BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ KỸ THUẬT
KHOA CƠ KHÍ CHẾ TẠO MÁY
MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO

HCMUTE

BÁO CÁO DỰ ÁN

NHẬN DIỆN NGÔN NGỮ KÝ HIỆU VIỆT NAM

ÁP DỤNG DEEP LEARNNING

MÃ MÔN HỌC: INAI226229
HỌC KỲ 1- ĐỢT 2 – NĂM HỌC 2025 – 2026
Thực hiện:

1. Nguyễn Văn An – 24134002
2. Phạm Nguyễn Trường Thịnh – 24134069
3. Nguyễn Khắc Minh Khôi – 24134035
4. Trần Văn Thư – 24134071

Giảng viên hướng dẫn: TS. Bùi Hà Đức

Thành phố Hồ Chí Minh, Tháng 1 năm 2026

MỤC LỤC

PHẦN 1. GIỚI THIỆU

1.1. Giới thiệu bối cảnh đề tài

Giao tiếp là nhu cầu thiết yếu, nhưng rào cản ngôn ngữ giữa người khiếm thính và cộng đồng vẫn là một thách thức lớn. Đề tài này được nhóm giải quyết bằng cách xây dựng một hệ thống thị giác máy tính sử dụng Deep Learning để nhận diện bảng chữ cái ngôn ngữ ký hiệu Việt Nam theo thời gian thực.

1.2 Mục tiêu hệ thống

Để giải quyết được bài toán này nhóm đã chia ra thành những bài toán nhỏ cụ thể cần giải quyết bao gồm

- Thu thập và xây dựng bộ dữ liệu đa dạng: Thu thập và làm giàu bộ dữ liệu ảnh tay cho bảng chữ cái tiếng Việt.
- Làm sạch bộ dữ liệu và tăng cường độ đa dạng cho dữ liệu: Áp dụng các công nghệ có sẵn để thực hiện tiền xử lý dữ liệu. Cũng như sử dụng các phương pháp tang cường dữ liệu (Data Augmention) để nâng cao chất lượng đầu vào cho mô hình.
- Huấn luyện mô hình: Tinh chỉnh kiến trúc mạng ResNet18 để tăng độ chính xác cho việc nhận diện ký hiệu bàn tay
- Triển khai: Xây dựng bản demo có khả năng nhận diện trực tiếp thông qua webcam

1.3 Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu :

+ Các cử chỉ tay biểu diễn các chữ cái trong bảng chữ cái tiếng Việt

+ Các thuật toán phát hiện bàn tày và cấu trúc mạng nơ ron tích chập CNN

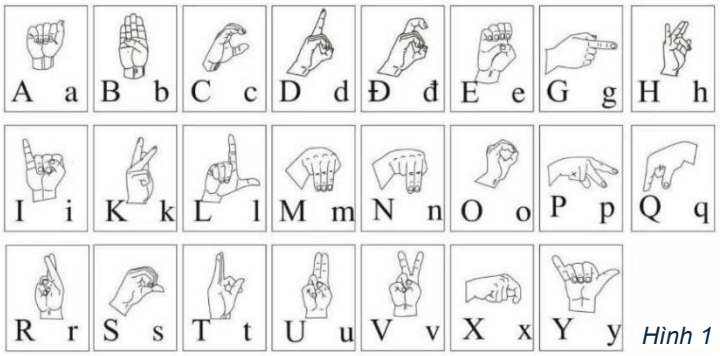
- Phạm vi nghiên cứu : Do giới hạn về thời gian cũng như kỹ thuật, nhóm giới hạn phạm vi nghiên cứu như sau :

+ Bộ ký tự: hệ thống nhận diện 22 chữ cái trong bảng chữ cái tiếng Việt, bao gồm: A, B, C, D, E, G, H, I, K, L, M, N, O, P, Q, R, S, T, U, V, X, Y

+ Dữ liệu: Sử dụng bộ dữ liệu được thu thập từ các thành viên trong nhóm cùng với sự trao đổi dữ liệu giữa các nhóm trong lớp với nhau.

+ Công nghệ: Python, PyTorch (ResNet18), OpenCV, MediPipe

Bảng chữ cái bằng bàn tay



Hình 1

1.4 Thách thức bài toán

Trong quá trình xây dựng hệ thống, nhóm gặp phải một số thách thức đặc thù mà bài toán nhận diện ngôn ngữ ký hiệu đặt ra:

+ Một số ký tự có độ tương đồng cao như chữ B và E, A và M, M và N.

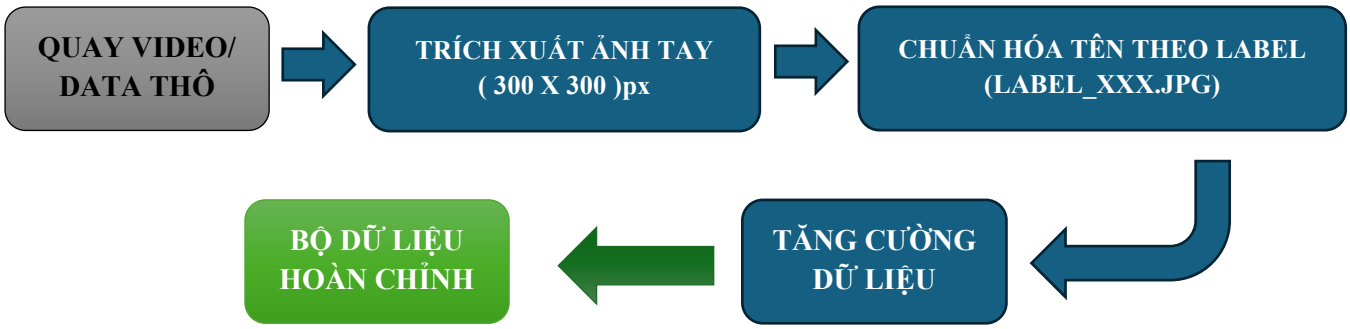
+ Việc nhận diện bàn tay và đưa ảnh về cùng kích thước đã dẫn đến việc một số ảnh bị bể, nén khiến dữ liệu bị nhiễu.

+ Dữ liệu còn hạn chế

PHẦN 2. PHƯƠNG PHÁP THỰC HIỆN

2.1. Xây dựng dữ liệu

Xây dựng dữ liệu chính là yếu tố quyết định hiệu suất của một mô hình Deep Learnin. Nhóm thực hiện thông qua 3 giai đoạn: Trích xuất -> chuẩn hóa -> tăng cường.



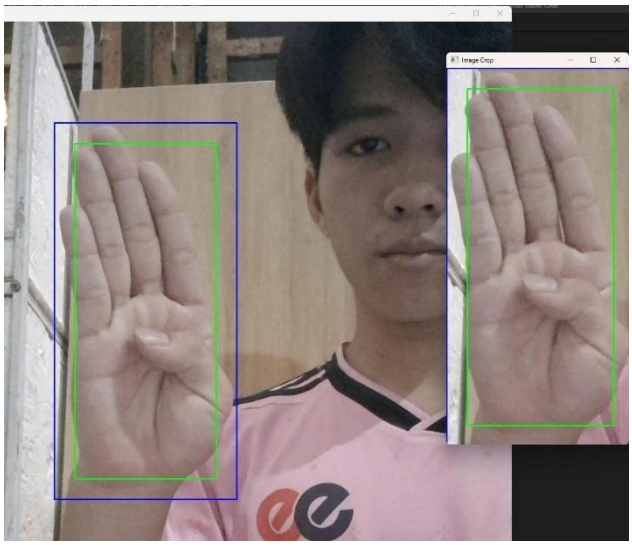
2.1.1 Trích xuất ảnh tay từ video

Các thành viên trong nhóm thực hiện quay video cho các ký tự với nhiều góc độ khác nhau của từng ký tự và mỗi thành viên sẽ là 1 kiểu tay khác nhau giúp mô hình được huấn luyện qua nhiều kiểu tay khác nhau (tăng độ chính xác). Tuy nhiên nếu sử dụng ảnh gốc sẽ dẫn đến việc nhiều background gây nhiễu dữ liệu. Do vậy cần loại bỏ và tập trung vào đối tượng chính là bàn tay.

Nhóm lựa chọn sử dụng thư viện cvzone.HandTrackingModule để phát hiện bàn tay từ các frames tiếp đó nhận được các thông tin về tọa độ mà thư viện trả về bao gồm chiều cao bắt đầu, chiều cao kết thúc, chiều rộng bắt đầu và chiều rộng kết thúc được đặt là các biến x, y, h, w. Với chiều cao sẽ là từ dòng y:y+h và chiều rộng sẽ là x:x+h.

Vị trí	Góc trên trái	Góc trên phải	Góc dưới trái	Góc dưới phải
Tọa độ	(x, y)	(x+w, y)	(x, y+w)	(x+w,y+h)

2.1.2 Chuẩn hóa ảnh giữ tỉ lệ



Hình 2

Với tọa độ ban đầu mà thư viện cung cấp ta sẽ có được box ảnh bàn tay như box màu xanh là ở hình 2. Tuy nhiên sẽ cắt rất sát bàn tay cho nên cần phải thay đổi kích thước cho phù hợp. Nhóm sử dụng biến tên là safe = 40 pixel để tăng vùng an toàn để đảm bảo bắt trọn vẹn được bàn tay. Cụ thể

$y_1 = y - safe, \quad y_2 = y + w + safe$

$x_1 = y - safe, \quad x_2 = x + w + safe$

để có được một box màu xanh như hình 2.

Nếu từ hình ảnh này trực tiếp đưa về kích thước 300x300px sẽ gây ra hiện tượng ảnh bị bể và nén ngoài (hình 3) ý muốn khiến viện huấn luyện mô hình gặp nhiều có khăn. Để giải quyết vấn đề này sau vài lần gặp khó khăn thì nhóm em lựa chọn thực hiện xây dựng một nền cố định là 300x300px.

Hình 3 Ảnh resize trực tiếp

Hình 4 Có nền trắng

*imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255*

Tính toán tỉ lệ giữa chiều cao và chiều rộng của ảnh đã được cắt. Nếu chiều cao > chiều rộng giữ nguyên chiều cao đặt chiều cao bằng 300px và chiều rộng được tính lại theo tỉ lệ tương ứng. Ảnh sau đó sẽ được dán vào nền trắng và căn giữa. Ngược lại nếu chiều rông > chiều cao thì sẽ đặt chiều rộng bằng 300px và chiều cao được tính lại. Ảnh được căn giữa theo chiều cao. Sau đó có thể lưu ảnh vào thư mục data_raw.

2.1.3 Tăng cường dữ liệu

Sau khi thu được dữ liệu anh thô nhóm thực hiện chuẩn hóa tên file đồng nhất dạng <Label_xxx.jpg> với Label là A -> Z. Tiếp theo sẽ đươc qua bước tăng cường dữ liệu nhằm tránh tình trạng overfitting khi train.

Nhóm dùng đoạn script tự động (Clean_data.py) để tạo ra các biến thể từ ảnh gốc thông qua các phép biến đổi một cách ngẫu nhiên.

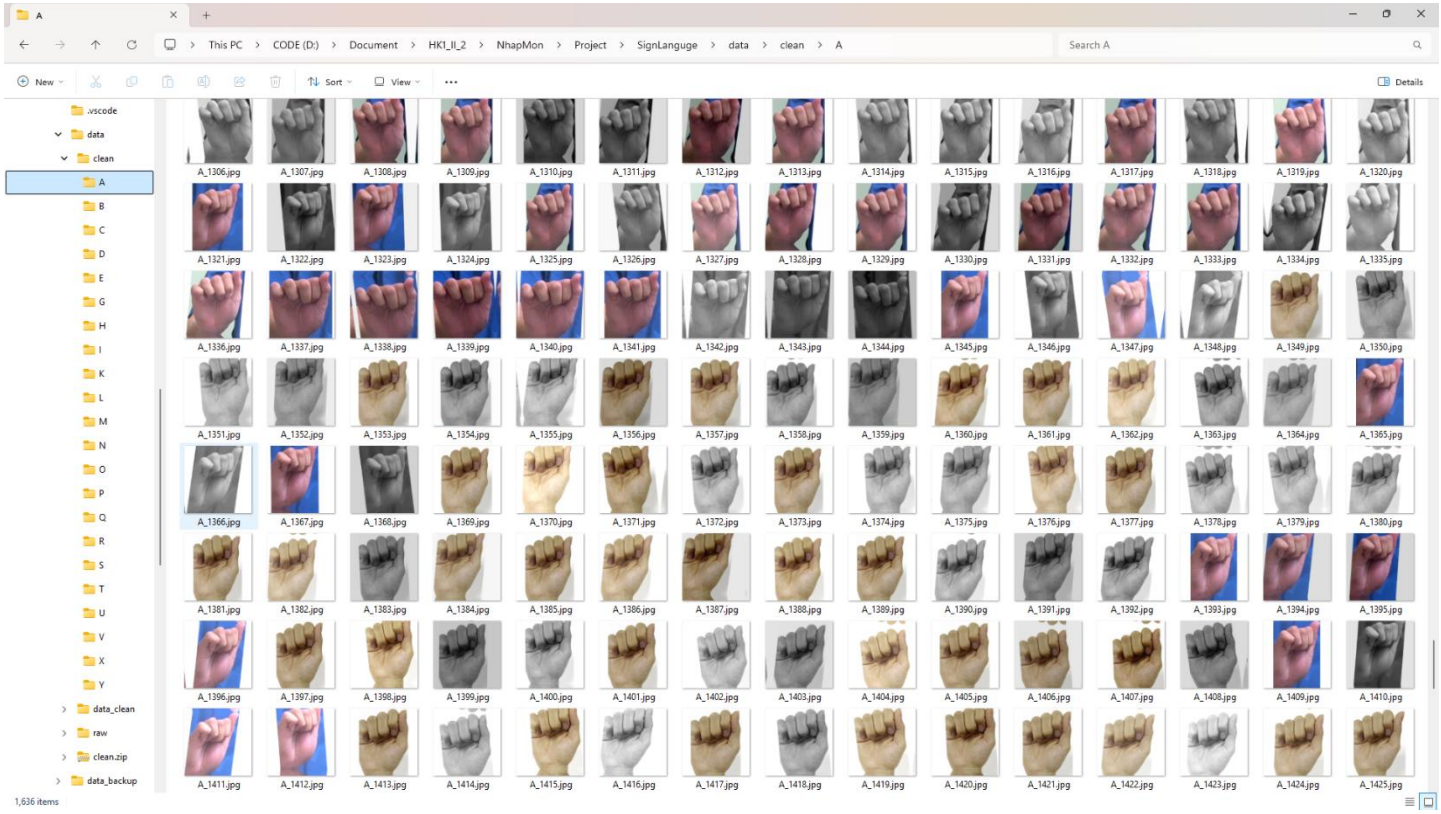
Biến đổi góc thực hiện ảnh từ -0.15 -> +0.15 (%) để giống như các góc máy khác nhau của camera

Dịch chuyển ảnh theo trục X,Y tối đa 10% kích thước để ảnh không phụ thuộc vào vj trí cố định của tay.

Tăng giảm độ sáng và dộ tương phản để thích nghi vào nhiều điều kiện môi trường và ánh sáng.

Chuyển ngẫy nhiên 50% số lượng ảnh sang ảnh xám để mô hình học tập trung và hình dáng thay vì màu sắc da

Sau cùng nhóm đã thu thập được bộ dữ liệu đa dạng phong phú với 2 img/s cho mỗi video sẵn sàng cho quá trình huấn luyện với số lượng ảnh là 41,544 ảnh



Hình 5 Ví dụ data đã qua xử lý

2.2 Phân bố dữ liệu trước khi huấn luyện

Sau quá trình thu thập dữ liệu nhóm đó có được một tập dữ liệu được trích xuất từ những videos gốc ra thành những hình ảnh được làm sạch background, không bi bể, méo nén ngoài ra còn được xử lý tăng cường thêm nhiều mô phỏng khác nhau. Cuối cùng tổng số ảnh trong dữ liệu hiện có là 41,544 ảnh.

Bộ dữ liệu bao gồm 22 nhãn lớp, tương ứng vơi s22 chữ cái trong bảng chữ cái tiếng Việt, bao gồm A, B, C, D, E, G, H, I, K, L, M, N, O, P, Q, R, S, T, U, V, X, Y.

Để tránh tình trạng overfitting hay còn gọi là họt vẹt khi cho model học 100% ảnh, nhóm không sử dụng toàn bộ dữ liệu để huấn luyện mà thay vào đó dùng bộ dữ liệu tổng thể để chia dữ liệu hiện tại ra làm 3 tập con độc lập nhau. Việc phân giúp đảm bảo cho mô hình có đầy đủ dữ liệu để học (Train), đủ dữ liệu để tinh chỉnh (Val) và một tập dữ liệu hoàn toàn mới để đánh giá kết quả cuối cùng (Test). Việc phân chia được nhóm thực hiện qua đoạn code *Create_file.py.* Nhóm áp dụng tỷ lệ được cộng đồng AI dùng nhiều là 70 – 15 – 15.

70% (Train): Dùng để tính toán gradient và cập nhật trọng số cho mạng ResNet18.

15% (Val): Dùng để đánh giá chéo ngay trong quá trình huấn luyện, giảm loss đồng thời nhóm dùng phương pháp lưu checkpoint để ghi lại phiên bản có loss thấp nhất.

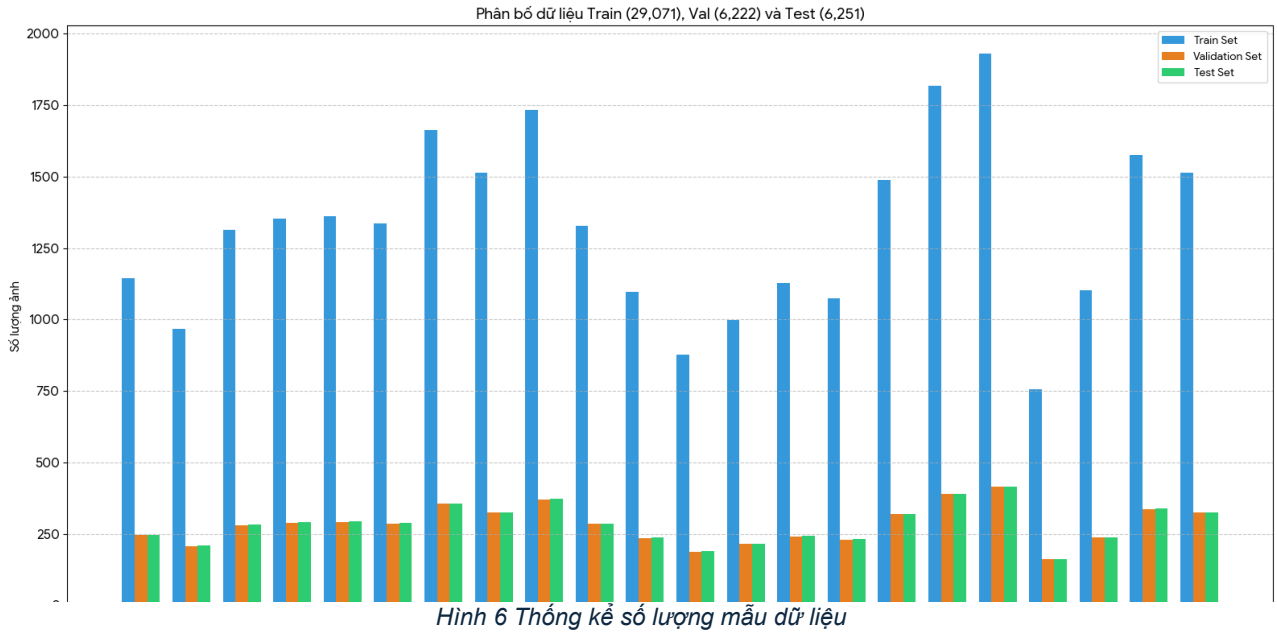
15% (Test): Được dùng một lần duy nhất khi hết vòng lặp train để đánh giá năng lực thực tế của model

Cơ chế ngẫu nhiên hóa được mô hình AI Gemini gợi ý. Nhóm em đã đã ám dụng để xóa trộn ngẫu nhiên ảnh của các lớp để đảm bảo các khung hình cùng 1 video sẽ không bị dồn vào cùng một tập train hay val hay test.

Kết quả phân chia cụ thể như sau

Tệp	Train	Val	Test
Số ảnh	29.071 (69,99%)	6.222 (14.97%)	6.251(15.04)%

Dưới đây là bảng thống kê số lượng mẫu dữ liệu cho từng nhãn ký tự trên cả 3 tập. Số liệu cho thấy sự bao phủ đầy đủ của dữ liệu lên không gian đặc trưng của bảng chữ cái.



Hình 6 Thống kê số lượng mẫu dữ liệu

Với tổng số lượng hơn 41K ảnh bộ dữ liệu đáp ứng đầy đủ yêu cầu để huấn luyện mạng nơ-ron Resnet18.

2.3 Xây dựng mô hình và huấn luyện

Nhóm lựa chọn phương pháp tràner Learing kết hợp vớ kỹ thuật Fine-tuning để sử dụng kiến trúc ResNet18 đã được huấn luyện trước tập dữ liệu ImageNet, để tận dụng các đặc trưng sẵn có và thích nghi với bài toán nhận diện ngôn ngữ ký hiệu

2.3.1 Kiến trúc ResNet18

Nhóm sử dụng mô hình *models.resnet18* được cung cấp bởi thư viện torchvision. ResNet18 là một phiên bản "nhẹ" nhất trong họ ResNet (chỉ khoảng 11 triệu tham số) Nó đủ phức tạp để trích xuất các đặc trưng hình học của bàn tay nhưng vẫn đảm bảo được tốc độ suy luận cho các thiết bị chạy thời gian thực.

Nhóm nạp mô hình ResNet18 với bộ trọng số (weights) đã được học sẵn trên ImageNet thay vì khởi tạo ngẫu nhiên.

```
model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
```

Hình 7

2.3.2 Fine-tuning

Thay vì khởi tạo trọng số ngẫu nhiên, mô hình được khởi tạo với bộ trọng số đã được huấn luyện trước (pre-trained weights) trên tập dữ liệu ImageNet. Tuy nhiên, ImageNet chủ yếu chứa các đối tượng tổng quát (chó, mèo, xe…), trong khi bài toán của nhóm tập trung vào nhận diện cử chỉ tay – một miền dữ liệu có đặc thù riêng.

Do đó, nhóm xây dựng chiến lược đóng băng – mở khóa có chọn lọc các lớp mạng trong file Train.py.

Đặt *requires_grad = False* cho các tham số của mô hình ban đầu. Nhằm giữ nguyên các khả năng trích xuất các đặc trung cơ bản của mô hình tránh làm mất đặc trưng tổng quát sẵn từ imageNett rong khi tập dữ liệu còn nhóm còn hạn chế.

Sau khi phân tích lựa chọn 2 khối tích chập cuối là layer3 và layer4 là những đặc trăng ở các tầng này mang tính đặc thù cao và cần điều chỉnh phù hợp với bài toán của nhóm.

```
for param in model.parameters():
    param.requires_grad = False
for param in model.layer3.parameters():
    param.requires_grad = True
for param in model.layer4.parameters():
    param.requires_grad = True
```

Hình 8

Lớp Fully Connected mặc định được thiết kế cho bài toán phân loại 1000 lớp của ImageNet. Do đó nhóm thay thế bằng một lớp phân lạo mới cho phù hợp với bài toán 22 lớp ký hiệu.

Trong quá trình thực nghiệm, nhóm nhận thấy mô hình có xu hướng overfitting do số lượng tham số lớn so với kích thước dữ liệu. Dropout giúp ngắt ngẫu nhiên 50% số kết nối trong quá trình huấn luyện, buộc mô hình phải học các đặc trưng bền vững hơn.

```
    num_ftrs = model.fc.in_features # lấy số features đầu vào
    model.fc = nn.Sequential(
        nn.Dropout(0.5), # dropout để tránh overfitting
        nn.Linear(num_ftrs, num_classes)
# số lớp đầu ra tương ứng số classes
    )
```

Hình 9

2.3.3 Cấu hình các tham số

Hàm mất mát (Loss Function) với Label Smoothing. Thay vì dùng CrossEntropy nhóm sử dụng *CrossEntropyLoss(label_smoothing=0.1)* thay vì ép mô hình học tuyệt đối [0;1], Label Smoothing giúp làm mềm đi nhãn mục tiêu giảm đi hiện tượng dự đoán tự tin và cải thiện trên dữ liệu chưa từng thấy

Learning rates nhóm sử dụng thuật toán Adam, kết hợp với nhiều mức learning rate khác nhau cho từng lớp mô hình. Các lớp 4,3 được huấn luyện tốt nên

```
criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
optimizer = optim.Adam([
    {'params': model.layer3.parameters(), 'lr': 1e-4},
    {'params': model.layer4.parameters(), 'lr': 1e-4},
    {'params': model.fc.parameters(), 'lr': 1e-3}
])
```

Hình 10

nhóm thay đổi đặt chỉ số LR rất nhẹ 0.0001. Ngược lại lớp cuối cùng là mới hoàn toàn nên cần thay đổi nhiều nên chỉ số LR được đặt là 0.001 gấp 10 lần để nhanh chóng học các mối liên kết mới.

2.3.4 Huấn luyện

Tiến hành huấn luyện với số Epoch là 30 đủ để mô hình hội tụ mà không tốn quá nhiều thời gian.

Quy trình trong mỗi epoch :

- Giai đoạn huấn luyện :

+ Ảnh đầu vào qua các lớp đã freeze để trích xuất đặc trưng, sau đó qua lớp phân loại để tạo ra vector dự đoán

+ Dùng hàm mấy mát để đo % độ chênh lệch dự đoán của mô hình với nhãn thật

+ Gradient của loss được lan truyền ngược, chỉ cập nhật các tham số của lớp phân loại, trong khi các tầng tích chập giữ nguyên

+ Thuật toán Adam cập nhật trọng số dựa trên gradient tính được

+ Lặp lại cho từng batch mỗi gói gồm 32 ảnh

- Giai đoạn kiểm định :

+ Tắt Dropout

+ Tắt tính đạo hàm

+ Mô hình thực hiện lan truyền xuôi để tính Loss và Accuracy trên một tập dữ liệu là mà nó chưa từng học

Sau mỗi epoch sẽ thực hiện tính toán Loss và Acc cho 2 giai đoạn lưu lại chỉ số để đánh giá mô hình

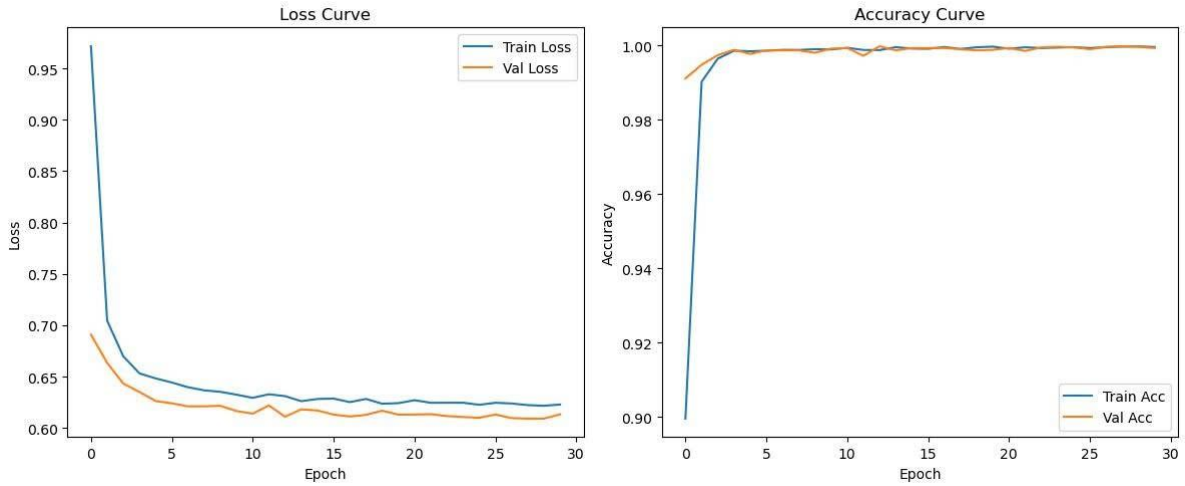
Ngoài ra nhóm còn áp dụng cơ chế lưu checkpoint tốt nhất hệ thống sẽ so sánh Acc của Val hiện tại với Best_acc. Kết quả sau khi train là mô hình tốt nhất có độ chính xác cao nhất.

Sau khi huấn luyện hoàn tất, mô hình được đánh giá trên tập test – tập dữ liệu hoàn toàn độc lập với quá trình huấn luyện. Kết quả trên tập test phản ánh chính xác khả năng ứng dụng của mô hình trong môi trường thực tế.

PHẦN 3. KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

3.1 Kết quả huấn luyện

Quá trình huấn luyện diễn ra trong 30 epochs với kích thước batch là 32. Kết quả được ghi nhận và trực quan hóa thông qua hai biểu đồ chính: Biểu đồ Hàm mất mát (Loss Curve) và Biểu đồ Độ chính xác (Accuracy Curve)



Hình 11 Áp dụng uấn luyên trên data thật

Biểu đồ hàm mất mát: Giá trị Loss giảm rất nhanh và sâu ngay trong 5 epoch đầu tiên (từ mức cao xuống dưới 0.7). Sau khhjoảng epoch thứ 15, đường Loss bắt đầu đi ngang (bão hòa) và tiệm cận về 0. Đường Val Loss (cam) bám rất sát và có xu hướng thấp hơn đường Train Loss (xanh) ở giai đoạn đầu. Điều này cho thấy mô hình tổng quát hóa tốt và không bị hiện tượng overfitting. Việc sử dụng kỹ thuật Label Smoothing và Dropout đã phát huy tác dụng hiệu quả.

Biểu đồ độ chính xác: Độ chính xác trên cả tập Train và Validation đều tăng vọt lên mức trên 98% chỉ sau vài epoch đầu và duy trì ổn định ở mức tiệm cận 99-100% trong suốt quá trình còn lại. Mô hình đạt độ hội tụ rất tốt. Việc fine-tuning các lớp sâu (layer3, layer4) giúp mô hình nắm bắt chính xác các đặc trưng của bàn tay.

Sau khi kết thúc quá trình huấn luyện, nhóm sử dụng mô hình tốt nhất (Best Checkpoint) để đánh giá trên Tập Test tập dữ liệu hoàn toàn độc lập chưa từng được sử dụng trước đó. Độ chính xác tổng thể (Overall Accuracy): Đạt mức > 99%. Kết quả này tương đồng với kết quả trên tập Validation, khẳng định độ tin cậy của mô hình. Hệ thống có khả năng nhận diện chính xác hầu hết các ký tự trong điều kiện ảnh tĩnh được cắt chuẩn xác.

3.2 Đánh giá thực tế

Nhóm đã triên khai mô hình nhận diện thời gian thực thông qua webcam. Hệ thống hoạt động với độ trễ thấp. Nhóm lọc ngưỡng trên 50% thì mới hiện kết quả.

Mặc dù độ chính xác khi train rất cao nhưng khi đem ra thực tế vẫn còn một số nhầm lẫn như ký tự E và B, A và M.

Do sự tương dồng về ký hiệu chữ E và B vì cả 2 đều có điểm chung là 4 ngón và ngón cái gập vào chỉ khác nhau là chữ E co lại B thì duỗi ra M và A đều là sự gập của các ngón tay nên khiến mô hình cũng khó nhận diện

Tuy nhiên nhóm em nhận thấy mô hình cũng đã hoạt động rất tốt trong thời gian thật. Đây là một dự án rất bổ ích cho xã hội nếu được xây dựng thêm khi có một lương dữ liệu đủ lớn và đa dạng.

PHỤ LỤC

Link video demo :

Link github : https://github.com/nguyenkhoiai116/SignLanguage.git

Link data :