

UỶ BAN NHÂN DÂN TP. HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

❑❑❑❑❑

BÁO CÁO CUỐI KỲ

MÔN HỌC:

KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG (841447)

ĐỀ TÀI:

ỨNG DỤNG THUẬT TOÁN XGBOOST PHÂN LOẠI CẢM XÚC DỰA TRÊN NỘI DUNG BÀI ĐĂNG

**Giảng** **viên:** ThS. Nguyễn Thanh Phước.

**Lớp** **:** DCT1223

**Thực hiện:** Trần Ngô Nhật Nam - 3122410253

*TPHCM, ngày 10 tháng 4 năm 2025*

***Mục lục***

[**Phần I: Giới thiệu** 4](#_Toc185369891)

[**1. Lý do chọn đề tài** 4](#_Toc185369892)

[**2. Vấn đề của bài toán** 4](#_Toc185369893)

[**3. Đặt giả thiết** 4](#_Toc185369894)

[**4. Các bài toán tương tự trong thực tế** 4](#_Toc185369895)

[**Phần II: Chuẩn bị dữ liệu** 5](#_Toc185369896)

[**1. Giới thiệu về bộ dữ liệu** 5](#_Toc185369897)

[**2. Mô tả các đặc trưng** 5](#_Toc185369898)

[**Phần III: Xử lý dữ liệu** 5](#_Toc185369899)

[**1. Xử lý dữ liệu thiếu** 5](#_Toc185369900)

[**2. Xử lý dữ liệu ngoại lệ** 5](#_Toc185369901)

[**3. Mã hóa dữ liệu** 5](#_Toc185369902)

[**3.1. Sử dụng tên thư mục làm nhãn:** 5](#_Toc185369903)

[**3.2. Lợi ích đạt được:** 5](#_Toc185369904)

[**Phần IV: Phân tích dữ liệu** 6](#_Toc185369905)

[**1. Mức độ xuất hiện của từng ký hiệu** 6](#_Toc185369906)

[**2. Mức độ đa dạng góc chụp** 6](#_Toc185369907)

[**Phần V: Trích xuất đặc trưng** 7](#_Toc185369908)

[**1. Xử lý dữ liệu** 7](#_Toc185369909)

[**1.1. Xóa nền ảnh và giữ lại vùng bàn tay** 7](#_Toc185369910)

[**1.2. Làm mờ ảnh bằng Gaussian Blur:** 7](#_Toc185369911)

[**1.3. Ngưỡng hóa ảnh (Thresholding):** 7](#_Toc185369912)

[**1.4. Ngưỡng hóa nhị phân (Binary Thresholding):** 7](#_Toc185369913)

[**1.5. Ngưỡng OTSU (cv2.THRESH\_OTSU):** 7](#_Toc185369914)

[**2. Nhận xét đánh giá** 7](#_Toc185369915)

[**2.1. Kết quả đạt được:** 7](#_Toc185369916)

[**2.2. Lợi ích của ảnh nhị phân (đen trắng)**: 8](#_Toc185369917)

[**Phần VI: Chuẩn hóa dữ liệu** 8](#_Toc185369918)

[**1. Chuyển đổi ảnh về mảng số numpy** 8](#_Toc185369919)

[**2. Chuẩn hóa cường độ pixel** 10](#_Toc185369920)

[**3 .Kích thước ảnh đồng nhất** 10](#_Toc185369921)

[**Phần VII: Huấn luyện mô hình** 10](#_Toc185369922)

[**1. Chia dữ liệu huấn luyện** 10](#_Toc185369923)

[**2. Kiến trúc mô hình LeNet-5 đã điều chỉnh** 11](#_Toc185369924)

[**2.1. Các lớp tích chập và gộp (Convolution + Pooling):** 11](#_Toc185369925)

[**2.2. Lớp chuyển đổi (Flatten):** 11](#_Toc185369926)

[**2.3. Các lớp kết nối đầy đủ (Fully Connected):** 11](#_Toc185369927)

[**2.4. Bộ tối ưu hóa và hàm mất mát:** 11](#_Toc185369928)

[**2.5. Đoạn code mô hình sau khi được tối ưu hóa để huấn luyện:** 11](#_Toc185369929)

[**3. Huấn luyện mô hình** 12](#_Toc185369930)

[**3.1. Số lượng epoch:** 12](#_Toc185369931)

[**3.2. Batch size:** 12](#_Toc185369932)

[**3.3. Callbacks sử dụng:** 12](#_Toc185369933)

[**3.4. Kết quả huấn luyện:** 13](#_Toc185369934)

[**Phần VIII: Đánh giá mô hình** 13](#_Toc185369935)

[**1. Phép đo được sử dụng** [2] 13](#_Toc185369936)

[**2. Đánh giá kết quả** 13](#_Toc185369937)

[**Phần IX: Điều chỉnh mô hình** 16](#_Toc185369938)

[**1. Mô hình LeNet-5 ban đầu:** 16](#_Toc185369939)

[**2. Quá trình điều chỉnh tạo ra mô hình Lenet-5 mới:** 18](#_Toc185369940)

[**2.1. Kiến trúc tích chập:** 18](#_Toc185369941)

[**2.2. Số lượng lớp:** 18](#_Toc185369942)

[**2.3. Số lượng nơ-ron và lớp đầu ra:** 18](#_Toc185369943)

[**2.4. Tối ưu hóa:** 18](#_Toc185369944)

[**3. Nhận xét đánh giá:** 19](#_Toc185369945)

[**3.1. Mục đích:** 19](#_Toc185369946)

[**3.2. Các cải tiến:** 19](#_Toc185369947)

[**3.3. Hạn chế tiềm năng:** 19](#_Toc185369948)

[**Phần X: Kết luận** 19](#_Toc185369949)

[**1. Nhận xét đánh giá:** 19](#_Toc185369950)

[**2.** **Tài liệu tham khảo** 20](#_Toc185369951)

**Mục lục hình ảnh**

[**Hình 1: 5 Ảnh ngẫu nhiên trong bộ dữ liệu 4**](#_Toc185367348)

[**Hình 2: 5 Ảnh ngẫu nhiên có nhãn trong bộ dữ liệu 4**](#_Toc185367349)

[**Hình 3: Tầng suất của các nhãn trong bộ dữ liệu 6**](#_Toc185367350)

[**Hình 4: 36 Folder của bộ dữ liệu 6**](#_Toc185367351)

[**Hình 5: Ảnh minh họa góc các góc chụp của bàn tay 6**](#_Toc185367352)

[**Hình 6: Ảnh minh họa quá trình tách nền 7**](#_Toc185367353)

[**Hình 7: Ảnh minh họa quá trình chuyển đổi sang ảnh nhị phân. 7**](#_Toc185367354)

[**Hình 8: Kết quả huấn luyện mô hình. 13**](#_Toc185367355)

[**Hình 9: Biểu đồ kết quả loss và accuracy. 13**](#_Toc185367356)

[**Hình 10: 5 Ảnh ngẫu nhiên của nhãn “H”. 14**](#_Toc185367357)

[**Hình 11: 5 Ảnh ngẫu nhiên của nhãn “G”. 15**](#_Toc185367358)

[**Hình 12: File ảnh thứ 33 trong tập “H” 15**](#_Toc185367359)

[**Hình 13: File ảnh thứ 33 trong tập “G" 15**](#_Toc185367360)

[**Hình 14: Kết quả SSIM của 2 ảnh "H" và "G" trong tập dữ liệu 15**](#_Toc185367361)

[**Hình 15: Kiến trúc mạng LeNet-5. 16**](#_Toc185367362)

# **Phần I: Giới thiệu**

## **Lý do chọn đề tài**

Trong thời đại số hóa hiện nay, mạng xã hội đóng vai trò ngày càng quan trọng trong việc phản ánh quan điểm, cảm xúc và hành vi của người dùng. Việc phân tích cảm xúc từ các bài đăng trên mạng xã hội không chỉ giúp các doanh nghiệp nắm bắt xu hướng người tiêu dùng mà còn hỗ trợ trong việc phát hiện khủng hoảng truyền thông, cải thiện dịch vụ khách hàng, và đưa ra chiến lược tiếp thị hiệu quả.

## **Mục đích của dự án**

Dự án này nhằm mục tiêu **phân tích và khai phá dữ liệu cảm xúc người dùng trên mạng xã hội**, thông qua việc áp dụng các kỹ thuật tiền xử lý, trích xuất đặc trưng văn bản (như **TF-IDF**), và mô hình học máy (như **XGBoost**) để phân loại cảm xúc của bài đăng thành các nhóm như **tích cực(positive)**, **tiêu cực(negative)** hoặc **trung tính(neutral)**.

## **Phạm vi của dự án**

Phạm vi của dự án tập trung vào việc xử lý dữ liệu văn bản, trích xuất đặc trưng, huấn luyện mô hình và đánh giá hiệu suất mô hình trong bài toán phân tích cảm xúc.

## **Các bài toán tương tự trong thực tế**

Một số bài toán tương tự với bài toán phân tích cảm xúc từ mạng xã hội. Các bài toán này đều nằm trong lĩnh vực **Khai phá dữ liệu (Data Mining)** và **Xử lý ngôn ngữ tự nhiên (NLP):**

1. **Phân loại bình luận sản phẩm (Product Review Classification)**

**Mô tả:** Dự đoán cảm xúc của người dùng thông qua đánh giá (review) trên các trang thương mại điện tử như Amazon, Tiki, Shopee,...

**Ứng dụng**: Xác định mức độ hài lòng của khách hàng, cải thiện chất lượng sản phẩm/dịch vụ.

1. **Phân loại tin tức (News Categorization)**

**Mô tả:** Phân loại bài báo hoặc tiêu đề tin tức vào các nhóm chủ đề như thể thao, chính trị, kinh tế, giải trí,...

**Ứng dụng:** Hệ thống gợi ý tin tức, phân luồng nội dung, lọc thông tin theo sở thích người dùng.

1. **Phát hiện phát ngôn thù ghét (Hate Speech Detection)**

**Mô tả:** Nhận diện các phát ngôn độc hại, phân biệt chủng tộc, giới tính, tôn giáo,... trên mạng xã hội.

**Ứng dụng:** Giúp các nền tảng mạng xã hội kiểm duyệt nội dung, bảo vệ cộng đồng.

Tuy các bài toán tương đối giống nhau nhưng quá trình rút trích đặc trưng lại có thể hoàn toàn khác tùy thuộc vào mô hình và đầu ra của bài toán.

# **Phần II: Mô tả dữ liệu**

## **Giới thiệu về bộ dữ liệu**

Bộ dữ liệu được cung cấp bởi người dùng **Kaggle có tên Kashish Parmar**, là tập hợp các bài đăng trên mạng xã hội được gán nhãn cảm xúc. Dữ liệu được chia sẻ công khai với mục đích phục vụ học thuật và nghiên cứu trong lĩnh vực khai phá dữ liệu và xử lý ngôn ngữ tự nhiên (NLP).

## **Cấu trúc bộ dữ liệu**

Bộ dữ liệu được xây dựng từ các bài đăng trên mạng xã hội, bao gồm nhiều thuộc tính mô tả nội dung, người dùng, thời gian và mức độ tương tác. Cụ thể, bộ dữ liệu bao gồm các trường sau:

|  |  |
| --- | --- |
| **Thuộc tính** | **Mô tả** |
| Text | Nội dung do người dùng tạo ra, phản ánh cảm xúc hoặc quan điểm. |
| Sentiment | Nhãn cảm xúc được gán cho bài đăng (tích cực, tiêu cực hoặc trung tính). |
| Timestamp | Thời gian cụ thể mà bài đăng được đăng tải. |
| User | Mã định danh duy nhất của người dùng đăng bài. |
| Platform | Nền tảng mạng xã hội nơi bài viết được đăng (như Twitter, Facebook, v.v.). |
| Hashtags | Các hashtag được sử dụng trong bài đăng, phản ánh chủ đề đang thịnh hành. |
| Likes | Số lượt thích, thể hiện mức độ tương tác từ người xem. |
| Retweets | Số lượt chia sẻ lại (retweet), thể hiện mức độ lan truyền của nội dung. |
| Country | Quốc gia nơi người dùng đăng bài. |
| Year | Năm bài viết được đăng. |
| Month | Tháng bài viết được đăng. |
| Day | Ngày bài viết được đăng. |
| Hour | Giờ bài viết được đăng. |

Bộ dữ liệu có kích thước khoảng hơn **700** dòng, với mỗi dòng tương ứng một bài đăng duy nhất. Các thuộc tính trong tập dữ liệu không chỉ phục vụ cho phân tích cảm xúc, mà còn hỗ trợ trong việc phân tích hành vi người dùng theo **thời gian, vị trí địa lý, nền tảng sử dụng, và mức độ tương tác với nội dung**. Chúng ta sẽ đặc biệt quan tâm 2 cột thuộc tính chính đó là **Text** và **Sentiment**, đây là 2 cột thuộc tính **input của bài toán.**

# **Phần III: Xử lý dữ liệu**

## **Xử lý dữ liệu thiếu**

Vì đây là bộ dữ liệu đã được xử lý sẵn các phần dữ liệu bị thiếu, sai nhãn hoặc dữ liệu bị mất nên quá trình xử lý này chúng ta không cần xử lý nữa

## **Xử lý dữ liệu ngoại lệ**

Tương tự như trên, chúng ta không cần phải xử lý lại dữ liệu ngoại lệ đối với tập dữ liệu này.

## **Mã hóa dữ liệu**

Nhãn của dữ liệu bao gồm cả chữ cái từ A-Z và các chữ số từ 0-9, đồng thời tên nhãn được xác định dựa trên tên thư mục tương ứng, nên việc chuẩn hóa nhãn không cần phải xử lý quá nhiều. Cụ thể:

### **Sử dụng tên thư mục làm nhãn:**

Tên các thư mục con trong đường dẫn dữ liệu được sử dụng để xác định nhãn (ví dụ: "A", "B", ..., "Z", "0", "1", ..., "9").

Mỗi thư mục tương ứng với một lớp, không cần chuẩn hóa nhãn thủ công.

### **Lợi ích đạt được:**

**Tự động hóa:** Tên thư mục được sử dụng trực tiếp, giảm sai sót khi chuẩn hóa thủ công.

**Tương thích:** Dữ liệu được chuẩn bị phù hợp cho mô hình với đầu ra là one-hot encoding(phần này sẽ được đề cập trong quá trình huấn luyện mô hình).

**Linh hoạt:** Phù hợp với mọi tập dữ liệu có cấu trúc tương tự (ảnh được lưu trong các thư mục con), dễ dàng lấy được folder ảnh dựa trên nhãn mà không cần phải nhớ xem folder có tên là gì.

# **Phần IV: Phân tích dữ liệu**

## **Mức độ xuất hiện của từng ký hiệu**

Vì là tập dữ liệu đã được thu thập và xử lý trước nên tầng suất các nhãn là đồng đều nhau, bộ dữ liệu ổn định về mặt số lượng. A graph of blue and black lines

Description automatically generated

Hình 3: Tầng suất của các nhãn trong bộ dữ liệu

A group of folders with a hole in the middle

Description automatically generated

Hình 4: 36 Folder của bộ dữ liệu

## **Mức độ đa dạng góc chụp**

Góc chụp bàn tay ở đây chỉ dùng góc chính diện của bàn tay trái (vị trí giống với bàn tay đầu tiên trong ảnh) lý do là vì khi thay đổi góc chụp có thể sẽ làm nhiễu đặc trưng “hình dạng” bàn tay mà chúng ta cần, chưa kể khi đổi góc chụp có thể sẽ khiến hình dạng của bàn tay khi rút trích đặc trưng bị trùng với ảnh khác nhãn gây nhiễu thông tin. 

Hình 5: Ảnh minh họa góc các góc chụp của bàn tay

# **Phần V: Trích xuất đặc trưng**

## **Xử lý dữ liệu**

Để trích xuất được đặc trưng “hình dạng” bàn tay của ảnh, chúng ta thực hiện xử lý ảnh theo từng bước:

### **Xóa nền ảnh và giữ lại vùng bàn tay**

Bằng cách sử dụng thư viện xóa nền ảnh có sẵn, chúng ta có thể dễ dàng tách dược ảnh vùng bàn tay

from rembg import remove  
rem = remove(image)

Sử dụng hàm xóa nền ảnh **remove** có sẵn từ thư viện **rembg**, đây là môt hàm xóa ảnh và giữ lại vật thể trung tâm rất tốt nhưng quá trình xử lý có đôi chút chậm. Kết quả đạt được là 1 tấm ảnh trích xuất được vùng bàn tay.

|  |
| --- |
| A hand making a peace sign  Description automatically generated |

Hình 6: Ảnh minh họa quá trình tách nền

### **Làm mờ ảnh bằng Gaussian Blur:**

blur = cv2.GaussianBlur(gray, (blurValue, blurValue), 0)

Sử dụng **Gaussian Blur** làm mịn ảnh, giúp giảm nhiễu. Với mô hình này sử dụng **blurValue** là 41 (mức độ làm mờ). Bộ lọc **Gaussian** với **kernel 41x41** sẽ làm mờ ảnh nhiều và mất chi tiết nhỏ nhưng hữu ích để giảm nhiễu và làm rõ các đặc trưng lớn.

### **Ngưỡng hóa ảnh (Thresholding):**

ret, thresh = cv2.threshold(blur, threshold, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

### **Ngưỡng hóa nhị phân (Binary Thresholding):**

Nếu giá trị pixel > ngưỡng (threshold), pixel trở thành trắng (255).

Nếu giá trị pixel ≤ ngưỡng, pixel trở thành đen (0).

### **Ngưỡng OTSU (cv2.THRESH\_OTSU):**

**OTSU sẽ** tự động tìm ngưỡng tối ưu để tách nền và đối tượng. **THRESH\_BINARY\_INV** sẽ đảo ngược màu (đối tượng -> trắng, nền -> đen).

## **Nhận xét đánh giá**

### **Kết quả đạt được:**

Ảnh màu ban đầu được chuyển đổi sang ảnh nhị phân (đen trắng) làm nổi bật được vùng quan tâm (bàn tay)

|  |
| --- |
|  |

Hình 7: Ảnh minh họa quá trình chuyển đổi sang ảnh nhị phân.

### **Lợi ích của ảnh nhị phân (đen trắng)**:

Giảm kích thước dữ liệu (từ 50×50×3 xuống 50×50).

Trích xuất được đặc chưng “hình dạng” bàn tay cần thiết cho mô hình

Loại bỏ thông tin màu sắc, các đặc trưng không cần thiết.

**Nhận xét:** Trong đề tài này, việc tối ưu hóa đặc trưng là rất quan trọng. Thay vì học các đặc trưng không cần thiết như màu sắc, độ sáng tối, đường nét hay các góc chụp khác nhau, mô hình chỉ cần tập trung vào đặc trưng "hình dạng" của bàn tay là đủ để dự đoán kết quả mong muốn.

# **Phần VI: Chuẩn hóa dữ liệu**

## **Chuyển đổi ảnh về mảng số numpy**

Vì mô hình sử dụng các thư viện học máy được xây dựng sẵn như **Tensorflow** và **keras**, các thư viện này đều sử dụng dữ liệu đầu vào là mảng **numpy**. Bên cạnh đó việc chuyển đổi dữ liệu về mảng **numpy** sẽ thuận lợi cho quá trình chuẩn hóa ảnh trước khi huấn luyện mô hình cũng như tiết kiệm bộ nhớ và tối ưu việc sử dụng tài nguyên tính toán **GPU**.

def load\_data(data\_dir):  
 images = []  
 labels = []  
 class\_names = os.listdir(data\_dir)for label in class\_names:  
 class\_dir = os.path.join(data\_dir, label)  
 if os.path.isdir(class\_dir):for filename in os.listdir(class\_dir):  
 if filename.endswith('.png')or filename.endswith('.jpg'): img\_path = os.path.join(class\_dir, filename)  
 image = Image.open(img\_path).convert('L')image = image.resize((32, 32))images.append(np.array(image))  
 labels.append(class\_names.index(label))return np.array(images), to\_categorical(np.array(labels), num\_classes=36)

**\*Lưu ý**: ở đây chúng ta sẽ đưa kích thước ảnh từ 50×50 xuống 32×32 vì những lý do sau đây:

**Giảm tải bộ nhớ cho mảng numpy:** Ảnh **50x50 có 2.500 pixel (50 \* 50)**, trong khi ảnh **32x32 chỉ có 1.024 pixel (32 \* 32)**, giảm bớt đáng kể số lượng pixel mà mô hình cần xử lý.

**Tăng tốc độ huấn luyện:** việc tính toán với dữ liệu có kích thước nhỏ hơn sẽ luôn nhanh hơn dữ liệu có kích thước lớn.

**Giữ lại các đặc trưng quan trọng:** Mặc dù việc giảm kích thước ảnh có thể làm mất một số chi tiết nhỏ, nhưng các đặc trưng quan trọng như hình dáng hoặc cấu trúc chính của đối tượng (trong trường hợp này là các ký hiệu tay) thường vẫn được giữ lại trong ảnh **32x32**. Việc giảm kích thước sẽ giúp mô hình tập trung vào những đặc trưng quan trọng hơn, thay vì những chi tiết không cần thiết.

**A screenshot of a computer screen

Description automatically generatedNhật xét:** Nhìn bằng mắt thường chúng ta cũng có thể thấy được rất rõ đặc trưng hình dạng của ảnh thông qua mảng numpy. A hand with a peace sign

Description automatically generated

## **Chuẩn hóa cường độ pixel**

Chia tất cả giá trị pixel cho **255** để đưa dữ liệu về khoảng **[0, 1]**. Để giảm ảnh hưởng của cường độ sáng.

*# Đảm bảo dữ liệu có định dạng đúng*X\_train = X\_train.astype('float32') / 255X\_test = X\_test.astype('float32') / 255

## **Kích thước ảnh đồng nhất**

Như đã nói trước đó ở phần chuyển đổi ảnh về mảng **numpy**, quá trình này sẽ thực thi trên toàn bộ tập dữ liệu nên sẽ đảm bảo tất cả ảnh đều có kích thước **32×32** để đưa vào mô hình.

# **Phần VII: Huấn luyện mô hình**

## **Chia dữ liệu huấn luyện**

**Dữ liệu được chia thành:**

**Tập huấn luyện**: 80% dữ liệu dùng để học.

**Tập kiểm tra**: 20% dữ liệu còn lại để đánh giá mô hình.

Dữ liệu ảnh đã được chuẩn hóa về khoảng giá trị **[0,1][0,1]** khi chuẩn hóa cường độ pixel trước đó bằng cách chia tất cả giá trị pixel cho **255**.

*# Danh sách tên lớp*class\_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9',  
 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',  
 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

*# Chuyển nhãn sang định dạng one-hot*labels\_one\_hot = to\_categorical(labels, num\_classes=num\_classes)  
*# Kết hợp ảnh và nhãn one-hot để lưu*data = np.hstack((images, labels\_one\_hot)) *# Gộp dữ liệu ảnh và nhãn vào cùng một mảng*data = np.array(data, dtype=np.float32)

Khi chuyển đổi nhãn sang dạng **one-hot encoding**, mỗi hình ảnh thuộc một lớp sẽ được biểu diễn bằng một vector nhị phân có số chiều bằng số lớp. Trong đó, chỉ có một phần tử tại vị trí tương ứng với lớp đó được gán giá trị 1, còn lại là 0.

**Ví dụ:**

'0' → [1, 0, 0, ..., 0] (36 chiều, phần tử đầu tiên là 1)

'A' → [0, 0, 0, ..., 0, 1, 0, ..., 0] (36 chiều, phần tử thứ 11 là 1)

'Z' → [0, 0, 0, ..., 0, 1] (36 chiều, phần tử cuối cùng là 1)

## **Kiến trúc mô hình LeNet-5 đã điều chỉnh**

Mô hình được xây dựng dựa trên kiến trúc **LeNet-5**, nhưng được điều chỉnh phù hợp với bài toán nhận diện ngôn ngữ ký hiệu tay:

### **Các lớp tích chập và gộp (Convolution + Pooling):**

**Lớp 1:**

**Tích chập:** 32 bộ lọc, kích thước 3×3, hàm kích hoạt ReLU.

**Average Pooling:** Kích thước 2×2.

**Batch Normalization:** Giúp tăng tốc độ hội tụ.

**Lớp 2:**

**Tích chập:** 64 bộ lọc, kích thước 3×3, hàm kích hoạt ReLU.

**Average Pooling:** Kích thước 2×2.

**Batch Normalization:** Ổn định đầu ra.

### **Lớp chuyển đổi (Flatten):**

Chuyển các đặc trưng hai chiều thành vector một chiều để đưa vào các lớp fully connected.

### **Các lớp kết nối đầy đủ (Fully Connected):**

**Lớp 1:** 128 nơ-ron, hàm kích hoạt ReLU, thêm Dropout (0.5) để giảm overfitting.

**Lớp output:** 36 nơ-ron (tương ứng với số lớp), hàm kích hoạt **softmax** để tính xác suất cho mỗi lớp.

### **Bộ tối ưu hóa và hàm mất mát:**

**Tối ưu hóa:** Adam, learning rate =0.001.

**Hàm mất mát:** Categorical Crossentropy.

### **Đoạn code mô hình sau khi được tối ưu hóa để huấn luyện:**

def train\_model\_lenet\_5(x\_train, y\_train, x\_test, y\_test):  
 model = Sequential([  
 *# Lớp tích chập đầu tiên* Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(32, 32, 3), padding='same'),  
 BatchNormalization(),  
 AveragePooling2D(),  
  
 *# Lớp tích chập thứ hai* Conv2D(64, kernel\_size=(3, 3), activation='relu', padding='same'),  
 BatchNormalization(),  
 AveragePooling2D(),  
  
 *# Chuyển đổi đặc trưng và lớp fully connected* Flatten(),  
 Dense(128, activation='relu'),  
 Dropout(0.5),  
  
 *# Lớp output* Dense(36, activation='softmax')  
 ])  
  
 model.compile(  
 optimizer=Adam(learning\_rate=0.001),  
 loss='categorical\_crossentropy',  
 metrics=['accuracy']  
 )  
  
 *# Callbacks* reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=5, min\_lr=0.00001)  
 early\_stop = EarlyStopping(monitor='val\_loss', patience=5, verbose=1, mode='min')  
  
 *# Huấn luyện mô hình* history = model.fit(  
 x\_train, y\_train,  
 epochs=20,  
 validation\_data=(x\_test, y\_test),  
 batch\_size=32,  
 callbacks=[checkpoint,reduce\_lr, early\_stop]  
 )  
  
 *# Lưu mô hình* model.save('../Scripts/hand\_sign\_recognition\_lenet5.h5')  
 return history

## **Huấn luyện mô hình**

* 1. **Số lượng epoch:**

Tổng cộng đã huấn luyện qua 20 epochs. Đây là một con số vừa phải để đảm bảo mô hình có đủ thời gian học mà không bị overfitting.

* 1. **Batch size:**

Với batch size: 32, mô hình xử lý và cập nhật trọng số sau mỗi 32 mẫu trong tập dữ liệu. Đây là một giá trị phổ biến, giúp cân bằng giữa tốc độ huấn luyện và độ chính xác.

* 1. **Callbacks sử dụng:**

**ModelCheckpoint:** Lưu mô hình tốt nhất dựa trên giá trị val\_loss.

**ReduceLROnPlateau:** Giảm learning rate khi val\_loss không cải thiện.

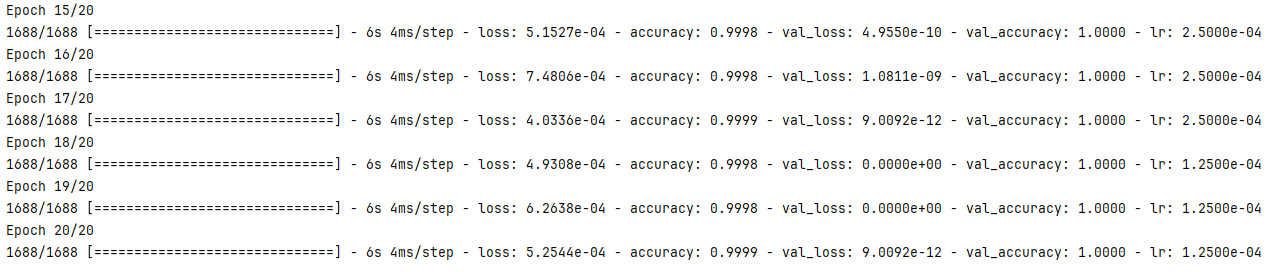
**EarlyStopping:** Dừng sớm nếu mô hình không cải thiện trong 5 epoch liên tiếp.

* 1. **Kết quả huấn luyện:**

Mô hình đạt được:

Độ chính xác trên tập huấn luyện (Training accuracy): **~99.98%.**

Độ chính xác trên tập kiểm tra (Validation accuracy): **~100%.**



Hình 8: Kết quả huấn luyện mô hình.

# **Phần VIII: Đánh giá mô hình**

## **Phép đo được sử dụng** [2]

**Loss:** Đánh giá sự khác biệt giữa dự đoán và thực tế.

**Accuracy:** Đo lường tỷ lệ dự đoán đúng trên tổng số mẫu.

## **Đánh giá kết quả**

A graph of a graph of a graph

Description automatically generated with medium confidence

Hình 9: Biểu đồ kết quả loss và accuracy.

**Loss:**

**Mất mát trong tập huấn luyện (Train loss)** giảm mạnh trong những epoch đầu tiên và gần như đạt mức rất thấp sau vài epoch. Điều này cho thấy mô hình đang học rất nhanh và cải thiện hiệu quả trong việc tối ưu hóa hàm mất mát.

**Mất mát trong tập kiểm tra (Validation loss)** giảm đều và đạt mức thấp ổn định. Mức độ ổn định này không có sự gia tăng rõ rệt sau đó, điều này có thể là dấu hiệu cho thấy mô hình đang tổng quát hóa tốt hơn trên dữ liệu kiểm tra, không bị overfitting như trong các mô hình khác..

**Accuracy:**

**Độ chính xác trong tập huấn luyện (Training accuracy)** đạt gần 1 (100%) rất nhanh sau vài epoch, chứng tỏ mô hình học rất nhanh và đạt hiệu suất cao trên tập huấn luyện.

**Độ chính xác trong tập kiểm tra (Validation accuracy)** cũng đạt gần 100% và ổn định trong suốt quá trình huấn luyện. Điều này cho thấy mô hình không bị overfit và có khả năng tổng quát hóa tốt trên dữ liệu kiểm tra, với độ chính xác cao và ít thay đổi qua các epoch.

**Đánh giá kết quả lý thuyết:** Mô hình đạt độ chính xác rất cao và giảm mất mát nhanh chóng trong quá trình huấn luyện, cả trên tập huấn luyện và kiểm tra.Mô hình tổng quát hóa tốt trên tập kiểm tra, với độ chính xác và mất mát ổn định. Điều này là dấu hiệu của việc huấn luyện tốt và không bị overfitting.

|  |  |
| --- | --- |
| A hand with a middle finger up  Description automatically generated | {  "name": "Predict image from local file",  "input": "l.jpg",  "predict": "L",  "completed": true,  "time": "2024-12-11T14:18:24.962922" } |

**Trường hợp dư đoán đúng (kết quả được lưu thông qua file json)**

|  |  |
| --- | --- |
| A hand making a hand gesture  Description automatically generated | {  "name": "Predict image from local file",  "input": "j.jpg",  "predict": "J",  "completed": true,  "time": "2024-12-11T14:18:10.049756" } |

**Trường hợp dư đoán sai (kết quả được lưu thông qua file json)**

|  |  |
| --- | --- |
|  | {  "name": "Predict image from local file",  "input": "h.jpg",  "predict": "G",  "completed": true,  "time": "2024-12-12T16:18:43.389042" } |

Đối với trường hợp dư đoán sai này chúng ta hãy cũng đánh giá dựa trên ảnh dữ liệu của ký tự **“H”** và **“G”**:

Một vài ảnh của tập **“H”:**

A hand holding a finger

Description automatically generated

Hình 10: 5 Ảnh ngẫu nhiên của nhãn “H”.

Một vài ảnh của tập **“G”:**

A hand pointing at something

Description automatically generated

Hình 11: 5 Ảnh ngẫu nhiên của nhãn “G”.

Chúng ta sẽ thử so sánh chỉ số SSIM của hai ảnh đen trắng “H” và “G” trong tập dữ liệu. Chỉ số Tương Đồng Cấu Trúc (SSIM) là một độ đo được sử dụng để so sánh sự tương đồng giữa hai hình ảnh. Điểm số SSIM dao động từ 0 đến 1, trong đó 1 chỉ ra sự khớp hoàn hảo giữa hai hình ảnh, và 0 chỉ ra không có sự tương đồng.

def compare\_images(imageA, imageB):  
 *# Chuyển ảnh sang grayscale* imageA = cv2.cvtColor(imageA, cv2.COLOR\_BGR2GRAY)  
 imageB = cv2.cvtColor(imageB, cv2.COLOR\_BGR2GRAY)  
   
 *# Tính chỉ số SSIM* score, \_ = ssim(imageA, imageB, full=True)  
 return score  
  
*# Đọc ảnh*image1 = cv2.imread("../Data/Gesture Image Pre-Processed Data/G/33.jpg")  
image2 = cv2.imread("../Data/Gesture Image Pre-Processed Data/H/33.jpg")  
  
*# Kiểm tra độ tương đồng*similarity\_score = compare\_images(image1, image2)  
print(f"SSIM Score: {similarity\_score}")

A white object in the dark

Description automatically generated

Hình 12: File ảnh thứ 33 trong tập “H”

A white pointy object in the middle of a black background

Description automatically generated

Hình 13: File ảnh thứ 33 trong tập “G"

Kết quả thu đươc:

A number on a white background

Description automatically generated

Hình 14: Kết quả SSIM của 2 ảnh "H" và "G" trong tập dữ liệu

**Phân tích:** Dựa trên những bức ảnh ngẫu nhiên được trích xuất từ hai tập dữ liệu chúng ta có thể nhận thấy nhiều điểm tương đồng về mặt hình dạng của hai mẫu này. Không những thế, chỉ số SSIM của hai ảnh đen trắng “H” và “G” trong tập dữ liệu là **khoảng ~0.709**, kết quả này cho thấy rằng mức độ tương đồng về cấu trúc hình dạng của hai mẫu này khá là cao, điều này đã giải thích tại sao kết quả lại có sự sai lệch.

**Đánh giá kết quả thực tế:** Mô hình cho thấy độ chính xác cao khi dự đoán ảnh thực tế. tuy nhiên, vẫn có sự chênh lệch nhỏ giữa ảnh thực tế và kết quả dự đoán chứng tỏ mô hình vẫn chưa vẫn chưa đạt được sự chính xác thuyệt đối. Tính chính xác của mô hình cần được kiểm tra nhiều hơn.

# **Phần IX: Điều chỉnh mô hình**

## **Mô hình LeNet-5 ban đầu:**

**LeNet-5** là một mô hình mạng nơ-ron tích chập (CNN) được thiết kế bởi Yann LeCun và các cộng sự vào năm 1998 [3], chủ yếu để nhận dạng chữ viết tay, đặc biệt là các chữ số viết tay trong cơ sở dữ liệu MNIST. Mô hình này là một trong những mạng CNN đầu tiên và đã góp phần quan trọng vào sự phát triển của học sâu (deep learning).

A group of white lines

Description automatically generated

Hình 15: Kiến trúc mạng LeNet-5.

**Mô hình Lenet-5 nguyên bản**   
def train\_model\_lenet\_5(x\_train, y\_train, x\_test, y\_test):  
 model = Sequential()  
  
 *# Lớp tích chập đầu tiên* model.add(Conv2D(6, kernel\_size=(5, 5), activation='tanh', input\_shape=(32, 32, 1), padding='valid'))  
 model.add(AveragePooling2D(pool\_size=(2, 2), strides=(2, 2)))  
  
 *# Lớp tích chập thứ hai* model.add(Conv2D(16, kernel\_size=(5, 5), activation='tanh', padding='valid'))  
 model.add(AveragePooling2D(pool\_size=(2, 2), strides=(2, 2)))  
  
 *# Lớp Flatten* model.add(Flatten())  
  
 *# Các lớp kết nối đầy đủ* model.add(Dense(120, activation='tanh'))  
 model.add(Dense(84, activation='tanh'))  
 model.add(Dense(10, activation='softmax')) *# Phân loại 10 lớp (MNIST)  
  
 # Biên dịch mô hình* model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  
  
 *# Thêm các callbacks để tăng hiệu quả huấn luyện (tùy chọn)* reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=5, min\_lr=0.00001)  
 early\_stop = EarlyStopping(monitor='val\_loss', patience=5, verbose=1, mode='min')  
  
 *# Huấn luyện mô hình* history = model.fit(  
 x\_train, y\_train,  
 epochs=20,  
 validation\_data=(x\_test, y\_test),  
 batch\_size=32,  
 callbacks=[reduce\_lr, early\_stop]  
 )  
  
 *# Lưu mô hình (tùy chọn)* model.save('../Scripts/original\_lenet5.h5')  
  
 return history

**Input:** Kích thước ảnh đầu vào là **32x32** (sau khi ảnh được padding thêm để đảm bảo kích thước chia hết cho **5x5** trong các lớp **convolution**).

**Output:** Đối với **MNIST**, đầu ra là 10 lớp (tương ứng với các chữ số từ 0 đến 9).

**Công nghệ:** **LeNet-5** sử dụng các kỹ thuật như **convolution**, **pooling**, và **fully** **connected layers**. Nó chủ yếu áp dụng các lớp **convolution** để tự động trích xuất các đặc trưng của ảnh mà không cần phải làm thủ công.

**Đặc điểm nổi bậc:** **LeNet-5** là một mô hình đơn giản nhưng hiệu quả, với cấu trúc khá đơn giản so với các mô hình **CNN** hiện đại, nhưng đã đạt được kết quả ấn tượng trên tập dữ liệu **MNIST** vào thời điểm đó. Các lớp **convolution** và **pooling** trong **LeNet-5** giúp mô hình học các đặc trưng phức tạp từ ảnh mà không cần quá nhiều sự can thiệp của con người. Nhờ khả năng này, **LeNet-5** đã được sử dụng rộng rãi trong nhận dạng chữ viết tay và trở thành một trong những ví dụ đầu tiên về hiệu quả của học sâu trong xử lý ảnh.

**Ứng dụng:** **LeNet-5** được thiết kế đặc biệt cho bài toán nhận dạng chữ viết tay và đã được triển khai trong các hệ thống nhận dạng ký tự quang học (**OCR**). Đây là một trong những mô hình **CNN** đầu tiên có ảnh hưởng lớn đến sự phát triển của các mạng học sâu (**deep learning**) ngày nay. Với khả năng học các đặc trưng phức tạp từ ảnh mà không cần quá nhiều sự can thiệp của con người, **LeNet-5** đã chứng minh hiệu quả vượt trội trong việc xử lý ảnh và nhận dạng chữ viết tay, đặt nền móng cho nhiều tiến bộ trong lĩnh vực này.

**Đối với đề tài nhận diện ngôn ngữ ký hiệu tay:** vì có mối liên hệ với việc nhận diện chữ viết tay và ảnh đầu vào là ảnh đen trắng, mô hình **Lenet-5** thật sự là mô hình thích hợp cho việc huấn luyện và dự đoán ngôn ngữ ký hiệu tay.

## **Quá trình điều chỉnh tạo ra** [**mô hình Lenet-5 mới**](#_Kiến_trúc_mô)**:**

### **Kiến trúc tích chập:**

**Mô hình gốc:**

Sử dụng **kernel size 5x5**

Kích hoạt **`tanh`**

Sử dụng **AveragePooling**

**Mô hình mới:**

**Kernel size 3x3**

Kích hoạt **`relu`**

Thêm **BatchNormalization**

Vẫn sử dụng **AveragePooling**

### **Số lượng lớp:**

**Mô hình gốc:**

2 lớp tích chập, 3 lớp kết nối đầy đủ

**Mô hình mới:**

3 lớp tích chập, thêm lớp Dropout

### **Số lượng nơ-ron và lớp đầu ra:**

**Mô hình gốc:**

**Các lớp ẩn:** 120 → 84 nơ-ron

**Lớp đầu ra:** 10 lớp (MNIST)

**Mô hình mới:**

**Lớp ẩn:** 128 nơ-ron

**Lớp đầu ra:** 36 lớp

Thêm Dropout để giảm overfitting

### **Tối ưu hóa:**

**Mô hình gốc:**

Optimizer cơ bản

**Mô hình mới:**

Adam với learning rate điều chỉnh

Kỹ thuật giảm learning rate (**ReduceLROnPlateau**)

Early stopping

## **Nhận xét đánh giá:**

### **Mục đích:**

Mô hình mới được thiết kế để nhận dạng nhiều lớp hơn (36 so với 10), mở rộng thêm lớp tích chấp, tăng số lượng nơron để học được nhiều đặc trưng hơn. Giảm thiểu overfitting thông qua **Dropout** và **BatchNormalization**, tiết kiệm thời gian huấn luyện bằng việc thêm **Early stopping**(dừng huấn luyện khi kết quả không cải thiện sau **5 epochs**)**.**

### **Các cải tiến:**

Sử dụng **ReLU** [4] thay cho **Tanh** [5], vì **ReLU** thường mang lại hiệu quả tốt hơn trong thực tế. Việc giảm kích thước **kernel** không chỉ giúp giảm số lượng tham số mà còn làm mô hình trở nên đơn giản hơn. Thêm **BatchNormalization** [6] vào mô hình sẽ giúp ổn định quá trình huấn luyện, đảm bảo mọi thứ diễn ra mượt mà. Cuối cùng, điều chỉnh **learning rate** động sẽ giúp mô hình thích nghi tốt hơn với các thay đổi trong quá trình học.

### **Hạn chế tiềm năng:**

Số lượng tầng ít hơn so với các kiến trúc CNN hiện đại. chúng ta có thể sẽ cần thêm nhiều lớp để xử lý các nhiệm vụ phức tạp. Nhưng đối với đề tài này, các đặc trưng không quá phức tạp nên chúng ta cũng không cần quá lo lắng về đề này.

# **Phần X: Kết luận**

## **Nhận xét đánh giá:**

**Ưu điểm:**

**Dữ liệu:** tập dữ liệu dễ dàng có được thông qua trang web kaggle, không phải mất công thu thập. Dữ liệu đã được tác giá tiền xử lý các bước cơ bản như xử lý dữ liệu thiếu, dữ liệu ngoại lệ. Việc rút trích đặc trưng diễn ra khá là thuận lợi, kết quả rút trích gần như không có sai sót, đặc trưng đơn giản và dễ hiểu không quá phức tạp với mô hình học máy.

**Mô hình LeNet-5:** mô hình đã được điều chỉnh để tối ưu hóa cho bài toán nhận diện ngôn ngữ ký hiệu tay, mang lại hiệu suất chính xác cao. Với cấu trúc đơn giản **LeNet-5** không chỉ dễ tiếp cận mà còn dễ dàng mở rộng. Đặc biệt, tập dữ liệu hình ảnh phong phú và đa dạng về hình dạng bàn tay, kết hợp với chất lượng ảnh ổn định, đã tạo điều kiện thuận lợi cho việc trích xuất đặc trưng, từ đó nâng cao hiệu quả của mô hình, giảm thiểu sai sót trong quá trình huấn luyện, đảm bảo độ tin cậy cao cho các kết quả cuối cùng.

**Nhược điểm:**

**Dữ liệu:** vì đặc trưng hình dạng không quá phức tạp nên khi xử lý dữ liệu sẽ có một vài mẫu có hình dạng tương đồng nhau. Điều này làm cho mô hình có sự nhầm lẫn trong kết quả dự đoán. Có thể nói mô hình bị nhiễu thông tin từ tập dữ liệu cũng có thể nói mô hình học chưa tốt.

**Mô hình LeNet-5:** Mô hình hiện tại chưa đạt được độ chính xác mong muốn khi kết quả huấn luyện và kết quả thực tế còn chênh lệch. Do chỉ xử lý ảnh đen trắng, mô hình dễ bị nhầm lẫn giữa các mẫu có đặc trưng gần tương đồng nhau. Điều này cho thấy mô hình có xu hướng quá phụ thuộc vào các đặc trưng cụ thể mà nó học được, dẫn đến giảm khả năng tổng quát hóa.

**Hướng phát triển:**

Thử nghiệm các kiến trúc hiện đại hơn như VGG [7] hoặc ResNet [8].

Tăng cường dữ liệu (data augmentation) để cải thiện độ chính xác, tìm kiếm các đặc trưng khác để phân biệt giữa các mẫu mà không phụ thuộc quá nhiều vào đặc trưng hình dạng.

Phát triển mô hình thành các biến thể nhận dạng ký hiệu tay để ứng dụng cho các mô hình smart home, hỗ trợ thuyết trình, hỗ trợ làm việc từ xa v.v…

## **Tài liệu tham khảo**

|  |  |
| --- | --- |
| [1] | Khan Ahmed, "Kaggle," 2019. [Online]. Available: https://www.kaggle.com/datasets/ahmedkhanak1995/sign-language-gesture-images-dataset . |
| [2] | Nguyễn Chiến thắng, "Mì AI," 12 6 2020. [Online]. Available: https://www.miai.vn/2020/06/12/oanh-gia-model-ai-theo-cach-mi-an-lien-chuong-1-loss-va-accuracy/ . |
| [3] | Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* 1998. |
| [4] | Xavier Glorot, Antoine Bordes, Yoshua Bengio, "Deep Sparse Rectifier Neural Networks," *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics,* 2011. |
| [5] | Bishop, Christopher M., Pattern Recognition and Machine Learning, 2006. |
| [6] | Sergey Ioffe, Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Machine Learning (cs.LG),* 2015. |
| [7] | Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (cs.CV),* 2014. |
| [8] | Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2015. |