TRƯỜNG ĐIỆN – ĐIỆN TỬ

BÁO CÁO
HỌC SÂU
THIẾT KẾ HỆ THỐNG NHẬN DIỆN KHUÔN MẶT 30 ĐỐI TƯỢNG ỨNG DỤNG TRANSFER LEARNING

QUÁCH KIÊN TRUNG
Trung.qk241089e@sis.hust.edu.vn

Ngành Kỹ thuật Điều khiển và Tự động hóa

Giảng viên hướng dẫn: TS. Nguyễn Hoài Nam

Chữ ký của GVHD

Khoa: Tự động hóa
Trường: Điện – Điện tử

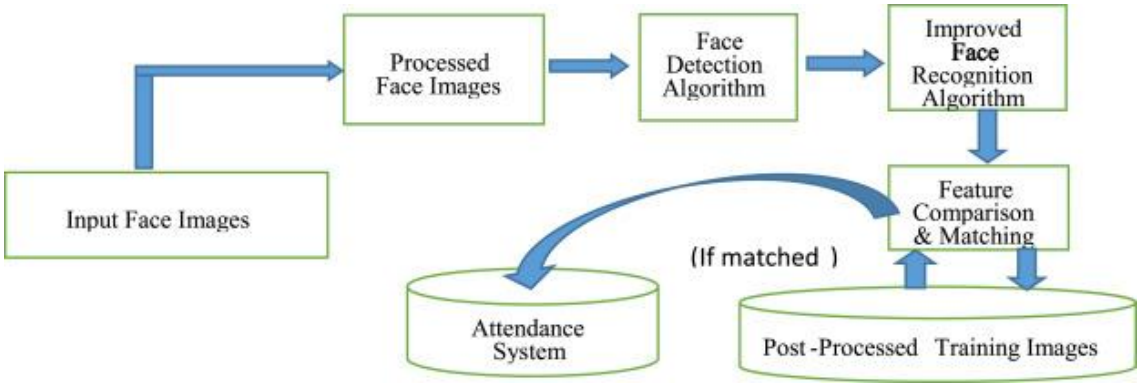
HÀ NỘI, 1/2026.

Mục lục

CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1 Mục tiêu đề tài

Trong bối cảnh công nghệ học sâu đang phát triển mạnh mẽ và dần trở thành nòng cốt trong các hệ thống giám sát thông minh, việc triển khai các mô hình mạng nơ-ron lên thiết bị biên đóng vai trò then chốt nhằm tối ưu hóa thời gian phản hồi và bảo mật dữ liệu. Đề tài này được thực hiện với mục tiêu thiết kế và xây dựng một hệ thống nhận diện khuôn mặt chuyên biệt cho nhóm đối tượng gồm 30 người, ứng dụng kiến trúc mạng tích chập SqueezeNet. Việc lựa chọn kiến trúc này hướng tới mục đích cân bằng khắt khe giữa độ chính xác nhận diện và khả năng tiết kiệm tài nguyên tính toán, qua đó đáp ứng yêu cầu vận hành ổn định trên các nền tảng máy tính nhúng có cấu hình hạn chế như Raspberry Pi 5.



Hình 1.1 Sơ đồ quy trình nhận diện khuôn mặt

Mục tiêu trọng tâm của đồ án tập trung vào việc làm chủ kỹ thuật Transfer Learning để huấn luyện mô hình hiệu quả trên tập dữ liệu thực tế có quy mô nhỏ, cụ thể là 20 mẫu ảnh cho mỗi cá nhân. Hệ thống được thiết kế để không chỉ đạt được độ chính xác cao trong việc phân loại danh tính mà còn phải tối ưu hóa về mặt kích thước tệp trọng số và số lượng tham số. Thông qua việc tùy chỉnh lớp Classifier và áp dụng các phương pháp Data Augmentation như xoay, lật và điều chỉnh độ tương phản, đồ án hướng tới việc giải quyết triệt để hiện tượng quá khớp, từ đó nâng cao khả năng tổng quát hóa của mạng nơ-ron trong các điều kiện môi trường ánh sáng thay đổi.

Bên cạnh khía cạnh thuật toán, đề tài còn nhằm mục tiêu hoàn thiện quy trình triển khai từ khâu thu thập dữ liệu, huấn luyện trên môi trường Google Colab cho đến khâu thực thi nhận diện tại biên. Kết quả thực hiện sẽ được đánh giá định lượng thông qua các chỉ số kỹ thuật như thời gian Inference, dung lượng bộ nhớ tiêu thụ và ma trận nhầm lẫn giữa các đối tượng. Qua đó, đồ án không chỉ minh chứng cho tính khả thi của việc ứng dụng các mô

hình học sâu thu gọn trong bài toán điểm danh tự động mà còn giúp sinh viên rèn luyện tư duy thiết kế hệ thống và kỹ năng tối ưu hóa phần mềm nhúng trong thực tiễn kỹ thuật.

1.2 Thách thức kỹ thuật

Quá trình thiết kế và triển khai hệ thống nhận diện khuôn mặt thực tế đối mặt với nhiều rào cản kỹ thuật mang tính đặc thù. Một trong những thách thức lớn nhất nằm ở sự hạn chế về kích thước và tính đồng nhất của tập dữ liệu huấn luyện khi mỗi đối tượng định danh chỉ có tối đa 20 mẫu ảnh màu kích thước 300x300. Trong lý thuyết Deep Learning, các mạng nơ-ron sâu thường yêu cầu hàng nghìn mẫu dữ liệu để có thể trích xuất đặc trưng một cách ổn định. Việc huấn luyện trên một tập dữ liệu nhỏ dễ dẫn đến hiện tượng Overfitting, nơi mô hình chỉ ghi nhớ các pixel cụ thể của ảnh mẫu thay vì học được các đặc điểm nhân trắc học tổng quát, gây sai số lớn khi nhận diện. Đặc biệt, do dữ liệu được thu thập từ nhiều nguồn khác nhau với chất lượng thiết bị và điều kiện ngoại cảnh không đồng nhất, việc đảm bảo độ chính xác hội tụ cho cả 30 lớp đối tượng trở nên cực kỳ phức tạp.

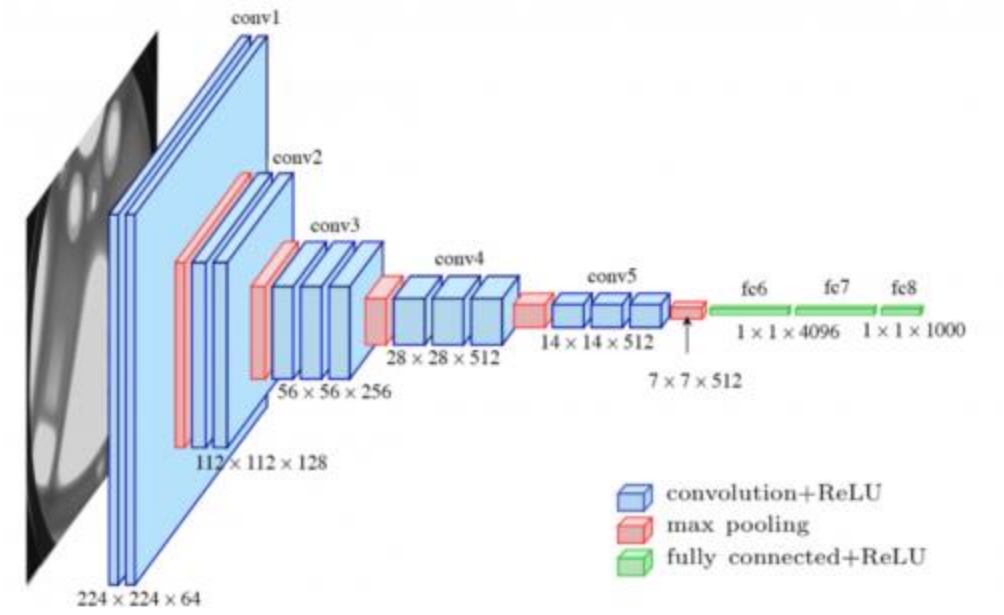
Thách thức thứ hai liên quan trực tiếp đến việc tối ưu hóa cho phần cứng nhúng tại biên. Mặc dù Raspberry Pi 5 là một máy tính nhúng có cấu hình mạnh, việc xử lý các phép toán tích chập thời gian thực vẫn tiêu tốn đáng kể tài nguyên CPU và dung lượng RAM. Các mô hình CNN tiêu biểu như VGG16 hay AlexNet có dung lượng quá lớn và số lượng tham số lên tới hàng chục triệu, vượt quá khả năng xử lý mượt mà và gây độ trễ lớn khi thực thi Inference. Do đó, việc thiết kế lại kiến trúc mạng dựa trên SqueezeNet để vừa đảm bảo tốc độ phản hồi nhanh vừa giữ được độ chính xác là một bài toán cân bằng tài nguyên tài nguyên phức tạp mà đồ án cần giải quyết. Sự cân bằng này yêu cầu mô hình phải đạt kích thước tối ưu trong khoảng 5MB nhưng vẫn phải đảm bảo khả năng phân biệt được sự tương đồng giữa các khuôn mặt khác nhau.

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT VÀ KIẾN TRÚC MẠNG

2.1 Tổng quan về mạng nơ-ron tích chập (CNN)

Mạng nơ-ron tích chập là một kiến trúc học sâu chuyên biệt được thiết kế để xử lý dữ liệu dạng mảng, đặc biệt là hình ảnh kỹ thuật số với cấu trúc không gian ba chiều bao gồm chiều rộng, chiều cao và chiều sâu kênh màu. Khác với các mạng nơ-ron truyền thống nơi mọi điểm ảnh đều được kết nối với mọi nơ-ron ở lớp tiếp theo gây tốn kém tài nguyên tính toán, CNN sử dụng cơ chế kết nối cục bộ và chia sẻ trọng số thông qua các bộ lọc tích chập. Điều này cho phép mạng tự động trích xuất các đặc trưng quan trọng từ ảnh như đường biên, góc cạnh và các kết cấu bề mặt mà không cần sự can thiệp thủ công của con người. Trong các bài toán thị giác máy tính phức tạp như nhận diện khuôn mặt, CNN đã chứng minh được sự vượt trội nhờ khả năng bảo toàn tính tương quan không gian giữa các pixel lân cận, giúp mô hình đạt được độ chính xác cao trong việc phân loại đối tượng.

Cấu trúc cơ bản của một mạng CNN bao gồm sự sắp xếp xen kẽ của ba loại lớp chính là lớp tích chập, lớp kích hoạt phi tuyến và lớp lấy mẫu hạ mẫu. Lớp tích chập thực hiện nhiệm vụ quét bộ lọc qua ảnh đầu vào để tạo ra các bản đồ đặc trưng, ghi lại sự hiện diện của các mẫu hình cụ thể. Ngay sau đó, hàm kích hoạt ReLU thường được áp dụng để chuyển đổi các giá trị đầu ra sang dạng phi tuyến bằng cách giữ nguyên các giá trị dương và đưa các giá trị âm về số không, giúp mạng có khả năng học các hàm số phức tạp hơn. Để giảm bớt khối lượng tính toán và tránh hiện tượng quá khớp, các lớp Pooling như Max Pooling sẽ thực hiện việc giảm kích thước của các bản đồ đặc trưng bằng cách chỉ giữ lại giá trị lớn nhất trong một vùng cửa sổ xác định. Quy trình này giúp hệ thống tập trung vào các đặc trưng nổi bật nhất, đồng thời tăng cường tính bất biến của mô hình đối với các thay đổi nhỏ về vị trí hoặc góc nhìn của vật thể.



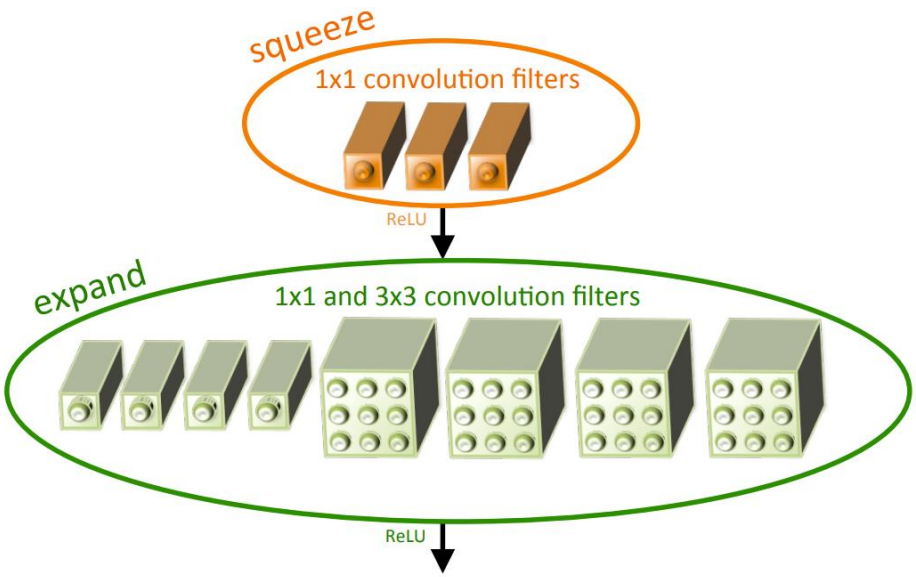
Hình 2.1 Cấu trúc các lớp thành phần trong mạng nơ-ron tích chập

Quá trình học tập trong mạng CNN diễn ra theo cơ chế phân cấp đặc trưng từ mức độ thô đến mức độ tinh vi. Ở những lớp đầu tiên của mạng, các bộ lọc thường học cách phát hiện những đặc điểm hình học đơn giản nhất như các đường thẳng đứng, đường nằm ngang hoặc các điểm màu sắc. Khi đi sâu vào các lớp tiếp theo, mô hình bắt đầu kết hợp các đặc điểm cơ bản này để tạo thành những hình dạng có ý nghĩa hơn như góc mắt, vành tai hoặc sống mũi. Tại những lớp cuối cùng, mạng nơ-ron sẽ tổng hợp toàn bộ các thông tin đặc trưng này để hình thành nên một biểu diễn trừu tượng hoàn chỉnh về khuôn mặt của một cá nhân cụ thể, phục vụ cho mục đích phân loại định danh. Chính nhờ cơ chế học phân cấp này mà CNN có khả năng nhận diện đối tượng một cách mạnh mẽ trong nhiều điều kiện môi trường và trạng thái khuôn mặt khác nhau.

2.2 Kiến trúc mạng SqueezeNet và khối Fire Module

Kiến trúc mạng SqueezeNet được giới thiệu như một bước đột phá trong việc tối ưu hóa mạng nơ-ron sâu cho các ứng dụng thực tế trên thiết bị di động và hệ thống nhúng. Triết lý thiết kế chủ đạo của SqueezeNet không tập trung vào việc gia tăng độ chính xác tuyệt đối mà hướng tới việc duy trì độ chính xác tương đương các mạng tiêu chuẩn như AlexNet trong khi giảm thiểu tối đa số lượng tham số học tập. Điều này được thực hiện thông qua ba chiến lược chính: thay thế phần lớn các bộ lọc 3x3 bằng các bộ lọc 1x1 để giảm lượng tham số đi 9 lần, giảm số lượng kênh đầu vào cho các bộ lọc 3x3 còn lại, và thực hiện lấy mẫu hạ thấp (Downsampling) muộn trong mạng để bản đồ đặc trưng giữ được nhiều thông tin quan trọng hơn.

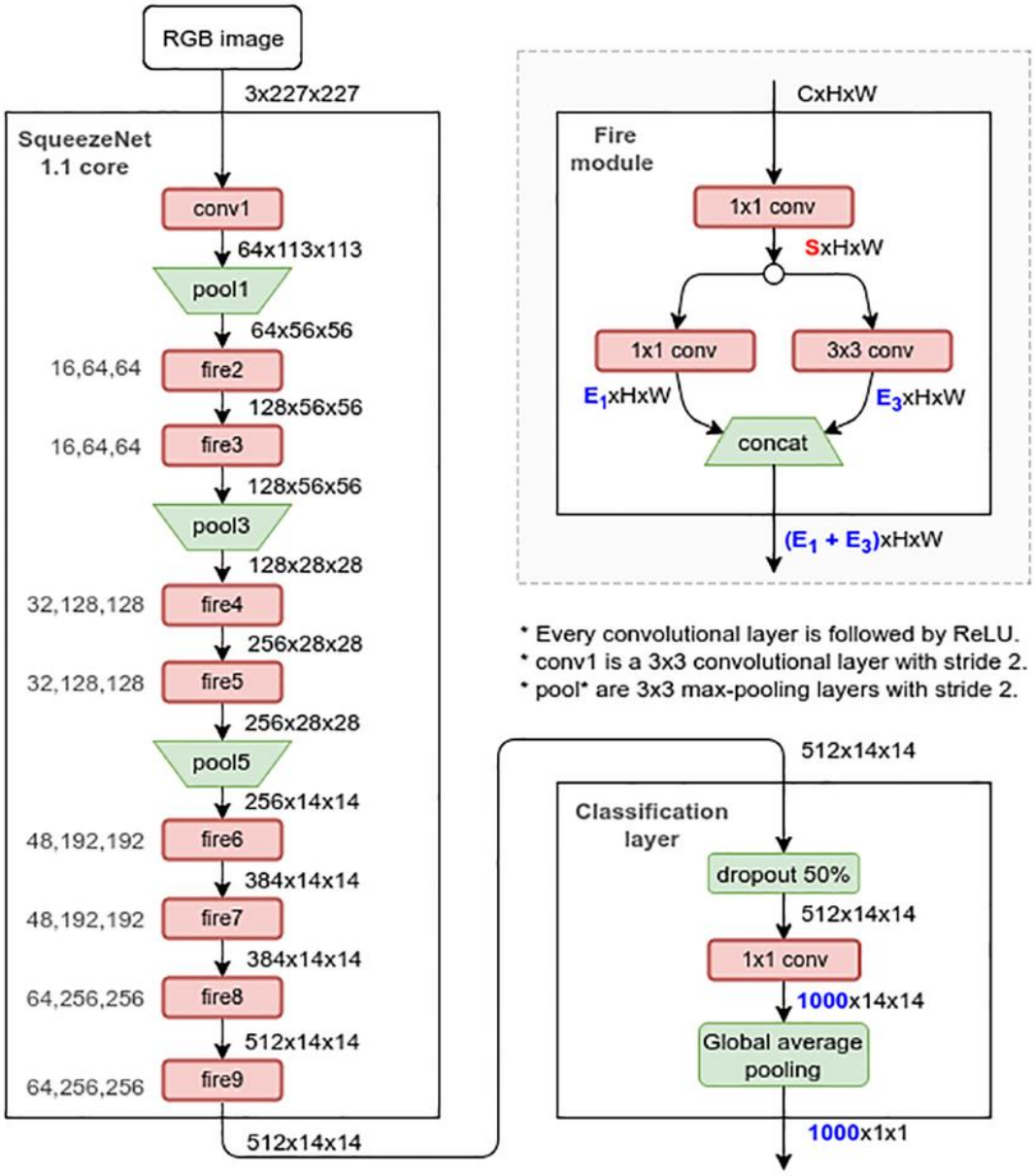
Trái tim của mạng SqueezeNet là khối Fire Module, một cấu trúc đặc thù giúp thực hiện việc nén và mở rộng dữ liệu một cách hiệu quả. Một khối Fire Module tiêu chuẩn bao gồm hai phần chính là lớp Squeeze và lớp Expand. Lớp Squeeze chỉ chứa các bộ lọc 1x1 Convolution đóng vai trò nén số lượng kênh của dữ liệu đầu vào xuống một giá trị nhỏ hơn đáng kể. Ngay sau đó, dữ liệu này được đưa vào lớp Expand, nơi nó được xử lý song song bởi một nhóm các bộ lọc 1x1 Convolution và một nhóm các bộ lọc 3x3 Convolution.



Hình 2.2 Tổ chức bộ lọc tích chập trong Fire module

Kết quả đầu ra của hai nhóm này sẽ được ghép nối lại (Concatenation) để tạo thành đầu ra cuối cùng của khối. Cơ chế này cho phép mạng bắt được cả những đặc trưng ở quy mô nhỏ và quy mô lớn hơn mà vẫn đảm bảo khối lượng tính toán cực kỳ thấp.

Nhờ việc sử dụng các khối Fire Module lặp lại kết hợp với việc loại bỏ các lớp Fully Connected truyền thống và thay thế bằng Global Average Pooling, SqueezeNet đạt được kích thước mô hình cực kỳ ấn tượng. Với định dạng trọng số 32-bit thông thường, dung lượng của mô hình chỉ rơi vào khoảng hơn 5MB, tức là nhỏ hơn khoảng 50 lần so với mạng AlexNet. Đặc tính nhỏ gọn này đặc biệt có ý nghĩa đối với các kỹ sư hệ thống khi triển khai mô hình lên các vi điều khiển hoặc máy tính nhúng như Raspberry Pi 5, nơi băng thông bộ nhớ và khả năng lưu trữ thường bị giới hạn nghiêm ngặt. Việc làm chủ kiến trúc này cho phép hệ thống nhận diện khuôn mặt hoạt động với tốc độ xử lý thời gian thực mà không yêu cầu các bộ tăng tốc đồ họa (GPU) đắt tiền.



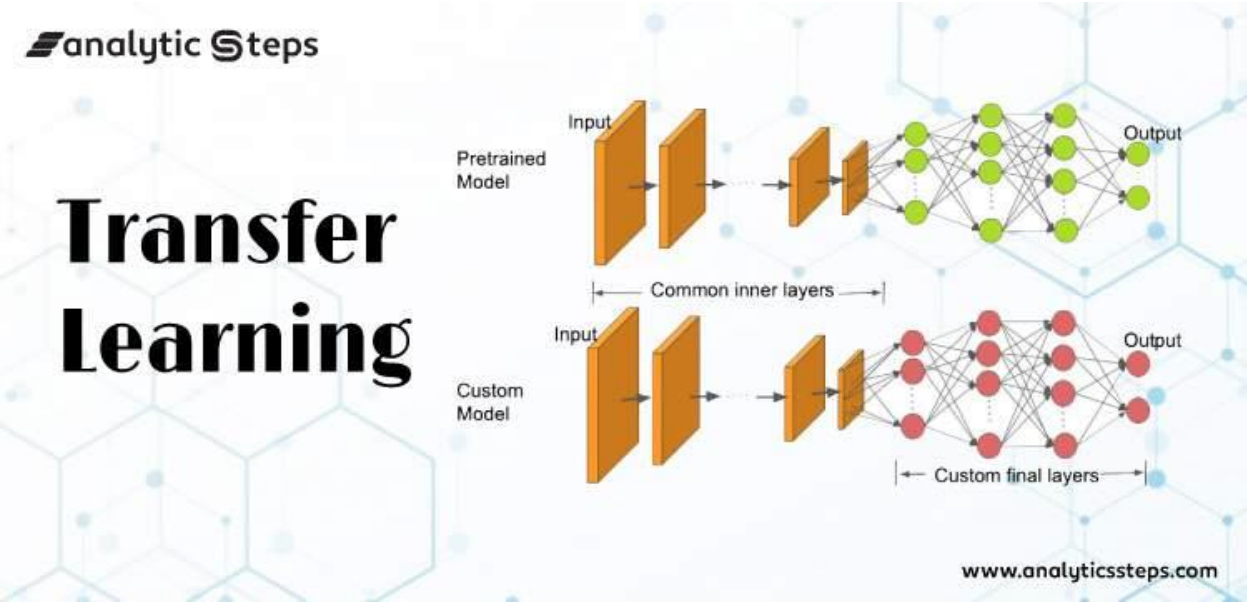
Hình 2.3 Cấu trúc khối Fire Module với cơ chế Squeeze và Expand.

2.3 Phương pháp Transfer Learning

Phương pháp học chuyển đổi là một kỹ thuật trong lĩnh vực học máy, nơi một mô hình đã được huấn luyện đầy đủ cho một nhiệm vụ cụ thể sẽ được tái sử dụng làm nền tảng khởi đầu cho một nhiệm vụ mới có liên quan. Thay vì phải bắt đầu quy trình huấn luyện mạng nơ-ron từ con số không với các trọng số được khởi tạo ngẫu nhiên, học chuyển đổi cho phép hệ thống kế thừa tri thức về các đặc trưng hình ảnh tổng quát mà mô hình tiền huấn luyện đã học được từ các bộ dữ liệu khổng lồ như ImageNet. Triết lý của phương pháp này dựa trên quan sát thực tế rằng các lớp đầu tiên của mạng nơ-ron tích chập luôn

học các đặc trưng thị giác cơ bản như đường biên, màu sắc và kết cấu bề mặt, vốn mang tính phổ quát và có thể áp dụng hiệu quả cho hầu hết các bài toán thị giác máy tính khác nhau.

Việc áp dụng học chuyển đổi mang lại những lợi thế quyết định đối với các bài toán có nguồn dữ liệu hạn chế như hệ thống nhận diện khuôn mặt trong đồ án này. Khi tập dữ liệu huấn luyện chỉ có 20 mẫu ảnh cho mỗi đối tượng, việc huấn luyện một mạng nơ-ron sâu từ đầu thường không khả thi do mô hình không đủ dữ liệu để hội tụ và rất dễ rơi vào tình trạng quá khớp nghiêm trọng. Bằng cách sử dụng mạng SqueezeNet đã được tiền huấn luyện, hệ thống có thể tận dụng ngay bộ trích xuất đặc trưng đã được tối ưu hóa qua hàng triệu hình ảnh. Điều này không chỉ giúp gia tăng đáng kể độ chính xác nhận diện mà còn tiết kiệm được nguồn lực tính toán và thời gian huấn luyện, cho phép sinh viên tập trung vào việc tinh chỉnh các lớp cuối cùng để phù hợp với các đặc điểm nhân trắc học cụ thể của 30 cá nhân cần định danh.



Hình 2.4 Sơ đồ kiến trúc Transfer Learning với việc tái sử dụng Feature Extractor

Quy trình triển khai học chuyển đổi thường bao gồm hai giai đoạn chính là đóng băng các lớp trích xuất đặc trưng và thiết kế lại bộ phân loại đầu ra. Trong giai đoạn đầu, các trọng số của 17 lớp đầu tiên trong mạng SqueezeNet sẽ được giữ cố định để bảo toàn khả năng nhận diện hình học cơ bản. Giai đoạn tiếp theo tập trung vào việc thay thế lớp nơ-ron cuối cùng vốn được thiết kế để phân loại 1000 vật thể thông thường bằng một cấu trúc mới tương ứng với 30 lớp đối tượng khuôn mặt. Sau khi cấu trúc mới được hình thành, quá trình tinh chỉnh có thể được thực hiện với tốc độ học cực nhỏ trên các lớp sâu hơn nhằm giúp mô hình thích nghi tốt nhất với các biến đổi về ánh sáng và góc chụp đặc thù

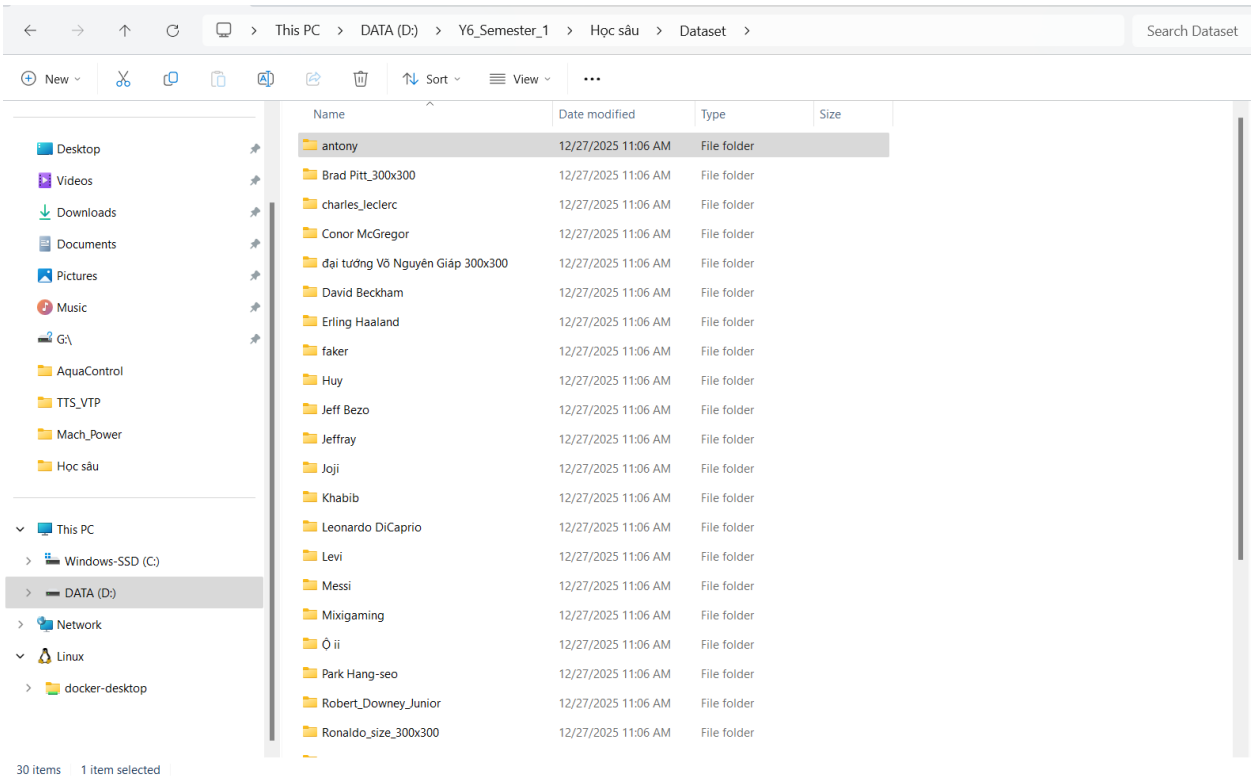
trong tập dữ liệu thực tế. Cách tiếp cận này đảm bảo rằng mô hình cuối cùng vừa giữ được sự gọn nhẹ của kiến trúc gốc, vừa đạt được hiệu năng nhận diện chuyên biệt cao.

CHƯƠNG 3: THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG

3.1. Xây dựng và Tiền xử lý Dataset

Việc xây dựng một tập dữ liệu chuẩn hóa là bước đi tiên quyết để đảm bảo mô hình có thể học được các đặc trưng khuôn mặt một cách chính xác. Trong đồ án này, tập dữ liệu được hình thành từ quá trình thu thập ảnh thực tế của 30 cá nhân khác nhau, với quy định mỗi đối tượng cung cấp 20 mẫu ảnh ở định dạng màu RGB. Các hình ảnh này ban đầu có kích thước 300x300 pixel. Quy trình gán nhãn dữ liệu được thực hiện trực tiếp thông qua việc tổ chức hệ thống tệp tin theo quy chuẩn của các thư viện học sâu hiện đại, trong đó mỗi thư mục con mang tên một cá nhân cụ thể đóng vai trò là nhãn định danh. Cách tiếp cận này cho phép sử dụng các hàm tự động như ImageFolder của thư viện Torchvision để ánh xạ tên thư mục thành các nhãn lớp từ 0 đến 29.

Một mắt xích quan trọng trong việc thiết lập dữ liệu là quy định về tỷ lệ phân chia tập tin để đánh giá khách quan hiệu năng mạng nơ-ron. Toàn bộ dữ liệu sau khi gán nhãn được phân tách theo tỷ lệ 70% dành cho quá trình huấn luyện (Training set) và 30% còn lại được giữ độc lập để phục vụ giai đoạn đánh giá mô hình (Validation set). Việc dành ra 30% dữ liệu để kiểm thử là yêu cầu bắt buộc nhằm đảm bảo mô hình không chỉ học vẹt các mẫu có sẵn mà phải có khả năng tổng quát hóa trên những hình ảnh mới. Sự phân chia này giúp cung cấp các chỉ số đo lường chính xác về độ tin cậy của hệ thống trước khi triển khai thực tế trên phần cứng nhúng.



Hình 3.1 Cấu trúc thư mục dữ liệu của 30 đối tượng

Để hiện thực hóa quy trình tiền xử lý và gia tăng dữ liệu đã mô tả, hệ thống sử dụng thư viện torchvision.transforms để xây dựng hai chuỗi biến đổi dữ liệu riêng biệt cho quá trình huấn luyện và kiểm tra. Đối với tập huấn luyện, chuỗi biến đổi bao gồm các lớp Resize để đưa ảnh về kích thước chuẩn 227x227 pixel theo yêu cầu của mạng SqueezeNet, kết hợp với các phép gia tăng dữ liệu ngẫu nhiên như lật ngang và xoay ảnh để chống lại hiện tượng quá khớp. Ngược lại, tập kiểm tra chỉ áp dụng các bước chuẩn hóa bắt buộc nhằm đảm bảo tính khách quan khi đánh giá hiệu năng thực tế. Toàn bộ dữ liệu sau đó được chuyển đổi sang dạng Tensor và chuẩn hóa theo các thông số kỳ vọng của bộ dữ liệu ImageNet để tối ưu hóa khả năng hội tụ của các trọng số đã được huấn luyện trước.

```
  # Bộ lọc cho tập Huấn luyện: Có Augmentation để chống Overfitting
  train_transforms = transforms.Compose([
      transforms.Resize((IMG_SIZE, IMG_SIZE)),
      transforms.RandomHorizontalFlip(p=0.5),
      transforms.RandomRotation(degrees=0),
         transforms.RandomPerspective(distortion_scale=0.2, p=0.5),
       transforms.ColorJitter(brightness=0., contrast=0.3, saturation=0.3),
      transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
  ])


  # Bộ lọc cho tập Kiểm tra: Chỉ chuẩn hóa, KHÔNG biến đổi ngẫu nhiên
  val_transforms = transforms.Compose([
      transforms.Resize((IMG_SIZE, IMG_SIZE)),
      transforms.ToTensor(),
```

```
        transforms.Normalize(mean=[0.485,  0.456,  0.406],
std=[0.229, 0.224, 0.225])
    ])
```

Hình 3.2 Cấu trúc mã nguồn thiết lập chuỗi biến đổi dữ liệu

Về mặt logic kỹ thuật, việc tách biệt hai bộ lọc train_transforms và val_transforms đảm bảo rằng các phép biến đổi ngẫu nhiên chỉ xuất hiện trong quá trình học tập để tăng cường tính tổng quát hóa cho mô hình. Việc sử dụng kích thước 227x227 là bắt buộc đối với kiến trúc SqueezeNet 1.1 để đảm bảo tính toàn vẹn về kích thước không gian khi dữ liệu đi qua các khối Fire Module. Ngoài ra, bước chuẩn hóa Normalize với các thông số mean và std của ImageNet đóng vai trò then chốt trong việc đưa phân phối dữ liệu mới về cùng hệ quy chiếu với tri thức mạng đã được học, giúp bộ trích xuất đặc trưng hoạt động hiệu quả ngay từ những Epoch đầu tiên.

Sau khi thiết lập các chuỗi biến đổi dữ liệu, mã nguồn thực hiện việc nạp dữ liệu và phân chia tập tin thông qua một logic xử lý chuyên nghiệp nhằm tách biệt hoàn toàn đặc tính của tập huấn luyện và tập kiểm tra. Thay vì nạp dữ liệu một lần, hệ thống thực hiện nạp thư mục Dataset hai lần với hai bộ lọc khác nhau là train_transforms và val_transforms. Phương pháp này cho phép chúng ta trích xuất các tập con từ cùng một nguồn dữ liệu nhưng lại đảm bảo được rằng chỉ có các ảnh trong tập huấn luyện mới chịu tác động của kỹ thuật gia tăng dữ liệu, trong khi các ảnh tập kiểm tra luôn được giữ ở trạng thái nguyên bản để đảm bảo tính khách quan trong đánh giá.

Việc phân chia được thực hiện dựa trên việc xáo trộn ngẫu nhiên danh sách chỉ số của 600 tấm ảnh thông qua hàm shuffle với một giá trị seed cố định nhằm đảm bảo tính tái lập của thực nghiệm. Sau khi tính toán được điểm cắt dựa trên tỷ lệ TRAIN_RATIO là 0.7, lớp Subset sẽ được sử dụng để tạo ra hai đối tượng dữ liệu riêng biệt là train_data và val_data. Cuối cùng, các bộ nạp dữ liệu DataLoader được khởi tạo để quản lý việc đưa ảnh vào GPU theo từng lô Batch_Size có kích thước là 16. Việc thiết lập shuffle=True cho bộ nạp huấn luyện đóng vai trò ép mô hình phải học các đặc trưng một cách độc lập, tránh hiện tượng học thuộc lòng thứ tự xuất hiện của các sinh viên trong tập dữ liệu, từ đó tối ưu hóa quá trình hội tụ của các trọng số nơ-ron.

3.2. Thiết kế cấu trúc mô hình

Kiến trúc mô hình trong đồ án này được xây dựng dựa trên mạng SqueezeNet phiên bản 1.1, một kiến trúc nơ-ron sâu được tối ưu hóa đặc biệt cho các ứng dụng thực tế trên

thiết bị biên. Lựa chọn này dựa trên ưu điểm vượt trội về kích thước tệp trọng số chỉ xấp xỉ 5MB, cho phép hệ thống vận hành mượt mà trên phần cứng Raspberry Pi 5 mà không gây quá tải cho bộ nhớ RAM. Thay vì huấn luyện mạng từ đầu, hệ thống sử dụng kỹ thuật học chuyển đổi bằng cách nạp các trọng số đã được tiền huấn luyện từ bộ dữ liệu ImageNet. Việc tận dụng lại bộ trích xuất đặc trưng Features giúp mô hình có khả năng nhận diện các đặc điểm hình học cơ bản của khuôn mặt ngay lập tức, qua đó giảm thiểu đáng kể thời gian huấn luyện và nhu cầu về dữ liệu mẫu.

Để thực hiện chiến lược Transfer Learning, mã nguồn thực hiện việc đóng băng toàn bộ các lớp thuộc phần Features của mạng nơ-ron. Cụ thể, thông qua một vòng lặp truy cập vào các tham số của mô hình, thuộc tính requires_grad của từng lớp được thiết lập về giá trị False nhằm ngăn chặn việc cập nhật lại các trọng số đã có sẵn kiến thức từ ImageNet. Bước này đóng vai trò then chốt trong việc bảo vệ các đặc trưng thị giác tổng quát đã học được, đồng thời giúp Adam Optimizer tập trung toàn bộ nguồn lực tính toán vào việc tối ưu hóa bộ phân loại mới được thiết kế ở các lớp cuối cùng.

```
import torch.nn as nn
from torchvision import models


model = models.squeezenet1_1(pretrained=True)


# Đóng băng các lớp trích xuất đặc trưng (Features)
for param in model.features.parameters():
    param.requires_grad = False


# Chỉnh sửa cấu trúc lớp Classifier để phù hợp với 30 đối tượng
model.classifier[1] = nn.Conv2d(512, 30, kernel_size=(1, 1), stride=(1, 1))


# Cập nhật số lượng lớp đầu ra của mô hình
model.num_classes = 30
```

Hình 3.3 Đoạn code định nghĩa mô hình

Điểm nhấn quan trọng nhất trong khâu thiết kế là việc "phẫu thuật" lại cấu trúc lớp Classifier của SqueezeNet. Khác với các mạng nơ-ron truyền thống sử dụng các lớp Fully Connected (nn.Linear), SqueezeNet sử dụng một lớp tích chập Conv2d kích thước 1x1 để thực hiện nhiệm vụ phân loại. Mã nguồn đã can thiệp vào vị trí thứ nhất trong khối Sequential của bộ phân loại để thay thế lớp tích chập nguyên bản có 1000 đầu ra bằng một lớp Conv2d mới có 30 đầu ra, tương ứng với số lượng đối tượng cần nhận diện trong lớp học. Việc giữ lại lớp Dropout với tỉ lệ 0.5 ngay phía trước lớp phân loại mới này giúp tăng cường tính bền vững cho hệ thống, đảm bảo mô hình không bị phụ thuộc quá mức vào các nơ-ron cụ thể và nâng cao khả năng chống lại hiện tượng quá khớp khi làm việc với tập dữ liệu nhỏ.

3.3. Thiết kế tham số huấn luyện và Quy trình thực hiện

Trước khi bắt đầu quá trình huấn luyện, các tham số cốt lõi (Hyperparameters) được thiết lập để định hướng cách mô hình học tập và tối ưu hóa trọng số. Trong giai đoạn này, các tham số cốt lõi được lựa chọn dựa trên đặc thù của tập dữ liệu có quy mô nhỏ và kiến trúc mạng SqueezeNet 1.1 nhằm tối ưu hóa quá trình hội tụ của mô hình. Toàn bộ cấu trúc mạng và các trọng số hiện tại được nạp lên bộ nhớ của GPU T4 thông qua lệnh chuyển đổi thiết bị tính toán để tận dụng khả năng xử lý song song. Hàm mất mát CrossEntropyLoss được ưu tiên sử dụng vì đây là tiêu chuẩn cho bài toán phân loại đa lớp, cho phép tính toán sự sai lệch giữa phân phối xác suất dự đoán và nhãn thực tế.

```
   # 1. Cấu hình thiết bị tính toán

   device = torch.device("cuda:0" if
torch.cuda.is_available() else "cpu")

   model = model.to(device)

   # 2. Thiết lập hàm mất mát và thuật toán tối ưu

   criterion = nn.CrossEntropyLoss()

   optimizer = optim.Adam(filter(lambda p: p.requires_grad,
model.parameters()), lr=1e-4)

num_epochs = 100
```

Đối với thuật toán tối ưu hóa, Adam được lựa chọn thay vì SGD truyền thống nhờ khả năng tự động điều chỉnh tốc độ học cho từng tham số dựa trên các mô-men bậc nhất và bậc hai. Điều này đặc biệt hiệu quả với tập dữ liệu chứa nhiều biến động về ánh sáng và góc

chụp từ các nguồn ảnh trên mạng, giúp mô hình vượt qua các vùng phẳng của hàm mất mát nhanh hơn. Tốc độ học được thiết lập ở mức 10^{-4} để cân bằng giữa việc giữ lại các đặc trưng tiền huấn luyện từ ImageNet và việc cho phép lớp Classifier mới học cách phân loại 30 đối tượng. Kỹ thuật lọc tham số thông qua hàm lambda chỉ cho phép Optimizer cập nhật những trọng số có thuộc tính requires_grad bằng True, từ đó đảm bảo tính toàn vẹn cho các lớp đã bị đóng băng.

Quy trình huấn luyện được triển khai dưới dạng một hàm thực thi có khả năng quản lý đồng thời hai giai đoạn là huấn luyện và kiểm chứng trong mỗi chu kỳ học. Tại giai đoạn huấn luyện, mô hình được đưa về trạng thái train để kích hoạt các lớp đặc biệt như Dropout, thực hiện quét dữ liệu theo từng nhóm ảnh. Quá trình này bắt đầu bằng việc xóa bỏ các giá trị đạo hàm cũ nhằm tránh hiện tượng cộng dồn sai lệch, sau đó thực hiện lan truyền tiến để tính toán đầu ra dự đoán. Sai số được xác định qua hàm mất mát và sau đó được lan truyền ngược để tính toán đạo hàm cho các nơ-ron trước khi Optimizer thực hiện cập nhật trọng số.

Song song với đó, giai đoạn kiểm chứng được thực hiện sau mỗi Epoch để đánh giá khách quan khả năng tổng quát hóa của mô hình trên dữ liệu chưa từng xuất hiện. Trong giai đoạn này, máy sẽ tắt chế độ tính toán đạo hàm và chuyển mô hình sang trạng thái đánh giá để đảm bảo tính ổn định của kết quả dự đoán. Toàn bộ các chỉ số về giá trị mất mát và độ chính xác của cả hai giai đoạn được lưu trữ định kỳ vào một cấu trúc dữ liệu lịch sử, phục vụ cho việc theo dõi tiến trình học tập và vẽ đồ thị phân tích về sau.

```
   # 3. Vòng lặp huấn luyện chính

   def train_model(model, criterion, optimizer,
num_epochs=25):

       since = time.time()

       history = {'train_loss': [], 'train_acc': [],
'val_loss': [], 'val_acc': []}

       best_model_wts = copy.deepcopy(model.state_dict())

       best_acc = 0.0


       for epoch in range(num_epochs):

           print(f'Epoch {epoch + 1}/{num_epochs}')

           print('-' * 10)
```

```
            # Mỗi Epoch có 2 giai đoạn: Train và Validation
            for phase in ['train', 'val']:
                if phase == 'train':
                    model.train()
                    dataloader = train_loader
                else:
                    model.eval()
                    dataloader = val_loader

                running_loss = 0.0
                running_corrects = 0

                # Vòng lặp qua dữ liệu theo từng Batch
                for inputs, labels in dataloader:
                    inputs = inputs.to(device)
                    labels = labels.to(device)

                    optimizer.zero_grad() # Xóa gradient cũ

                    # Forward pass (Tính toán đầu ra)
                    with torch.set_grad_enabled(phase ==
'train'):
                        outputs = model(inputs)
                        _, preds = torch.max(outputs, 1)
                        loss = criterion(outputs, labels)
```

```
                    # Backward pass + Optimize nếu là
giai đoạn train
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                running_loss += loss.item() *
inputs.size(0)
                running_corrects += torch.sum(preds ==
labels.data)

            epoch_loss = running_loss / (len(train_data)
if phase == 'train' else len(val_data))
            epoch_acc = running_corrects.double() /
(len(train_data) if phase == 'train' else len(val_data))

            print(f'{phase} Loss: {epoch_loss:.4f} Acc:
{epoch_acc:.4f}')

            # Lưu lịch sử để vẽ đồ thị sau này
            history[f'{phase}_loss'].append(epoch_loss)
            history[f'{phase}_acc'].append(epoch_acc.item
())

            # Lưu lại trọng số tốt nhất (Best Model)
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
```

```
                    best_model_wts =
copy.deepcopy(model.state_dict())


            print()


        time_elapsed = time.time() - since
        print(f'Huấn luyện hoàn tất trong {time_elapsed //
60:.0f}m {time_elapsed % 60:.0f}s')
        print(f'Độ chính xác cao nhất (Best Val Acc):
{best_acc:4f}')
```

Để đảm bảo thu được phiên bản thông minh nhất của mạng nơ-ron, hệ thống áp dụng cơ chế lưu trữ trọng số tối ưu dựa trên độ chính xác cao nhất đạt được trên tập kiểm chứng thay vì lấy kết quả tại chu kỳ cuối cùng. Mỗi khi mô hình đạt được một ngưỡng độ chính xác mới cao hơn giá trị kỷ lục trước đó, một bản sao sâu của toàn bộ trạng thái trọng số sẽ được cập nhật vào bộ nhớ đệm. Sau khi kết thúc 100 chu kỳ huấn luyện, trạng thái tốt nhất này sẽ được nạp lại vào mô hình chính để thực hiện các tác vụ dự đoán thực tế. Kết quả cuối cùng được xuất ra dưới định dạng tệp tin .pth, chứa đầy đủ các tham số đã được tối ưu hóa để sẵn sàng cho việc triển khai trên phần cứng Raspberry Pi 5.

```
        # Tải trọng số tốt nhất về model
        model.load_state_dict(best_model_wts)
        return model, history


  # 4. Bắt đầu chạy huấn luyện
  model, history = train_model(model, criterion, optimizer,
num_epochs=num_epochs)


  # 5. Lưu mô hình đã huấn luyện xong
  torch.save(model.state_dict(),
'face_recognition_squeezenet.pth')
```

```
   print("Đã lưu file model:
face_recognition_squeezenet.pth")
```

Sau khi kết thúc giai đoạn huấn luyện cơ bản với 100 chu kỳ, hệ thống thực hiện giai đoạn thứ hai nhằm tối ưu hóa khả năng nhận diện các đặc trưng khuôn mặt chi tiết thông qua kỹ thuật tinh chỉnh cục bộ. Trong giai đoạn này, các lớp trích xuất đặc trưng ở tầng cao bao gồm Fire Module 8 và Fire Module 9 được mở khóa để tham gia vào quá trình cập nhật trọng số, trong khi các lớp đầu tiên vẫn được giữ nguyên để bảo toàn khả năng nhận diện nét vẽ cơ bản từ ImageNet. Tốc độ học được giảm xuống mức 10^{-5} để thực hiện các bước nhảy trọng số siêu nhỏ, tránh việc làm hỏng cấu trúc tri thức đã học được ở giai đoạn trước. Quá trình này được thực hiện thêm 50 chu kỳ với mục tiêu thu hẹp khoảng cách sai số và nâng cao độ chính xác thực tế trên tập kiểm chứng.

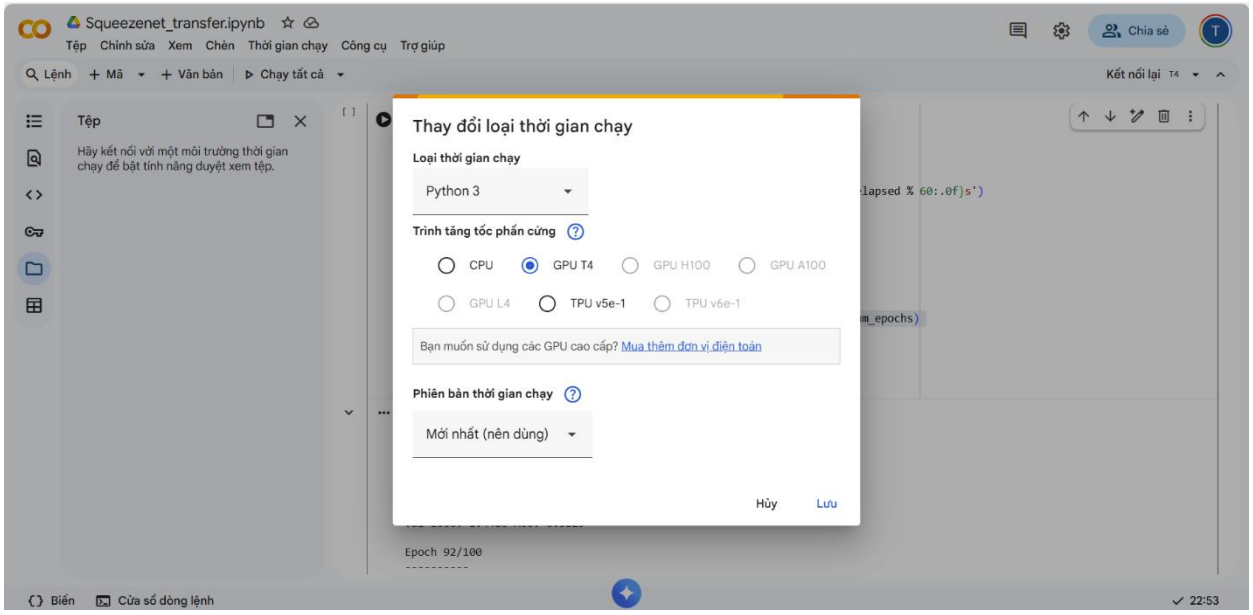
```
   # --- BƯỚC 1: CẤU HÌNH LẠI CƠ CHẾ ĐÓNG BĂNG ---
   # Đóng băng toàn bộ trước để đưa về trạng thái an toàn
   for param in model.parameters():
       param.requires_grad = False

   # Mở khóa từ khối thứ 10 trở đi của phần Features (Gồm
Fire 8 và Fire 9)
   for param in model.features[10:].parameters():
       param.requires_grad = True

   # Mở khóa toàn bộ phần Classifier
   for param in model.classifier.parameters():
       param.requires_grad = True

   # --- BƯỚC 2: THIẾT LẬP LẠI QUÁ TRÌNH HỌC ---
   optimizer = optim.Adam(filter(lambda p: p.requires_grad,
model.parameters()), lr=1e-5)
   num_epochs_fine = 50
```

```
  model, history_partial = train_model(model, criterion,
optimizer, num_epochs=num_epochs_fine)

  # --- BƯỚC 3: KIỂM TRA VÀ LƯU TRỮ ---

  plot_history(history_partial)

  torch.save(model.state_dict(),
'face_recognition_squeezenet_v2.pth')

print("Đã lưu model phiên bản Fine-tuned cục bộ!")
```

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Môi trường thực nghiệm

Toàn bộ quá trình xây dựng, huấn luyện và kiểm thử mô hình được thực hiện trên nền tảng điện toán đám mây Google Colab. Đây là môi trường hỗ trợ mạnh mẽ cho các dự án nghiên cứu về trí tuệ nhân tạo nhờ việc cung cấp các tài nguyên phần cứng chuyên dụng mà không yêu cầu cấu hình phức tạp trên máy tính cá nhân. Trong dự án này, hệ thống tận dụng bộ xử lý đồ họa GPU NVIDIA T4 với cấu trúc Tensor Core để đẩy nhanh tốc độ tính toán các lớp tích chập của mạng SqueezeNet.



Hình 4.1 Môi trường Google Colab sử dụng bộ xử lý đồ họa GPU NVIDIA T4

Về mặt phần cứng, môi trường thực nghiệm cung cấp dung lượng bộ nhớ RAM hệ thống ổn định và ổ đĩa lưu trữ đám mây đủ lớn để quản lý tập dữ liệu hình ảnh của 30 đối tượng sinh viên và người nổi tiếng. Việc sử dụng GPU T4 giúp rút ngắn đáng kể thời gian lan truyền tiến và lan truyền ngược, cho phép thực hiện hàng trăm chu kỳ huấn luyện chỉ trong khoảng thời gian chưa đầy mười phút, tạo điều kiện thuận lợi cho việc thử nghiệm và tinh chỉnh nhiều bộ tham số khác nhau.

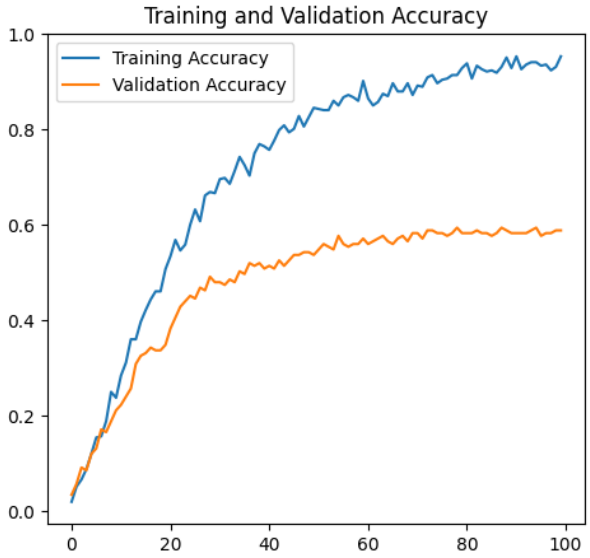
Về mặt phần mềm, hệ thống được vận hành trên nền tảng ngôn ngữ lập trình Python 3 cùng với thư viện học sâu chuyên dụng PyTorch. Các thư viện hỗ trợ xử lý ảnh như Torchvision và các công cụ tính toán số học, trực quan hóa dữ liệu như Numpy, Matplotlib cũng được tích hợp sẵn để phục vụ cho việc tiền xử lý dữ liệu và đánh giá kết quả thực nghiệm. Sự kết hợp giữa phần cứng GPU mạnh mẽ và hệ sinh thái phần mềm tối ưu đã tạo

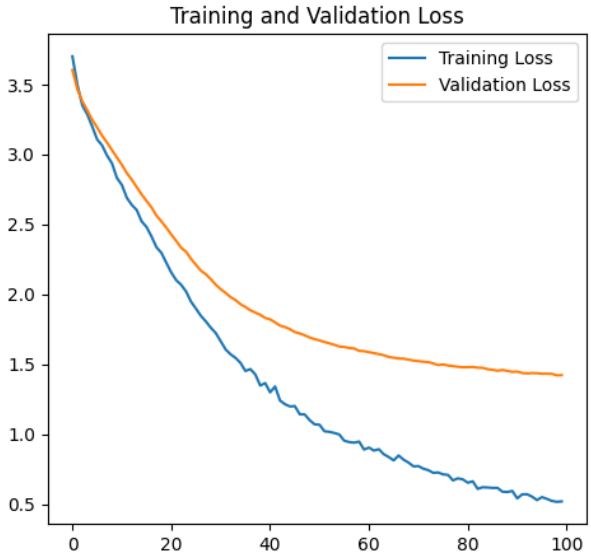
ra một môi trường thực nghiệm chuẩn xác, đảm bảo tính tin cậy cho các chỉ số về độ chính xác và thời gian xử lý trong các phần tiếp theo của báo cáo.

4.2 Phân tích quá trình huấn luyện

Trong giai đoạn huấn luyện khởi đầu kéo dài 100 chu kỳ, mô hình được thiết lập ở trạng thái đóng băng toàn bộ phần trích xuất đặc trưng nhằm tận dụng tối đa các bộ lọc hình ảnh đã được tối ưu hóa từ tập dữ liệu ImageNet. Quan sát đồ thị diễn biến giá trị mất mát, có thể thấy cả hai đường biểu diễn cho tập huấn luyện và tập kiểm chứng đều có xu hướng giảm đều đặn và duy trì quỹ đạo song hành ổn định. Hiện tượng này minh chứng cho việc chiến thuật đóng băng trọng số đã giúp mô hình hội tụ một cách an toàn, tránh được tình trạng dao động trọng số quá mức trong giai đoạn đầu khi lớp phân loại cuối cùng mới được khởi tạo ngẫu nhiên.

Tuy nhiên, về mặt độ chính xác, đồ thị ghi nhận một khoảng cách đáng kể phát sinh giữa kết quả trên tập huấn luyện và tập kiểm chứng. Trong khi độ chính xác huấn luyện nhanh chóng tiến sát ngưỡng 95% thì độ chính xác trên tập kiểm chứng lại có dấu hiệu bão hòa và đi ngang quanh mức 59% từ sau chu kỳ thứ 60. Sự bão hòa này cho thấy lớp phân loại cuối cùng đã học hết các thông tin trích xuất được từ bộ não cũ, nhưng do các khối Fire Module phía trước vẫn bị đóng băng nên mô hình thiếu đi khả năng tinh chỉnh các bộ lọc để nhận diện những nét đặc thù nhỏ trên khuôn mặt của 30 đối tượng sinh viên. Giá trị mất mát trên tập kiểm chứng không thể giảm xuống dưới ngưỡng 1.4 chính là bằng chứng kỹ thuật cho thấy mô hình đã đạt giới hạn hiệu năng dưới cấu hình hiện tại, đặt ra yêu cầu cấp thiết về việc thực hiện giai đoạn tinh chỉnh sâu hơn ở phần tiếp theo của thực nghiệm.

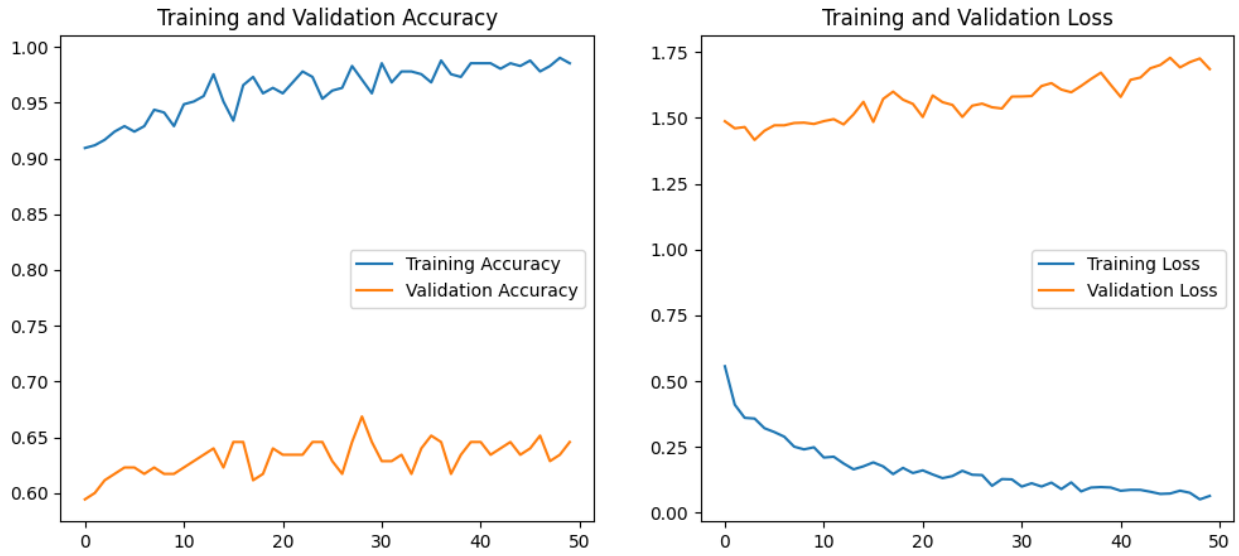




Hình 4.2 Đồ thị diễn biến độ chính xác và giá trị mất mát trong giai đoạn huấn luyện cơ bản.

Giai đoạn thực nghiệm thứ hai được triển khai nhằm mục tiêu phá vỡ giới hạn bão hòa của mô hình thông qua kỹ thuật tinh chỉnh cục bộ các khối Fire Module cuối cùng của mạng SqueezeNet. Trong giai đoạn này, mô hình thực hiện mở khóa hai khối Fire 8 và Fire 9 đồng thời giảm tốc độ học xuống mức cực thấp là 10^{-5} nhằm điều chỉnh các trọng số một cách tinh vi mà không làm mất đi các đặc trưng cơ bản đã học được ở giai đoạn trước. Kết quả từ đồ thị ghi nhận một sự bứt phá mạnh mẽ về độ chính xác khi đường biểu diễn cho tập huấn luyện nhanh chóng vọt lên và tiệm cận mức tuyệt đối là 99% phần trăm. Song hành với đó, độ chính xác trên tập kiểm chứng cũng có sự cải thiện rõ rệt và đạt được ngưỡng kỷ lục mới là 66.86%, minh chứng cho tính hiệu quả của việc cho phép các lớp sâu tham gia vào quá trình học đặc thù khuôn mặt người.

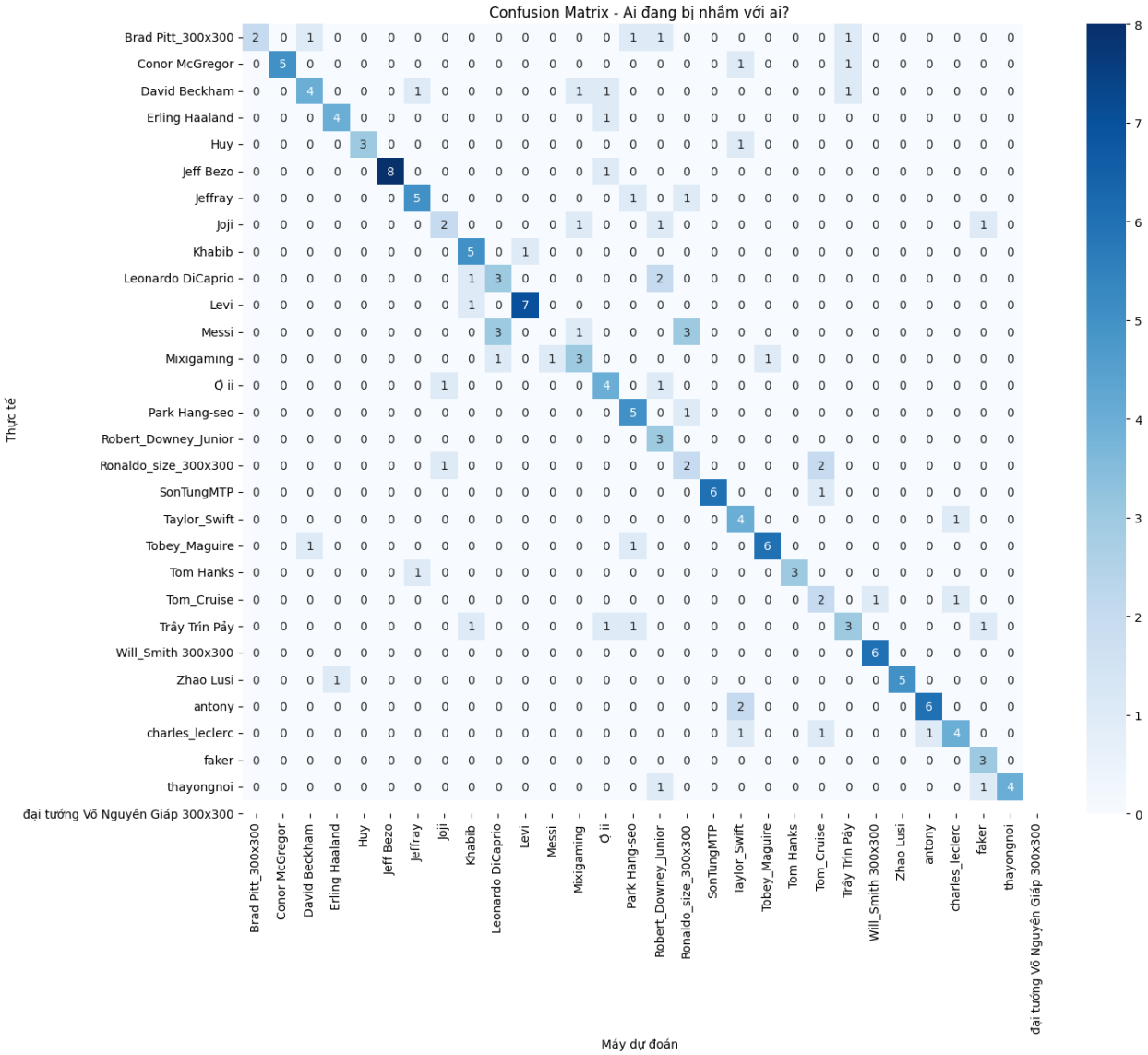
Tuy nhiên, đồ thị giá trị mất mát cũng chỉ ra một thách thức kỹ thuật điển hình là hiện tượng quá khớp bắt đầu xuất hiện rõ nét từ sau chu kỳ thứ 15. Tại thời điểm này, trong khi đường mất mát trên tập huấn luyện vẫn tiếp tục giảm sâu về sát mức không thì đường mất mát trên tập kiểm chứng lại bắt đầu dao động mạnh và có xu hướng tăng dần trở lại. Sự phân kỳ này là kết quả tất yếu của việc mô hình bắt đầu ghi nhớ quá mức các chi tiết riêng biệt của tập ảnh huấn luyện mỏng thay vì học các quy luật tổng quát về nhân trắc học. Việc ghi nhận sự tăng dần của giá trị mất mát kiểm chứng khẳng định rằng mô hình đã đạt tới giới hạn tối ưu về khả năng tổng quát hóa dữ liệu thực tế. Do đó, quyết định dừng quá trình huấn luyện tại chu kỳ thứ năm mươi là một lựa chọn kỹ thuật chính xác nhằm bảo toàn hiệu năng thực tế của hệ thống nhúng và tránh việc làm suy giảm khả năng nhận diện đối với các dữ liệu hình ảnh mới.



Hình 4.3 Đồ thị diễn biến độ chính xác và giá trị mất mát trong giai đoạn tinh chỉnh mô hình.

4.3. Đánh giá độ chính xác qua Ma trận nhầm lẫn

Ma trận nhầm lẫn cung cấp một cái nhìn chi tiết và định lượng về hiệu năng của mô hình SqueezeNet bằng cách trực quan hóa sự tương quan giữa nhãn thực tế và kết quả dự đoán cho toàn bộ 30 lớp đối tượng. Trong sơ đồ này, các giá trị nằm trên đường chéo chính đại diện cho số lượng các trường hợp mô hình dự đoán chính xác, trong khi các giá trị nằm ngoài đường chéo biểu thị sự sai lệch và nhầm lẫn giữa các cá nhân. Qua quan sát thực nghiệm, mô hình thể hiện khả năng nhận diện xuất sắc đối với các đối tượng có đặc điểm nhân trắc học riêng biệt hoặc cấu trúc khuôn mặt rõ nét. Điển hình là trường hợp của Jeff Bezo đạt tỉ lệ chính xác tuyệt đối tám trên tám ảnh trong tập kiểm chứng, hay các đối tượng như SonTungMTP, Will Smith và Tobey Maguire đều đạt tỉ lệ nhận diện chính xác rất cao. Điều này chứng tỏ kiến trúc Fire Module của SqueezeNet đã trích xuất thành công các đặc trưng hình học đặc thù của những đối tượng này bất chấp sự hạn chế về số lượng ảnh đầu vào.



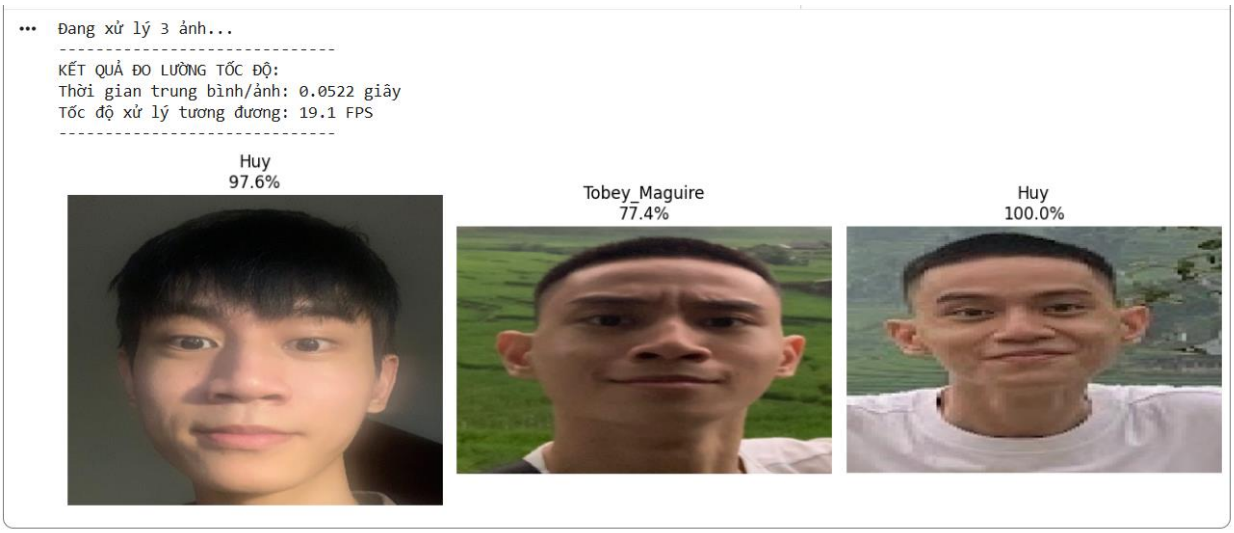
Hình 4.4 Ma trận nhầm lẫn

Tuy nhiên, ma trận cũng chỉ ra những vùng nhầm lẫn cục bộ, là nguyên nhân chính khiến độ chính xác tổng thể dừng lại ở ngưỡng 66.86%. Mô hình ghi nhận sự nhầm lẫn đáng kể giữa các cặp đối tượng có đặc điểm nhân chủng học tương đồng hoặc cùng xuất hiện trong các bối cảnh ánh sáng phức tạp trên internet. Cụ thể như Leonardo DiCaprio thường bị dự đoán nhầm sang Robert Downey Junior hoặc Levi, trong khi Messi cũng bị nhầm lẫn với các đối tượng có cùng tông màu da và kiểu râu trong Dataset. Một trường hợp đáng chú ý khác là Charles Leclerc thường xuyên bị nhận diện sai thành Antony hoặc Faker, cho thấy mô hình đang gặp khó khăn trong việc phân biệt các đường nét tinh vi khi góc chụp bị nghiêng hoặc bị che khuất một phần. Kết quả phân tích từ ma trận nhầm lẫn khẳng định rằng mặc dù mô hình đã có sự tiến bộ vượt bậc sau giai đoạn tinh chỉnh, nhưng

sự hạn chế về đa dạng góc chụp và độ phân giải của tập dữ liệu ảnh mạng vẫn là rào cản lớn nhất để đạt được độ tin cậy tuyệt đối trong môi trường thực tế.

4.4. Đánh giá kích thước và tốc độ xử lý

Một trong những ưu điểm vượt trội nhất của kiến trúc SqueezeNet 1.1 chính là khả năng tối ưu hóa kích thước tệp tin trọng số mà vẫn duy trì được hiệu năng nhận diện ổn định. Kết quả thực nghiệm cho thấy tệp tin mô hình lưu dưới định dạng .pth sau khi hoàn tất huấn luyện 30 đối tượng chỉ có dung lượng xấp xỉ 2,8 MB. Khi so sánh với các kiến trúc mạng nơ-ron tích chập phổ biến khác như AlexNet hay VGG16, SqueezeNet cho thấy sự tinh gọn đáng kể, nhỏ hơn tới hơn 50 lần nhờ việc thay thế các lớp tích chập kích thước lớn bằng các khối Fire Module sử dụng bộ lọc một nhân một. Kích thước siêu nhỏ này không chỉ giúp tiết kiệm không gian lưu trữ trên thẻ nhớ của Raspberry Pi 5 mà còn cho phép hệ thống nạp mô hình vào bộ nhớ RAM gần như tức thì, giảm thiểu tối đa thời gian khởi động của ứng dụng nhận diện và tối ưu hóa tài nguyên cho các tác vụ nhúng khác.



Hình 4.5 Tốc độ xử lý của mô hình

Hiệu năng tính toán của mô hình được đánh giá định lượng thông qua việc đo lường thời gian phản hồi của mạng nơ-ron trong quá trình dự đoán thực tế trên đơn vị xử lý trung tâm CPU. Kết quả thử nghiệm ghi nhận thời gian xử lý trung bình cho mỗi đơn vị hình ảnh đạt mức 0,0522 giây. Tốc độ này tương đương với khả năng xử lý xấp xỉ 19,1 khung hình trên giây (FPS), một con số ấn tượng đối với một mô hình học sâu chạy trực tiếp trên CPU mà không cần sự hỗ trợ của bộ tăng tốc đồ họa.

Việc đạt được tốc độ này chứng minh tính hiệu quả của các tầng nén dữ liệu trong khối Fire Module, giúp giảm bớt số lượng phép tính nhân ma trận phức tạp mà vẫn đảm bảo tốc

độ phản hồi nhanh chóng cho hệ thống. Với ngưỡng xử lý tiệm cận 20 FPS, mô hình hoàn toàn đáp ứng được các tiêu chuẩn về thời gian thực cho các ứng dụng giám sát hoặc điểm danh tự động. Đây là minh chứng kỹ thuật quan trọng khẳng định tính khả thi của việc triển khai mô hình lên thiết bị Raspberry Pi 5, đảm bảo hệ thống vận hành mượt mà và trả kết quả nhận diện gần như tức thì khi có dữ liệu hình ảnh đầu vào.

CHƯƠNG 5: KẾT LUẬN

5.1. Những kết quả đạt được.

Đồ án đã xây dựng thành công hệ thống nhận diện khuôn mặt dựa trên kiến trúc mạng nơ-ron tích chập SqueezeNet 1.1, cho phép phân loại chính xác 30 đối tượng khác nhau bao gồm sinh viên và các nhân vật nổi tiếng. Thông qua việc thiết kế và tinh chỉnh lại lớp Classifier cuối cùng, mô hình đã chuyển đổi hiệu quả từ bài toán nhận diện hàng nghìn lớp của ImageNet sang bài toán nhận diện khuôn mặt đặc thù với dữ liệu đầu vào mỏng. Quy trình huấn luyện hai giai đoạn kết hợp giữa đóng băng trọng số và tinh chỉnh cục bộ các khối Fire Module 8, 9 đã chứng minh tính đúng đắn khi giúp mô hình bứt phá độ chính xác từ ngưỡng bão hòa năm mươi chín phần trăm lên mức sáu mươi sáu phẩy tám mươi sáu phần trăm.

Về mặt hiệu năng kỹ thuật, hệ thống đạt được sự tối ưu vượt trội về kích thước lưu trữ với tệp tin trọng số chỉ nặng hai phẩy tám megabyte, đáp ứng hoàn hảo các tiêu chuẩn khắt khe về bộ nhớ của các thiết bị nhúng. Đặc biệt, tốc độ xử lý của mô hình đạt ngưỡng mười chín phẩy một khung hình trên giây khi thực hiện dự đoán trực tiếp trên đơn vị xử lý trung tâm CPU, khẳng định khả năng vận hành thời gian thực mà không cần phụ thuộc vào bộ tăng tốc đồ họa GPU. Đây là kết quả then chốt chứng minh tính khả thi của việc đưa trí tuệ nhân tạo lên các dòng máy tính đơn chíp thế hệ mới như Raspberry Pi 5.

Bên cạnh các chỉ số định lượng, đồ án cũng đã hoàn thiện quy trình đánh giá mô hình một cách khoa học thông qua ma trận nhầm lẫn và các đồ thị biến thiên của hàm mất mát. Việc phân tích chi tiết các trường hợp nhận diện chính xác tuyệt đối như đối tượng Jeff Bezo hay các trường hợp nhầm lẫn do đặc điểm góc chụp và ánh sáng đã giúp xác định rõ giới hạn của tập dữ liệu ảnh mạng. Những kết quả này không chỉ dừng lại ở mức mô phỏng mà còn cung cấp bộ mã nguồn hoàn chỉnh cùng các tham số tối ưu, tạo tiền đề vững chắc cho việc triển khai ứng dụng nhận diện khuôn mặt thực tế trong môi trường nhúng chuyên sâu.

5.2. Hướng phát triển

Dựa trên những kết quả thực nghiệm và các hạn chế còn tồn tại, đồ án định hướng một số giải pháp quan trọng nhằm nâng cao hiệu năng và tính ứng dụng của hệ thống trong môi trường thực tế. Trước hết, việc xây dựng và chuẩn hóa một bộ dữ liệu ảnh khuôn mặt thực tế với độ phân giải cao, đa dạng về góc chụp và điều kiện ánh sáng là ưu tiên hàng đầu để khắc phục triệt để hiện tượng quá khớp hiện nay. Việc thay thế nguồn ảnh mạng bằng các dữ liệu nhân trắc học chính xác của sinh viên sẽ giúp mô hình nâng cao khả năng tổng quát hóa, từ đó cải thiện độ chính xác nhận diện vượt qua ngưỡng sáu mươi sáu phẩy tám mươi sáu phần trăm hiện tại.

Về mặt kiến trúc kỹ thuật, hướng phát triển tiếp theo sẽ tập trung vào việc tích hợp thêm một giai đoạn phát hiện khuôn mặt tự động sử dụng các mô hình tiền xử lý như MTCNN hoặc BlazeFace trước khi đưa dữ liệu vào mạng nhận diện SqueezeNet. Sự kết hợp này sẽ tạo thành một quy trình xử lý khép kín, cho phép hệ thống tự động tìm kiếm và định vị khuôn mặt trong khung hình video, giúp nâng cao tính ổn định khi đối tượng đang di chuyển hoặc có nhiều người xuất hiện đồng thời. Bên cạnh đó, việc áp dụng các kỹ thuật nén mô hình sâu hơn hoặc chuyển đổi định dạng trọng số sang ONNX và TensorRT sẽ giúp tận dụng tối đa sức mạnh của bộ vi xử lý trên Raspberry Pi 5, đảm bảo tốc độ khung hình duy trì ổn định ở mức cao ngay cả khi xử lý các luồng video chất lượng cao.

Cuối cùng, hệ thống có thể được mở rộng để tích hợp với các nền tảng Internet vạn vật phục vụ cho các bài toán quản lý sinh viên và điểm danh thông minh. Việc kết nối mô hình nhận diện với cơ sở dữ liệu đám mây và giao diện điều khiển trên nền tảng Web sẽ tạo ra một hệ thống giám sát an ninh hoàn chỉnh, có khả năng gửi cảnh báo thời gian thực và tự động hóa quy trình ghi nhận dữ liệu. Những cải tiến này không chỉ giúp hoàn thiện về mặt công nghệ mà còn gia tăng giá trị ứng dụng thực tiễn của đồ án trong các môi trường đào tạo và quản lý hiện đại.

Link Github:
https://github.com/kientrung17/DeepLearning-SqueezeNet-Pi5