**UBND TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

A blue circle with text

Description automatically generated

**BÁO CÁO CUỐI KỲ**

**MÔN:**

NHẬP MÔN MÁY HỌC

**ĐỀ TÀI:**

XÂY DỰNG MÔ HÌNH NHẬN DIỆN

NGÔN NGỮ KÝ HIỆU TAY

**BẬC:** ĐẠI HỌC

**HÌNH THỨC:** CHÍNH QUY

**SINH VIÊN THỰC HIỆN:**

TRẦN DUY HOÀNH

MSSV: 3122410132

TRẦN NGÔ NHẬT NAM

MSSV: 3122410253

**GIẢNG VIÊN HƯỚNG DẪN:** ThS. Nguyễn Thanh Phước

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2024**

**Mục lục**

[**1** **Giới Thiệu** 3](#_Toc184817746)

[**2** **Nội dung** 3](#_Toc184817747)

[**2.1** **Lựa chọn dữ liệu** 3](#_Toc184817748)

[**2.2** **Tiền xử lý dữ liệu** 3](#_Toc184817749)

[2.2.1 Giới thiệu bộ dữ liệu 3](#_Toc184817750)

[2.2.2 Mô tả đặc trưng 3](#_Toc184817751)

[2.2.3 Xử lý nền 3](#_Toc184817752)

[2.2.4 Xử lý tay 3](#_Toc184817753)

[**2.3** **Huấn luyện mô hình** 3](#_Toc184817754)

[2.3.1 Mô hình LenNet -5 3](#_Toc184817755)

[2.3.2 Mô hình resnet-5 5](#_Toc184817756)

[2.3.3 Mô hình CNN. 7](#_Toc184817757)

[**2.4** **Kết quả thực tế** 9](#_Toc184817758)

[**3** **Kết luận** 9](#_Toc184817759)

# **Giới Thiệu**

Ngôn ngữ thông thường của chúng ta dùng chữ viết, âm thanh để biểu thị thông tin. Trong khi ngôn ngữ kí hiệu tay lại dùng tay như một công cụ biểu thị thông tin giúp cho họ có thể trao đổi thông tin. Trong quá trình học phân tích nhận dạng mẫu nhóm em đã biết được tiền xử lý dữ liệu kết với học học sâu thì nhóm em đã xây dựng được mô hình nhận dạng được 24 chữ cái từ A - Z và 10 kí hiệu số từ 0-9.

# **Nội dung**

## **Lựa chọn dữ liệu**

Trong quá trình thực hiện dự án. Nhóm em có rất nhiều nguồn tham khảo tài liệu chẳng hạn như Kaggle, Github, và Mendeley Data. Và có rất nhiều nguồn data đáng tin cậy như là file scv gồm 748 đặc trưng. Mỗi đặc trưng chứa các số từ 1 tới 255 thể hiện độ sáng của điểm ảnh. File ảnh Trắng đen gồm 36 foldel chứa các ảnh thể hiện ý nghĩa của chữ cái đó. Với nên đen và ảnh bà tay màu trắng. File ảnh xám cũng như tương tự ảnh trắng đen nhưng màu sác là xám thể hiện độ sáng tại mỗi điểm…. Còn rất nhiều dữ liệu khá hay. Nhưng nhóm em đã quyết định lữa chọn ảnh thô. Bởi mục tiêu dự án lần này là nhóm em sẽ xây dựng mô hình từ điểm đầu.

## **Tiền xử lý dữ liệu**

### **Giới thiệu bộ dữ liệu**

Dữ liệu gồm 36 folder, trong mỗi folder chứa 1500 tấm ảnh thể hiện ý nghĩa chữ cái đó (Hình 2). Được đặt tên từ 1 – 1500.Tên file của mỗi ảnh là file .jpg. Kích cở của dữ liệu là 99 MB (hình 1).

A group of white and yellow boxes

Description automatically generated

Hình 1: Các folder chứa ảnh.

A screenshot of a computer screen

Description automatically generated

Hình 2: Ảnh trong folder “0”

### **Mô tả đặc trưng**

Khi làm việc với dữ liệu là hình ảnh trong bài toán nhận diện ngôn ngữ ký hiệu tay, các đặc trưng quan trọng cần được trích xuất từ tấm ảnh bao gồm:

1. **Hình dạng bàn tay**

Đường viền và khung bàn tay: Các đường viền của bàn tay thể hiện hình dạng tổng quát và cấu trúc của cử chỉ tay.

Kích thước và tỷ lệ: Phân tích kích thước bàn tay so với toàn bộ ảnh hoặc so với các ngón tay.

1. **Tư thế và vị trí ngón tay**

Góc độ của ngón tay: Xác định góc mở của từng ngón tay để phân biệt các ký hiệu khác nhau.

Vị trí tương đối: Vị trí giữa các ngón tay hoặc giữa bàn tay với các ngón tạo nên đặc trưng độc nhất của ký hiệu.

1. **Kết cấu và màu sắc**

Màu da và sắc độ: Có thể dùng để phân biệt bàn tay với nền ảnh hoặc các đối tượng khác trong khung hình.

Chi tiết bề mặt: Đặc trưng như nếp nhăn, bóng đổ, hoặc đường nét có thể hữu ích trong phân loại nâng cao.

1. **Tọa độ không gian**

Vị trí của bàn tay trong khung hình: Xác định tọa độ bàn tay để theo dõi và nhận diện chính xác.

Chiều sâu (nếu có ảnh 3D): Tính năng này hữu ích khi nhận diện chuyển động hoặc các ký hiệu phức tạp.

1. **Bối cảnh**

**Nền ảnh:** Loại bỏ nhiễu từ môi trường xung quanh để tập trung vào bàn tay.

**Đối tượng khác trong ảnh:** Tránh nhầm lẫn giữa tay và các vật thể khác trong khung hình.

Trong các đặc trưng này chúng ta không thể dung hết và cần loại bỏ bớt.Để giảm độ phức tạp và hiệu quả bài toán. Vì bài toán mục đích là nhận diện diện kí cử chỉ tay và đoán đó là chữ cái nào hoặc số nào thì tôi nghĩ cần loại bỏ một số thuộc tính.

**Những thuộc tính cần loại bỏ:**

* **Kết cấu và màu sắc**

Các thông tin về màu da, sắc độ và màu nền bị mất hoàn toàn khi ảnh được chuyển sang dạng trắng đen.

Không thể phân biệt dựa trên màu sắc giữa bàn tay và các đối tượng khác (nếu có).

* **Tọa độ không gian**

Các chi tiết nhỏ như nếp nhăn trên tay hoặc sắc thái của bóng đổ (dựa vào màu) có thể bị giảm độ rõ ràng.

### **Xử lý nền**

Xử lý nền rất quan trọng để quá trình học trở nên hiệu quả và chính xác. Một bức ảnh có rất nhiều đặc trưng mà nó không cần thiết cho quá trình huấn luyện. Cần được loại bỏ để quá trình học trở nên hiệu quả và chính xác. Thậm chí, nó còn giúp cải thiện bộ nhớ, tốc độ học.

def remove\_background(image):gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
blurred = cv2.GaussianBlur(gray, (5, 5), 0)  
  
\_, mask = cv2.threshold(blurred, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  
mask\_inv = cv2.bitwise\_not(mask)  
background\_removed = cv2.bitwise\_and(image, image, mask=mask\_inv)  
  
 return background\_removed

Hàm này có tham số đầu vào là một ảnh. Sau đó sẽ chuyển ảnh đó thành ảnh xám. Tại vì ảnh thô là những tấm ảnh màu. Sau đó sẽ áp dụng GaussianBlur để làm mịn ảnh. Áp dụng phân ngưỡng nhị phân (binary thresholding) để tạo mặt nạ nhị phân (mask) từ ảnh đã làm mịn. Sử dụng phương pháp Otsu để tự động tìm ngưỡng tốt nhất phân biệt giữa nền (background) và đối tượng (foreground). Tạo mặt nạ ngược (mask\_inv) từ mặt nạ ban đầu. Áp dụng mặt nạ ngược (mask\_inv) lên ảnh gốc (image) để giữ lại vùng đối tượng (foreground).Nền (khu vực có giá trị pixel 0 trong mask\_inv) sẽ được thay bằng màu đen.

Ảnh mới (background\_removed) giữ lại đối tượng và xóa nền (thành màu đen).

Kết quả trả về sẽ là 1 tấm ảnh nền đen với 1 bàn tay được giữ nguyên.



Hình 3: Hình “kí hiệu 2” trước và sau khi xóa nền

### **Xử lý tay**

Sau khi xử lý nền. Tấm hình giờ đây có mỗi bàn tay có nhiều kênh màu.Cần chuyển đổi nó thành màu trắng.

def prepare\_image(image):  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 blur = cv2.GaussianBlur(gray, (blurValue, blurValue), 0)ret, thresh = cv2.threshold(blur, threshold, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)  
 return ret, thresh

Chuyển ảnh màu (BGR) thành ảnh thang độ xám để đơn giản hóa dữ liệu đầu vào. Dòng đầu sẽ cho ra ảnh thang độ xám (gray), trong đó mỗi pixel có giá trị từ 0 (đen) đến 255 (trắng). Áp dụng bộ lọc Gaussian để làm mịn ảnh, giảm nhiễu, và làm mềm các cạnh sắc nét. Tham số blurValue là kích thước kernel (ma trận Gaussian). Nó phải là một số lẻ.

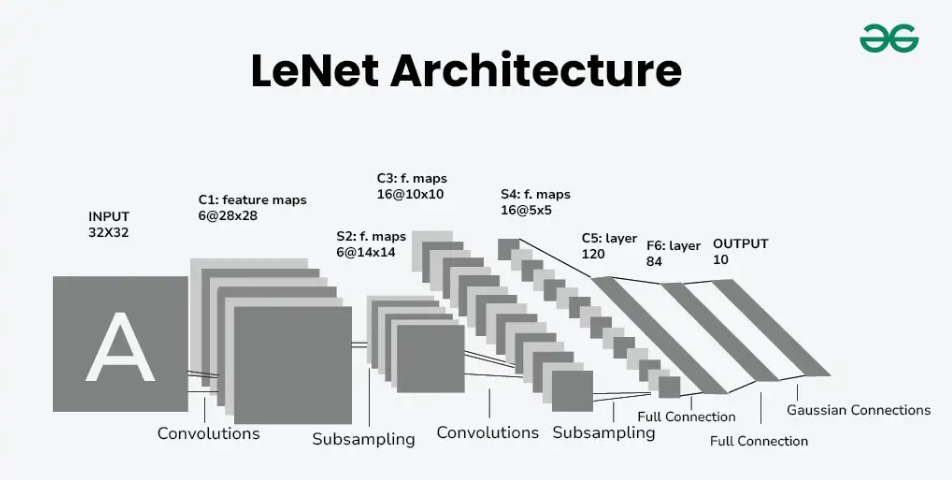
Hàm này sẽ trả về ret và thresh. ret là giá trị ngưỡng tối ưu mà Otsu tính được. thresh là ảnh nhị phân chứa bàn tay màu trắng còn trên nền đen



Hình 4: Hình “kí hiệu 2” trước và sau khi đổi màu tay.

## **Huấn luyện mô hình**

### **Mô hình LenNet -5**

Mô hình được phát triển bởi Yann LeCunn (Director of AI Research Facebook) cùng với Léon Bottou, Yoshua Bengio (đồng tác giả với Ian Goodfellow của cuốn “Deep Learning”) và Patrick Haffner. LeNet-5 là kiến ​​trúc mạng nơ-ron tích chập (CNN) giới thiệu một số tính năng và cải tiến chính đã trở thành tiêu chuẩn trong học sâu hiện đại. Nó chứng minh hiệu quả của CNN đối với các tác vụ nhận dạng hình ảnh và giới thiệu các khái niệm chính như tích chập, gộp và trích xuất tính năng phân cấp làm nền tảng cho các mô hình học sâu hiện đại. Ban đầu thiết kế để nhận diện chữ số viết tay. Sau này nó áp dung rất nhiều trong việc nhận diện ảnh 2D. Và chính vì thế nó đã có mặt trong đồ án này.

Hình 5: Kiến trúc LeNet-5. Nguồn GeeksForGeeks

**Input:** Dữ liệu đầu vào (ảnh) được đưa qua các tầng tích chập và giảm kích thước để trích xuất các đặc trưng quan trọng.

Feature Extraction: Các tầng convolution và subsampling giúp trích xuất các đặc trưng từ đơn giản (như cạnh) đến phức tạp hơn (như hình dạng).

**Classification:** Các tầng fully connected học cách kết hợp các đặc trưng và dự đoán lớp tương ứng.

**Xây dựng mô hình:**

def train\_model\_lenet\_5(x\_train, y\_train, x\_test, y\_test):  
 model = Sequential()  
  
 *# Lớp tích chập đầu tiên* model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(32, 32, 3), padding='same')) *# 3 kênh màu* model.add(BatchNormalization())  
 model.add(AveragePooling2D())  
  
 *# Lớp tích chập thứ hai* model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu', padding='same'))  
 model.add(BatchNormalization())  
 model.add(AveragePooling2D())  
  
 *# Lớp tích chập thứ ba  
  
 # Chuyển đổi đặc trưng thành vector* model.add(Flatten())model.add(Dense(128, activation='relu')) *# Tăng số lượng nơ-ron* model.add(Dropout(0.5)) *# Thêm lớp Dropout để giảm overfitting  
  
 # Lớp output với số lớp tương ứng* model.add(Dense(36, activation='softmax')) *# 36 lớp  
  
 # Biên dịch mô hình* model.compile(optimizer=Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])  
  
 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

A comparison of a graph

Description automatically generatedHình 6: Biểu đồ loss qua mỗi lần học (trái) và độ chính xác qua mỗi lần học (phải)

Hai Biểu đồ loss có độ lỗi qua mỗi lần học đều giảm qua từng lần học cho thấy môi hình học khá tốt. Khji đến epochs 10 thì mô hình đã hội tụ. Gần như không thể phân biết hai đường cam và xanh dương. Tương tự biểu đồ độ chính xác khi đến epochs 10 thì gần như có độ chính xác tuyết đối. Cho thấy mô hình học khá tốt trên tập test. Chọn mô hình này làm mô hình chính.

### **Mô hình resnet-5**

ResNet (Residual Network) là một trong những kiến trúc mạng nơ-ron sâu (Deep Neural Network) nổi tiếng, được giới thiệu vào năm 2015 bởi nhóm nghiên cứu của Microsoft Research trong bài báo khoa học “Deep Residual Learning for Image Recognition”. ResNet đã giành chiến thắng trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2015, đạt hiệu suất vượt trội trong các bài toán phân loại hình ảnh và nhận diện đối tượng. Mô hình này củng cực kì mạnh mẻ trong việc nhận dạng ảnh 2D. Trong mô hình náy có nhiều loại biến thể chẳng hạn ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-15...Với tên là ResNet theo sau là một số chỉ kiến trúc ResNet với số lớp nhất định. Trong số này sẽ chọn ResNer-50 làm mô hình cho bài toán này.

A diagram of a block diagram

Description automatically generated

Hình 7: cấu trúc mô hình ResNet-50

Tóm tắt cấu trúc:

1. **Nhận input.**
2. **Các giai đoạn residual:**
3. ResNet-50 chia thành 4 giai đoạn, mỗi giai đoạn bao gồm các Residual Blocks.
4. Mỗi block có ba lớp:

* 1x1 Convolution để giảm chiều kênh (compression).
* 3x3 Convolution để trích xuất đặc trưng.
* 1x1 Convolution để phục hồi chiều kênh ban đầu.

1. Các giai đoạn này bao gồm các số lượng block khác nhau, ví dụ, giai đoạn đầu có 3 block, giai đoạn tiếp theo có 4 block, v.v.
2. **Residual Learning:**

Sử dụng skip connections, giúp tín hiệu đầu vào 𝑥 trực tiếp cộng vào đầu ra của mạng, khắc phục vấn đề gradient biến mất.

1. **Output:**

Global Average Pooling (GAP) tóm tắt các đặc trưng từ các khối tích chập.

Fully Connected Layer có 1000 lớp đầu ra (thường sử dụng trong bài toán nhận diện ảnh ImageNet).

**Xây dựng mô hình:**

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(32, 32, 3))  
x = base\_model.output  
x = GlobalAveragePooling2D()(x) *# Tạo đầu ra 1 chiều từ đầu ra 2 chiều của ResNet50*x = Dense(1024, activation='relu')(x) *# Lớp FC đầu tiên*predictions = Dense(36, activation='softmax')(x) *# Lớp dự đoán (36 lớp)*model = Model(inputs=base\_model.input, outputs=predictions)for layer in base\_model.layers:  
 layer.trainable = False  
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  
early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)  
history = model.fit(  
 X\_train,  
 Y\_train,  
 epochs=30,  
 validation\_data=(X\_test, Y\_test),  
 callbacks=[early\_stopping]  
)  
model.save('../Scripts/hand\_sign\_recognition\_ResNet-50.h5')

A graph of different types of data

Description automatically generated with medium confidence

Hình 8: Biểu đồ loss qua mỗi lần học (trái) và độ chính xác qua mỗi lần học (phải)

**1. Biểu đồ Loss**

Loss Train và Loss Validation đều giảm mạnh ở giai đoạn đầu, cho thấy mô hình học tốt từ dữ liệu ban đầu.

Đường train cho thấy mô hình này học khá tốt và có dấu hiệu hội tụ. Nhưng đường validation lại cho thấy mô hình đoán toàn sai.

**2. Biểu đồ Accuracy**

Tương tự thì mô hình có biểu đồ chính xác như biểu đồ Loss.

Đánh giá tổng quan: Tuy mô hình đã hội tụ nhưng lại sai quá và có độ chính xác thấp trên validation. Dẫn đến mô hình này không đạt.

### **Mô hình CNN.**

Bản chất thì cấu trúc lenet-5 được xây dựng dựa trên mô hình CNN. Cho nên cấu trúc hai cái này tương tự nhau. Chỉ khác số lượng lớp và số lượng kênh.

Xây dựng mô hình:

model = Sequential([  
 Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)),  
 MaxPooling2D(2, 2),  
 Conv2D(64, (3, 3), activation='relu'),  
 MaxPooling2D(2, 2),  
 Flatten(),  
 Dense(128, activation='relu'),  
 Dense(36, activation='softmax')  
])  
  
  
def train():  
 *# Thêm EarlyStopping với patience = 5 (tạm dừng sau 5 epoch nếu độ lỗi không cải thiện)* early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)  
 model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  
 history = model.fit(  
 X\_train,  
 Y\_train,  
 epochs=20,  
 validation\_data=(X\_test, Y\_test),  
 batch\_size=32,  
 callbacks=[early\_stopping] *# Thêm early stopping vào callback* )  
  
 *# Lưu model sau khi huấn luyện xong* model.save('../Scripts/hand\_sign\_recognition\_cnns.h5')

A graph of a graph of a graph

Description automatically generated with medium confidence

Hình 9: Biểu đồ loss qua mỗi lần học (trái) và độ chính xác qua mỗi lần học (phải)

**Biểu đồ Mất mát (Loss):**

Loss (Train) (màu xanh dương) bắt đầu với giá trị rất cao ở epoch đầu tiên, sau đó giảm mạnh xuống mức rất thấp chỉ trong vài epoch đầu. Điều này có thể chỉ ra rằng mô hình đã học rất nhanh và cải thiện hiệu suất ngay từ những bước đầu tiên.

Loss (Validation) (màu cam) có xu hướng dao động và tăng nhẹ ở những epoch sau. Điều này có thể cho thấy rằng mặc dù mô hình đang học tốt trên dữ liệu huấn luyện, nhưng có thể gặp một số vấn đề với việc tổng quát hóa trên dữ liệu kiểm tra (validation), đặc biệt là sau epoch

**Biểu đồ Độ chính xác (Accuracy):**

Accuracy (Train) bắt đầu từ một mức độ chính xác khá thấp và nhanh chóng tăng lên gần 1.0 (100%) trong vài epoch đầu, cho thấy mô hình học tốt trên dữ liệu huấn luyện.

Accuracy (Validation) có sự giảm sút đột ngột trong epoch 12, nhưng sau đó lại cải thiện và có xu hướng ổn định. Tuy nhiên, việc dao động mạnh trong accuracy validation có thể chỉ ra rằng mô hình đang overfitting (quá khớp) với dữ liệu huấn luyện, tức là mô hình học quá kỹ vào dữ liệu huấn luyện nhưng lại không tổng quát tốt trên dữ liệu kiểm tra.

**Đánh giá:**

Overfitting: Mô hình có dấu hiệu overfitting, khi độ chính xác trên dữ liệu huấn luyện đạt gần 100%, nhưng độ chính xác trên dữ liệu validation có sự dao động lớn, đặc biệt là sau epoch 12. Điều này chỉ ra rằng mô hình có thể học quá nhiều từ đặc điểm cụ thể của dữ liệu huấn luyện mà không tổng quát tốt trên dữ liệu mới.

Mô hình này học không tốt.

## **Kết quả thực tế**

|  |  |
| --- | --- |
| 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": "m.jpg",  "predict": "C",  "completed": true,  "time": "2024-12-11T14:20:10.359847" } |

|  |  |
| --- | --- |
|  | {  "name": "Predict image from local file",  "input": "v.jpg",  "predict": "C",  "completed": true,  "time": "2024-12-11T14:15:04.339381" } |

# **Kết luận**

Mặc dù trên lý thuyết quá trình huấn luyện đã cho ra kết quả dự đoán khá tốt nhưng thực tế các trường hợp sai xót vẫn còn nhiều, đây có thể là dấu hiệu của overfitting.

Bên cạnh đó trong quá trình thực hiện đề tài, nhóm em nhận thấy rằng mặc dù đây là một đề tài khá hay nhưng tính ứng dụng thực tế chưa được cao lắm, nguyên nhân là vì vẫn chưa có nhiều tình huống cụ thể để sử dụng mô hình.