**: Đề xuất các hướng phát triển tiếp theo để cải thiện hệ thống nhận dạng cảm xúc, bao gồm việc nâng cao độ chính xác, mở rộng phạm vi ứng dụng, và tích hợp với các công nghệ mới.

- Việc xây dựng hệ thống nhận diện cảm xúc khuôn mặt là một nhiệm vụ quan trọng nhằm nâng cao hiệu quả các hoạt động trong cuộc sống. Dự án này sẽ giúp chúng em hiểu rõ hơn về mô hình học máy và những kiến thức liên quan đến việc huấn luyện mô hình.

- Với việc lựa chọn sử dụng các mô hình huấn luyện phù hợp, chúng em tin tưởng rằng hệ thống sẽ được huấn luyện một cách hiệu quả và đáp ứng được các yêu cầu của bài toán đề ra.

- Quá trình thực hiện dự án được chia thành các giai đoạn rõ ràng nhằm đảm bảo tiến độ và chất lượng của sản phẩm cuối cùng. Chúng em sẽ nỗ lực hoàn thành dự án đúng kế hoạch và mang lại giá trị thiết thực.

PHẦN II. PHƯƠNG PHÁP THỰC HIỆN

1. Đặt vấn đề:

- Mục tiêu của việc xây dựng hệ thống này là phát triển một giải pháp hiệu quả, chính xác và nhanh chóng trong việc nhận diện khuôn mặt trong các điều kiện ánh sáng, góc nhìn và môi trường khác nhau.

- Bài toán yêu cầu xây dựng một hệ thống phân loại cảm xúc từ hình ảnh khuôn mặt.

- **Input**: Một ảnh xám kích thước 48x48 pixel, thể hiện khuôn mặt người.
- **Output**: Một trong 7 loại cảm xúc: *Giận dữ, Ghê tởm, Sợ hãi, Hạnh phúc, Buồn, Ngạc nhiên, Bình thường*.

2. Một số mô hình giải quyết bài toán:

- Đề xuất một số phương pháp giải quyết bài toán:

1) Mạng nơ-ron tích chập (CNN - Convolutional Neural Network): Giới thiệu các kiến trúc CNN phổ biến như VGG, ResNet, hoặc các mô hình tùy chỉnh cho bài toán nhận diện cảm xúc.

2) Deep Learning: Sử dụng các mô hình deep learning tiên tiến như Inception, EfficientNet, hoặc các mô hình Transformer.

3) Mô hình FaceNet: Sử dụng mạng nơ-ron sâu để chuyển đổi hình ảnh khuôn mặt thành các vector đặc trưng trong không gian tính toán, cho phép so sánh sự tương đồng giữa các khuôn mặt.

- Trong đề tài này, chúng em sẽ sử dụng mô hình Mạng nơ-ron tích chập (CNN). Bởi vì mạng nơ-ron tích chập (CNN) là một công cụ mạnh mẽ trong lĩnh vực học sâu, đặc biệt là cho các tác vụ liên quan đến hình ảnh, dưới đây là 1 trong số những lý do nhóm chứng em chọn mô hình này:

1) **Khả năng nhận diện mẫu**: CNN rất hiệu quả trong việc nhận diện và phân loại các mẫu trong hình ảnh nhờ vào cấu trúc nhiều lớp, giúp trích xuất các đặc trưng quan trọng.

2) **Tính chính xác cao**: Các mô hình CNN thường đạt được độ chính xác cao trong các bài toán phân loại hình ảnh và phát hiện đối tượng, nhờ vào khả năng học được các đặc trưng phức tạp.

3) **Tính tự động hóa**: CNN có khả năng tự động học các đặc trưng từ dữ liệu mà không cần phải thiết kế các đặc trưng thủ công, giúp tiết kiệm thời gian và công sức.

4) **Khả năng xử lý dữ liệu lớn**: CNN có khả năng xử lý lượng lớn dữ liệu hình ảnh, phù hợp với các bài toán yêu cầu phân tích dữ liệu lớn.

5) **Ứng dụng đa dạng**: CNN có thể áp dụng cho nhiều lĩnh vực như nhận diện khuôn mặt, phân loại hình ảnh y tế, và nhiều ứng dụng khác trong thị giác máy tính.

- Tại sao không dùng hồi quy tuyến tính hoặc các phương pháp cổ điển khác?

- Hồi quy tuyến tính hoặc các mô hình ML truyền thống không thể tự động trích xuất đặc trưng từ ảnh.

- CNN có khả năng tự động học các đặc trưng phức tạp từ hình ảnh mà không cần xử lý thủ công.

3. Kết luận

- Dự án xây dựng hệ thống nhận diện cảm xúc khuôn mặt mà nhóm chúng em thực hiện, nhóm lựa chọn sử dụng mô hình CNN không chỉ nhằm mục đích giải quyết các vấn đề đã đặt ra mà còn hướng tới khả năng mở rộng và tùy chỉnh theo nhu cầu thực tế. Qua việc thực hiện các chức năng cơ bản và nâng cao, việc huấn luyện mô hình sẽ tiết kiệm thời gian, đồng thời đáp ứng được các yêu cầu đa dạng của bài toán. Chúng em tin rằng với nền tảng vững chắc và khả năng mở rộng linh hoạt, dự án này sẽ trở thành một công cụ hữu ích và thiết thực trong đời sống.

PHẦN III. THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

1. Dữ liệu thực nghiệm.

- Dữ liệu được sử dụng từ file `fer2013.csv`, trong đó mỗi dòng chứa:

  - **Emotion**: Nhãn cảm xúc (giá trị từ 0 đến 6 tương ứng với 7 cảm xúc).
  - **Pixels**: Chuỗi các giá trị pixel (2304 giá trị, tương ứng kích thước 48x48).

- Dữ liệu được chia thành tập huấn luyện (80%) và tập kiểm tra (20%).

- Dữ liệu FER-2013 được công bố bởi trang Kaggle trong khuôn khổ workshop của hội thảo ICML 2013. Dữ liệu gồm 35,887 grayscale ảnh khuôn mặt có kích thước 48x48 pixels, chỉ gồm khuôn mặt hầu như được căn giữa ảnh và tỉ lệ khuôn mặt được điều chỉnh chiếm phần lớn diện tích của ảnh. Một ảnh sẽ được gán nhãn nằm một trong bảy loại cảm xúc giá trị từ 0 đến 6 (0: giận dữ, 1: ghê tởm, 2: sợ hãi, 3: hạnh phúc, 4: buồn rầu, 5: bất ngờ, 6: trung lập)

- Chúng em sử dụng bộ dữ liệu này cho mô hình thử nghiệm vì bộ dữ liệu có số mẫu khá lớn, phù hợp với việc huấn luyện với mạng học sâu, vốn đòi hỏi nhiều mẫu hơn các phương pháp học máy thông thường. Ngoài ra, bộ dữ liệu được cấu trúc dễ dàng xử lý bởi thư viện Keras/TensorFlow và có nhiều kết quả đối chứng khi thực hiện so sánh mô hình của chúng em với các kết quả của những nhóm nghiên cứu khác.

2. Triển khai thực nghiệm.

- Dưới đây là mô tả Quy trình triển khai mô hình:

2..1, Import các thư viện

- Bước đầu tiên trong xây dựng hệ thống là import các thư viện cần thiết vào chương trình. Dưới đây là tổng quan về các thư viện phổ biến và vai trò của chúng:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from sklearn.model_selection import train_test_split
```

1. NumPy

- NumPy cung cấp các cấu trúc dữ liệu mạnh mẽ như ma trận và mảng đa chiều. Những tính năng nổi bật:

  - Tối ưu hóa hiệu suất cho các phép toán ma trận.
  - Thực hiện nhanh chóng các phép tính đại số tuyến tính, xử lý dữ liệu số hiệu quả.

- Numpy cung cấp nhiều cấu trúc dữ liệu mạnh mẽ để tính toán tối ưu mảng và ma trận.

2. Matplotlib

- Matplotlib là thư viện vẽ đồ thị mạnh mẽ, hỗ trợ trực quan hóa dữ liệu:

- Pyplot Module: Giao diện tương tự MATLAB, dễ sử dụng và tương thích tốt với Python.

- Matplotlib là một trong những giải pháp để thực hiện các suy luận thống kê cần thiết, cần phải trực quan hóa dữ liệu cho người dùng Python. Nó là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với Python và NumPy. Module được sử dụng nhiều nhất của Matplotib là Pyplot cung cấp giao diện như MATLAB nhưng thay vào đó, nó sử dụng Python và nó là nguồn mở.

3. Pandas

- Pandas cung cấp các cấu trúc dữ liệu dễ sử dụng, được tối ưu hóa để làm việc với dữ liệu có cấu trúc:

- DataFrame: Dễ dàng xử lý dữ liệu dạng bảng (2D) và dữ liệu thời gian.

- Ứng dụng: Phân tích và xử lý dữ liệu thực tế, làm sạch và tổ chức dữ liệu.

- Pandas là thư viện linh hoạt, được xem như công cụ chính trong phân tích dữ liệu Python.

4. Scikit-learn (Sklearn)

- Scikit-learn là thư viện học máy hàng đầu trong Python.

- Ứng dụng: Xử lý các bài toán machine learning từ cơ bản đến nâng cao.

5. TensorFlow

- TensorFlow chủ yếu được sử dụng cho các tác vụ liên quan đến học sâu như phân loại ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên (NLP), và nhiều ứng dụng AI khác.

- **Tensor** trong TensorFlow là cấu trúc dữ liệu chính, nó đại diện cho một mảng đa chiều (có thể là ma trận hoặc tensor bậc cao) chứa dữ liệu.
- **Flow** mô tả cách các phép toán được thực thi trên các tensor, tức là dòng chảy dữ liệu và phép toán.

- **TensorFlow**: Thư viện cơ bản mạnh mẽ, hỗ trợ nhiều kỹ thuật học máy và học sâu.

2.2: Đọc và khai phá dữ liệu , tiền xử lý dữ liệu .

Mục đích của phần này là:

- Biết dữ liệu chứa gì và cách thức dữ liệu được cấu trúc.
- Đảm bảo dữ liệu đã được làm sạch, chuẩn hóa và có thể đưa vào mô hình.
- **T**ối ưu hóa các đặc tính dữ liệu để mô hình có thể học tốt hơn.
- Chia dữ liệu thành các bộ huấn luyện và kiểm tra để đánh giá mô hình một cách hiệu quả.

2.2.1, Đọc và khai phá dữ liệu

```
data = pd.read_csv('fer2013.csv')
print("Dataset preview:")
print(data.head())

# Đánh giá thông tin cơ bản về dữ liệu
print("\nDataset info:")
print(data.info())

# Kiểm tra số lượng mỗi nhãn cảm xúc
print("\nEmotion counts:")
print(data['emotion'].value_counts())
# 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral
```

- Ở bước này chương trình thực hiện các bước cơ bản để khám phá và đánh giá một tập dữ liệu cảm xúc từ file `fer2013.csv`

```
data = pd.read_csv('fer2013.csv')
print("Dataset preview:")
print(data.head())

# Đánh giá thông tin cơ bản về dữ liệu
print("\nDataset info:")
print(data.info())
```

- Hàm `head()` hiển thị 5 dòng đầu tiên của DataFrame để có cái nhìn tổng quan về cấu trúc và nội dung của dữ liệu.

- Hàm `info()` cung cấp thông tin về kiểu dữ liệu của từng cột, số lượng giá trị không null, và kích thước của DataFrame, giúp đánh giá chất lượng dữ liệu.

```
Dataset preview:
   emotion                                             pixels     Usage
0        0  70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...  Training
1        0  151 150 147 155 148 133 111 140 170 174 182 15...  Training
2        2  231 212 156 164 174 138 161 173 182 200 106 38...  Training
3        4  24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...  Training
4        6  4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...  Training

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35887 entries, 0 to 35886
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   emotion  35887 non-null  int64
 1   pixels   35887 non-null  object
 2   Usage    35887 non-null  object
dtypes: int64(1), object(2)
memory usage: 841.2+ KB
None

Emotion counts:
emotion
3    8989
6    6198
4    6077
2    5121
0    4953
5    4002
1     547
Name: count, dtype: int64
```

Kết quả khi chạy chương trình đọc và khai phá dữ liệu

2.2.2, Tiền xử lý dữ liệu

```python
# Bước 2: Tiền xử lý dữ liệu
# Chuyển đổi ảnh từ chuỗi pixel sang mảng 2D
def process_image(pixels):
    pixels = np.array(pixels.split(), dtype='float32')
    return pixels.reshape(48, 48)  # FER-2013 sử dụng kích thước 48x48

data['pixels'] = data['pixels'].apply(process_image)

# Chuẩn hóa dữ liệu
data['pixels'] = data['pixels'] / 255.0  # Chuẩn hóa pixel thành [0,1]
```

- Đoạn chương trình này xử lý hình ảnh trong tập dữ liệu FER-2013 bằng cách chuyển đổi chuỗi pixel thành mảng 2D và chuẩn hóa các giá trị pixel, giúp chuẩn bị dữ liệu cho các bước tiếp theo trong quá trình học máy.

```python
def process_image(pixels):
    pixels = np.array(pixels.split(), dtype='float32')
    return pixels.reshape(48, 48)  # FER-2013 sử dụng kích thước 48x48
```

- **Mục đích**: Chuyển đổi chuỗi pixel (mỗi pixel được biểu diễn dưới dạng chuỗi) thành một mảng 2D.

- **Chi tiết**:

- `pixels.split()`: Tách chuỗi pixel thành một danh sách các giá trị pixel.
- `np.array(..., dtype='float32')`: Chuyển danh sách này thành một mảng NumPy với kiểu dữ liệu `float32`.
- `reshape(48, 48)`: Định hình lại mảng thành kích thước 48x48, phù hợp với kích thước của hình ảnh trong tập dữ liệu FER-2013.

```
data['pixels'] = data['pixels'].apply(process_image)

# Chuẩn hóa dữ liệu
data['pixels'] = data['pixels'] / 255.0  # Chuẩn hóa pixel thành [0,1]
```

- Dòng lệnh đầu nhằm áp dụng hàm `process_image` cho từng giá trị trong cột `pixels` của DataFrame `data`. Kết quả là cột `pixels` giờ đây chứa các mảng 2D đại diện cho hình ảnh.
- Dòng lệnh thứ 2 nhằm chuẩn hóa các giá trị pixel để nằm trong khoảng [0, 1]

+ **Chi tiết**:

Việc chia cho 255 có các lợi ích sau :
+Chuẩn hóa dữ liệu (Normalization)
+Tăng tốc độ huấn luyện(quá trình lan truyền ngược và tối ưu hóa trở nên ổn định và nhanh chóng hơn.)
+Giảm rủi ro overfitting(giá trị pixel có phạm vi quá rộng, mạng có thể gặp khó khăn trong việc học các đặc trưng từ dữ liệu và dễ bị overfitting (quá khớp))
+Sử dụng cho các hàm kích hoạt (Activation functions)(Relu hoạt động tốt hơn khi đầu vào nằm trong phạm vi nhỏ hơn)

```
fig, axes = plt.subplots(len(emotions), 5, figsize=(15, 20))  # 7 hàng, 5 cột
for i, emotion in enumerate(emotions):
    for j in range(5):
        ax = axes[i, j]
        img = data[data['emotion'] == emotion].iloc[j]['pixels']
        ax.imshow(img, cmap='gray')
        ax.set_title(f'Emotion: {emotion}')
        ax.axis('off')
plt.tight_layout()
plt.show()
```

2.3, Phân tích và trực quan hoá dữ liệu

- Đoạn code này giúp trực quan hóa các mẫu hình ảnh cho từng loại cảm xúc trong tập dữ liệu FER-2013, cung cấp cái nhìn trực quan về dữ liệu và sự phân bố của các cảm xúc.

- fig, axes = plt.subplots(len(emotions), 5, figsize=(15, 20)): Tạo biểu đồ với số hàng bằng số cảm xúc (len(emotions)), mỗi hàng có 5 cột. Kích thước toàn bộ là 15x20 inch.
- for i, emotion in enumerate(emotions): Duyệt qua từng cảm xúc (emotion) và chỉ số (i).
- for j in range(5): Với mỗi cảm xúc, lấy 5 ảnh đầu tiên.
- img = data[data['emotion'] == emotion].iloc[j]['pixels']: Lấy dữ liệu ảnh thuộc cảm xúc hiện tại (emotion) từ DataFrame.
- ax.imshow(img, cmap='gray'): Hiển thị ảnh dưới dạng grayscale.
- ax.set_title(f'Emotion: {emotion}'): Đặt tiêu đề cho từng ảnh theo cảm xúc.
- ax.axis('off'): Ẩn trục của biểu đồ.
- plt.tight_layout(): Điều chỉnh khoảng cách giữa các biểu đồ để không bị chồng chéo.
- plt.show(): Hiển thị tất cả các biểu đồ.

Kết quả khi chạy chương trình phân tích và trực quan hoá dữ liệu

2.4. Chuẩn bị dữ liệu cho mô hình

=>Bắt đầu từ bước này, chúng em phân tách nó ra thành nhiều cách khác nhau để có thể phân tích, xem xét và so sánh:

Cách 1:

```
# Bước 4: Chuẩn bị dữ liệu cho mô hình
# Chuyển đổi dữ liệu thành mảng 4D
X = np.array(data['pixels'].tolist())
X = X.reshape(-1, 48, 48, 1)  # Định dạng cho CNN
y = tf.keras.utils.to_categorical(data['emotion'], num_classes=7)

# Chia dữ liệu thành train và test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Đoạn chương trình này chuẩn bị dữ liệu cho mô hình học sâu bằng cách chuyển đổi hình ảnh thành định dạng phù hợp cho CNN, chuyển đổi nhãn cảm xúc thành định dạng phân loại một nóng, và chia dữ liệu thành tập huấn luyện và tập kiểm tra. Điều này là rất quan trọng để đảm bảo mô hình có thể học và đánh giá chính xác.

```
y = tf.keras.utils.to_categorical(data['emotion'], num_classes=7)
```

- Mục đích: Chuyển đổi nhãn cảm xúc từ định dạng số nguyên thành định dạng phân loại một nóng (one-hot encoding).

+ **Chi tiết:**

- `tf.keras.utils.to_categorical(...)`: Hàm này chuyển đổi mỗi nhãn từ 0 đến 6 thành một mảng nhị phân, trong đó chỉ có một phần tử là 1 (đại diện cho cảm xúc tương ứng) và các phần tử còn lại là 0. `num_classes=7` chỉ định rằng có 7 lớp cảm xúc.

```
# Chia dữ liệu thành train và test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Mục đích: Chia dữ liệu thành hai phần: một phần để huấn luyện mô hình và một phần để kiểm tra độ chính xác của mô hình.

+ **Chi tiết**:

- `train_test_split(...)`: Hàm này từ thư viện `sklearn` chia dữ liệu thành tập huấn luyện và tập kiểm tra.
- `test_size=0.2`: 20% dữ liệu sẽ được sử dụng cho tập kiểm tra, còn lại 80% cho tập huấn luyện.
- `random_state=42`: Đặt giá trị ngẫu nhiên cố định để đảm bảo rằng mỗi lần chạy đều cho kết quả giống nhau (tính tái lập).

Cách 2:

```
import random

# Đặt seed cho ngẫu nhiên hóa để đảm bảo tính nhất quán
seed = 42
np.random.seed(seed)  # Thiết lập seed cho NumPy
tf.random.set_seed(seed)  # Thiết lập seed cho TensorFlow
random.seed(seed)  # Thiết lập seed cho Python random module

# Chuyển đổi dữ liệu thành mảng 4D
X = np.array(data['pixels'].tolist())  # Chuyển cột 'pixels' thành mảng NumPy
X = X.reshape(-1, 48, 48, 1)  # Định dạng lại thành (số ảnh, 48, 48, 1) cho CNN

y = tf.keras.utils.to_categorical(data['emotion'], num_classes=7)  # One-hot encoding cho nhãn cảm xúc

# Chia dữ liệu thành train và test
```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=seed) # Chia 80% train, 20% test

- **Đặt seed:** Đảm bảo kết quả ngẫu nhiên được tái tạo trong các lần chạy khác nhau (giúp so sánh và debug).
- **Chuyển dữ liệu:** Chuyển đổi cột 'pixels' từ chuỗi thành mảng NumPy 4D (48x48 ảnh đen trắng).
- **One-hot encoding:** Chuyển nhãn cảm xúc thành dạng one-hot với 7 lớp.
- **Chia dữ liệu:** Chia dữ liệu thành tập huấn luyện (80%) và kiểm tra (20%).

Cách 3:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.utils import shuffle

# Chuyển đổi dữ liệu thành array 4D và one-hot encoding nhãn
X = np.array(data['pixels'].tolist()).reshape(-1, 48, 48, 1)  # Chuyển cột 'pixels' thành mảng 4D, mỗi ảnh có kích thước 48x48 và 1 kênh (ảnh đen trắng)
y = tf.keras.utils.to_categorical(data['emotion'], num_classes=7)  # One-hot encoding nhãn cảm xúc (7 lớp)

# Shuffle dữ liệu trước khi chia để đảm bảo dữ liệu được phân phối ngẫu nhiên
X, y = shuffle(X, y, random_state=42)  # Xáo trộn dữ liệu (X, y) để đảm bảo phân phối ngẫu nhiên

# Sử dụng StratifiedShuffleSplit để chia dữ liệu phân tầng
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)  # Chia dữ liệu thành 2 tập: train (80%) và test (20%) với phân tầng
for train_index, test_index in sss.split(X, np.argmax(y, axis=1)):  # np.argmax để lấy nhãn từ one-hot encoding
    X_train, X_test = X[train_index], X[test_index]  # Lấy tập huấn luyện và kiểm tra từ index phân tầng
    y_train, y_test = y[train_index], y[test_index]

# Tăng cường dữ liệu:
datagen = ImageDataGenerator(
    rotation_range=10,          # Xoay ảnh tối đa 10 độ
    width_shift_range=0.1,      # Dịch chuyển ngang tối đa 10% chiều rộng ảnh
    height_shift_range=0.1,     # Dịch chuyển dọc tối đa 10% chiều cao ảnh
    horizontal_flip=True        # Lật ngang ảnh
)
```

-Tăng cường Dữ liệu (Data Augmentation) rất quan trọng vì nó giúp mô hình học sâu (CNN) cải thiện hiệu suất bằng cách tạo ra các biến thể mới từ dữ liệu huấn luyện gốc mà không cần thu thập thêm dữ liệu. Điều này có thể giảm overfitting và cải thiện độ chính xác của mô hình khi xử lý dữ liệu thực tế.

1. **`rotation_range=10`**:

- Tham số này cho phép xoay ảnh ngẫu nhiên trong phạm vi 10 độ. Việc xoay ảnh giúp mô hình học cách nhận diện các đối tượng (trong trường hợp này là cảm xúc khuôn mặt) khi chúng bị xoay trong thực tế, điều này làm cho mô hình trở nên mạnh mẽ hơn khi gặp phải dữ liệu chưa được chuẩn hóa hoặc có các biến thể.

2. **`width_shift_range=0.1` và `height_shift_range=0.1`**:
   - Cả hai tham số này sẽ dịch chuyển ảnh ngẫu nhiên theo chiều rộng và chiều cao với tỉ lệ 10%. Điều này giúp mô hình học cách nhận diện đối tượng dù chúng không luôn ở vị trí chính giữa ảnh. Trong thực tế, các đối tượng có thể bị dịch chuyển, và việc dịch chuyển này giúp mô hình học được sự linh hoạt trong việc phát hiện các đặc điểm bất kỳ vị trí nào trong ảnh.
3. **`horizontal_flip=True`**:
   - Tham số này cho phép lật ảnh theo chiều ngang. Việc lật ảnh ngẫu nhiên giúp mô hình nhận diện cảm xúc không phụ thuộc vào hướng của khuôn mặt (ví dụ, khuôn mặt nhìn về bên trái hay phải). Điều này làm cho mô hình trở nên bền vững hơn đối với các biến thể trong thực tế.

4. `datagen.fit(X_train)`:

- Phương thức `fit()` này áp dụng các phép biến đổi trên ảnh trong tập huấn luyện (`X_train`). Việc này giúp đối tượng `datagen` hiểu và lưu trữ các tham số cần thiết để biến đổi các ảnh trong quá trình huấn luyện mà không làm thay đổi dữ liệu gốc.

Lý do sử dụng Tăng cường Dữ liệu:

1. **Giảm overfitting**:
   - Khi sử dụng tăng cường dữ liệu, mô hình sẽ không chỉ học trên các ảnh gốc mà còn học từ các biến thể của ảnh đó. Điều này giúp mô hình không bị "học thuộc" (overfit) các đặc điểm cụ thể của ảnh gốc và trở nên mạnh mẽ hơn với các trường hợp mới trong thực tế.
2. **Tăng độ chính xác**:
   - Việc tạo ra các biến thể ảnh giúp mô hình nhận diện cảm xúc từ khuôn mặt ở các góc độ và tình huống khác nhau, từ đó cải thiện độ chính xác khi xử lý các ảnh mới mà mô hình chưa thấy trước đó.
3. **Tăng tính linh hoạt của mô hình**:
   - Khi mô hình phải học từ nhiều dạng dữ liệu khác nhau (sau khi đã tăng cường), nó sẽ có khả năng nhận diện các biểu cảm khuôn mặt dưới nhiều điều kiện khác nhau, chẳng hạn như khi khuôn mặt bị nghiêng, bị dịch chuyển, hay bị đảo ngược.

2.5, Xây dựng mô hình CNN

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),
    MaxPooling2D(2, 2),
    Dropout(0.25),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Dropout(0.25),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Dropout(0.25),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(7, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

- Đoạn code này thiết lập một mô hình CNN đơn giản nhưng hiệu quả để phân loại cảm xúc từ hình ảnh. Mô hình được cấu trúc hợp lý với các lớp tích chập, lớp pooling, lớp đầy đủ và dropout, nhằm tối ưu hóa khả năng học và giảm thiểu overfitting.

Các lớp trong mô hình:

1. **Conv2D**:
    a. `Conv2D(32, (3,3), activation='relu', input_shape=(48, 48, 1))`
        i. Tạo một lớp tích chập với 32 bộ lọc, mỗi bộ lọc có kích thước 3x3.
        ii. Sử dụng hàm kích hoạt ReLU (Rectified Linear Unit).
        iii. `input_shape=(48, 48, 1)` chỉ định kích thước đầu vào của hình ảnh (chiều cao, chiều rộng, số kênh).
2. **MaxPooling2D**:
    a. `MaxPooling2D(2,2)`
        a. Thực hiện giảm kích thước không gian của đầu ra từ lớp tích chập bằng cách lấy giá trị lớn nhất trong mỗi khối 2x2.
        b. Giúp giảm số lượng tham số và tính toán, đồng thời tăng cường tính bất biến của mô hình đối với các biến thể nhỏ trong hình ảnh.
3. **Lớp tích chập thêm**:
    a. `Conv2D(64, (3,3), activation='relu')`
        i. Tạo một lớp tích chập thứ hai với 64 bộ lọc.
    b. `Conv2D(128, (3,3), activation='relu')`
        ii. Tạo một lớp tích chập thứ hai với 128 bộ lọc.
4. **Lớp Flatten**:
    a. `Flatten()`
        a. Chuyển đổi đầu ra 2D từ lớp tích chập cuối cùng thành một mảng 1D để có thể kết nối với các lớp Dense.
5. **Lớp Dense**:

a. `Dense(128, activation='relu')`
    a. Tạo một lớp đầy đủ với 128 nơ-ron và hàm kích hoạt ReLU.

6. **Dropout**:
    a. `Dropout(0.5)`
        a. Giúp giảm thiểu overfitting bằng cách ngẫu nhiên loại bỏ 50% nơ-ron trong quá trình huấn luyện.
7. **Lớp đầu ra**:
    a. `Dense(7, activation='softmax')`
        a. Lớp đầu ra với 7 nơ-ron (tương ứng với 7 loại cảm xúc) và hàm kích hoạt softmax, cung cấp xác suất cho mỗi lớp.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

- Mục đích: Thiết lập cấu hình cho quá trình huấn luyện mô hình.

+ Chi tiết:

- `optimizer='adam'`: Sử dụng thuật toán tối ưu Adam, thường được ưa chuộng vì hiệu suất tốt và tốc độ hội tụ nhanh.
- `loss='categorical_crossentropy'`: Hàm mất mát được sử dụng cho bài toán phân loại nhiều lớp.
- `metrics=['accuracy']`: Theo dõi độ chính xác của mô hình trong quá trình huấn luyện và kiểm tra.

- model.summary() : Hiển thị thông tin tóm tắt về cấu trúc của mô hình, bao gồm số lượng các lớp, kích thước đầu vào, số lượng tham số, và dạng đầu ra

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 46, 46, 32)	320
max_pooling2d_4 (MaxPooling2D)	(None, 23, 23, 32)	0
dropout_2 (Dropout)	(None, 23, 23, 32)	0
conv2d_5 (Conv2D)	(None, 21, 21, 64)	18,496
max_pooling2d_5 (MaxPooling2D)	(None, 10, 10, 64)	0
dropout_3 (Dropout)	(None, 10, 10, 64)	0
conv2d_6 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_6 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_4 (Dropout)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 128)	262,272
dropout_5 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 7)	903

Total params: 355,847 (1.36 MB)

Trainable params: 355,847 (1.36 MB)

Non-trainable params: 0 (0.00 B)

Kết quả khi chạy chương trình xây dựng mô hình CNN

2.6, Huấn luyện và đánh giá mô hình

```
# Bước 6: Huấn luyện và đánh giá mô hình
history = model.fit(X_train, y_train, epochs=50, batch_size=64, validation_data=(X_test, y_test))

# Đánh giá mô hình
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
```

- Huấn luyện: Mô hình được huấn luyện qua 50 epochs với kích thước batch 64, với việc theo dõi hiệu suất trên tập kiểm tra.
- Đánh giá: Mô hình được đánh giá trên tập kiểm tra và kết quả được in ra bao gồm tổn thất và độ chính xác.

- Quá trình này giúp xác định khả năng nhận diện cảm xúc của mô hình trên dữ liệu chưa thấy, từ đó đánh giá tính hiệu quả và độ chính xác của mô hình.

2.7, Lưu vào model sau khi huấn luyện mô hình

```
#  Lưu sau khi huấn luyện mô hình
model.save('emotion_model.h5')
```

- Đoạn code model.save('emotion_model.h5') sẽ lưu toàn bộ cấu trúc mô hình, trọng số đã được học, và các cài đặt khác của mô hình vào một tệp HDF5 có tên emotion_model.h5.

2.8, Kỹ thuật đánh giá

- Kỹ thuật đánh giá mô hình sử dụng . Cụ thể:

i, Đánh giá mô hình trên tập kiểm tra:

- Mô hình được đánh giá bằng cách sử dụng model.evaluate() trên tập kiểm tra (X_test và y_test). Điều này giúp bạn đo lường độ chính xác và mất mát của mô hình trên dữ liệu mà mô hình chưa từng thấy trong quá trình huấn luyện.
- Kết quả trả về bao gồm giá trị **test loss** và **test accuracy**, là hai chỉ số cơ bản để đánh giá hiệu suất của mô hình sau khi huấn luyện.

```
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')
```

Độ chính xác (Accuracy)

- Độ chính xác (accuracy) là một trong những chỉ số phổ biến nhất để đánh giá hiệu suất của mô hình phân loại. Nó đo lường tỷ lệ dự đoán chính xác so với tổng số dự đoán.

Công thức tính độ chính xác:

$$\text{Accuracy} = \frac{\text{Số lượng dự đoán đúng}}{\text{Tổng số dự đoán}}$$

- **Số lượng dự đoán đúng** là số lượng mẫu mà mô hình phân loại đúng (tức là nhãn dự đoán của mô hình trùng khớp với nhãn thực tế).
- **Tổng số dự đoán** là tổng số mẫu mà mô hình dự đoán (cả đúng và sai).

- Trong Keras, khi biên dịch mô hình, chỉ cần chỉ định metrics=['accuracy'], và Keras sẽ tự động tính toán độ chính xác cho ta trong mỗi epoch, cả khi huấn luyện (history.history['accuracy']) và khi kiểm tra trên tập validation (history.history['val_accuracy']).

Quy trình tính độ chính xác trong quá trình huấn luyện:

- **So sánh giữa giá trị thực tế và giá trị dự đoán**: Mô hình sẽ tính toán số lượng dự đoán đúng (sự khớp giữa nhãn thực tế và nhãn dự đoán) trong mỗi batch.
- **Tính toán tổng tỷ lệ chính xác**: Mỗi epoch, Keras sẽ tính toán tỷ lệ chính xác cho tất cả các mẫu trong tập huấn luyện và tập validation.

- **Ví dụ**: Giả sử, trong một batch huấn luyện gồm 10 mẫu, nếu mô hình dự đoán đúng 7 mẫu, thì tỷ lệ chính xác cho batch đó là 70%.

ii, Mất mát (Loss)

- Mất mát (loss) là chỉ số đo lường sự khác biệt giữa giá trị dự đoán của mô hình và giá trị thực tế. Hàm mất mát là yếu tố chính giúp tối ưu hóa mô hình trong quá trình huấn luyện, nhằm giảm dần sự sai lệch giữa giá trị thực tế và giá trị dự đoán.

Công thức tính categorical crossentropy:

- Trong bài toán phân loại nhiều lớp, Keras sử dụng hàm **categorical crossentropy** để tính mất mát. Đây là công thức thường dùng trong các bài toán phân loại mà nhãn là dạng one-hot encoded (mỗi lớp được biểu diễn bằng một vector nhị phân).

$$\text{Loss} = -\sum_{i} y_i \log(p_i)$$

Trong đó:

- y_i là giá trị thực tế (one-hot encoded) cho lớp i,
- p_i là xác suất dự đoán của mô hình cho lớp i.

- Công thức này tính toán mất mát cho mỗi mẫu trong tập huấn luyện. Mỗi phần tử yiy_iyi là 1 nếu lớp đó là lớp đúng, và 0 nếu không phải. Do đó, hàm mất mát chỉ xem xét log⁡(pi)\log(p_i)log(pi) tại vị trí của lớp đúng.

Mất mát trong Keras:

- Khi biên dịch mô hình, hàm mất mát được chỉ định như sau:

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

- Kết quả mất mát sẽ được lưu trữ trong history.history['loss'] cho huấn luyện và history.history['val_loss'] cho tập kiểm tra.

iii, Biểu đồ huấn luyện:

- Các biểu đồ chính xác và mất mát được vẽ trong suốt quá trình huấn luyện và validation. Biểu đồ độ chính xác cho thấy sự thay đổi về độ chính xác trên cả tập huấn luyện và tập validation theo từng epoch.

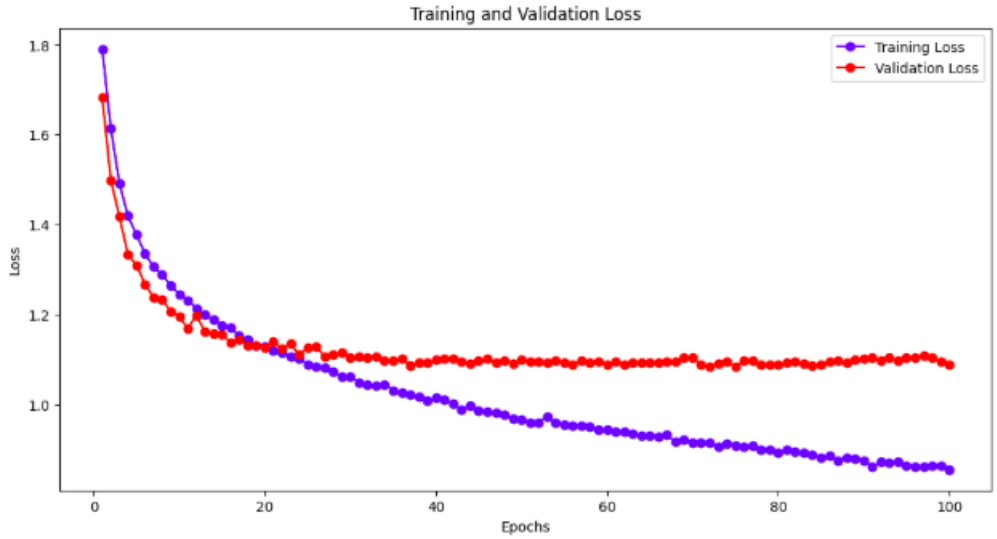
- Biểu đồ mất mát (loss) cho thấy mức độ giảm mất mát của mô hình trên cả tập huấn luyện và tập validation trong suốt quá trình huấn luyện.

- Các biểu đồ này giúp theo dõi và đánh giá mức độ học của mô hình qua các epoch, đặc biệt là xem mô hình có bị **overfitting** (quá khớp với dữ liệu huấn luyện) hay không, dựa trên sự thay đổi của độ chính xác và mất mát trên tập validation.

plt.subplot(1, 2, 1) # Biểu đồ chính xác

plt.subplot(1, 2, 2) # Biểu đồ mất mát

2.9, Số liệu đánh giá .



- Đây là hai đồ thị thể hiện kết quả huấn luyện mô hình sau 100 epochs đầu:

1. **Đồ thị Accuracy (Độ chính xác)**:

   - Từ epochs thứ 25 trở lên:

   + Training accuracy tăng

   + Validation accuracy chững lại

-Nhận xét: Độ chính xác của tập huấn luyện tăng nhanh và ổn định. Tuy nhiên, độ chính xác trên tập kiểm tra có vẻ đạt đỉnh sớm và sau đó dao động nhẹ. Điều này có thể chỉ ra rằng mô hình bắt đầu bị overfitting (học quá tốt trên tập huấn luyện, nhưng không tổng quát tốt trên dữ liệu mới).

2. **Đồ thị Loss (Mất mát)**:

- Từ epochs thứ 20 trở lên:

+ Training loss giảm

+ Validation loss chững lại

-Nhận xét: Mất mát của tập huấn luyện giảm mạnh, điều này chứng tỏ mô hình đang học tốt hơn theo thời gian. Tuy nhiên, mất mát của tập kiểm tra có xu hướng giảm chậm hơn và dao động sau một thời gian, điều này cho thấy mô hình có thể đang gặp khó khăn trong việc tổng quát hóa trên tập kiểm tra.

2.10. Phân tích và so sánh giữa các cách:

Các cách/ tên file	Sử dụng	Mất mát Độ chính xác (accuracy)	Nhận xét
Cách1: train_model	B4: hàm train_test_split	0.54207301139831 54 1.60082554817199 7 (Máy đức)	
train_model	B5: Mô hình CNN gồm 8 lớp: • Conv2D: Lớp tích chập 2D - 2 lớp • MaxPooling2D: Lớp giảm kích thước - 2 lớp • Flatten: Làm phẳng đầu ra • Dense: Lớp kết nối đầy đủ - 2 lớp • Dropout: Giảm overfitting - optimizer 'adam' và hàm mất mát 'categorical_crossentropy'	1 lần khác chạy: Test Loss: 1.44631588459014 9 Test Accuracy: 0.53956532478332 52 ở epochs 25: accuracy: 0.6792 - loss: 0.7952 - val_accuracy: 0.5396 - val_loss: 1.4463	
train_model	Thêm 1 lớp tích chập, 1 lớp maxpooling	Test Loss: 1.09228563308715	Khá tốt : -acc tăng nhưng ko cao:

		82 Test Accuracy: 0.591947615146637 ở epoch 1: accuracy: 0.2440 - loss: 1.8236 - val_accuracy: 0.3721 - val_loss: 1.6160 ở epoch 25: accuracy: 0.5963 - loss: 1.0665 - val_accuracy: 0.5919 - val_loss: 1.0923	đạt 0.5933(sau tất cả vs dữ liệu ktra) - loss giảm đều -val-acc: nhìn chung tăng đều -val-loss: nhìn chung giảm đều
train_model	Thêm 50 epoch	Test Loss: 1.0821202993392944 Test Accuracy: 0.6047645807266235 accuracy: 0.6006 - loss: 1.0805 epoch 1(hoặc 26?) accuracy: 0.5964 - loss: 1.0492 - val_accuracy: 0.5853 - val_loss: 1.1061 epoch 50 accuracy: 0.6708 - loss: 0.8642 - val_accuracy: 0.6048 - val_loss: 1.0821	Tốt hơn trước : -acc tăng: đạt 0.6006(sau tất cả vs dữ liệu ktra) - loss giảm đều -val-acc: nhìn chung tăng đều -val-loss: nhìn chung giảm đều
Cách 3 /training	# Chuyển đổi dữ liệu thành array 4D và one-hot encoding nhãn (20) Shuffle dữ liệu trước khi chia Sử dụng StratifiedShuffleSplit Tăng cường dữ liệu B5: Vẫn là mô hình CNN 8 lớp như lần 1	Test Loss: 1.5553220510482788 Test Accuracy: 0.5430482029914856	

training	Huấn luyện tiếp của lần 2 thêm 10 epochs	Test Loss after continued training: 1.9709891080856323 Test Accuracy after continued training: 0.5488994121551514	
training	SD như lần 2 nhưng thay đổi thành one-hot encoding nhãn (10)	Test Loss: 1.5031154155731201 Test Accuracy: 0.540679872035980 2	- Từ epoch 1=>13: Nhìn chung: acc tăng, loss giảm. val-acc tăng, loss giảm -Nhưng từ 13-25: +Mặc dù acc tăng, loss giảm +val-acc chững lại (ko tăng mấy), loss chững lại và tăng
	Thêm lớp tích chập, maxpooling	Test Loss: 1.0903279781341553 Test Accuracy: 0.58846473693847 66 ở epoch 1: accuracy: 0.2364 - loss: 1.8396 - val_accuracy: 0.3123 - val_loss: 1.6818 ở epoch 25: accuracy: 0.5985 - loss: 1.0640 - val_accuracy: 0.5885 - val_loss: 1.0903	Khá tốt : -acc tăng nhưng ko cao: đạt 0.584(sau tất cả với dữ liệu ktra) - loss giảm đều -val-acc: nhìn chung tăng đều -val-loss: nhìn chung giảm đều
	Thêm 70 epoch	Test Loss: 1.0842759609222412 Test Accuracy: 0.611312329769134 5 accuracy: 0.6029 - loss: 1.1023	Tốt hơn: -acc tăng: đạt 0.6029(sau tất cả vs dữ liệu ktra) - loss giảm đều -val-acc: nhìn chung tăng đều -val-loss: nhìn chung giảm đều

		ở epoch 1 accuracy: 0.5968 - loss: 1.0599 - val_accuracy: 0.5926 - val_loss: 1.0916 ở epoch 70 accuracy: 0.6781 - loss: 0.8388 - val_accuracy: 0.6113 - val_loss: 1.0843	
Cách 2 (test)	Giống lần 1 nhưng thêm Seed vào: (Seed giúp cố định đánh giá, không thay đổi KQ)	Test Loss: 1.4679433107376099 Test Accuracy: 0.5323209762573242	
	Thay đổi thêm lớp tích chập, lớp maxpooling,, thêm lớp BatchNormalization	Test Loss: 1.1902110576629639 Test Accuracy: 0.5533574819564819	Đều hơn: -acc tăng nhưng ko cao đạt 0.5992, loss giảm -val-acc: lúc đầu tăng, nhưng về sau biến động -val-loss:lúc đầu giảm mạnh(1.6946->1.1918), về sau chững lại ~ 1.13-1.19)
	bỏ lớp BatchNormalization -	Test Loss: 1.1266142129898071 Test Accuracy: 0.5792700052261353 accuracy: 0.2471 - loss: 1.8252 - val_accuracy: 0.3207 - val_loss: 1.6827 accuracy: 0.5897 - loss: 1.0786 - val_accuracy: 0.5793 - val_loss: 1.1266	Tạm ổn : -acc tăng nhưng ko cao: đạt 0.5809(sau tất cả vs dữ liệu ktra) - loss giảm đều Nó thay đổi mạnh ở những epoch đầu(1-7 đầu tiên) nhưng về sau chững quanh khoảng giá trị: -val-acc: nhìn chung tăng đều (0.57~0.58) -val-loss: nhìn chung giảm đều(0.12~0.2)
	Mình thấy ở trước đó nó có xu	Test Loss:	Về cơ bản, mình thấy tốt

	hướng tăng accuracy nhưng do có 25 epoch nên chưa phát huy. => nên mình Tăng số lượng epoch lên 50	1.1000391244888306 Test Accuracy: 0.6022568941116333 epoch 50: accuracy: 0.6363 - loss: 0.9611 - val_accuracy: 0.6023 - val_loss: 1.1000 Cuối cùng: tập ktra: accuracy: 0.6059 - loss: 1.0994	hơn tất cả những lần thử: -acc,val-acc khá cao(>=0.6) và xêm nhau, ko bị chênh lệch nhiều -loss,val-loss cũng giảm đáng kể và xêm nhau (0.96-1.1)
	Tăng lên 100 epoch	Test Loss: 1.0887961387634277 Test Accuracy: 0.6086653470993042 epoch 100: accuracy: 0.6842 - loss: 0.8382 - val_accuracy: 0.6087 - val_loss: 1.0888 Cuối cùng trên tập ktra: accuracy: 0.6088 - loss: 1.0827	Có tăng accuracy nhưng có sự chững lại của val-acc và val-loss(từ khoảng epoch 50 trở đi): -acc,val-acc bắt đầu có sự chênh lệch(xấp xỉ 0.08) -loss,val-loss cũng bắt đầu chênh lệch (xấp xỉ 0.2)

2.11.Phân tích các chỉ số:

- Ở đây, chúng em phân tích chỉ số của file test (Cách 2) sau những lần thử nghiệm.

1. **Test Loss**:
   - **Test Loss** (Mất mát kiểm tra) là giá trị mất mát (loss) của mô hình khi chạy trên tập dữ liệu kiểm tra (test set). Nó đo lường sự khác biệt giữa các dự đoán của mô hình và nhãn thực tế của dữ liệu kiểm tra.
   - **Giải thích**: Một giá trị **loss** thấp cho thấy mô hình có khả năng dự đoán chính xác, trong khi một giá trị cao chỉ ra rằng mô hình chưa học đúng hoặc chưa tối ưu hóa đủ tốt. Mức **loss** là 1.08879 trong trường hợp này, có thể

cho thấy mô hình vẫn còn khoảng cách để cải thiện, nhưng không phải là quá tệ.
    - **Mục tiêu**: Khi huấn luyện mô hình, bạn sẽ muốn giá trị **loss** giảm xuống qua các epoch, cho thấy mô hình đang học tốt hơn và tối ưu hóa dần dần.
2. **Test Accuracy**:
    - **Test Accuracy** (Độ chính xác kiểm tra) đo lường tỷ lệ phần trăm dự đoán chính xác trên tổng số dự đoán trong tập dữ liệu kiểm tra. Đây là một chỉ số quan trọng để đánh giá hiệu quả của mô hình.
    - **Giải thích**: Độ chính xác **0.6086**(tức là 60.86%) có thể cho thấy rằng mô hình đã học được một số mẫu cảm xúc khuôn mặt từ dữ liệu huấn luyện, nhưng vẫn có thể cải thiện được. Một độ chính xác 60.86% là thấp đối với một bài toán phân loại có nhiều lớp (7 lớp cảm xúc), và có thể cho thấy mô hình cần thêm thời gian huấn luyện hoặc cải thiện trong việc tối ưu hóa các tham số.

1. Đánh giá chung:

- **Test Loss**: 1.08879 cho thấy mô hình chưa đạt được kết quả tối ưu, nhưng vẫn có khả năng cải thiện trong quá trình huấn luyện thêm.
- **Test Accuracy**: 60.86% là mức độ chính xác vừa phải. Mô hình có thể chưa được huấn luyện đủ lâu hoặc có thể cần tinh chỉnh thêm (ví dụ như thay đổi kiến trúc mô hình hoặc tối ưu hóa tham số).

2. Định hướng những bước tiếp theo để cải thiện mô hình:

1. **Tăng số lượng epoch**: Đào tạo mô hình lâu hơn có thể giúp cải thiện độ chính xác. Bạn có thể thử huấn luyện trong nhiều epoch hơn để giúp mô hình học được các đặc điểm phức tạp hơn của dữ liệu.
2. **Điều chỉnh siêu tham số**: Thử nghiệm với các giá trị khác nhau của learning rate, batch size, số lớp (layers) của mô hình hoặc số lượng filters trong các lớp convolutional.
3. **Sử dụng các kỹ thuật tăng cường dữ liệu**: Nếu chưa đủ mạnh, bạn có thể áp dụng thêm các kỹ thuật tăng cường dữ liệu như biến đổi ảnh mạnh hơn (ví dụ: thay đổi độ sáng, độ tương phản, vv) để làm phong phú thêm dữ liệu huấn luyện.
4. **Kiến trúc mô hình phức tạp hơn**: Nếu mô hình hiện tại chưa đủ mạnh, có thể thử thêm các lớp convolutional hoặc thêm các lớp học sâu (deep learning) để làm phong phú thêm khả năng học của mô hình.
5. **Sử dụng các kỹ thuật giảm thiểu overfitting**: Nếu có hiện tượng overfitting (hiệu suất trên tập huấn luyện tốt nhưng trên tập kiểm tra lại thấp), bạn có thể thử dùng các kỹ thuật như Dropout hoặc Regularization để giúp mô hình tổng quát tốt hơn.

3. Đánh giá kết quả

- Chúng em đã xây dựng thành công mô hình học sâu sử dụng mạng nơ-ron tích chập (CNN) cho bài toán nhận diện cảm xúc khuôn mặt, áp dụng bộ dữ liệu FER-2013. Mô hình sử dụng các lớp Conv2D, MaxPooling2D và các lớp fully connected (Dense) để phân loại 7 loại cảm xúc.

- Sau khi hoàn thành việc xây dựng mô hình, chúng em tiến hành huấn luyện trong 40 epoch với kỹ thuật tăng cường dữ liệu (data augmentation) để cải thiện hiệu suất và giảm overfitting. Mô hình đã được đánh giá với các tham số tối ưu và sử dụng **ModelCheckpoint** để lưu lại mô hình tốt nhất trong suốt quá trình huấn luyện.

- Kết quả đạt được là độ chính xác trên tập kiểm tra đạt 0.6086 và mất mát (loss) trên tập kiểm tra là 1.08879. Các biểu đồ cho thấy mô hình dần dần cải thiện trong quá trình huấn luyện và đạt được kết quả tốt nhất vào cuối 40 epoch.

PHẦN IV. KẾT LUẬN

1, Những điều đã làm được

- Sau khi thực hiện đề tài "Xây dựng hệ thống nhận diện cảm xúc khuôn mặt," chúng em đã đạt được một số kết quả đáng kể như sau:

+) Chương 1: Chúng em đã nghiên cứu tổng quan về nhận diện cảm xúc khuôn mặt, hiểu được các khái niệm cơ bản và các ứng dụng của công nghệ này trong đời sống, như phát hiện cảm xúc trong các cuộc hội thoại, chăm sóc khách hàng, và cải thiện trải nghiệm người dùng trong các sản phẩm công nghệ.

+) Chương 2: Chúng em đã tìm hiểu các phương pháp học máy và học sâu, đặc biệt là ứng dụng mạng nơ-ron tích chập (CNN) trong nhận diện cảm xúc khuôn mặt. Đồng thời, chúng em đã phân tích bộ dữ liệu **fer2013** và ứng dụng các phương pháp tiền xử lý dữ liệu, như chuẩn hóa hình ảnh và tăng cường dữ liệu (augmentation), để cải thiện hiệu quả nhận diện.

+) Chương 3: Chúng em đã xây dựng thành công mô hình học sâu sử dụng **Convolutional Neural Networks (CNN)** để nhận diện các cảm xúc từ khuôn mặt. Sau khi huấn luyện, mô hình đã cho ra kết quả với độ chính xác khoảng 60.866**%** trên tập dữ liệu kiểm tra, chứng minh hiệu quả của phương pháp học sâu trong bài toán này.

2, Những điều chưa làm được

- Bên cạnh những kết quả đạt được, vẫn còn một số hạn chế trong báo cáo cần được cải thiện:

+) Báo cáo chủ yếu tập trung vào lý thuyết và các mô hình học sâu cơ bản, chưa thực sự xây dựng được một ứng dụng hoàn chỉnh để triển khai trong môi trường thực tế với dữ liệu thực.

+) Mặc dù đã áp dụng mô hình CNN, dữ liệu phân tích vẫn chưa hoàn toàn tối ưu và có dấu hiệu **overfitting**, tức là mô hình có thể học quá kỹ các đặc điểm của dữ liệu huấn luyện mà không tổng quát tốt với dữ liệu mới. Cần phải tinh chỉnh thêm trong quá trình tiền xử lý dữ liệu và áp dụng các kỹ thuật cải thiện như dropout hay regularization.

+) Việc cải thiện độ chính xác của mô hình hiện tại còn là một thách thức, vì mô hình chỉ đạt độ chính xác khoảng **60.866%**.(Đối với cách 2) Hơn nữa, còn nhiều loại cảm xúc khác nhau cần được nhận diện với độ chính xác cao hơn.

3, Hướng phát triển trong tương lai

- Dựa trên các kết quả đạt được và các hạn chế hiện tại, chúng em đưa ra một số hướng phát triển trong tương lai:

+) Nghiên cứu sâu hơn về các mô hình học sâu: Chúng em dự định tìm hiểu và áp dụng các mô hình học sâu phức tạp hơn như **ResNet**, **VGGNet**, hoặc **Inception** để cải thiện độ chính xác của hệ thống nhận diện cảm xúc. Các mô hình này có khả năng nhận diện các đặc điểm phức tạp trong ảnh và có thể giúp giảm thiểu hiện tượng overfitting.

+) Tối ưu hóa dữ liệu đầu vào: Một phần quan trọng trong việc cải thiện độ chính xác của mô hình là tối ưu hóa dữ liệu đầu vào. Chúng em sẽ áp dụng các kỹ thuật như **augmentation** dữ liệu thêm, cải tiến chất lượng hình ảnh và áp dụng các phương pháp tiền xử lý khác như histogram equalization để tăng cường chất lượng đầu vào cho mô hình.

+) Ứng dụng trong thực tế: Chúng em muốn phát triển hệ thống nhận diện cảm xúc khuôn mặt thành một ứng dụng thực tế, có thể triển khai trên các thiết bị như điện thoại thông minh hoặc máy tính, phục vụ cho các mục đích như hỗ trợ chăm sóc khách hàng, nhận diện cảm xúc trong các video hội nghị, hoặc giúp cải thiện trải nghiệm người dùng trong các hệ thống AI.

+) Nâng cao độ chính xác của mô hình: Chúng em cũng dự định tiếp tục cải thiện mô hình bằng cách tinh chỉnh các siêu tham số (hyperparameters), áp dụng các kỹ thuật như **transfer learning**, và thử nghiệm với các bộ dữ liệu lớn hơn để có thể nâng cao độ chính xác và khả năng nhận diện các cảm xúc phức tạp hơn.

TÀI LIỆU THAM KHẢO

[1] Rui Xu, Donald C. Wunsch II, Survey of clustering algorithms. IEEE Trans. Neural Networks 16(3): 645-678, 2005.

[2] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl: Constrained K-means Clustering with Background Knowledge. ICML 2001: 577-584, 2001.

[3] Dataset kaggle FER-2013 https://www.kaggle.com/datasets/msambare/fer2013/data

PHỤ LỤC