HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Võ Minh Mạnh

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI - 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Võ Minh Mạnh

ỨNG DỤNG MẠNG NƠ-RON NHÂN TẠO VÀ PHÉP BIẾN ĐỔI FOURIER ĐỂ PHÁT HIỆN HÌNH ẢNH THẺ CĂN CƯỚC GIẢ MẠO

CHUYÊN NGÀNH : KHOA HỌC MÁY TÍNH

MÃ SỐ: 8.48.01.01

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN TẤT THẮNG

HÀ NỘI - 2025

LỜI CAM ĐOAN

Tôi cam đoan rằng đề án: **"Ứng dụng mạng nơ-ron nhân tạo và phép biến đổi Fourier để phát hiện hình ảnh thẻ căn cước giả mạo"** là công trình nghiên cứu của chính tôi. Những kết quả nghiên cứu được trình bày trong đề án này là công trình của riêng tôi dưới sự hướng dẫn của TS. Nguyễn Tất Thắng, giảng viên khoa Công nghệ thông tin 1, Học viện Công nghệ Bưu chính Viễn thông.

Tôi cam đoan các số liệu, kết quả nêu trong đề án này là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác. Không có sản phẩm, nghiên cứu nào của người khác được sử dụng trong đề án này mà không được trích dẫn theo đúng quy định.

Hà Nội, ngày tháng năm

Người thực hiện đề án

Võ Minh Mạnh

LỜI CẢM ƠN

Để hoàn thành được đề án tốt nghiệp thạc sĩ, bên cạnh sự nghiên cứu và phát triển của bản thân, là sự giúp đỡ nhiệt tình của các thầy cô giáo. Đầu tiên em xin gửi lời cảm ơn đến các thầy cô giáo trong khoa Công nghệ thông tin 1 nói riêng và Học viện Công nghệ Bưu chính Viễn thông nói chung. Các thầy cô là những người đã truyền tải cho em rất nhiều kiến thức quý báu học suốt hai năm học vừa qua, những kiến thức đó là nền móng vững chắc để em xây dựng đề án tốt nghiệp này.

Tiếp theo, em xin chân thành cảm ơn giảng viên hướng dẫn TS. Nguyễn Tất Thắng đã định hướng và giúp đỡ em tận tình trong suốt quá trình thực hiện đồ án này.

Cuối cùng, xin cảm ơn thầy cô và độc giả đã dành thời gian quý báu để nghiên cứu, phản biện đồ án tốt nghiệp. Những nhận xét, đóng góp của thầy cô và độc giả dù nhỏ nhất cũng rất quan trọng và có giá trị.

Chúc các thầy cô và độc giả nhiều sức khỏe!

MỤC LỤC

LỜI CAM ĐOAN i
LỜI CẢM ƠN ii
MỤC LỤC iii
DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT v
DANH MỤC BẢNG vi
DANH MỤC HÌNH ẢNH vii
MỞ ĐẦU 1
1. Lý do chọn đề tài 1
2. Tổng quan về vấn đề nghiên cứu 2
3. Mục đích nghiên cứu 3
4. Đối tượng và phạm vi nghiên cứu 4
4.1. Đối tượng nghiên cứu 4
4.2. Phạm vi nghiên cứu 4
5. Phương pháp nghiên cứu 4
CHƯƠNG 1 - TỔNG QUAN VỀ VẤN ĐỀ HÌNH ẢNH THẺ CĂN CƯỚC GIẢ MẠO 5
1.1. Giới thiệu vấn đề phát hiện hình ảnh thẻ căn cước giả mạo. 5
1.2. Các nghiên cứu liên quan 5
1.3. Giải pháp giải quyết vấn đề của đề án 8
1.4. Kết luận 9
CHƯƠNG 2 - PHƯƠNG PHÁP PHÁT HIỆN HÌNH ẢNH THẺ CĂN CƯỚC GIẢ MẠO 10
2.1. Giới thiệu về phép biến đổi Fourier 10
2.1.1. Phép biến đổi Fourier rời rạc hai chiều. 10
2.1.2. Nguyên lý và cách thực hiện lọc tần số bằng phép biến đổi Fourier trên ảnh. 11
2.2. Giới thiệu về mô hình mạng nơ-ron nhân tạo VGG16 15
2.2.1. Giới thiệu về học sâu, mạng nơ-ron nhân tạo. 15

2.2.2. Giới thiệu về mạng nơ-ron tích chập 19
2.2.3. Mô hình mạng nơ-ron VGG16.....21
2.3. Giới thiệu về mô-đun tự chú ý 24
2.4. Giới thiệu về mô hình VGG16 kết hợp phép biến đổi Fourier. 26
2.5. Kết luận 28
CHƯƠNG 3 - XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH.....29
3.1. Phân tích bài toán phát hiện thẻ căn cước giả mạo 29
3.2. Xây dựng tập dữ liệu.....30
3.3. Xây dựng kiến trúc mạng học sâu.....34
3.3.1. Kiến trúc tổng quan của mạng 34
3.3.2. Mô-đun tiền xử lý: phép biến đổi Fourier rời rạc nhanh kết hợp cơ chế lọc tần số cao.....36
3.3.3. Mô-đun mạng nơ-ron tích chập khung sử dụng kiến trúc VGG16.....39
3.3.4. Mô-đun chú ý chéo đa tầng.....40
3.3.5. Mô-đun phân loại 43
3.4. Hàm mất mát và quá trình tối ưu hóa.....44
3.5. Huấn luyện và đánh giá mô hình. 45
3.5.1. Cài đặt và huấn luyện mô hình.....45
3.5.2. Đánh giá mô hình. 47
3.6. Kết luận 51
KẾT LUẬN.....52
TÀI LIỆU THAM KHẢO.....54
PHỤ LỤC I 58

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

Từ viết tắt	Nghĩa Tiếng Anh	Nghĩa Tiếng Việt
EER	Equal Error Rate	Tỉ lệ lỗi cân bằng
ACER	Average Classification Error Rate	Tỉ lệ lỗi phân loại trung bình
HTER	Half Total Error Rate	Nửa tổng tỷ lệ lỗi
APCER	Attack Presentation Classification Error Rate	Tỷ lệ phân loại sai ảnh giả
BPCER	Bona Fide Presentation Classification Error Rate	Tỷ lệ phân loại sai ảnh thật
FFT	Fast Fourier Transform	Phép biến đổi Fourier nhanh
IFFT	Inverse Fast Fourier Transform	Phép biến đổi Fourier ngược
VGG16+FFT	VGG16 artificial neural network incorporating Fourier transform	Mạng nơ-ron nhân tạo VGG16 kết hợp phép biến đổi Fourier.
tensor		một đối tượng toán học gồm một mảng các thành phần là hàm của tọa độ trong không gian.
LCD	Liquid Crystal Display	Màn hình tinh thể lỏng
LED	Light Emitting Diode	Màn hình Đi-ốt phát quang
MSE	Mean squared error	Sai số toàn phương trung bình

DANH MỤC BẢNG

Bảng 2.1: Các cấu hình khác nhau của mạng nơ-ron tích chập theo Simonyan và Zisserman 22
Bảng 2.2: Chi tiết cấu trúc trong mạng nơ-ron VGG16.....23
Bảng 3.1: Thông tin về tập dữ liệu huấn luyện và kiểm thử.....33
Bảng 3.2: Chi tiết cấu trúc mạng nơ-ron tích chập khung sử dụng kiến trúc VGG16 39
Bảng 3.3: Kết quả đo trên mô hình DLC-2021 cơ sở, Bộ dò nhiễu theo biên và VGG16+FFT 47
Bảng 3.4: Kết quả thời gian thực thực thi trung bình trên CPU Intel Xeon @ 2.20 GHz (x86-64), 6 nhân / 12 luồng 48
Bảng 3.5: Đánh giá kết quả các nghiên cứu loại bỏ thành phần 50

DANH MỤC HÌNH ẢNH

Hình 1.1: Minh họa quá trình định danh điện tử bằng thẻ căn cước thật (bên trái) và thẻ căn cước giả mạo (bên phải) 5
Hình 2.1: Ảnh xám (bên phải) thu được sau khi chuyển đổi từ ảnh màu (bên trái) .12
Hình 2.2: Ảnh xám gốc. 13
Hình 2.3: Ảnh phổ tần số Fourier.....13
Hình 2.4: Ảnh phổ tần số Fourier đã dịch chuyển tần số 0 về tâm 13
Hình 2.5: Ảnh phổ tần số Fourier ban đầu 14
Hình 2.6: Mặt nạ tần số cao với bán kính bằng 20. 14
Hình 2.7: Ảnh phổ tần số Fourier sau khi áp dụng mặt nạ 14
Hình 2.8: Ảnh phổ tần số Fourier sau khi đã áp dụng mặt nạ.....15
Hình 2.9: Ảnh phổ tần số Fourier khi dịch ngược 15
Hình 2.10: Ảnh phổ tần số sau khi biến đổi Fourier rời rạc ngược 15
Hình 2.11: Các sọc vân màn hình được làm nổi bật với bộ lọc tần số cao với mặt nạ có bán kính bằng 8 27
Hình 3.1: Một số hình ảnh có nhãn “ảnh thật”.....33
Hình 3.2: Một số hình ảnh có nhãn “ảnh giả”.....33
Hình 3.3: Sơ đồ tổng thể về mô hình mạng VGG16 kết hợp phép biến đổi Fourier 34
Hình 3.4: Minh họa các lớp trong mô-đun chú ý chéo đa tầng.....40
Hình 3.5: Minh họa các lớp trong mô-đun phân loại 43
Hình 3.6: Biểu đồ thể hiện giá trị của hàm mất mát của tập dữ liệu huấn luyện (trái) và tập dữ liệu kiểm tra (phải) trong quá trình huấn luyện mạng.....46

MỞ ĐẦU

1. Lý do chọn đề tài

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ trên toàn cầu, việc sử dụng giấy tờ tùy thân như giấy chứng minh nhân dân, thẻ căn cước công dân, thẻ căn cước, hộ chiếu để xác thực danh tính trên môi trường điện tử đã trở thành một xu hướng phổ biến. Các hệ thống định danh điện tử hiện nay đang được ứng dụng trong nhiều lĩnh vực thiết yếu của cuộc sống như ngân hàng số, ví điện tử, thương mại điện tử và các dịch vụ công trực tuyến. Theo Hoàng Lan và Hải Vân (2023) [1], đã có khoảng 11,9 triệu tài khoản ngân hàng được tạo thông qua hình thức định danh điện tử và hầu hết các ngân hàng tại Việt Nam đều đã triển khai quy trình này. Thay vì yêu cầu người dùng phải đến trực tiếp các điểm giao dịch, hệ thống định danh điện tử cho phép thực hiện toàn bộ quy trình xác thực từ xa thông qua các thiết bị điện tử có kết nối Internet như điện thoại thông minh hoặc máy tính. Việc định danh điện tử nhanh chóng, tiện lợi và tiết kiệm chi phí hơn nhiều so với phương thức truyền thống như gặp mặt trực tiếp hay đối chiếu hồ sơ giấy. Điều này giúp nâng cao trải nghiệm người dùng và góp phần thúc đẩy quá trình số hóa dịch vụ, mở rộng khả năng tiếp cận đến nhiều đối tượng ở khu vực vùng sâu, vùng xa. Ngoài ra, định danh điện tử còn đóng vai trò quan trọng trong việc tăng cường tính minh bạch, nâng cao hiệu quả quản lý và kiểm soát rủi ro trong các giao dịch số. Các công nghệ như nhận diện khuôn mặt, trích xuất ký tự quang học và phát hiện giả mạo hình ảnh đang được ứng dụng rộng rãi nhằm đảm bảo tính chính xác và an toàn cho quy trình định danh điện tử.

Tuy nhiên, bên cạnh những lợi ích kể trên thì quy trình định danh điện tử cũng tiềm ẩn nhiều rủi ro về an toàn bảo mật và gian lận. Theo Phạm Vinh (2024) [2], chỉ trong năm 2023, 8.000 - 10.000 tỉ đồng là tổng số tiền người dân bị các đối tượng lừa đảo chiếm đoạt trên không gian mạng, trong đó các vụ lừa đảo liên quan đến lĩnh vực tài chính chiếm đến 91%. Đây là con số báo động, phản ánh mức độ phức tạp và tinh vi của các hành vi lừa đảo trực tuyến. Kẻ gian thường sử dụng các tài khoản ngân hàng giả mạo để thực hiện hành vi chiếm đoạt tài sản. Sau khi nhận được tiền từ nạn

nhân, chúng nhanh chóng chuyển khoản qua nhiều tài khoản khác nhau nhằm xóa dấu vết, gây khó khăn lớn cho lực lượng chức năng trong quá trình điều tra và truy vết dòng tiền. Các đối tượng xấu thường lợi dụng những lỗ hổng trong quy trình kiểm tra và xác thực giấy tờ để thực hiện hành vi giả mạo, nhằm vượt qua hệ thống định danh tự động. Mục tiêu của các hành vi này là giả mạo danh tính của nạn nhân, để truy cập trái phép vào các dịch vụ tài chính hoặc chiếm đoạt thông tin cá nhân phục vụ cho mục đích lừa đảo. Do đó, việc tăng cường năng lực phát hiện các hình thức giả mạo trên môi trường định danh điện tử đang trở thành một yêu cầu cấp thiết trong quá trình triển khai hệ thống định danh điện tử. Các giải pháp kết hợp giữa thị giác máy tính, xử lý ảnh số và trí tuệ nhân tạo đang được nghiên cứu và ứng dụng nhằm nâng cao độ chính xác, giảm thiểu rủi ro và bảo vệ các hệ thống khỏi các mối đe dọa ngày càng phức tạp.

Hiện nay, các công nghệ trí tuệ nhân tạo như mạng nơ-ron nhân tạo ngày càng phổ biến. Ngoài ra, phép biến đổi Fourier cũng đã được chứng minh là công cụ mạnh mẽ trong xử lý tín hiệu số và hình ảnh, giúp phân tích cấu trúc tần số và phát hiện các dấu hiệu bất thường trong hình ảnh [3]. Việc kết hợp hai công cụ này với nhau và ứng dụng vào việc phòng chống gian lận trên môi trường điện tử sẽ có thể giúp giảm thiểu các rủi ro về tài chính cho người dân. Chính vì vậy, việc nghiên cứu và thực nghiệm đề tài "Ứng dụng mạng nơ-ron nhân tạo và phép biến đổi Fourier để phát hiện hình ảnh thẻ căn cước giả mạo" nhằm áp dụng các mô hình học máy tiên tiến, kết hợp với phép biến đổi Fourier trong Xử lý ảnh để phát hiện những dấu hiệu giả mạo trong hình ảnh thẻ căn cước, làm cơ sở để cải tiến hệ thống định danh điện tử.

2. Tổng quan về vấn đề nghiên cứu

Các tổ chức cần định danh điện tử người dùng sẽ phát triển các ứng dụng trên thiết bị di động. Quy trình định danh điện tử sẽ thường sẽ trải qua ba bước. Bước một: thu nhận ảnh giấy tờ tùy thân: hệ thống yêu cầu người dùng đặt giấy tờ tùy thân lên mặt phẳng, sau đó chụp giấy tờ tùy thân đó bằng camera của thiết bị (bắt buộc chụp trực tiếp khi dùng ứng dụng, không cho phép người dùng tải ảnh có sẵn lên). Hệ thống

kiểm tra xem ảnh đó có phải là ảnh thật hay không. Ảnh thật được hiểu là ảnh được người dùng chụp trực tiếp giấy tờ tùy thân. Khi ảnh đầu vào hợp lệ, người dùng sẽ được chuyển qua bước hai: kiểm tra sinh trắc học khuôn mặt. Quá trình này có lôgic hoạt động tương tự như bước thứ nhất – yêu cầu người dùng trực tiếp quét khuôn mặt của mình bằng camera sau đó kiểm tra tính chân thực của bức ảnh đã chụp. Bước ba, trích xuất thông tin quang học từ giấy tờ tùy thân và hình ảnh khuôn mặt nếu cả hai bước trên đều hợp lệ. Đề tài tập trung vào bài toán phát hiện hình ảnh giấy tờ tùy thân không phải là ảnh thật, cụ thể là phát hiện các ảnh thẻ căn cước được chụp gián tiếp qua màn hình thiết bị điện tử khác. Việc nghiên cứu được tiến hành qua hai nghiên cứu thành phần là **(i)** nghiên cứu các ưu điểm, nhược điểm khi sử dụng mạng nơ-ron nhân tạo VGG16, phép biến đổi Fourier và các bộ lọc tần số trong việc phát hiện các yếu tố giả mạo trong thẻ căn cước và **(ii)** nghiên cứu, xây dựng và đánh giá tính khả thi, tính hiệu quả một mạng nơ-ron nhân tạo kết hợp giữa VGG16 và phép biến đổi Fourier để phát hiện các dấu hiệu giả mạo của hình ảnh thẻ căn cước. Trên cơ sở đó, đề án tốt nghiệp được trình bày theo ba chương như sau:

Chương 1: Tổng quan về vấn đề hình ảnh thẻ căn cước giả mạo.

Chương 2: Phương pháp phát hiện hình ảnh thẻ căn cước giả mạo

Chương 3: Xây dựng và đánh giá mô hình

3. Mục đích nghiên cứu

Mục đích của nghiên cứu là thử nghiệm và đánh giá mức độ hiệu quả của việc sử dụng mạng nơ-ron nhân tạo kết hợp với phép biến đổi Fourier trong phát hiện hình ảnh thẻ căn cước giả mạo. Để hoàn thành mục tiêu này, nghiên cứu sẽ tiến hành các nhiệm vụ sau:

- Nghiên cứu đặc trưng của hình ảnh thẻ căn cước và các dấu hiệu nhận biết giữa thẻ thật và thẻ giả.

- Xây dựng bộ dữ liệu hình ảnh thẻ căn cước thật và giả phục vụ cho quá trình huấn luyện và kiểm thử mô hình.

- Nghiên cứu chi tiết mô hình mạng nơ-ron nhân tạo VGG16 và cách thức nó có thể được áp dụng trong nhận dạng hình ảnh.

- Nghiên cứu phép biến đổi Fourier và ứng dụng của nó trong xử lý ảnh, đặc biệt là trong việc phát hiện những bất thường về tần số của hình ảnh.

- Tìm hiểu cách kết hợp giữa mạng nơ-ron nhân tạo VGG16 và phép biến đổi Fourier để tối ưu hóa quá trình phát hiện hình ảnh giả mạo.

- Xây dựng mô hình, tiến hành huấn luyện mạng nơ-ron, và đánh giá kết quả qua các tiêu chí như độ chính xác và hiệu suất của mô hình

Nghiên cứu này hướng đến việc cung cấp một giải pháp mới trong việc chống giả mạo giấy tờ tùy thân khi thực hiện định danh điện tử, góp phần nâng cao tính bảo mật cho các hệ thống này.

4. Đối tượng và phạm vi nghiên cứu

4.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là các hình ảnh giấy tờ tùy thân, bao gồm: giấy chứng minh nhân dân, thẻ căn cước công dân, thẻ căn cước, hộ chiếu bao gồm cả hình ảnh thật và hình ảnh giả mạo.

4.2. Phạm vi nghiên cứu

a. Phạm vi không gian: Bộ dữ liệu mã nguồn mở hình ảnh thẻ căn cước công khai trên Internet khoảng 72.793 ảnh.

b. Phạm vi thời gian: Từ tháng 8 năm 2024 đến tháng 11 năm 2025.

5. Phương pháp nghiên cứu

Sử dụng phương pháp nghiên cứu định lượng, bao gồm thu thập các thông tin và dữ liệu dưới dạng số học, số liệu có tính chất thống kê để có được những thông tin cơ bản, tổng quát về đối tượng nghiên cứu nhằm phục vụ mục đích thống kê, phân tích.

CHƯƠNG 1 - TỔNG QUAN VỀ VẤN ĐỀ HÌNH ẢNH THẺ CĂN CƯỚC GIẢ MẠO

1.1. Giới thiệu vấn đề phát hiện hình ảnh thẻ căn cước giả mạo.

Một trong những điểm chính của các vụ lừa đảo trên môi trường điện tử là việc kẻ gian tạo ra các giấy tờ giả mạo để mở tài khoản ngân hàng hoặc giả mạo danh tính nạn nhân. Với sự hỗ trợ của công nghệ hiện đại, những giấy tờ giả này ngày càng tinh vi, khó phân biệt với giấy tờ thật. Hiện nay, kỹ thuật gian lận phổ biến thường được kẻ gian sử dụng là sử dụng hình ảnh thẻ căn cước gián tiếp qua màn hình thiết bị khác. Kẻ gian dùng các thiết bị như điện thoại, máy tính bảng hoặc máy tính để mở hình ảnh của giấy tờ rồi đánh lừa hệ thống rằng đó là một chiếc thẻ căn cước thật để qua mặt các ứng dụng định danh điện tử. Ví dụ minh họa cho hành vi này được thể hiện trong hình 1.1. Hành vi trên không chỉ gây thiệt hại nghiêm trọng về kinh tế mà còn làm suy giảm niềm tin của người dân vào các hệ thống tài chính và dịch vụ trực tuyến. Vì vậy, việc phát hiện và ngăn chặn kịp thời các giấy tờ giả sẽ đóng vai trò quan trọng trong việc giảm thiểu các vụ lừa đảo trên không gian mạng.



Hình 1.1: Minh họa quá trình định danh điện tử bằng thẻ căn cước thật (bên trái) và thẻ căn cước giả mạo (bên phải)

1.2. Các nghiên cứu liên quan

Bài toán phát hiện giả mạo nói chung và bài toán phát hiện thẻ căn cước giả mạo nói riêng là lĩnh vực thu hút nhiều sự quan tâm của cộng đồng khoa học. Đã có

nhiều nghiên cứu về lĩnh vực này, từ phương pháp trích xuất đặc trưng đơn giản đến những phương pháp học sâu tiên tiến.

Polevoy và cộng sự (2022) [4] đã sử dụng kiến trúc ResNet-50 [5] cho mục đích phân loại hai nhãn: ảnh thật (ảnh thẻ căn cước được chụp trực tiếp) và ảnh giả mạo (ảnh được chụp lại từ ảnh căn cước hiển thị trên thiết bị điện tử). Sử dụng tham số được huấn luyện sẵn trên tập dữ liệu ImageNet [6]; đóng băng 49 trên tổng 50 lớp; thay thế hàm softmax cuối thành hai chiều; sử dụng hàm mất mát: binary cross-entropy; sử dụng hàm tối ưu: Adam với tốc độ học (learning rate) 0.1; Tác giả sử dụng bộ dữ liệu DLC-2021 [4] và MIDV-2020 [7] cho quá trình huấn luyện và đánh giá mô hình này. Trong đó, quá trình huấn huyện sử dụng 19.543 ảnh giả, và 25.980 ảnh thật, quá trình đánh giá sử dụng 11.006 ảnh giả và 16.264 ảnh thật. Kết quả cho thấy Resnet-50 đã thu được cho hiệu năng tốt, với độ chính xác (precision): 85.89%, độ phủ (recall): 89.03%, và độ chính xác (accuracy) là 89.67%.

Kunina và cộng sự [8] giới thiệu một bộ phát hiện ảnh chụp màn hình giả mạo tập trung vào vùng biên của thẻ căn cước. Các dải hẹp dọc theo mép ảnh căn cước được làm nổi bật và áp dụng biến đổi Hough nhanh để làm lộ các vân nhiễu (moiré) dạng sọc. Trên cùng một bộ dữ liệu của Polevoy và cộng sự [4], phương pháp này đạt độ chính xác theo lớp dương (precision) 93,5%, độ bao phủ (recall) 90,5% và độ chính xác tổng thể (accuracy) đạt 92,1%, đồng thời chạy nhanh hơn khoảng 1,5–2 lần.

Markham và cộng sự (2023) [9] đã sử dụng bốn mô hình tạo sinh để tạo ra các hình ảnh thẻ căn cước giả mạo, sau đó huấn luyện mạng MobileNetV2 [10] để phân loại thật giả. Dữ liệu kiểm thử được xây dựng bằng cách tổng hợp từ các bộ dữ liệu công khai như: MIDV-2020 [7] và DLC-2021 [4]. Trên tập dữ liệu kiểm thử bao gồm ảnh thật và ảnh giả mạo theo kiểu in ấn, hệ thống này đạt tỷ lệ lỗi cân bằng (EER) chỉ khoảng 3.2% còn trên tập dữ liệu kiểm thử bao gồm ảnh thật và ảnh giả mạo theo kiểu ảnh màn hình, tỷ lệ lỗi cân bằng (EER) đạt 5.8%.

Al-Ghadi và cộng sự [11] giới thiệu IDTrust - một khung học sâu để phân biệt ảnh thẻ căn cước thật với thẻ giả mạo mà không cần các mẫu tham chiếu. Nhóm đề xuất hai biến thể: DeepQD, sử dụng bộ mã hoá CNN kèm một đầu phân loại nhị phân; và GuidedDeepQD, bổ sung bước lọc thông dải dựa trên FFT ở tiền xử lý để nhấn mạnh nền và các hoạ tiết guilloche bị suy giảm do quá trình quét trước khi đưa vào CNN. Mô hình đạt độ chính xác gần mức trần trên bộ dữ liệu MIDV-2020 [7].

Zhang và cộng sự [12] thực hiện một tổng hợp về các phương pháp phát hiện ảnh thẻ căn cước giả mạo dựa trên trí tuệ nhân tạo. Zhang và cộng sự tổng hợp 43 nghiên cứu, phân loại các hướng tiếp cận hiện có, đồng thời chỉ ra những ưu điểm, hạn chế và các thách thức còn bỏ ngỏ.

Mở rộng hơn, đối với bài toán phát hiện khuôn mặt giả mạo, Zeng và cộng sự (CVPRW 2023) [13] giải quyết bài toán này bằng mô hình hai nhánh mạng nơ-ron tích chập EfficientFormerV2 [14] trong đó một nhánh dùng ảnh gốc, nhánh còn lại đã được lọc qua bộ lọc Gaussian và phép biến đổi Fourier, sau đó dùng mô-đun chú ý chéo đa tầng để kết hợp đặc trưng. Trên bộ dữ liệu Face Anti-Spoofing Challenge@CVPR2023, mô hình này đạt mô hình này đạt tỷ lệ lỗi trung bình (ACER) 6.22% và xếp thứ tư trong cuộc thi.

Yu và cộng sự. (2020) [15] có cách tiếp cận bài toán phát hiện khuôn mặt giả mạo bằng cách thay thế các lớp mạng tích chập thông thường bằng mạng tích chập chênh lệch tập trung (Central Difference Convolution) và kết hợp thông tin cường độ, véc-tơ lượng giảm (gradient) tại mỗi điểm ảnh. Kiến trúc này cho kết quả tốt, đạt tỷ lệ lỗi trung bình (ACER) thấp (0.2%) trên bộ dữ liệu OULU-NPU nhánh Protocol-1 [16] và tỷ lệ lỗi HTER 6.5% khi đánh giá chéo trên bộ dữ liệu CASIA-MFSD [17].

Các phương pháp phát hiện giả mạo này có xu hướng ưu tiên sử dụng các mô hình học sâu tiên tiến, mới được công bố gần đây. Những mô hình này thường có quy mô lớn, cấu trúc phức tạp, và yêu cầu tài nguyên tính toán đáng kể để huấn luyện và triển khai. Các nghiên cứu này vẫn chưa quan tâm nhiều đến việc khai thác các kiến

trúc mạng cơ bản, ổn định, dễ triển khai và dễ tinh chỉnh trong các hệ thống có yêu cầu cao về tốc độ xử lý thời gian thực và tối ưu tài nguyên phần cứng.

1.3. Giải pháp giải quyết vấn đề của đề án

Việc nghiên cứu các mô hình truyền thống và cơ bản như VGG16 nhưng chú trọng vào việc tinh chỉnh kiến trúc mô hình, tối ưu hóa tham số hoàn toàn có thể mang lại hiệu quả rõ rệt, tương đương với việc sử dụng các mô hình mới. Đây là một hướng tiếp cận khả thi. Do đó, đề án tập trung vào việc tối ưu hóa các mô hình cơ bản, tăng cường thiết kế kiến trúc mới dựa trên những mô hình đó, xây dựng chiến lược huấn luyện hiệu quả và sử dụng thêm các kỹ thuật xử lý ảnh để nâng cao tính hiệu quả cho bài toán phát hiện hình ảnh thẻ căn cước giả mạo.

Mạng nơ-ron nhân tạo, đặc biệt là mạng VGG16 thường được sử dụng trong việc trích xuất và phân biệt các đặc trưng hình ảnh. Trong khi đó phép biến đổi Fourier và các bộ lọc tần số thường được ứng dụng để làm nổi bật các đặc trưng về tần số, tinh vi, khó phát hiện trong hình ảnh thông thường. Vì vậy đề án trình bày ý tưởng kết hợp mạng nơ-ron nhân tạo VGG16 với phép biến đổi Fourier để tăng cường khả năng phát hiện các dấu hiệu giả mạo trong hình ảnh thẻ căn cước nói riêng và giấy tờ tùy thân nói chung.

Đề án được triển khai theo ba giai đoạn. Đầu tiên, xây dựng tập dữ liệu hình ảnh thẻ căn cước, bao gồm dữ liệu huấn luyện, kiểm tra và kiểm thử. Tiếp theo, xây dựng, huấn luyện và tinh chỉnh mô hình. Quá trình này sử dụng ngôn ngữ lập trình Python, thư viện học sâu Pytorch, thư viện xử lý ảnh OpenCV và các công cụ hỗ trợ khác. Cuối cùng, đánh giá mô hình dưới hai khía cạnh là độ chính xác và hiệu năng hoạt động, đồng thời so sánh với một số mạng khác để kiểm chứng độ hiệu quả.

Hướng nghiên cứu này mở ra hướng tiếp cận mới trong việc triển khai các giải pháp định danh điện tử chi phí thấp nhưng độ chính xác cao, phù hợp với bối cảnh tại Việt Nam và các quốc gia đang phát triển.

1.4. Kết luận

Chương này đã khái quát bài toán giả mạo thẻ căn cước trên môi trường định danh điện tử, làm rõ hình thức gian lận giấy tờ tùy thân bằng phương pháp chụp gián tiếp qua màn hình thiết bị khác, sau đó nêu các nghiên cứu liên quan đến bài toán này và phương pháp giải quyết vấn đề của đề án. Các nghiên cứu hiện nay có hạn chế là thường tập trung vào mô hình mới mà chưa chú trọng đến việc tinh chỉnh, nâng cao hiệu suất của các mô hình cơ bản và phát huy các lợi thế của các phương pháp xử lý ảnh như phép biến đổi Fourier rời rạc. Giải pháp giải quyết vấn đề của đề án là tập trung vào mạng nơ-ron VGG16 và phép biến đổi Fourier, đồng thời nghiên cứu phương án tích hợp hai chiến lược này để tối ưu việc phát hiện thẻ căn cước giả mạo. Đây là cơ sở lý thuyết cơ bản cho các chương sau.

CHƯƠNG 2 - PHƯƠNG PHÁP PHÁT HIỆN HÌNH ẢNH THẺ CĂN CƯỚC GIẢ MẠO

2.1. Giới thiệu về phép biến đổi Fourier

Phép biến đổi Fourier là một phương pháp toán học được sử dụng trong lĩnh vực xử lý ảnh, nén hình ảnh, khôi phục hình ảnh, xử lý âm thanh, truyền thông số, và nhiều lĩnh vực kỹ thuật – khoa học khác.

2.1.1. Phép biến đổi Fourier rời rạc hai chiều.

Phép biến đổi Fourier rời rạc hai chiều là phép tính đưa tín hiệu như tín hiệu âm thanh hoặc tín hiệu hình ảnh từ miền thời gian hoặc không gian sang miền tần số, mục đích là phân giải tín hiệu đó thành tổng các thành phần sóng sin hoặc cos với các tần số khác nhau [18]. Giả sử đầu vào là một bức ảnh xám $p(x, y)$, kích thước $M \times N$, phép biến đổi Fourier rời rạc hai chiều cho bức ảnh này như sau:

$$Fp(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} p(x, y) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \quad (2.3)$$

Với $p(x, y)$ là một giá trị pixel ở hàng x cột y; $Fp(u, v)$ là ma trận phổ tần số tương ứng với $u = 0, 1, 2, \ldots, M-1$; $v = 0, 1, 2, \ldots, N-1$. Phép biến đổi Fourier rời rạc ngược để chuyển $Fp(u, v)$ thành $p(x, y)$ như sau:

$$p(x, y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} Fp(u, v) e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \quad (2.4)$$

Hai phương trình trên tạo thành một cặp biến đổi Fourier rời rạc, ký hiệu là $p(x, y) \Leftrightarrow Fp(u, v)$.

Cooley và Tukey đề xuất thuật toán biến đổi Fourier rời rạc nhanh (Fast Fourier Transform - FFT), giúp tối ưu hóa chi phí tính toán trên máy tính. FFT áp

dụng chiến lược "chia để trị", giúp giảm đáng kể độ phức tạp tính toán, từ $O(N^2)$ xuống còn $O(NlogN)$ [18].

2.1.2. Nguyên lý và cách thực hiện lọc tần số bằng phép biến đổi Fourier trên ảnh.

2.1.2.1. Các đặc trưng tần số trong ảnh

Phổ tần số cung cấp thông tin quan trọng về cấu trúc và tính toàn vẹn của hình ảnh mà miền không gian khó hiển thị. Theo R.C. Gonzalez, và R. E. Woods [3], sự phân bố năng lượng trong miền tần số phản ánh trực tiếp các đặc tính hình thái của ảnh:

- Tần số thấp có liên quan đến các thành phần cường độ thay đổi chậm trong hình ảnh (ví dụ: các bức tường của một căn phòng, bầu trời không mây).

- Tần số cao là do sự chuyển đổi đột ngột về cường độ (ví dụ: các cạnh biên, vân nhiễu).

Trong một bức ảnh rõ, có nhiều vùng tần số cao hơn. Trong một bức ảnh mờ, có nhiều vùng tần số thấp hơn. Từ đó có thể sử dụng miền tần số để phân biệt ảnh rõ nét và ảnh mờ.

2.1.2.2. Khái niệm lọc tần số

Trong miền tần số, bộ lọc là một mặt nạ phủ lên phổ tần số của ảnh để giữ lại hoặc loại bỏ một dải tần số nhất định. Bộ lọc tần số cao có tác dụng loại bớt hoặc triệt tiêu các thành phần tần số thấp, chỉ giữ lại các thành phần tần số cao, đồng nghĩa các đường nét, chi tiết kết cấu, đường biên, cạnh sẽ được giữ lại. Ngược lại, bộ lọc tần số thấp giúp loại bỏ các chi tiết tần số cao và chỉ giữ lại các thành phần có tần số thấp.

2.1.2.3. Cách thực hiện bộ lọc tần số

Cho bức ảnh đầu vào là ảnh xám có kích thước $M \times N$, mỗi điểm ảnh tại hàng x, cột y có giá trị $p(x, y)$. Trong trường hợp bức ảnh đầu vào là ảnh màu (ảnh RGB), cần

thực hiện việc chuyển ảnh sang ảnh xám theo công thức (2.5). Hình 2.1 minh họa quá trình chuyển đổi từ ảnh màu sang ảnh xám.

$$p(x,y) = Y'(x,y) = 0.299R'(x,y) + 0.587G'(x,y) + 0.114B'(x,y) \quad (2.5)$$



Hình 2.1: Ảnh xám (bên phải) thu được sau khi chuyển đổi từ ảnh màu (bên trái)

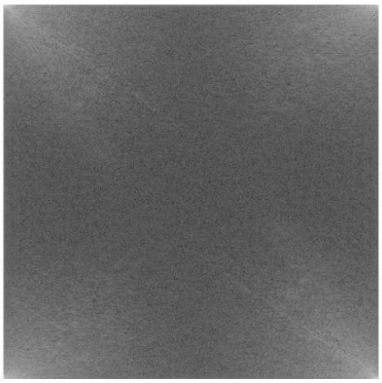
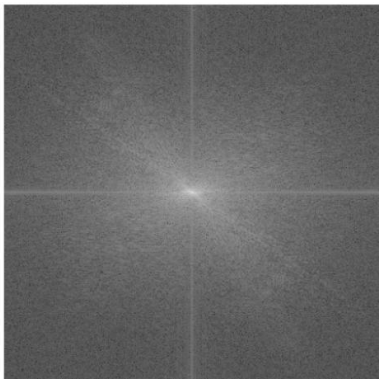
Để thực hiện bộ lọc tần số trên bức ảnh này cần thực hiện các bước sau:

Bước 1: Chuyển ảnh từ miền không gian sang miền tần số

Áp dụng công thức (2.4) vào bức ảnh xám đầu vào, thu được ma trận phổ tần số đầu ra. Sau đó, tiến hành dịch phổ tần số theo công thức (2.6) để các tần số bằng 0 sẽ nằm tại vị trí trung tâm của ma trận, việc dịch phổ này giúp dễ dàng quan sát mặt nạ tần số dưới dạng vòng tròn đồng tâm, các tần số thấp nằm xung quanh tâm của phổ, các tần số cao nằm ở biên, cũng như dễ dàng thao tác với các bộ lọc ở bước tiếp theo.

$$Fp_{shift}(u,v) = Fp\left(\left(u + \frac{M}{2}\right) \bmod M, \left(v + \frac{N}{2}\right) \bmod N\right) \quad (2.6)$$

Quá trình này được minh họa trong hình 2.2, hình 2.3 và hình 2.4 dưới đây.

(2.2) *(2.3)* *(2.4)*

Hình 2.2: Ảnh xám gốc.
Hình 2.3: Ảnh phổ tần số Fourier
Hình 2.4: Ảnh phổ tần số Fourier đã dịch chuyển tần số 0 về tâm

Bước 2: Thiết kế bộ lọc tần số

Có ba loại bộ lọc phổ biến cho bộ lọc tần số là bộ lọc lý tưởng, bộ lọc butterworth và bộ lọc Gaussian. Trong phạm vi của đề án, chỉ giới thiệu về bộ lọc lý tưởng.

Định nghĩa bộ lọc thông thấp lý tưởng:

$$H(u,v) = \begin{cases} 1, D(u,v) \le D_0, \\ 0, D(u,v) > D_0, \end{cases} \quad (2.7)$$

Định nghĩa bộ lọc thông cao lý tưởng:

$$H(u,v) = \begin{cases} 0, D(u,v) \le D_0, \\ 1, D(u,v) > D_0, \end{cases} \quad (2.8)$$

Với:

$$D(u,v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2} \quad (2.9)$$

Và D_0 là bán kính phần giữ lại.

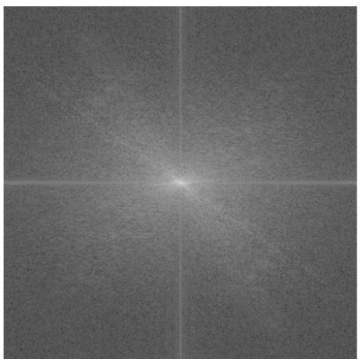
Cần lưu ý chọn giá trị D_0 phù hợp. Trong bộ lọc tần số thấp, nếu D_0 quá nhỏ thì ảnh đầu ra sẽ rất mờ, ngược lại nếu D_0 quá lớn thì chỉ loại bỏ được rất ít phần nhiễu. Trong bộ lọc tần số cao, nếu D_0 quá nhỏ thì sẽ giữ lại gần như toàn bộ tần số cao, việc lọc không hiệu quả, ngược lại nếu D_0 quá lớn thì sẽ chỉ làm nổi bật được những chi tiết đột ngột, cá chi tiết thực tế khác sẽ bị loại bỏ hết.

Về ưu điểm, bộ lọc tần số lý tưởng sẽ cắt đứt hoàn toàn các thành phần tần số vượt ngưỡng, tuy nhiên, nhược điểm là sẽ tạo nên một ranh giới giữa phần đã được loại bỏ và phần còn lại tương đối rõ ràng, dẫn đến hiện tượng ringing, hay còn gọi là hiệu ứng Gibbs.

Bước 3. Áp dụng bộ lọc vào ảnh tần số và biến đổi Fourier ngược

Sau khi đã có bộ lọc $H(u, v)$, có thể là lọc tần số thấp hoặc tần số cao tùy nhu cầu sử dụng. Tiến hành áp dụng bộ lọc vào ảnh theo công thức (2.10). Kết quả thu được $G(u, v)$ là phổ tần số đã được lọc, tức là chỉ còn các thành phần tần số mong muốn. Quá trình này được minh họa trong hình 2.5, hình 2.6 và hình 2.7.

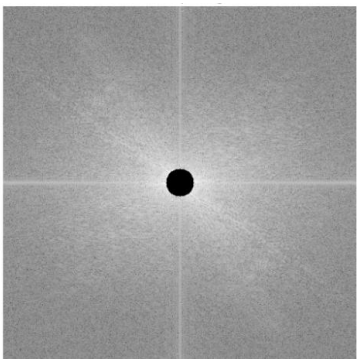
$$G(u, v) = H(u, v) \times Fp_{shift}(u, v) \tag{2.10}$$



(2.5)



(2.6)



(2.7)

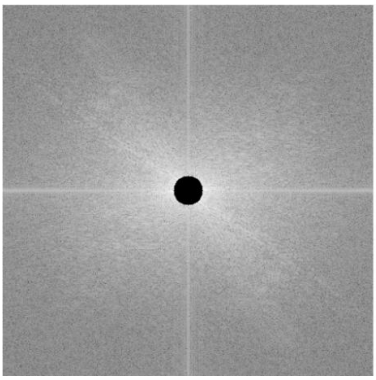
Hình 2.5: Ảnh phổ tần số Fourier ban đầu
Hình 2.6: Mặt nạ tần số cao với bán kính bằng 20.
Hình 2.7: Ảnh phổ tần số Fourier sau khi áp dụng mặt nạ

Để có thể biến đổi Fourier rời rạc ngược, cần tiến hành đưa các điểm tần số bằng 0 về góc theo công thức (2.11)

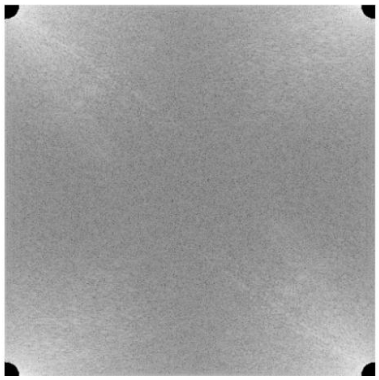
$$G_{unshift}(u,v) = G\left(\left(u+\frac{M}{2}\right) \bmod M, \left(v+\frac{N}{2}\right) \bmod N\right) \qquad (2.11)$$

Áp dụng công thức biến đổi Fourier rời rạc ngược (2.12) vào $G_{unshift}(u,v)$. Kết quả thu được $g(x,y)$ là ảnh đã lọc trong miền không gian. Quá trình này được minh họa trong hình 2.8, hình 2.9 và hình 2.10 dưới đây.

$$g(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} G_{unshift}(u,v)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \qquad (2.12)$$



(2.8)



(2.9)



(2.10)

Hình 2.8: Ảnh phổ tần số Fourier sau khi đã áp dụng mặt nạ.

Hình 2.9: Ảnh phổ tần số Fourier khi dịch ngược

Hình 2.10: Ảnh phổ tần số sau khi biến đổi Fourier rời rạc ngược

2.2. Giới thiệu về mô hình mạng nơ-ron nhân tạo VGG16

2.2.1. Giới thiệu về học sâu, mạng nơ-ron nhân tạo.

Học sâu là một nhánh của học máy, cho phép xây dựng các mạng nơ-ron nhân tạo để biểu diễn dữ liệu và biến đổi dữ liệu ở các mức độ trừu tượng khác nhau. Mạng nơ-ron nhân tạo gồm nhiều lớp chồng lên nhau. Các lớp phía sau liên tiếp biến đổi

đầu ra của lớp liền trước thành các biểu diễn đặc trưng ngày càng rõ ràng và sắc nét hơn. Để tạo ra một mạng nơ-ron nhân tạo, thường sẽ cần hai quá trình cơ bản: Quá trình huấn luyện và quá trình đánh giá mạng.

2.2.1.1. Quá trình huấn luyện, hàm mất mát và quá trình tối ưu hóa mạng.

Trong một lô gồm B mẫu được sử dụng để huấn luyện mạng, gọi $X = (x_1, x_2, \dots, x_d), X \in B$ là một mẫu đầu vào của mạng. Mục tiêu của quá trình huấn luyện là tìm bộ tham số tối ưu $W = (w_1, w_2, \dots, w_m)$ sao cho nhãn đầu ra dự đoán $\hat{y}$ thu được khi đi qua mạng nơ-ron: $\hat{y} = f(X, W)$ càng gần với nhãn thực tế y càng tốt trên tập dữ liệu.

Để đánh giá độ khác biệt giữa các nhãn đầu ra dự đoán $\hat{y}$ với nhãn thực tế y, cần sử dụng một hàm đánh giá được gọi là hàm mất mát. Còn quá trình tối ưu hóa để nhãn dự đoán gần với nhãn thực tế sau mỗi bước huấn luyện được gọi là quá trình tối ưu hóa véc-tơ lượng giảm (gradient). Hàm mất mát được định nghĩa như sau:

$$L = \frac{1}{B}\sum_{n=1}^{B} l(\hat{y}_n, y_n) \tag{2.13}$$

Trong đó $l(\hat{y}_n, y_n)$ là hàm mất mát cho một mẫu, hàm mất mát thường được sử dụng có thể là Cross-Entropy, MSE,…

Sau đó, sử dụng giá trị L để tìm ra véc-tơ lượng giảm (hay còn gọi là véc-tơ gradient), véc-tơ này chỉ hướng tăng nhanh nhất của hàm mất mát:

$$g = \frac{\partial L}{\partial W} \tag{2.14}$$

Cuối cùng, tham số của mạng được cập nhật theo một phương pháp tối ưu hóa gradient:

$$W_{t+1} = W_t - \alpha \times \Delta(g_t) \tag{2.15}$$

Trong đó α là hệ số tốc độ học, W_t là véc-tơ tham số tại bước thứ t, g_t là véc-tơ gradient tại bước thứ t. $\Delta(g_t)$ là thuật toán biên dịch gradient, có thể là: Gradient Descent, SGD, Adam, RMSProp.

Quá trình này lặp đi lặp lại, mạng sẽ dần tìm được bộ tham số W tối ưu. Quá trình kết thúc khi đủ số lượng chu kỳ đã cài đặt hoặc đạt một điều kiện đánh giá mạng được cài đặt nào đó.

2.2.1.2. Quá trình đánh giá mạng

Quá trình đánh giá mạng là bước quan trọng để xác định mức độ chính xác của mô hình trên những dữ liệu chưa từng thấy trong quá trình huấn luyện. Trong các mạng nơ-ron giải quyết bài toán phân loại nhị phân, đầu ra của mạng sẽ gồm hai nhãn: nhãn dương (positive) hoặc nhãn âm (negative). Dựa vào nhãn dự đoán và nhãn thực tế, có thể chia kết quả dự đoán thành bốn phần, bao gồm:

- TP (True Positive): dự đoán đúng mẫu dương.

- FP (False Positive): dự đoán sai, dự đoán dương, thực tế là mẫu âm.

- TN (True Negative): dự đoán đúng mẫu âm.

- FN (False Negative): dự đoán sai, dự đoán âm, thực tế là mẫu dương.

Từ những chỉ số trên, có thể lập thành các chỉ số đánh giá khác nhau để đánh giá mạng trên tập dữ liệu kiểm thử, tùy vào tính chất bài toán phân loại, thường sử dụng một hoặc kết hợp các chỉ số đánh giá sau:

- Ma trận nhầm lẫn (confusion matrix): là một ma trận kết hợp từ bốn chỉ số trên, thể hiện số lượng trường hợp đúng/sai dự đoán theo mỗi chỉ số.

- Độ chính xác tổng thể (Accuracy):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

- Độ chính xác của mẫu dương (Precision):

$$Precision = \frac{TP}{TP + FP} \tag{2.17}$$

- Độ phủ của mẫu dương (Recall):

$$Recall = \frac{TP}{TP + FN} \tag{2.18}$$

- Điểm trung bình điều hòa của mẫu dương (F1 Score):

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.19}$$

Ngoài ra, trong các bài toán phát hiện giả mạo, theo ISO/IEC 30107-3:2023 [19], APCER (Attack Presentation Classification Error Rate) tức tỉ lệ mẫu giả bị phân loại sai thành nhãn thật và BPCER (Bona Fide Presentation Classification Error Rate) tức tỉ lệ mẫu thật bị phân loại sai thành nhãn giả là hai chỉ số quan trọng giúp đánh giá độ chính xác và tin cậy của mạng nơ-ron nhân tạo.

$$APCER = \frac{FN}{TP + FN} \tag{2.20}$$

$$BPCER = \frac{FP}{TN + FP} \tag{2.21}$$

Ngoài việc đo lường các chỉ số về độ chính xác của mạng, việc đo lường các thông số về hiệu năng cũng rất quan trọng, các chỉ số này bao gồm: số lượng tham số (parameters), độ phức tạp tính toán (floating-point operations - FLOPs), tốc độ thực thi (speed).

Quá trình đánh giá mạng là bước trọng yếu trong vòng đời phát triển mô hình học sâu. Việc lựa chọn chỉ số đánh giá phù hợp đảm bảo mô hình đạt hiệu suất cao và hoạt động hiệu quả trên dữ liệu thực tế.

2.2.2. Giới thiệu về mạng nơ-ron tích chập

Mạng nơ-ron tích chập là một kiến trúc được thiết kế để xử lý dữ liệu hình ảnh và các dữ liệu có cấu trúc lưới ma trận tín hiệu thời gian. Điểm khác biệt cơ bản của mạng nơ-ron tích chập so với mạng nơ-ron truyền thống khác nằm ở cách thức xử lý thông tin đầu vào: thay vì trải phẳng véc-tơ ảnh đầu vào sau đó kết nối tuyến tính toàn bộ điểm dữ liệu vào từng nơ-ron thì mạng nơ-ron tích chập tổ chức các nơ-ron thành những véc-tơ đặc trưng không gian và thực hiện phép tích chập giữa bộ lọc và các vùng cục bộ của ảnh. Cách tiếp cận này giúp mạng học được các đặc trưng không gian như cạnh, góc, họa tiết, đồng thời giảm thiểu số lượng tham số cần huấn luyện.

Thành phần cơ bản của mạng nơ-ron tích chập bao gồm: lớp tích chập (convolutional layers), các lớp kích hoạt (activation layers), lớp gộp (pooling layers), lớp kết nối đầy đủ (fully connected layers), lớp chuẩn hóa (batch normalization layers), lớp lược bỏ (dropout).

2.2.2.1. Lớp tích chập.

Lớp tích chập (convolutional - Conv) là lớp phổ biến nhất trong mạng nơ-ron tích chập. Giả sử, véc-tơ ảnh đầu vào có kích thước $H \times W \times C$ trong đó H là chiều cao, W là chiều rộng, và C là số kênh màu. Một lớp tích chập sử dụng K bộ lọc (kernel/filter) F_i kích thước $f \times f$, $i \leq K$ mỗi bộ lọc có số kênh bằng C. Quá trình tích chập là phép lặp mỗi bộ lọc F_i trên toàn véc-tơ đặc trưng ảnh, tính tích vô hướng giữa trọng số của bộ lọc và vùng con tương ứng trên véc-tơ ảnh, sau đó cộng thêm hệ số chêch lệch (bias) và đưa qua hàm kích hoạt. Kết quả tạo ra một véc-tơ đầu ra có kích thước$(H' \times W' \times K)$, trong đó:

$$H' = \left\lfloor \frac{H + 2p - f}{s} \right\rfloor + 1 \tag{2.23}$$

$$W' = \left\lfloor \frac{W + 2p - f}{s} \right\rfloor + 1 \tag{2.24}$$

với s là hệ số trượt (slide), p là hệ số kích thước biên (padding).

Mỗi bộ lọc trong lớp tích chập chuyên trách nhận diện một đặc trưng cụ thể, ví dụ cạnh, góc, hoặc họa tiết. Ban đầu, trọng số của các bộ lọc được khởi tạo ngẫu nhiên; sau đó, thông qua quá trình huấn luyện, mạng sẽ điều chỉnh các bộ lọc này sao cho véc-tơ đầu ra thể hiện những đặc trưng phù hợp. Nhờ áp dụng nhiều bộ lọc song song, lớp tích chập có thể phát hiện đồng thời nhiều đường biên, kết cấu, cũng như các mẫu vệt phức tạp.

2.2.2.2. Lớp kích hoạt.

Sau khi thực hiện phép tích chập, đầu ra thường được đưa qua lớp kích hoạt như hàm ReLU (Rectified Linear Unit), hàm Leaky ReLU, hàm Sigmoid, hàm Softmax, hàm Tanh,…. Lớp kích hoạt có tác dụng giúp mạng học được những đặc trưng phức tạp hơn, hoặc giảm tải quá trình tính toán,… Ví dụ, hàm ReLU được định nghĩa là $f(x) = max(0, x)$ có tác dụng loại bỏ giá trị âm, giúp mô hình hội tụ nhanh và giảm thiểu vấn đề tiêu biến gradient.

2.2.2.3. Lớp gộp

Lớp gộp có nhiệm vụ giảm kích thước của các véc-tơ đặc trưng, từ đó giảm số lượng tham số, tăng khả năng khái quát hóa và giảm thiểu vấn đề quá khớp vào tập huấn luyện. Hai dạng phổ biến nhất của lớp gộp là Max Pooling và Average Pooling.

2.2.2.4. Lớp kết nối đầy đủ

Sau khi véc-tơ ảnh đầu vào đi qua chuỗi các lớp mạng trong khối mạng tích chập trích xuất đặc trưng (chủ yếu là lớp tích chập và lớp gộp), một véc-tơ đặc trưng không gian đầu ra đã được trích xuất và cô đọng. Tiếp theo, cần một hoặc nhiều lớp kết nối đầy đủ (fully-connected) là bộ phân loại ở cuối kiến trúc CNN. Lớp cuối cùng thường có số lượng nơ-ron tương ứng với số lượng nhãn của bài toán. Hàm kích hoạt Softmax được áp dụng tại đầu ra để chuyển đổi các giá trị logit thành phân phối xác suất dự đoán cho từng phân lớp.

2.2.2.5. Lớp lược bỏ

Lớp lược bỏ (dropout) nhằm ngăn chặn tình trạng quá khớp, bằng cách ngẫu nhiên vô hiệu hóa một tỷ lệ p nơ-ron trong quá trình huấn luyện. Khi đó, mà phải học được đặc trưng phân tán ở nhiều nơ-ron khác nhau thay vì tập trung vào một hoặc một vài nơ-ron cụ thể nào đó.

2.2.2.6. Lớp chuẩn hóa tích chập

Lớp chuẩn hóa tích chập nhận dữ liệu đầu vào cho mỗi lô nhỏ (mini-batch) và chuẩn hóa giá trị sao cho có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1, sau đó nhân với hệ số tỷ lệ γ và cộng thêm hệ số dịch β. Mục đích chính là để giảm hiện tượng thay đổi phân phối đầu vào của mỗi lớp theo quá trình huấn luyện. Điều này cho phép dùng quá trình huấn luyện có thể dùng hệ số tỉ lệ học (learning rate) lớn hơn, tối ưu mô hình nhanh hơn.

2.2.3. Mô hình mạng nơ-ron VGG16

2.2.3.1. Lịch sử ra đời.

Simonyan và Zisserman (2014) [20] đã giới thiệu về các cấu hình mạng khác nhau của mô hình mạng nơ-ron tích chập. Đặc trưng của mạng này là việc tăng số lượng lớp tích chập lên so với phiên bản trước đây, và sử dụng những cửa sổ lọc có kích thước rất nhỏ (3x3) thay vì sử dụng cửa sổ lọc có kích thước lớn hơn như (5x5) hay (7x7). Mục đích của việc này là, **(i)** để mô hình có khả năng phân hóa tốt hơn, và **(ii)** để giảm số lượng tham số trong mạng. Các cấu trúc mạng được Simonyan và Zisserman giới thiệu được biểu thị ở bảng dưới đây.

Tên cấu hình	**Số lớp**
A	11 lớp bao gồm 8 lớp tích chập và 3 lớp kết nối đầy đủ

A-LRN	11 lớp bao gồm 8 lớp tích chập và 3 lớp kết nối đầy đủ nhưng bổ sung thêm 2 lớp Chuẩn hóa phản hồi cục bộ (Local Response Normalization - LRN) sau mỗi hai lớp tích chập đầu tiên.
B	13 lớp bao gồm 10 lớp tích chập và 3 lớp kết nối đầy đủ
C	16 lớp bao gồm 13 lớp tích chập và 3 lớp kết nối đầy đủ
D	16 lớp bao gồm 13 lớp tích chập và 3 lớp kết nối đầy đủ
E	19 lớp bao gồm 16 lớp tích chập và 3 lớp kết nối đầy đủ

Bảng 2.1: Các cấu hình khác nhau của mạng nơ-ron tích chập theo Simonyan và Zisserman

Tên gọi VGG16 là tên gọi không chính thức, bắt nguồn từ việc sử dụng mạng nơ-ron tích chập với cấu hình D. VGG viết tắt của tổ chức "Visual Geometry Group", một tổ chức khoa học kỹ thuật thuộc đại học Oxford, nơi công tác của tác giả. 16 là số lượng lớp nơ-ron trong mạng, bao gồm 13 lớp mạng tích chập và 3 lớp mạng kết nối đầy đủ. VGG16 là một mô hình tích chập hiệu quả trong việc nhận diện và phân loại hình ảnh. Theo Simonyan & Zisserman (2014) [20], VGG16 đạt được độ chính xác 92.7% tập kiểm thử trên tập dữ liệu ImageNet [6], xét theo chỉ số Top-5.

Tham số mạng được huấn luyện sẵn với bộ dữ liệu ImageNet [6] thường được sử dụng trong kỹ thuật học chuyển giao nhằm rút ngắn thời gian huấn luyện mạng.

2.2.3.2 Cấu trúc mạng

VGG16 có khoảng 138 triệu tham số, trong đó 13 lớp tích chập có khoảng 18 triệu tham số, còn lại 120 triệu tham số nằm ở ba lớp kết nối đầy đủ cuối cùng. Hình ảnh đầu vào được chuẩn hóa về kích thước cố định 224×224 pixel với 3 kênh màu (RGB). Trước khi đưa vào mạng, mỗi pixel sẽ được trừ đi giá trị trung bình RGB tính trên toàn bộ tập huấn luyện. Tất cả các lớp tích chập đều có hệ số trượt (stride) = 1 và hệ số đệm biên (padding) = 1 để giữ nguyên kích thước không gian. Lớp gộp sử

dụng Max Pooling với cửa sổ 2×2 và hệ số trượt (stride) = 2, làm giảm một nửa chiều rộng và chiều cao mỗi lần gộp.

Chi tiết về các lớp mạng của VGG16 được thể hiện trong bảng 2.2

Khối	**Lớp**	**Kích thước đầu ra**
1	Conv (3×3, 64) + BatchNorm + ReLU	224 × 224 × 64
	Conv (3×3, 64) + BN + ReLU + MaxPool (2×2, s=2)	112 × 112 × 64
2	Conv (3×3, 128) + BN + ReLU	112 × 112 × 128
	Conv (3×3, 128) + BN + ReLU + MaxPool (2×2, s=2)	56 × 56 × 128
3	Conv (3×3, 256) + BN + ReLU	56 × 56 × 256
	Conv (3×3, 256) + BN + ReLU	56 × 56 × 256
	Conv (3×3, 256) + BN + ReLU + MaxPool (2×2, s=2)	28 × 28 × 256
4	Conv (3×3, 512) + BN + ReLU	28 × 28 × 512
	Conv (3×3, 512) + BN + ReLU	28 × 28 × 512
	Conv (3×3, 512) + BN + ReLU + MaxPool (2×2, s=2)	14 × 14 × 512
5	Conv (3×3, 512) + BN + ReLU	14 × 14 × 512
	Conv (3×3, 512) + BN + ReLU	14 × 14 × 512
	Conv (3×3, 512) + BN + ReLU + MaxPool (2×2, s=2)	7 × 7 × 512
6	Làm phẳng + Kết nối đầy đủ (4096) + ReLU + Dropout	4096
	Kết nối đầy đủ (4096) + ReLU + Dropout	4096
	Kết nối đầy đủ (1000) + Softmax	1000

Bảng 2.2: Chi tiết cấu trúc trong mạng nơ-ron VGG16

2.2.3.3 Ưu điểm và nhược điểm của VGG16

Về ưu điểm, VGG16 đã tăng số lớp tích chập giúp mạng học được các đặc trưng ngày càng trừu tượng hơn, dẫn đến độ chính xác cao trên ImageNet [6]. Hơn nữa, toàn bộ lớp tích chập đều có kích thước bộ lọc 3×3, toàn bộ lớp kết nối đầy đủ

đều có 4096 nơ-ron ở 2 lớp đầu, khiến việc cài đặt và điều chỉnh dễ dàng hơn so với các mạng pha trộn nhiều kích thước bộ lọc.

Về nhược điểm, kích thước tham số của VGG16 khá lớn, chiếm khoảng 533MB với 138 triệu tham số. Để triển khai mô hình cần bộ nhớ lớn và khả năng tính toán cao. Vì vậy, VGG16 không phù hợp cho những hệ thống yêu cầu độ trễ thấp hoặc bộ nhớ hạn chế.

2.2.3.4. Ứng dụng của VGG16

Mặc dù ra đời đã lâu, tuy nhiên VGG16 vẫn được sử dụng trong các bài toán phân loại ảnh, phát hiện vật thể, trích xuất đặc trưng. VGG16 được huấn luyện để phân loại 1000 đối tượng trên bộ dữ liệu ImageNet [6], đạt độ chính xác khoảng 71.3% theo chỉ số Top-1 và khoảng 92.7% với chỉ số Top-5. Đối với các bộ dữ liệu nhỏ hơn như CIFAR-10, CIFAR-100 [21], VGG16 thường được sử dụng trong phương pháp học chuyển giao từ tham số được huấn luyện sẵn, chỉ cần tinh chỉnh một số lớp cuối cùng để thích hợp với số lượng đối tượng mới. Ngoài ra, VGG16 còn được sử dụng như một bộ trích xuất đặc trưng bằng cách sử dụng các khối tích chập đầu tiên để trích xuất véc-tơ đặc trưng, sau đó làm phẳng để sử dụng trong các bài toán khác như bài toán huấn luyện "zero-shot". Nhờ kết quả tốt và cấu trúc đơn giản, VGG16 nhanh chóng trở thành nền tảng cho nhiều nghiên cứu sau này [22] [23] [24], đặc biệt trong các ứng dụng học chuyển giao (transfer learning) và các bài toán xử lý ảnh y tế, xe tự lái, v.v.

2.3. Giới thiệu về mô-đun tự chú ý

Vaswani và cộng sự (2017) [25] đã giới thiệu mô-đun tự chú ý (self-attention). Nó đã trở thành thành phần quan trọng trong nhiều kiến trúc mạng nơ-ron, nhất là trong lĩnh vực xử lý ngôn ngữ tự nhiên và thị giác máy tính. Mô-đun tự chú ý cho phép mô hình tự động xác định mức độ liên quan giữa các thành phần khác nhau trong cùng một chuỗi đầu vào, từ đó cải thiện khả năng học biểu diễn và mối quan hệ ngữ cảnh.

Vaswani và cộng sự nhận thấy các kiến trúc mạng nơ-ron hồi quy truyền thống dựa vào quá trình tính toán tuần tự theo bước thời gian sẽ dẫn đến việc huấn luyện song song gặp nhiều khó khăn, đặc biệt với các chuỗi đầu vào dài. Thêm vào đó, việc vận chuyển thông tin qua nhiều lớp khiến mạng nơ-ron hồi quy truyền thống dễ gặp vấn đề về tiêu biến gradient hoặc bùng nổ gradient. Vì vậy, Vaswani và cộng sự đã thay thế hoàn toàn cơ chế xử lý tuần tự bằng một phép toán cho phép xử lý đồng thời mọi vị trí trong chuỗi, giúp cải thiện hiệu năng tính toán và khả năng học ngữ cảnh dài hạn.

Mô-đun tự chú ý có mục tiêu tính toán bộ trọng số α_{ij} (hay còn gọi là trọng số chú ý) từ tập các véc-tơ đặc trưng đầu vào $\{x_1, x_2, \dots, x_n\}$ nhằm biểu thị mức độ "chú ý" mà phần tử thứ i dành cho phần tử thứ j. Bộ trọng số này bao gồm ba thành phần chính: ma trận Q (Query), K (Key) và V (Value).

- Query (Q): Tập hợp các véc-tơ $\{q_1, q_2, \dots, q_n\}$ được sinh ra từ đầu vào, đóng vai trò như câu truy vấn thông tin.

- Key (K): Tập hợp các véc-tơ $\{k_1, k_2, \dots, k_n\}$ cũng được tính từ cùng đầu vào, đóng vai trò như từ khóa để tìm truy vấn phù hợp.

- Value (V): Tập hợp các véc-tơ $\{v_1, v_2, \dots, v_n\}$ chứa thông tin thực sự được truyền đến tầng tiếp theo.

Giả sử ma trận đầu vào $X \in \mathbb{R}^{n \times d_{model}}$, lô-gic của mô-đun tự chú ý (self-attention) trải qua các bước sau:

Bước 1: Thực hiện phép chiếu tuyến tính (Linear Projection) để xây dựng các ma trận Q, K, V từ véc-tơ đầu vào:

$$Q = XW^Q;\ K = XW^K;\ V = XW^V \tag{2.25}$$

với $W^Q, W^K, W^V \in \mathbb{R}^{d_{model} \times d_k}$

Bước 2: Tính tích vô hướng tỉ lệ (Scaled Dot-Product):

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \in \mathbb{R}^{n\times d_k} \quad (2.26)$$

trong đó $\sqrt{d_k}$ là hệ số chuẩn hóa nhằm tránh giá trị tích vô hướng quá lớn khi kích thước chiều tăng, dẫn đến độ dốc của softmax quá nhỏ. Kết quả là ma trận trọng số chú ý $\alpha = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$ với $\alpha \in \mathbb{R}^{d_{model}\times d_k}$. Thông thường, các tầng của mô-đun tự chú ý (self-attention) sẽ được nối hoặc cộng để tạo thành mô-đun tự chú ý đa tầng (multi-head attention).

Bên cạnh cơ chế tự chú ý, Vaswani và cộng sự cũng giới thiệu mô-đun chú ý chéo (cross-attention). Mô-đun chú ý chéo có cấu trúc tương tự mô-đun tự chú ý, nhưng cho phép một chuỗi truy vấn tập trung vào các vị trí quan trọng của một chuỗi ngữ cảnh khác. Cơ chế này đặc biệt quan trọng trong các bài toán ánh xạ từ chuỗi nguồn sang chuỗi đích, như trong bài toán dịch máy, khi đầu ra tại mỗi thời điểm cần quan sát các vị trí phù hợp trong câu nguồn để sinh token tiếp theo.

2.4. Giới thiệu về mô hình VGG16 kết hợp phép biến đổi Fourier.

Các bức ảnh giả mạo thường chứa các đặc trưng khác biệt. Hình ảnh giả mạo thường chứa các chi tiết như viền màn hình điện thoại, mép khung máy tính. Những đường nét này sẽ được mạng nơ-ron VGG16 trích xuất đặc trưng và phân loại. Yếu tố này đóng vai trò lớn trong việc phát hiện những bất thường trong ảnh chụp thẻ căn cước.

Tuy nhiên, nếu chỉ dựa vào các đặc trưng không gian truyền thống từ ảnh đầu vào có thể bỏ sót các dấu hiệu nhỏ trong ảnh giả mạo như độ mờ rõ của ảnh, các vân nhiễu,…Để khắc phục hạn chế này, tiến hành xử lý hình ảnh theo miền tần số nhằm giúp những khác biệt giữa ảnh thật và giả có thể được khuếch đại và nhận diện rõ ràng hơn. Hình ảnh thẻ căn cước giả mạo thường mang theo các vân sọc từ màn hình

thiết bị hiển thị, và những cấu trúc tần số cao này chỉ lộ rõ khi áp dụng phép biến đổi Fourier kết hợp với bộ lọc tần số cao. Cụ thể, sau khi tiến hành phân tích Fourier, những dải vân ngang, dọc có chu kỳ ngắn, một đặc trưng của nhiễu màn hình LCD hay LED, được khuếch đại hơn, nhờ đó dễ dàng phát hiện dấu hiệu giả mạo. Hình 2.11 là ví dụ về việc sử dụng bộ lọc tần số cao để làm nổi bật các sọc vân màn hình trong ảnh giả mạo. Một đặc trưng khác là việc chụp gián tiếp qua màn hình làm cho hình ảnh mờ và nhòe hơn nhiều so với việc chụp trực tiếp. Dùng bộ lọc tần số cao thông qua phép biến đổi Fourier giúp cho mô hình có thể phân biệt giữa ảnh mờ và ảnh sắc nét. Đây cũng một đặc trưng quan trọng để phân biệt ảnh giả mạo.



Hình 2.11: Các sọc vân màn hình được làm nổi bật với bộ lọc tần số cao với mặt nạ có bán kính bằng 8

Sau đó, các véc-tơ đặc trưng đầu ra từ hai nhánh nêu trên sẽ được tích hợp với nhau bằng cách sử dụng mô-đun chú ý chéo đa tầng (multi-heads cross-attention). Phương pháp này sử dụng nhiều tầng của mô-đun tự chú ý để tổng hợp và chọn lọc những đặc trưng giữa hai nguồn thông tin: Cơ chế tự chú ý trong từng nhánh nhằm nhấn mạnh những vị trí quan trọng trong mỗi không gian đặc trưng. Cơ chế chú ý chéo giữa hai nhánh để khai thác mối quan hệ tương hỗ giữa thông tin không gian và tần số. Cơ chế này giúp mô hình tự động học được cách chọn lọc và ưu tiên các đặc trưng tốt nhất, qua đó cải thiện hiệu năng nhận dạng.

Như vậy, với cách tiếp cận bằng mạng nơ-ron nhân tạo VGG16 kết hợp với sử dụng phép biến đổi Fourier và bộ lọc tần số cao sẽ là công cụ hiệu quả nâng cao độ chính xác cho bài toán phát hiện giả mạo ảnh căn cước.

Kiến trúc VGG16 kết hợp phép biến đổi Fourier và bộ lọc tần số cao đã mở ra một hướng đi hiệu quả trong việc tăng cường khả năng phát hiện giả mạo trên hình ảnh kỹ thuật số, hỗ trợ hiệu quả cho các công cụ định danh điện tử.

2.5. Kết luận

Chương này đề cập đến các lý thuyết cơ sở và quá trình nghiên cứu các thành phần tạo nên mô hình mạng nơ-ron để phát hiện thẻ căn cước giả mạo. Đầu tiên là lý thuyết về phép biến đổi Fourier rời rạc cho ảnh kỹ thuật số và bộ lọc lý tưởng tần số cao. Thứ hai là lý thuyết về mạng nơ-ron nhân tạo VGG16 và các thành phần của mạng này, cùng với phương pháp huấn luyện và kiểm thử. Thứ ba là lý thuyết về mô-đun tự chú ý. Cuối cùng là nêu phương pháp kết hợp các thành phần kể trên lại với nhau để tạo thành một kiến trúc mạng nơ-ron nhân tạo tổng thể, phát huy hiệu quả từng thành phần, vừa khai thác được đặc trưng không gian bằng VGG16, vừa tận dụng thành phần tần số cao qua phép biến đổi Fourier để phát hiện dấu hiệu giả mạo tinh vi. Chương này đã cung cấp nền tảng lý thuyết đầy đủ và minh họa cách thức kết hợp các thành phần, làm cơ sở triển khai thực nghiệm ở chương sau.

CHƯƠNG 3 - XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH

Như đã đề cập ở chương 1, vấn đề giả mạo thẻ căn cước trên môi trường điện tử là một vấn đề cấp thiết trong bối cảnh chuyển đổi số rộng rãi như hiện nay. Việc một kẻ tấn công sử dụng ảnh thẻ căn cước đã chụp từ trước để sử dụng vào các mục đích phi pháp như mạo danh trên môi trường mạng, gian lận tài chính đang trở thành thách thức lớn. Ở Việt Nam và nhiều quốc gia trên thế giới, thẻ căn cước là giấy tờ tùy thân chính thức và được chấp nhận rộng rãi. Do đó, nếu không có cơ chế kiểm tra tính hợp lệ của thẻ thì sẽ tiềm ẩn nguy cơ lẫn lộn giữa ảnh thẻ thật (được chụp trực tiếp) và ảnh thẻ giả (ảnh chụp lại từ màn hình thiết bị khác), gây ra rất nhiều hệ lụy cho người dùng. Căn cứ vào những lý thuyết đã phân tích ở chương 2, chương này tập trung vào việc xây dựng và đánh giá mô hình phát hiện hình ảnh giả mạo từ dữ liệu thẻ căn cước. Kết quả cần đạt là một mạng nơ-ron nhân tạo kết hợp được ưu điểm của mạng VGG16 và phép biến đổi Fourier, mạng này có kiến trúc đơn giản, tốc độ suy luận tốt và có hiệu quả tốt trong việc phát hiện ảnh thẻ giả mạo. Ngoài ra, mô hình cũng phải có khả năng mở rộng, dễ cải tiến và có thể bảo trì lâu dài.

3.1. Phân tích bài toán phát hiện thẻ căn cước giả mạo

Bài toán phát hiện thẻ căn cước giả mạo là một bài toán phân loại nhị phân (binary classification), trong đó: Ảnh thật là những bức ảnh được chụp trực tiếp và chân thực, tức là thẻ căn cước được đặt trước camera, không có bất kỳ bước trung gian nào gây suy giảm chất lượng. Ảnh giả là ảnh thẻ căn cước hiển thị trên thiết bị điện tử (điện thoại, máy tính bảng…) và chụp lại bằng thiết bị thu nhận ảnh. Đầu vào của bài toán này là một bức ảnh kỹ thuật số của thẻ căn cước định dạng RGB. Đầu ra là một nhãn phân loại rời rạc, trong đó: nhãn âm tương đương với “thật” (“genuine”); nhãn dương tương đương với nhãn “giả mạo” (“spoof”).

Để khẳng định tính đúng đắn và hiệu quả của mô hình đề xuất, quá trình thực nghiệm trong đề án được thiết kế với hai kịch bản thử nghiệm chính: (1) đánh giá so sánh với các nghiên cứu trước đây và (2) nghiên cứu loại bỏ thành phần.

Kịch bản 1: đánh giá so sánh hiệu quả và hiệu năng mô hình. Tiến hành so sánh trực tiếp mô hình đề xuất với hai mô hình tham chiếu: (i) mô hình DLC-2021 cơ sở [4] và (ii) mô hình Bộ dò nhiễu theo biên [8]. Về tiêu chí đánh giá độ chính xác: sử dụng các chỉ số Precision, Recall và Accuracy. Ngoài ra, đánh giá thêm các chỉ số APCER (tỷ lệ phân loại sai ảnh giả) và BPCER (tỷ lệ phân loại sai ảnh thật). Về tiêu chí đánh giá hiệu năng tính toán: đo lường thời gian thực thi trung bình trên CPU, số lượng tham số và độ phức tạp tính toán (FLOPs). Kịch bản này nhằm chứng minh rằng mô hình đề xuất không chỉ đạt độ chính xác tương đương hoặc vượt trội so với các phương pháp hiện có mà còn đảm bảo được tốc độ xử lý nhanh, tiêu tốn ít tài nguyên và phù hợp để triển khai trên các thiết bị sử dụng CPU.

Kịch bản 2: nghiên cứu loại bỏ thành phần. Tiến hành thay đổi hoặc lược bỏ các thành phần cốt lõi trong kiến trúc đề xuất để tạo ra 03 mô hình biến thể nhằm kiểm chứng vai trò của từng bộ phận với ba biến thể là (i) thay thế mô-đun chú ý chéo đa tầng (Multi-head Cross Attention) bằng cơ chế ghép nối đơn giản (ConcatGAP); (ii) Loại bỏ nhánh xử lý tần số (FFT), thay thế bằng kiến trúc hai nhánh không gian VGG16 song song và (iii) Chỉ sử dụng một nhánh đơn duy nhất với kiến trúc khung VGG16, loại bỏ hoàn toàn cơ chế kết hợp và nhánh FFT. Thử nghiệm này nhằm chứng minh vai trò của nhánh FFT: Khẳng định việc bổ sung thông tin miền tần số giúp mô hình nhận diện tốt hơn các nhiễu ảnh giả mạo mà nhánh không gian đơn thuần có thể bỏ sót. Đồng thời chứng minh hiệu quả của mô-đun chú ý chéo: Khẳng định cơ chế này giúp hợp nhất linh hoạt các tín hiệu bổ trợ giữa miền không gian và tần số tốt hơn so với việc ghép nối đơn giản.

3.2. Xây dựng tập dữ liệu

DLC-2021 (Document Liveness Challenge 2021) [4] là một bộ dữ liệu mở được khởi xướng trong khuôn khổ cuộc thi “Document Liveness Challenge” tổ chức năm 2021. Mục tiêu chính của DLC-2021 là cung cấp một tập ảnh thực và giả về các loại giấy tờ tùy thân, bao gồm thẻ căn cước và hộ chiếu để nghiên cứu và phát triển các phương pháp phát hiện giả mạo của các loại giấy tờ này.

Về cấu trúc của bộ dữ liệu DLC-2021, sẽ có bốn thư mục: “or” tương ứng với “original” (ảnh thật); “re” tương ứng với “replay” (ảnh giả chụp từ màn hình); “cc” tương ứng với “color unlaminated copies” (ảnh in màu không ép plastic, sau đó chụp lại từ camera); “cg” tương ứng với “grayscale unlaminated copies” (ảnh in đen trắng không ép plastic, sau đó chụp lại từ camera); Trong mỗi thư mục, có hai thư mục con là “images” (ảnh) và “annotations” (tọa độ chính xác của thẻ căn cước trong mỗi ảnh). Trong mỗi thư mục or/images và re/images, có các thư mục con, mỗi thư mục con đặt tên theo mã_quốc_gia + loại_giấy_tờ, cụ thể:

- alb_id (Thẻ căn cước Albania)
- esp_id (Thẻ căn cước Spain)
- in_id (Thẻ căn cước Finland)
- est_id (Thẻ căn cước Estonia)
- svk_id (Thẻ căn cước Slovakia)
- aze_passport (Hộ chiếu Azerbaijan)
- lva_passport (Hộ chiếu Latvia)
- grc_passport (Hộ chiếu Greece)
- srb_passport (Hộ chiếu Serbia)
- rus_internalpassport (Hộ chiếu quốc tế Nga)

mỗi thư mục này lại chứa từng thư mục con ví dụ re0001, re0002,… tương ứng với từng video hay lô khung hình, trong đó có các hình ảnh JPEG là từng khung hình đã được trích xuất. Ngoài ra, thư mục “annotations” chứa các thông tin bổ như vị trí hình học của tài liệu và khuôn mặt trong từng khung hình. Nhờ đó, có thể trích từng khung hình, lấy vị trí tọa độ chính xác của giấy tờ trong bức ảnh. Tập dữ liệu sử dụng 2 độ phân giải khung hình là $(1080px \times 1920px)$ và $(2160px \times 3840px)$

MIDV-2020 (Mobile Identity Document Video-2020) [7] là một bộ dữ liệu mở về phân tích giấy tờ tùy thân, được xây dựng từ 1.000 giấy tờ thuộc 10 loại tài liệu khác nhau (thẻ căn cước và hộ chiếu của 10 quốc gia). Bộ dữ liệu cung cấp đồng thời 1.000 video clip quay bằng smartphone, 2.000 ảnh scan độ phân giải cao và 1.000 ảnh chụp. Về cấu trúc thư mục, bộ dữ liệu này có cấu trúc thư mục tương tự với DLC-2021. Nhờ phạm vi bao phủ rộng và chú trọng vào tính đa dạng của nội dung cũng như điều kiện chụp, MIDV-2020 thường được dùng như một tập dữ liệu cơ sở cho các bài toán phát hiện, phân loại loại giấy tờ, nhận dạng chữ viết và kiểm tra tính thật - giả của giấy tờ tùy thân.

Trong khuôn khổ của đề án, chỉ có nhãn "or" (ảnh thật) và "re" (ảnh giả chụp từ màn hình) được sử dụng để huấn luyện và đánh giá mô hình. Nhãn "or" sẽ được gọi là ảnh thật (hay nhãn "genuine") và nhãn "re" được gọi là ảnh giả (hay nhãn "spoof"). Chia bộ dữ liệu thành ba phần: huấn luyện, kiểm tra và kiểm thử. Cách chia tuân thủ theo đề xuất của DLC-2021 như sau: Tập huấn luyện và kiểm tra sử dụng 25.980 ảnh thật lấy từ thư mục "MIDV-2020/clips", 19.543 ảnh giả mạo lấy từ 3 thư mục: esp_id, lva_passport, và rus_internalpassport, sau đó chia ngẫu nhiên theo tỉ lệ tập huấn luyện 80%, tập kiểm tra 20%. Tập kiểm thử bao gồm các dữ liệu được liệt kê trong danh sách: "screen_positive_test.lst" và "screen_negative_test.lst", tức 11.006 ảnh giả mạo (nhãn dương) và 16.264 ảnh thật (nhãn âm), những ảnh này được rút ra từ các loại giấy tờ còn lại của DLC-2021 [4]. Các ví dụ tiêu biểu từ bộ dữ liệu được trình bày lần lượt ở hình 3.1 và hình 3.2.

Cấu trúc thư mục mới và số lượng ảnh mỗi thư mục được phân bố như sau:

Tên	Chi tiết	Tổng
Tập huấn luyện	**Ảnh thật** Đường dẫn: "/dlc-2021/train/genuine" Số lượng ảnh: 20784	
	Ảnh giả Đường dẫn: "/dlc-2021/train/spoof"	36419

	Số lượng ảnh: 15635	
Tập kiểm tra	**Ảnh thật** Đường dẫn: "/dlc-2021/val/genuine" Số lượng ảnh: 5196	9104
	Ảnh giả Đường dẫn: "/dlc-2021/val/spoof" Số lượng ảnh: 3908	
Tập kiểm thử	**Ảnh thật** Đường dẫn: "/dlc-2021/test/genuine" Số lượng ảnh: 11006	27270
	Ảnh giả Đường dẫn: "/dlc-2021/test/spoof" Số lượng ảnh: 16264	

Bảng 3.1: Thông tin về tập dữ liệu huấn luyện và kiểm thử



Hình 3.1: Một số hình ảnh có nhãn "ảnh thật"



Hình 3.2: Một số hình ảnh có nhãn "ảnh giả"

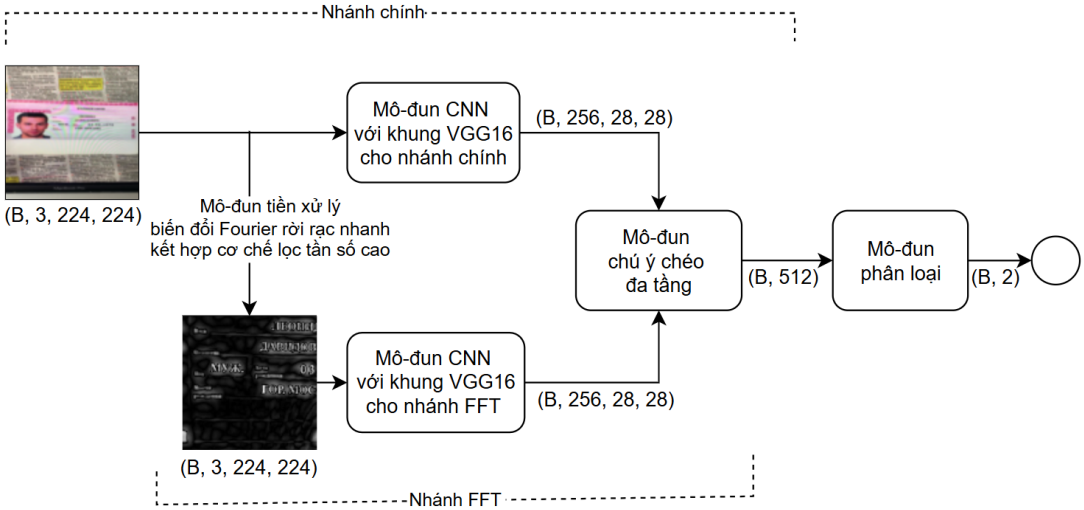
3.3. Xây dựng kiến trúc mạng học sâu.

Đề án sử dụng ngôn ngữ lập trình Python và các thư viện học sâu PyTorch, scikit-learn để xây dựng kiến trúc mạng học sâu cho bài toán phát hiện ảnh thẻ căn cước giả mạo.

3.3.1. Kiến trúc tổng quan của mạng

Kiến trúc mạng đề xuất gồm hai nhánh: nhánh thứ nhất để xử lý với ảnh gốc (được gọi là nhánh chính), song song với nhánh này là nhánh xử lý với phép biến đổi Fourier (được gọi là nhánh FFT). Hai nhánh này sẽ được kết nối lại với nhau bằng một mô-đun tổng hợp thông tin (mô-đun chú ý chéo đa tầng). Và cuối cùng phân loại đầu ra bằng một mô đun phân loại. Sơ đồ tổng thể về luồng của mô hình được minh họa trong hình 3.3.

Đầu vào của mạng là một lô gồm B ảnh có kích thước tiêu chuẩn $(1080px \times 1920px)$ hoặc $(2160px \times 3840px)$ (định dạng 3 kênh màu RGB), và chuyển phổ điểm ảnh từ $[0; 255]$ trở thành $[0; 1]$.



Hình 3.3: Sơ đồ tổng thể về mô hình mạng VGG16 kết hợp phép biến đổi Fourier

Trong nhánh chính, ảnh đầu vào (ảnh gốc, định dạng RGB) sẽ được tiền xử lý bằng cách cắt một hình vuông quanh tâm căn cước với kích thước lớn (896×896) nhằm cắt được nhiều chi tiết tổng thể của ảnh, rồi đưa vào mạng CNN với khung VGG16. Các lớp tích chập giúp mô hình khả năng học được các đặc trưng như cạnh, hình dạng, kết cấu. Đối với bài toán phân biệt hình ảnh căn cước thật – giả, các đặc trưng từ nhánh này giúp mô hình nắm bắt hình thái tổng thể của đối tượng trong ảnh. Tiếp theo. tiến hành chuẩn hóa đầu vào theo chuẩn ImageNet [6] với trung bình $\mu = [0.485, 0.456, 0.406]$ và độ lệch chuẩn $\sigma = [0.229, 0.224, 0.225]$), hoán vị chiều, điều chỉnh kích thước ảnh về kích thước (B, 3, 224, 224) và đưa về định dạng tensor. Sau đó đưa vào mạng nơ-ron khung VGG16 (cửa sổ: 3×3, lớp gộp max-pooling sau mỗi khối), chỉ giữ lại đến hết khối thứ 3 để giảm thiểu kích thước của mạng. Đầu ra của nhánh là một tensor đặc trưng với kích thước (B, 256, 28, 28).

Ở nhánh phụ, còn gọi là nhánh FFT, ảnh sẽ được tiền xử lý trước bằng cách cắt quanh tâm căn cước một ô vuông kích thước nhỏ hơn (448×448), chuyển thành ảnh xám và xử lý qua phép biến đổi Fourier rời rạc hai chiều và bộ lọc lý tưởng tần số cao để giữ lại thành phần tần số cao (thực nghiệm chọn bán kính mặt nạ $H = 8$), mục đích là tăng cường các đường nét nhỏ, khó nắm bắt như vân sọc nhiễu, và làm nổi bật hơn sự chênh lệch về độ sắc nét trong ảnh thật và ảnh giả, còn các thành phần tần số thấp - biểu diễn thông tin tổng thể - sẽ bị loại bỏ. Sau đó, ảnh phổ tần số sau khi áp dụng bộ lọc được biến đổi ngược trở lại miền không gian thông qua phép biến đổi Fourier ngược, tạo thành ảnh mới chỉ còn những đặc trưng tần số cao. Ảnh này được đưa qua nhánh mạng nơ-ron VGG16 thứ hai, nhánh này giúp mô hình nhận biết một biểu diễn bổ sung, hỗ trợ phát hiện các dấu hiệu không dễ quan sát trong không gian thị giác thông thường. Đầu ra là tensor đặc trưng cũng có kích thước (B, 256, 28, 28).

Sau khi có được hai tensor đặc trưng, tiếp tục đưa hai tensor này vào mô-đun chú ý chéo đa tầng để mô hình học các tương tác, mối liên hệ giữa đặc trưng miền không gian (đặc trưng từ nhánh chính) và những đặc trưng được khuếch đại bằng bộ

lọc tần số cao (đặc trưng từ nhánh FFT). Đầu ra là tensor có kích thước (B, 512) tích hợp thông tin hai nhánh, sẵn sàng đưa vào tầng kết nối đầy đủ để phân loại nhị phân.

Tầng phân loại nhị phân có đầu vào: tensor đặc trưng có độ dài 512 từ bước trên. Xử lý qua ba lớp kết nối đầy đủ (có thể kèm lớp chuẩn hóa tích chập), rồi đến lớp kích hoạt softmax để trả về đầu ra cho 2 lớp ("genuine" và "spoof").

Về ưu điểm của mô hình hai nhánh VGG16 kết hợp phép biến đổi Fourier, việc kết hợp đặc trưng hình ảnh theo cả hai miền không gian và tần số giúp tăng khả năng phân biệt các mẫu khó, giảm thiểu số tham số cần huấn luyện, tăng tốc độ huấn luyện cho mô hình.

Chi tiết xử lý trong từng mô-đun được trình bày trong các phần dưới.

3.3.2. Mô-đun tiền xử lý: phép biến đổi Fourier rời rạc nhanh kết hợp cơ chế lọc tần số cao

Để phục vụ cho nhánh FFT, cần chuẩn bị đầu vào cho nhánh này. Sau khi xử lý, sẽ thu được ảnh đã lọc tần số cao. Tiến hành áp dụng tuần tự các bước sau:

- Bước 1: Cắt và chuẩn hóa vùng cần quan tâm

Thay vì xử lý toàn bộ khung hình, tiến hành xác định vị trí tâm của thẻ căn cước trong tấm ảnh, tiếp theo cắt vùng ô vuông nằm ở tâm ảnh với kích thước như sau: một vùng lớn $L_1 = 896 \times 896$ và một vùng nhỏ hơn $L_2 = 448 \times 448$. Cả hai vùng cắt sau đó đều được chỉnh kích thước về 224×224, sẵn sàng đưa vào mô hình để xử lý. Ảnh được cắt theo L_1 được đưa vào nhánh không gian (VGG16), trong khi ảnh được cắt theo L_2 được đưa vào nhánh tần số (FFT). L_1 đóng vai trò cung cấp vùng ngữ cảnh rộng hơn cho các đặc trưng không gian, còn L_2 tập trung vào các chi tiết vi mô (ví dụ: vân nhiễu), từ đó giúp mô hình học được các đặc trưng phân biệt rõ hơn giữa hình ảnh giấy tờ định danh thật và giả mạo.

- Bước 2: Biến đổi Fourier rời rạc nhanh, dịch ảnh và áp dụng bộ lọc

Đặt $H = 224, W = 224$ lần lượt là kích thước chiều dài và chiều rộng của ảnh. Tiến hành chuyển ảnh thành ảnh xám theo công thức (3.1)

$$x_{gray}(i,j) = 0.2989R(i,j) + 0.5870G(i,j) + 0.1140B(i,j) \quad (3.1)$$

với $i \in [0, H-1];\ j \in [0, W-1];\ x \in [0\ ,1]$

Kết quả thu được tensor x_{gray} kích thước $(B, 1, H, W)$. Áp dụng phép biến đổi Fourier rời rạc nhanh và phép dịch FFTShift lên ảnh xám bằng công thức (3.2) và (3.3)

$$X = FFT\left(x_{gray}\right) \quad (3.2)$$

$$Xs(u,v) = FFTShift\left(X(u,v)\right) \quad (3.3)$$

với $u \in \left[-\frac{H}{2}, \frac{H}{2} - 1\right];\ v \in \left[-\frac{W}{2}, \frac{W}{2} - 1\right]$

Xây dựng lưới toạ độ U, V với $U_{mn} = m - \frac{H}{2}, V_{mn} = n - \frac{W}{2}$ trong đó $m \in [0, H-1];\ n \in [0, W-1]$.

Tạo mặt nạ lọc dạng nhị phân theo công thức (3.4) với bán kính của mặt nạ $r = 8$:

$$mask2d(m,n) = \begin{cases} 1, U_{mn}^2 + V_{mn}^2 \geq r^2 \\ 0, U_{mn}^2 + V_{mn}^2 < r^2 \end{cases} \quad (3.4)$$

Áp dụng mặt nạ $mask2d$ lên ảnh để giữ lại các thành phần tần số cao theo công thức (3.5)

$$X_{hp}^s = Xs \odot mask2d \quad (3.5)$$

- Bước 3: Dịch ngược và biến đổi ngược:

Tiếp tục thực hiện dịch ngược trong miền tần số (IFFTShift), sau đó áp dụng Biến đổi Fourier ngược (IFFT) để tái tạo lại ảnh thành phần tần số cao theo (3.6)

$$x_{hp} = IFFT\left(IFFTShift\left(X_{hp}^{s}\right)\right) \tag{3.6}$$

Kết quả thu được x_{hp} là một tensor phức gồm phần thực và phần ảo. Để sử dụng cho mô hình học sâu, lấy độ lớn (magnitude) của tensor này theo công thức (3.7)

$$mag\left(x_{hp}\right) = \left|Re\left(x_{hp}\right) + i\ Im\left(x_{hp}\right)\right| \tag{3.7}$$

Cuối cùng, áp dụng biến đổi log để nén miền giá trị của ảnh tần số cao. Phép biến đổi này giúp giảm ảnh hưởng của các giá trị biên quá lớn và làm nổi bật chi tiết nhỏ trong ảnh.

$$hp_{log} = \ln\left(1 + mag\left(x_{hp}\right)\right) \tag{3.8}$$

- Bước 4: Chuẩn hóa cực tiểu – cực đại

Sau khi thực hiện các phép biến đổi Fourier, ảnh có các giá trị không nằm trong khoảng cố định (có thể rất lớn hoặc rất nhỏ). Vì vậy, chuẩn hóa cực tiểu – cực đại giúp nén giá trị ảnh về khoảng [0, 1], giúp mô hình học sâu dễ xử lý hơn. Tiến hành làm phẳng toàn bộ hp_{log} về tensor bằng cách tính m_{min} theo (3.9). Sau đó chuẩn hóa theo công thức (3.10).

$$m_{min} = \min_{(i,j)} hp_{log}(i,j)\ m_{max} = \max_{(i,j)} hp_{log}(i,j) \tag{3.9}$$

$$hp_{gray} = \frac{hp_{log} - m_{min}}{m_{max} - m_{min} + \ \varepsilon} \tag{3.10}$$

Với $\varepsilon = \ 10^{-8}$ để tránh chia cho 0. Kết quả thu được tensor (B, 1, 224, 224) nằm trong khoảng $[0\ ,1]$. Do mạng VGG16 yêu cầu đầu vào 3 kênh, tiến hành lặp 3 lần tensor bằng công thức (3.11) để thu được đầu vào hợp lệ có kích thước $(B, 3, 224, 224)$.

$$hp_{rgb} = tile\left(hp_{gray}, [1, 3, 1, 1]\right) \tag{3.11}$$

Cuối cùng chuẩn hóa theo chuẩn ImageNet với trung bình $\mu = [\mu_R, \mu_G, \mu_B] = [0.485, 0.456, 0.406]$ và độ lệch chuẩn $\sigma = [\sigma_R, \sigma_G, \sigma_B] = [0.229, 0.224, 0.225]$ theo từng kênh R, G, B như sau:

$$hp_{rgb_{norm}} = \frac{hp_{rgb} - \mu}{\sigma} \tag{3.12}$$

Sau 4 bước thu được tensor $(B, 3, 224, 224)$ sẵn sàng đưa vào nhánh FFT.

3.3.3. Mô-đun mạng nơ-ron tích chập khung sử dụng kiến trúc VGG16

Trong mô-đun này, sử dụng kiến trúc VGG16 và chỉ lấy 3 lớp đầu tiên trong mạng để giảm thiểu kích thước. Kích thước chi tiết các lớp được thể hiện trong bảng 3.2.

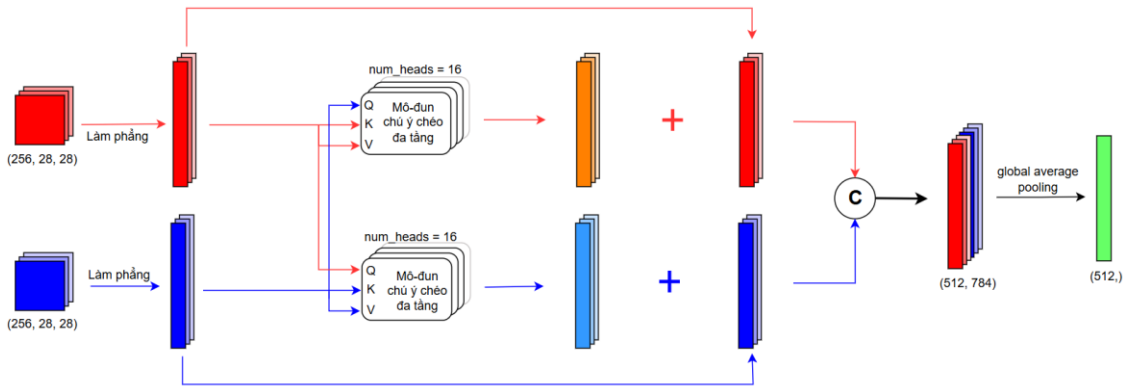
Khối	**Lớp**	**Thông số kỹ thuật**	**Kích thước đầu ra**
Đầu vào	-	-	(B, 3, 224, 224)
1	Conv2D	Conv (3×3, 64)	(B, 64, 224, 224)
	Conv2D	Conv (3×3, 64)	(B, 64, 224, 224)
	MaxPool	MaxPool (2×2, s=2)	(B, 64, 112, 112)
2	Conv2D	Conv (3×3, 128)	(B, 128, 112, 112)
	Conv2D	Conv (3×3, 128)	(B, 128, 112, 112)
	MaxPool	MaxPool (2×2, s=2)	(B, 128, 56, 56)
3	Conv2D	Conv (3×3, 256)	(B, 256, 56, 56)
	Conv2D	Conv (3×3, 256)	(B, 256, 56, 56)
	Conv2D	Conv (3×3, 256)	(B, 256, 56, 56)
	MaxPool	MaxPool (2×2, s=2)	(B, 256, 28, 28)

Bảng 3.2: Chi tiết cấu trúc mạng nơ-ron tích chập khung sử dụng kiến trúc VGG16

Cả hai nhánh đều dùng phương pháp học chuyển giao với tham số được huấn luyện sẵn trên trên ImageNet [6] nhằm tận dụng khả năng trích đặc trưng tổng quát sẵn có và tiết kiệm thời gian huấn luyện.

3.3.4. Mô-đun chú ý chéo đa tầng

Sau khi có hai tensor đặc trưng đều có kích thước $(B, 256, 28, 28)$, để mạng có khả năng học mối quan hệ tương hỗ giữa đặc trưng trong ảnh gốc và đặc trưng và thông tin tần số cao, cần xây dựng một mô-đun sử dụng kỹ thuật chú ý chéo đa tầng. Mô-đun này cho phép mỗi đơn vị trong ảnh RGB "chú ý" đến những vị trí tương ứng hoặc liên quan trong ảnh đã lọc tần số cao, và ngược lại, qua đó tạo mối liên kết và tổng hợp từ hai nguồn ngữ nghĩa. Sau cùng, xuất ra một tensor có kích thước $D_{out} = 512$ để đưa vào mô-đun phân loại. Sơ đồ các lớp trong mô-đun chú ý chéo đa tầng được thể hiện trong hình 3.4.



Hình 3.4: Minh họa các lớp trong mô-đun chú ý chéo đa tầng

Giả sử đầu vào của mô-đun này là hai tensor từ bước 3.3.3 có dạng $F_{RGB} \in \mathbb{R}^{B\times C\times H\times W}$, $F_{FFT} \in \mathbb{R}^{B\times C\times H\times W}$ trong đó B là kích thước lô, $C = 256, H = W = 28$, quá trình hoạt động của mô-đun này như sau:

1. Chuyển đổi định dạng của tensor đầu vào từ $B \times C \times H \times W$ thành $(HW) \times B \times C$ bằng cách sử dụng phép làm phẳng (flatten) và hoán vị chiều (permute) để sử dụng với mô-đun chú ý chéo đa tầng.

$$F_{RGB}^{flat} = Permute\big(Flatten(F_{RGB})\big) \in \mathbb{R}^{(HW)\times B\times C} \quad (3.13)$$

$$F_{FFT}^{flat} = Permute\big(Flatten(F_{FFT})\big) \in \mathbb{R}^{(HW)\times B\times C} \quad (3.14)$$

2. Tạo các tensor đầu vào Query, Key, Value

$$Q_{RGB} = F_{RGB}^{flat}\, W_{Q\,|\,RGB} \quad (3.15)$$

$$K_{RGB} = F_{RGB}^{flat}\, W_{K\,|\,RBG} \quad (3.16)$$

$$V_{RGB} = F_{RGB}^{flat}\, W_{V\,|\,RBG} \quad (3.17)$$

Và

$$Q_{FFT} = F_{FFT}^{flat}\, W_{Q\,|\,FFT} \quad (3.18)$$

$$K_{FFT} = F_{FFT}^{flat}\, W_{K\,|\,FFT} \quad (3.19)$$

$$V_{FFT} = F_{FFT}^{flat}\, W_{V\,|\,FFT} \quad (3.20)$$

3. Tính giá trị chú ý chéo đa tầng:

Để mạng có khả năng ghi nhớ được nhiều mối quan hệ liên kết, sử dụng mạng chú ý chéo đa tầng (multi-heads cross-attention) với $n_{heads} = 16$ và $d_k = d_v = \frac{C}{n_{heads}}$.

a. Tính giá trị chú ý chéo của ảnh FFT đến ảnh RGB:

$$Attention_{RGB\leftarrow FFT} = softmax\left(\frac{Q_{RGB}K_{FFT}^{T}}{\sqrt{d_k}}\right)V_{FFT} \in \mathbb{R}^{(HW)\times B\times d_v} \quad (3.21)$$

b. Tính giá trị chú ý chéo của ảnh RGB đến ảnh FFT:

$$Attention_{FFT\leftarrow RGB} = softmax\left(\frac{Q_{FFT}K_{RGB}^{T}}{\sqrt{d_k}}\right)V_{RGB} \in \mathbb{R}^{(HW)\times B\times d_v} \quad (3.22)$$

4. Lớp tuyến tính kết hợp kết quả attention

Sau khi tính $Attention_{RGB\leftarrow FFT}$ và $Attention_{FFT\leftarrow RGB}$, mỗi mô-đun Attention thu được n_{heads} tầng, mỗi khối có kích thước $(HW) \times B \times d_v$. Để có kích thước $(HW) \times B \times C$ như ban đầu, cần phải làm phẳng chúng bằng phép làm phẳng (flatten). Sau đó, sử dụng một lớp tuyến tính để kết hợp đầu ra của các tầng.

$$Z_{RGB} = \text{Linear}\big(\text{Flatten}(Attention_{RGB\leftarrow FFT})\big) \in \mathbb{R}^{(HW)\times B\times C} \quad (3.23)$$

$$Z_{FFT} = \text{Linear}\big(\text{Flatten}(Attention_{FFT\leftarrow RGB})\big) \in \mathbb{R}^{(HW)\times B\times C} \quad (3.24)$$

5. Hậu xử lý các tensor.

Áp dụng kỹ thuật cộng trực tiếp đầu vào với đầu ra của mô-đun học nhằm bảo toàn thông tin ban đầu trong trường hợp mô hình chưa học tốt:

$$Z_{RGB} = Z_{RGB} + F_{RGB}^{flat} \quad (3.25)$$

$$Z_{FFT} = Z_{FFT} + F_{FFT}^{flat} \quad (3.26)$$

Cách làm này giúp giảm nguy cơ mất mát đặc trưng quan trọng trong quá trình huấn luyện. Sau đó, tiếp tục hoán vị chiều của hai tensor để chuẩn bị cho bước tiếp theo:

$$Z_{RGB} = Permute(Z_{RGB}) \in \mathbb{R}^{B\times C\times (HW)} \quad (3.27)$$

$$Z_{FFT} = Permute(Z_{FFT}) \in \mathbb{R}^{B\times C\times (HW)} \quad (3.28)$$

Áp dụng phép gộp trung bình toàn cục (global average pooling) để thu được tensor kích thước $B \times C$

$$Z_{RGB} = GAP(Z_{RGB}, dim = 2) \in \mathbb{R}^{B\times C} \quad (3.29)$$

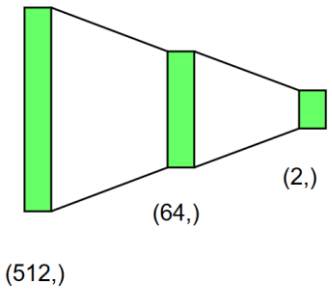
$$Z_{FFT} = GAP(Z_{FFT}, dim = 2) \in \mathbb{R}^{B \times C} \quad (3.30)$$

Cuối cùng nối hai tensor để thu được một tensor chung:

$$\mathrm{Z} = \mathrm{Concat}(Z_{RGB}, Z_{FFT}) \in \mathbb{R}^{B \times 2C} \quad (3.31)$$

3.3.5. Mô-đun phân loại

Sau khi hoàn tất quá trình kết hợp đặc trưng từ hai nhánh chính và nhánh FFT, thu được tensor đầu ra Z có kích thước $(B, 2C)$, trong đó $C = 256$ là số chiều đặc trưng từ mỗi nhánh. Tensor Z sau đó được đưa vào một mô-đun phân loại gồm ba tầng kết nối đầy đủ, có vai trò trích lọc và phân tách đặc trưng cuối cùng để phục vụ cho nhiệm vụ phân loại. Minh họa về các lớp trong mô-đun này được thể hiện trong hình 3.5.



Hình 3.5: Minh họa các lớp trong mô-đun phân loại

Cấu trúc của mô-đun này như sau:

1. Khối kết nối đầy đủ 1:

$$h_1 = DropOut\big(ReLU\big(BatchNorm(W_1 Z + b_1)\big), \varepsilon\big) \quad (3.32)$$

với $W_1 \in \mathbb{R}^{K \times 2C}$ (chọn $K = 512$); b_1 là véc-tơ hệ số chênh lệch (bias); ε là tỉ lệ lượng nơ-ron trong lớp đó bị loại bỏ (có thể chọn $\varepsilon = 0.5$)

2. Khối kết nối đầy đủ 2:

$$h_2 = DropOut\big(ReLU\big(BatchNorm(W_2 h_1 + b_2)\big), \varepsilon\big) \quad (3.33)$$

với $W_2 \in \mathbb{R}^{H \times K}$ là tham số của lớp (có thể chọn $H = 64$); b_2 véc-tơ hệ số chênh lệch (bias); ε là tỉ lệ lượng nơ-ron trong lớp đó bị loại bỏ (có thể chọn $\varepsilon = 0.5$)

3. Khối kết nối đầy đủ 3:

$$output = \ h_3 \ = W_3 h_2 + b_3 \tag{3.34}$$

với $W_3 \in \mathbb{R}^{2 \times H}$ là tham số của lớp, trong đó 2 là số lượng đầu ra, tương ứng với nhãn "thật" và "giả"; b_3 là véc-tơ hệ số chênh lệch (bias).

3.4. Hàm mất mát và quá trình tối ưu hóa

Sau khi qua mô-đun phân loại, đầu ra của mô hình là một tensor có hai phần tử, tương ứng 'thật và 'giả, giả sử $z = (z_0, z_1)$ và nhãn thực tế $y \in \{0,1\}$. Áp dụng hàm Softmax để chuyển đổi thành xác suất:

$$p_i = \frac{e^{z_i}}{e^{z_0} + e^{z_1}}, \qquad với\ i \in \{0,1\} \tag{3.35}$$

Hàm mất mát được sử dụng là Cross-Entropy Loss nhị phân:

$$L = \frac{1}{B} \sum_{n=1}^{B} l(z_n, y_n) \tag{3.36}$$

$$l(z_n, y_n) = CELoss(z_n, y_n) = -\log\left(p_{y_n}\right) \tag{3.37}$$

Sau đó, mô hình sẽ tính tensor lượng giảm (gradient) dựa vào hàm mất mát vừa tìm được. Công thức tìm tensor tham số tại bước thứ t như sau:

$$g_t = \frac{\partial l}{\partial w_t} \tag{3.38}$$

Tiếp theo, tiến hành tối ưu hóa hàm mất mát theo phương pháp Adam. Đặt hệ số tốc độ học (learning rate) là $\alpha = 10^{-4}$ và $\varepsilon = 10^{-8}$ để tránh mẫu số bằng 0, công thức tối ưu hóa như sau:

$$w_t = w_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{3.39}$$

Với:

$$\hat{m}_t = \frac{m_t}{1 - (\beta_1)^t} \; ; \; m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$\hat{v}_t = \frac{v_t}{1 - (\beta_2)^t} \; ; \; v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t)^2$$

Trong đó, m_t là hàm tích lũy moment bậc nhất, và v_t là hàm tích lũy moment bậc hai. g_t là tensor gradient. $\beta_1 \in [0,1)$ thường có giá trị $\beta_1 = 0.9, \beta_2 \in [0,1)$ thường có giá trị $\beta_2 = 0.999$. Khi $t = 0$, hàm tích lũy moment bậc nhất và hàm tích lũy moment bậc hai có giá trị khởi tạo là $m_0 = 0$ và $v_0 = 0$.

Với cách thiết lập này, phương pháp Adam là sự kết hợp những ưu điểm từ hai thuật toán tối ưu hóa RMSProp và SGD, nhờ đó mô hình có thể học được những đặc trưng phức tạp từ hai nhánh, quá trình hội tụ nhanh mà không quá bị quá khớp vào tập huấn luyện.

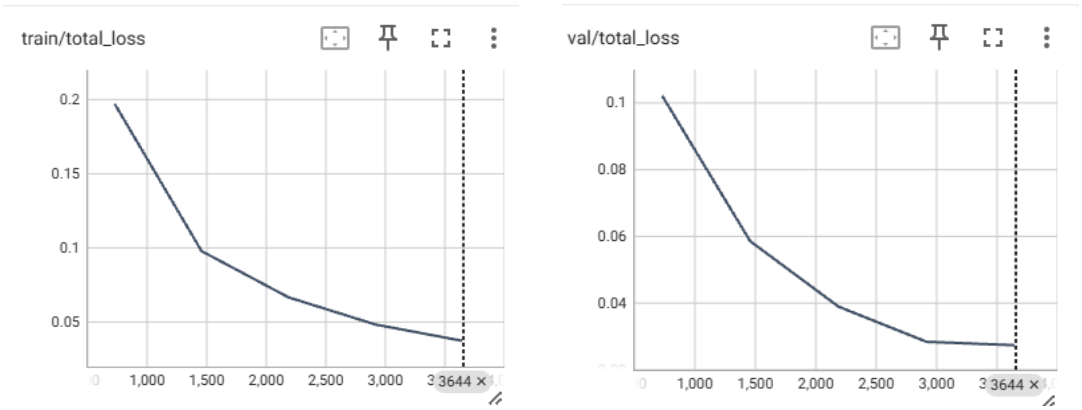
3.5. Huấn luyện và đánh giá mô hình.

3.5.1. Cài đặt và huấn luyện mô hình.

Trong quá trình huấn luyện, tiến hành thiết lập kích thước lô (batch size) $B = 64$. Dữ liệu được nạp song song qua 7 luồng (workers) để tối ưu hóa tốc độ đọc và tiền xử lý, đồng thời áp dụng phép hoán vị ngẫu nhiên (shuffle) trước khi xếp các lô để đảm bảo tính đa dạng và giảm thiểu sự phụ thuộc trình tự mẫu, qua đó tăng tính khái quát và giúp mô hình tránh học các mẫu lặp lại một cách máy móc. Quá trình

huấn luyện được triển khai trên nền tảng Google Colab với GPU L4 (24 GB bộ nhớ GDDR6, Hỗ trợ đầy đủ CUDA 11.x+, cuDNN và TensorRT) [26]. Với việc tích hợp CUDA driver và PyTorch Lightning, GPU này được truy cập trực tiếp qua API torch.cuda trong quá trình huấn luyện, giúp giảm đáng kể thời gian chạy so với CPU truyền thống.

Mô hình kết hợp giữa VGG16 và nhánh xử lý tần số (FFT Branch) đã được huấn luyện trong 5 chu kỳ (epochs), với tổng cộng 3644 bước (steps), sử dụng khoảng 20.5 GB VRAM GPU để tăng tốc xử lý. Trong quá trình huấn luyện, mô hình sử dụng tối ưu hoá AdamW với tốc độ học cố định $lr = 10^{-5}$. Trong quá trình huấn luyện mạng, tiến hành theo dõi giá trị của hàm mất mát. Trong quá trình huấn luyện, mô hình sử dụng chiến lược dừng sớm (early stopping) để tránh quá khớp. Ở mỗi epoch, nếu giá trị hàm mất mát của hàm kiểm tra không giảm so với giá trị tốt nhất trước đó trong 2 chu kỳ liên tiếp (patience = 2), quá trình huấn luyện sẽ tự động dừng lại. Xu hướng thay đổi của hàm mất mát của tập dữ liệu huấn luyện và tập dữ liệu kiểm tra được biểu thị trong hình 3.6.



Hình 3.6: Biểu đồ thể hiện giá trị của hàm mất mát của tập dữ liệu huấn luyện (trái) và tập dữ liệu kiểm tra (phải) trong quá trình huấn luyện mạng

Biểu đồ thể hiện rõ xu hướng giảm ổn định của hàm mất mát của tập dữ liệu huấn luyện và kiểm tra theo thời gian huấn luyện. Giai đoạn đầu (khoảng 1500 bước) hàm mất mát của tập kiểm tra giảm rất nhanh từ giá trị ban đầu khoảng 0.2 xuống dưới 0.1. Đây là giai đoạn mà mô hình tiếp thu những đặc trưng cơ bản từ dữ liệu

huấn luyện. Giai đoạn giữa (1500 bước tiếp theo) tốc độ giảm chậm lại, dao động nhẹ xung quanh mức 0.1 – 0.03, cho thấy mô hình đang học những đặc trưng phức tạp hơn. Giai đoạn cuối (600 bước) hàm mất mát dao động rất nhẹ xung quanh 0.02. Tổng thời gian huấn luyện khoảng 50 phút. Mức độ hội tụ ổn định, không xuất hiện dao động bất thường hoặc tăng giá trị mất mát trở lại. Mô hình đạt hiệu suất huấn luyện tốt, thể hiện qua đường cong hàm mất mát giảm đều và hội tụ nhanh.

3.5.2. Đánh giá mô hình.

Sau khi hoàn tất giai đoạn huấn luyện, sử dụng tập dữ liệu kiểm thử theo đúng khuôn mẫu của DLC-2021 [4] để đánh giá hiệu năng mô hình. Việc đánh giá được tiến hành với các chỉ số: precision, recall và accuracy. Việc đánh giá các chỉ số này nhằm đồng bộ với các mô hình tham chiếu như DLC-2021 cơ sở [4] và phương pháp dò nhiễu theo biên [8]. Ngoài ra, các chỉ số FLOPs, số lượng tham số (Param), Tốc độ suy luận trung bình (Speed) cũng được đánh giá. Kết quả kiểm thử các chỉ số trên ba mô hình: DLC-2021 cơ sở, Bộ dò nhiễu theo biên và VGG16+FFT được thể hiện trong bảng 3.3.

Model	**Precision (%)**	**Recall (%)**	**Accuracy (%)**
DLC-2021 cơ sở	85.89	89.03	89.67
Bộ dò nhiễu theo biên	93.5	90.2	92.1
VGG16+FFT	90.62	92.49	91.71

Bảng 3.3: Kết quả đo trên mô hình DLC-2021 cơ sở, Bộ dò nhiễu theo biên và VGG16+FFT

Với cùng bộ dữ liệu và phương pháp đánh giá, nhìn chung, cả ba mô hình đều cho kết quả tốt. VGG16+FFT có accuracy đạt 91.71%, precision đạt 90,62% và recall đạt 92.49%, vượt trội hơn so với mô hình DLC-2021 cơ sở. Khi so sánh với Bộ dò nhiễu theo biên, VGG16+FFT chỉ kém nhẹ về accuracy (91.71% so với 92.1%) và precision (90.62% so với 93.5%), nhưng đạt recall cao hơn (92.49% so với 90.2%), cho thấy mô hình đề xuất ưu tiên khả năng phát hiện đầy đủ các mẫu giả mạo trong

khi vẫn duy trì độ chính xác tổng thể. Nhìn chung, các kết quả ở Bảng 3.3 khẳng định VGG16+FFT là một phương án mô hình hóa hiệu quả và ổn định cho bài toán phát hiện giả mạo trên tập dữ liệu DLC-2021.

Để đánh giá tốc độ thực thi của mô hình, tiến hành đo thời gian xử lý trung bình trên hai thư mục grc_passport/01.or0004 (thư mục chứa ảnh hộ chiếu Greece thật) và grc_passport/01.re0001 (thư mục chứa ảnh hộ chiếu Greece giả mạo). CPU Intel Xeon @ 2.20 GHz (x86-64), 6 nhân / 12 luồng được sử dụng làm môi trường kiểm thử cho từng mô hình. Trên mỗi thư mục, các mô hình được chạy với cùng quy trình tiền xử lý và cấu hình suy luận như trong kiến trúc mà mô hình đề xuất; thời gian xử lý cho từng ảnh được ghi nhận, sau đó lấy trung bình để thu được tốc độ suy luận tính theo giây/ảnh. Kết quả tổng hợp được trình bày trong Bảng 3.4.

Model	**grc_passport/01.or0004**	**grc_passport/01.re0001**
DLC-2021 cơ sở	0.147	0.151
Bộ dò nhiễu theo biên	-	-
VGG16+FFT	0.120	0.120

Bảng 3.4: Kết quả thời gian thực thực thi trung bình trên CPU Intel Xeon @ 2.20 GHz (x86-64), 6 nhân / 12 luồng

Kết quả cho thấy, trên CPU Intel Xeon @ 2.20 GHz (x86-64), 6 nhân/12 luồng, VGG16+FFT tốn 0.120 s để xử lý một ảnh cho cả trường hợp ảnh thật và giả mạo, vượt trội so với DLC-2021 cơ sở (0.147–0.151 s/ảnh trên cùng các tệp). Thời gian chạy của mô hình Bộ dò nhiễu theo biên không có sẵn trong cấu hình này (Bảng 2).

Về số lượng tham số, VGG16+FFT chỉ sử dụng khoảng 4 triệu tham số, chiếm khoảng 42MB, ít hơn rất nhiều so với VGG16 gốc. Mô hình VGG16 + FFT có độ phức tạp khoảng 40,95 GFLOPs, chủ yếu do kiến trúc hai nhánh kết hợp với mô-đun chú ý chéo đa tầng, vốn yêu cầu nhiều phép biến đổi số cường độ cao. VGG16 + FFT đạt BPCER (mẫu thật bị phân loại nhầm thành giả mạo) thấp, chỉ 9.03% và APCER

(mẫu giả mạo bị phân loại nhầm thành thật) ở mức 7.51%, cho thấy mô hình có độ nhạy cao với các dấu hiệu giả mạo.

Tóm lại, trong cùng giao thức đánh giá, VGG16+FFT đạt kết quả tốt, độ chính xác xấp xỉ phương pháp dò nhiễu theo biên và tốt hơn đáng kể so với DLC-2021 cơ sở trên các mẫu đã đo. Những kết quả này cho thấy mô hình đề xuất phù hợp với các kịch bản triển khai đòi hỏi cả độ tin cậy và suy luận hiệu quả trên CPU.

3.5.3. Nghiên cứu loại bỏ thành phần

Tất cả các khảo sát loại bỏ thành phần đều tuân theo cùng một bộ dữ liệu huấn luyện, kiểm tra, kiểm thử và phương pháp đánh giá (theo DLC-2021 [4]). Các chỉ số đanh giá bao gồm precision, recall và accuracy. Lấy VGG16+FFT (mô hình đề xuất) với precision 90.62%, recall 92.49%, accuracy 91.71% làm mốc tham chiếu.

Các khảo sát sau đã được thực hiện:

1. Thay mô-đun chú ý chéo đa tầng bằng mô đun ghép nối đơn giản (ConcatGAP): Ghép nối đơn giản rồi áp dụng Global Average Pooling để trộn hai nhánh không gian và nhánh FFT lại với nhau.

2. Thay VGG16+FFT bằng hai kênh VGG16: Loại bỏ nhánh FFT; thay bằng hai nhánh không gian (VGG16).

3. Chỉ sử dụng một nhánh đơn sử dụng kiến trúc khung VGG16: Loại bỏ VGG16+FFT và mô-đun chú ý chéo, chỉ giữ một nhánh VGG16 không gian, theo sau là cùng mô-đun phân loại tương tự trong mô hình tham chiếu. Tất cả thiết lập huấn luyện và đánh giá được giữ nguyên.

Kết quả kiểm thử của từng phương pháp được thể hiện ở bảng 3.5.

Model	**Precision** (%)	**Recall** (%)	**Accuracy** (%)
Thay mô-đun chú ý chéo đa tầng bằng mô-đun ghép nối đơn giản	91.29	83.44	88.10
Thay VGG16+FFT bằng hai kênh VGG16	91.38	84.80	88.73
Chỉ sử dụng một nhánh đơn sử dụng kiến trúc khung VGG16	87.67	86.35	87.48

Bảng 3.5: Đánh giá kết quả các nghiên cứu loại bỏ thành phần

Phương pháp sử dụng mô-đun ghép nối đơn giản (ConcatGAP) làm giảm precision xuống 91,29%, recall xuống 83,44% và accuracy xuống 88,10%. Việc mất sự căn chỉnh linh hoạt giữa các miền, vốn có nhờ mô-đun chú ý chéo khiến các trường hợp giả mạo bị bỏ sót. Điều này cho thấy cơ chế chú ý chéo đa tầng quan trọng để hợp nhất các tín hiệu bổ trợ giữa không gian và tần số.

Phương pháp nhân đôi nhánh không gian mà không sử dụng nhánh FFT đạt precision 91,38%, recall 84,80% và accuracy 88,73%. Thiếu bằng chứng từ miền tần số, mô hình đánh giá thiếu các nhiễu, khẳng định đặc trưng FFT là không dư thừa so với đặc trưng không gian.

Phương pháp loại bỏ hoàn toàn kiến trúc VGG16+FFT và cơ chế hợp nhất bằng mô-đun chú ý chéo đa tầng và chỉ sử dụng VGG16 làm kiến trúc khung làm giảm recall và accuracy tổng thể so với mốc tham chiếu, do thiếu vắng cả các gợi ý tần số bổ sung lẫn một cơ chế căn chỉnh, tổng hợp thông tin. Mô-đun chú ý chéo đa tầng không thể áp dụng nếu kiến trúc chỉ có một nhánh, vì vậy so sánh này nhằm chứng minh phần đóng góp của nhánh FFT và làm rõ tác động của việc loại bỏ nhánh này.

3.6. Kết luận

Chương này đã trình bày chi tiết về cấu trúc của mạng nơ-ron nhân tạo VGG16 kết hợp với phép biến đổi Fourier trong bài toán phát hiện thẻ căn cước giả mạo, đồng thời nêu rõ phương án huấn luyện và kết quả đánh giá mô hình. Kết quả đánh giá đã chứng minh hiệu quả của kiến trúc VGG16 kết hợp phép biến đổi Fourier trong phát hiện thẻ căn cước giả mạo. Thực nghiệm cho thấy mô hình hội tụ ổn định, đạt các chỉ số F1 cao và tỉ lệ lỗi APCER, BPCER ở mức rất thấp trên tập kiểm thử, phản ánh khả năng phân biệt chính xác giữa mẫu thật và mẫu giả. Thiết kế nhánh FFT bổ sung thông tin tần số đã tăng cường khả năng phát hiện các dấu hiệu tinh vi trên hình ảnh thẻ căn cước. Đồng thời, quy trình huấn luyện được triển khai đầy đủ, quá trình đánh giá đa chiều nhằm đảm bảo kết quả khách quan, giảm thiểu rủi ro quá khớp trên tập huấn luyện. Những kết quả này khẳng định tính phù hợp của phương án đề xuất và tạo nền tảng vững chắc cho các bước cải tiến về sau.

KẾT LUẬN

Vấn đề phát hiện hình ảnh thẻ căn cước giả mạo là một bài toán có tính thực tiễn cao. Đề án đã nghiên cứu và triển khai mô hình kết hợp giữa mạng nơ-ron nhân tạo VGG16 và phép biến đổi Fourier để phát hiện các dấu hiệu giả mạo trong hình ảnh thẻ căn cước. Kết quả kiểm thử cho thấy rằng mô hình đã đáp ứng tốt các yêu cầu về nhận diện các đặc trưng giả mạo, tốc độ thực thi nhanh chóng. Mô hình này sẽ giúp cho các hệ thống định danh điện tử nâng cao độ chính xác và hiệu quả cao trong việc phát hiện các hình ảnh giả, giúp giảm thiểu rủi ro gian lận. Việc xác thực hình ảnh thẻ căn cước trước khi tiến hành các giai đoạn tiếp theo của quá trình định danh điện tử giúp nâng cao trải nghiệm người dùng, phòng chống rủi ro và cải thiện hiệu suất cho các hệ thống. Thành công của đề án không chỉ nằm ở việc nâng cao độ chính xác phát hiện giả mạo, mà còn chứng minh rằng việc kết hợp các phương pháp truyền thống có tính nền tảng với các kỹ thuật xử lý ảnh hoàn toàn có thể tạo ra một mô hình hiệu quả, nhẹ, và dễ ứng dụng thực tế. Đây là một đóng góp mang tính gợi mở, đồng thời đặt nền móng cho các hệ thống định danh điện tử an toàn và hiệu quả hơn trong tương lai.

Về hướng nghiên cứu tiếp theo, tiếp tục mở rộng nghiên cứu ra các biến thể giả mạo thẻ căn cước khác như giả mạo bằng ảnh in màu, ảnh in không màu, không ép plastic, có ép plastic; Mở rộng tập dữ liệu huấn luyện và kiểm thử đa dạng hơn, nhiều quốc gia hơn; tối ưu hóa mô hình kết hợp mạng nơ-ron VGG16 và phép biến đổi Fourier. Bên cạnh việc nâng cao hiệu suất của mô hình, một hướng thử nghiệm khả thi là sử dụng các mạng nơ-ron khung khác để thay thế kiến trúc khung VGG16. Các phương pháp trên sẽ góp phần tăng cường độ chính xác và tốc độ xử lý của mô hình.

Bài toán định danh điện tử đã và đang trở thành bước tiến quan trọng trong quá trình số hóa dịch vụ và nâng cao hiệu quả quản lý xã hội. Đề tài “Ứng dụng mạng nơ-ron nhân tạo và phép biến đổi Fourier để phát hiện hình ảnh thẻ căn cước giả mạo”

là một minh chứng cho khả năng ứng dụng các công cụ học sâu để phục vụ cho các hệ thống chuyển đổi số, mang lại cuộc sống hiện đại, văn minh và an toàn hơn.

TÀI LIỆU THAM KHẢO

[1] Hoàng Lan, Hải Vân, "Có 40 ngân hàng chính thức triển khai quy trình mở tài khoản thanh toán eKYC," Tạp chí điện tử của Hội Khoa học Kinh tế Việt Nam, 26 05 2023. URL: https://vneconomy.vn/co-40-ngan-hang-chinh-thuc-trien-khai-quy-trinh-mo-tai-khoan-thanh-toan-ekyc.htm.

[2] Phạm Vinh, "Người dân bị lừa đảo khoảng 10.000 tỷ đồng trên không gian mạng," Tạp chí điện tử của Hội Khoa học Kinh tế Việt Nam, 15 06 2024. URL: https://vneconomy.vn/nguoi-dan-bi-lua-dao-khoang-10-000-ty-dong-tren-khong-gian-mang.htm.

[3] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, Pearson Education, 2018.

[4] Polevoy, Dmitry V. and Sigareva, Irina V. and Ershova, Daria M. and Arlazarov, Vladimir V. and Nikolaev, Dmitry P. and Ming, Zuheng and Luqman, Muhammad Muzzamil and Burie, Jean-Christophe, "Document Liveness Challenge Dataset (DLC-2021)," *Journal of Imaging,* tập 8, 2022. DOI: https://doi.org/10.3390/jimaging8070181

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," *Computer Vision and Pattern Recognition (cs.CV),* 2015. DOI: https://doi.org/10.48550/arXiv.1512.03385

[6] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition,* pp. 248-255, 2009. DOI: https://doi.org/10.1109/CVPR.2009.5206848

[7] Konstantin Bulatov, Ekaterina Emelianova, Daniil Tropin, Natalya Skoryukina, Yulia Chernyshova, Alexander Sheshkus, Sergey Usilin, Zuheng Ming, Jean-Christophe Burie, Muhammad Muzzamil Luqman, Vladimir V. Arlazarov, "MIDV-2020: A Comprehensive Benchmark Dataset for Identity Document Analysis," *Computer Optics,* tập 46, số 2, pp. 252-270, 2022. DOI: https://doi.org/10.18287/2412-6179-CO-1006.

[8] I.A. Kunina, A.V. Sher, D.P. Nikolaev, “Screen recapture detection based on color-texture analysis of document boundary regions,” *Computer Optics,* tập 47, số 4, pp. 650-657, 2023. DOI: https://doi.org/10.18287/2412-6179-CO-1237.

[9] Reuben Markham, Juan M. Espín, Mario Nieto-Hidalgo, Juan E. Tapia, “Open-Set: ID Card Presentation Attack Detection using Neural Transfer Style,” *Computer Vision and Pattern Recognition (cs.CV),* 2023. DOI: https://doi.org/10.48550/arXiv.2312.13993.

[10] A. H. M. Z. A. Z. L.-C. C. Mark Sandler, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 4510-4520, 2018. DOI: https://doi.org/10.48550/arXiv.1801.04381.

[11] Musab Al-Ghadi; Joris Voerman, Mickaël Coustaty, Olivier Lessard, Nicolas Sidere, “IDTrust: Deep Identity Document Quality Detection with Bandpass Filtering,” *2025 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW),* 2025. DOI: https://doi.org/10.1109/WACVW65960.2025.00081

[12] Chuo Jun Zhang, Asif Q. Gill, Bo Liu, Memoona J. Anwar, “AI-based Identity Fraud Detection: A Systematic Review,” *arXiv:2501.09239.*

[13] Dingheng Zeng, Liang Gao, Hao Fang, Guohui Xiang, Yue Feng, Quan Lu, "Bandpass Filter Based Dual-stream Network for Face Anti-spoofing," 2023. DOI: https://doi.org/10.48550/arXiv.2501.09239.

[14] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, Jian Ren, “Rethinking Vision Transformers for MobileNet Size and Speed,” *Computer Vision and Pattern Recognition,* 2022. DOI: https://doi.org/10.48550/arXiv.2212.08059

[15] Zitong Yu, Chenxu Zhao, Zezheng Wang, Yunxiao Qin, “Searching Central Difference Convolutional Networks for Face Anti-Spoofing,” 2020. DOI: https://doi.org/10.48550/arXiv.2003.04092

[16] Boulkenafet, Zinelabinde and Komulainen, Jukka and Li, Lei and Feng, Xiaoyi and Hadid, Abdenour, "{OULU-NPU}: A Mobile Face Presentation Attack Database with Real-World Variations," trong *12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, 2017, pp. 612-618. DOI: https://doi.org/10.1109/FG.2017.77

[17] Zhiwei Zhang; Junjie Yan; Sifei Liu; Zhen Lei; Dong Yi; Stan Z. Li, "A face antispoofing database with diverse attacks," trong *2012 5th IAPR International Conference on Biometrics (ICB)*, IEEE. DOI: https://doi.org/10.1109/ICB.2012.6199754

[18] Mark S. Nixon & Alberto S. Aguado, Feature Extraction and Image Processing, Academic Press publications, 2008.

[19] ISO, "ISO/IEC 30107-3:2023 - Information technology — Biometric presentation attack detection," 2023.

[20] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Computer Vision and Pattern Recognition (cs.CV),* 2014. DOI: https://doi.org/10.48550/arXiv.1409.1556.

[21] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[22] Khamsa Djaroudib, Pascal Lorenz, Rime Belkacem Bouzida, Hanine Merzougui, "Skin Cancer Diagnosis Using VGG16 and Transfer Learning: Analyzing the Effects of Data Quality over Quantity on Model Efficiency," *Applied Sciences,* tập 14, số 17, 2024. DOI: https://doi.org/10.3390/app14177447

[23] Prerepa Gayathri, Aiswarya Dhavileswarapu, Sufyan Ibrahim, Rahul Paul, Reena Gupta, "Exploring the Potential of VGG-16 Architecture for Accurate Brain Tumor Detection Using Deep Learning," *JCMM,* tập 2, số 2, 2023. DOI: https://doi.org/10.57159/gadl.jcmm.2.2.23056

[24] G Jignesh Chowdary, Suganya G, Premalatha M, Karunamurthy K, "Class dependency based learning using Bi-LSTM coupled with the transfer learning

of VGG16 for the diagnosis of Tuberculosis from chest x-rays," *Image and Video Processing,* 2021. DOI: https://doi.org/10.48550/arXiv.2108.04329

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is All You Need," *Computation and Language (cs.CL); Machine Learning (cs.LG),* 2017. DOI: https://doi.org/10.48550/arXiv.1706.03762

[26] Google Cloud Platform, "GPUs on Compute Engine – GPU machine types," Google, 2024. [Trực tuyến]. Available: https://cloud.google.com/compute/docs/gpus.

PHỤ LỤC I

Mã nguồn quá trình chuẩn bị dữ liệu, huấn luyện và đánh giá mạng VGG16+FFT

Phụ lục này trình bày mô tả và mã nguồn các bước thực hiện quá trình huấn luyện mạng VGG16+FFT.

1. Tải xuống bộ dữ liệu DLC-2021

```
import kagglehub

# Download latest version
dlc_2021_path = kagglehub.dataset_download("kontheeboonmeeprakob/dlc-dataset",
force_download=True)

print("Path to dataset files:", dlc_2021_path)
```

2. Tải xuống bộ dữ liệu MIDV-2020 (thư mục “clips”)

```
# Tải bộ dữ liệu MIDV-2020 bằng SFTP

# Cài đặt thư viện:
# pip install paramiko tqdm

from pathlib import Path
import paramiko
from tqdm import tqdm

# Thông tin kết nối SFTP
HOST = "l3i-share.univ-lr.fr"
PORT = 22
USERNAME = "userMIDV2020"
PASSWORD = "W3C!p5{y7"

# Đường dẫn file trên server và thư mục lưu ở máy local
REMOTE_FILES = [
    "/MIDV2020/dataset/clips.tar",
    # Có thể thêm các file khác tại đây nếu cần
]

LOCAL_DIR = Path("/content/midv2020")
LOCAL_DIR.mkdir(parents=True, exist_ok=True)

# Kết nối tới server SFTP
client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
client.connect(HOST, port=PORT, username=USERNAME, password=PASSWORD,
timeout=60)
sftp = client.open_sftp()
print("Connected to SFTP server")
```

```
# Tải file
for remote_path in REMOTE_FILES:
    local_path = LOCAL_DIR / Path(remote_path).name

    try:
        stat = sftp.stat(remote_path)
        file_size = stat.st_size
    except FileNotFoundError:
        print(f"File not found on server: {remote_path}")
        continue

    print(f"Downloading {local_path.name} ...")
    with tqdm(total=file_size, unit="B", unit_scale=True, desc=local_path.name)
as pbar:
        def progress_callback(transferred, total):
            # cập nhật phần đã tải thêm
            pbar.update(transferred - pbar.n)

        sftp.get(remote_path, str(local_path), callback=progress_callback)
        print(f"Saved to {local_path}")

# Đóng kết nối
sftp.close()
client.close()
print("All downloads completed. Local directory:", LOCAL_DIR)
midv_2020_path = LOCAL_DIR
```

3. Tải các tệp DLC2021 experimental_baselines bao gồm:

- /content/experimental_baselines/screen_positive_train.lst
- /content/experimental_baselines/screen_negative_train.lst
- /content/experimental_baselines/screen_positive_test.lst
- /content/experimental_baselines/screen_negative_test.lst

```
import os
import zipfile
import requests

# Thư mục lưu
os.makedirs("/content/experimental_baselines", exist_ok=True)
zip_path = os.path.join("/content", "experimental_baseline.zip")

url =
"https://zenodo.org/records/7467028/files/experimental_baseline.zip?download=1"

# Tải file zip
with requests.get(url, stream=True) as r:
    r.raise_for_status()
```

```
    with open(zip_path, "wb") as f:
        for chunk in r.iter_content(chunk_size=8192):
            if chunk:
                f.write(chunk)

print("Downloaded to:", zip_path)

# Giải nén
lst_dir = "/content/experimental_baselines"
with zipfile.ZipFile(zip_path, "r") as z:
    z.extractall(lst_dir)

print("Extracted to: /content/experimental_baselines")
print("Các file/thu mục trong experimental_baselines:",
os.listdir("experimental_baselines"))
```

4. Tạo tập dữ liệu train/val/test từ 2 nguồn DLC-2021 và MIDV-2020.

Trước hết, chương trình đọc bốn tệp danh sách screen_*_train/test.lst để lấy ra từng tệp cần dùng. Dựa trên tên tệp, mỗi mẫu sẽ được ánh xạ tới đúng thư mục nguồn: ảnh thật train lấy từ MIDV-2020/images, ảnh giả và ảnh test lấy từ DLC-2021/or và DLC-2021/re. Với mỗi ảnh, các bước xử lý như sau: (i) xác định tâm cắt (điểm trung tâm thẻ căn cước theo siêu tham số trong các tệp .lst), rồi cắt hai cửa sổ vuông 448×448 và 896×896 quanh tâm, bảo đảm không bị vượt biên, sau đó (ii) resize cả hai về cùng kích thước 224×224. Hai ảnh đã cắt được lưu lần lượt vào cây thư mục crops/<split>/<label>/ và images/<split>/<label>/. Sau khi sinh toàn bộ tập train và test, hàm move_first_percent_train_to_val thực hiện tách một tỷ lệ 20% mẫu từ train sang val, tạo ra ba tập con train/val/test nhất quán về cấu trúc thư mục, sẵn sàng cho huấn luyện và đánh giá mô hình.

```
import os, json, shutil
from pathlib import Path
from PIL import Image, ImageOps
from tqdm.auto import tqdm

# Cấu hình crop/resize
CROP_SIZE_SMALL = (448, 448)
CROP_SIZE_LARGE = (896, 896)
RESIZE_TO = (224, 224)

try:
    RESAMPLE_FILTER = Image.Resampling.LANCZOS
except AttributeError:
    RESAMPLE_FILTER = Image.LANCZOS
```

```
SUBJECT_MAP = {
    "alb_id": "00", "esp_id": "01", "fin_id": "02", "est_id": "03", "svk_id":
"04",
    "aze_passport": "05", "lva_passport": "06", "grc_passport": "07",
    "srb_passport": "08", "rus_internalpassport": "09",
}

def crop_center_keep_in_bounds(im, cx, cy, size_wh):
    w, h = im.size
    tw, th = size_wh
    half_w, half_h = tw // 2, th // 2
    left, top = cx - half_w, cy - half_h
    right, bottom = left + tw, top + th
    if left < 0:
        right -= left; left = 0
    if top < 0:
        bottom -= top; top = 0
    if right > w:
        shift = right - w; left = max(0, left - shift); right = w
    if bottom > h:
        shift = bottom - h; top = max(0, top - shift); bottom = h
    crop = im.crop((left, top, right, bottom))
    if crop.size != size_wh:
        crop = crop.resize(size_wh, RESAMPLE_FILTER)
    return crop

def ensure_dirs(output_dir):
    for kind in ("annotations", "crops", "images"):
        for split in ("train", "val", "test"):
            for label in ("genuine", "spoof"):
                (output_dir / kind / split / label).mkdir(parents=True,
exist_ok=True)

def move_first_percent_train_to_val(output_dir, ratio=0.2):
    for kind in ("annotations", "crops", "images"):
        for label in ("genuine", "spoof"):
            src_dir = output_dir / kind / "train" / label
            if not src_dir.exists():
                continue
            dst_dir = output_dir / kind / "val" / label
            dst_dir.mkdir(parents=True, exist_ok=True)
            files = sorted(p for p in src_dir.iterdir() if p.is_file())
            n_move = int(len(files) * ratio)
            for p in files[:n_move]:
                shutil.move(str(p), str(dst_dir / p.name))

def load_screen_splits(lst_dir: Path):
    items = []
    def _load_one(fname, split_name, label_name):
        path = lst_dir / fname
        if not path.is_file():
            print(f"[WARN] missing {path}")
            return
        with path.open("r", encoding="utf-8") as f:
            for line in f:
```

```
                line = line.strip()
                if not line:
                    continue
                parts = line.split()
                rel = parts[0].lstrip("/")
                try:
                    country, clip, filename = rel.split("/", 2)
                except ValueError:
                    continue
                cx = cy = None
                if len(parts) >= 3:
                    try:
                        cx = int(float(parts[1])); cy = int(float(parts[2]))
                    except Exception:
                        pass
                items.append((country, clip, filename, split_name, label_name,
cx, cy))
    _load_one("screen_positive_train.lst", "train", "spoof")
    _load_one("screen_negative_train.lst", "train", "genuine")
    _load_one("screen_positive_test.lst", "test", "spoof")
    _load_one("screen_negative_test.lst", "test", "genuine")
    print(f"Loaded {len(items)} lines from .lst")
    return items

def split_and_convert_dlc(dlc_or_img_dir, dlc_re_img_dir, output_dir,
                          lst_dir, midv_clips_root, val_ratio=0.2):
    dlc_or_img_dir = Path(dlc_or_img_dir)
    dlc_re_img_dir = Path(dlc_re_img_dir)
    output_dir = Path(output_dir)
    lst_dir = Path(lst_dir)
    midv_root = Path(midv_clips_root)

    items = load_screen_splits(lst_dir)
    if not items:
        print("No items loaded; abort.")
        return

    ensure_dirs(output_dir)

    roots = {
        ("train", "genuine"): midv_root / "images",
        ("train", "spoof"): dlc_re_img_dir,
        ("test", "genuine"): dlc_or_img_dir,
        ("test", "spoof"): dlc_re_img_dir,
    }

    desc = f"Center crops {CROP_SIZE_SMALL[0]} & {CROP_SIZE_LARGE[0]} ->
{RESIZE_TO[0]}"
    for country, clip, fname, split_use, label_use, cx, cy in tqdm(items,
desc=desc, unit="img"):
        img_root = roots.get((split_use, label_use))
        if img_root is None:
            continue
        src_img = img_root / country / clip / fname
        if not src_img.is_file():
            continue
```

```
        country_id = SUBJECT_MAP.get(country, "xx")
        stem = fname.rsplit(".", 1)[0]

        try:
            with Image.open(src_img) as im:
                im = ImageOps.exif_transpose(im).convert("RGB")
                w, h = im.size
                cx0, cy0 = (w // 2, h // 2) if cx is None or cy is None else
(cx, cy)

                crop_small = crop_center_keep_in_bounds(im, cx0, cy0,
CROP_SIZE_SMALL)
                crop_small = crop_small.resize(RESIZE_TO, RESAMPLE_FILTER)

                crop_large = crop_center_keep_in_bounds(im, cx0, cy0,
CROP_SIZE_LARGE)
                crop_large = crop_large.resize(RESIZE_TO, RESAMPLE_FILTER)

                base =
f"{label_use}_{country}_{country_id}_{clip}_{stem}_cx{cx0:04d}_cy{cy0:04d}"

                p_crop = output_dir / "crops" / split_use / label_use /
f"{base}.jpg"
                crop_small.save(p_crop, format="JPEG", quality=90,
subsampling="4:2:0")

                p_img = output_dir / "images" / split_use / label_use /
f"{base}.jpg"
                crop_large.save(p_img, format="JPEG", quality=90,
subsampling="4:2:0")

                p_ann = output_dir / "annotations" / split_use / label_use /
f"{base}.json"
                ann = {
                    "filename": fname,
                    "subject": country,
                    "subject_id": country_id,
                    "clip": clip,
                    "center": {"cx": cx0, "cy": cy0},
                    "original_size": [w, h],
                }
                p_ann.write_text(json.dumps(ann, ensure_ascii=False),
encoding="utf-8")
        except Exception as e:
            print(f"Error processing {src_img}: {e}")

    if 0.0 < val_ratio < 1.0:
        move_first_percent_train_to_val(output_dir, ratio=val_ratio)

    for kind in ("annotations", "images", "crops"):
        print(f"--- {kind} ---")
        for split in ("train", "val", "test"):
            for label in ("genuine", "spoof"):
                folder = output_dir / kind / split / label
```

```
                cnt = sum(1 for p in folder.iterdir() if p.is_file()) if
folder.exists() else 0
                print(f"{kind}/{split}/{label}: {cnt:6d}")
        print()

if __name__ == "__main__":
    dlc_2021_path = "/kaggle/input/dlc-dataset"
    dlc_2021_dir = os.path.join(dlc_2021_path, "DLC-2021")
    dlc_or_img = os.path.join(dlc_2021_dir, "or", "images")
    dlc_re_img = os.path.join(dlc_2021_dir, "re", "images")

    midv_2020_path = "/content/midv2020"
    midv_2020_dir = os.path.join(midv_2020_path, "clips")

    lst_dir = "/content/experimental_baselines"
    out_dir = "/content/dlc-2021-split"

    split_and_convert_dlc(
        dlc_or_img_dir=dlc_or_img,
        dlc_re_img_dir=dlc_re_img,
        output_dir=Path(out_dir),
        lst_dir=Path(lst_dir),
        midv_clips_root=midv_2020_dir,
        val_ratio=0.2,
    )
```

5. Cài thư viện cần thiết cho quá trình xây dựng và huấn luyện mạng.

- Cài thư viện pytorch_lightning

```
! pip install pytorch-lightning
```

6. Tiến hành huấn luyện mạng

6.1. Xây dựng mô hình VGG16+FFT

Mạng gồm hai nhánh chia sẻ kiến trúc VGG16-BN: nhánh không gian (VGG16Branch) nhận ảnh màu đã chuẩn hoá; nhánh tần số (FFTBranch) áp dụng biến đổi Fourier 2D lên ảnh gốc, dùng bộ lọc thông cao (high-pass), rồi chuẩn hoá và trích đặc trưng bằng cùng backbone VGG16. Hai đầu ra đặc trưng được đưa vào mô-đun chú ý chéo đa tầng: CrossAttentionFusion, thực hiện cross-attention hai chiều và gộp thành một vector đặc trưng chung. Class LightningModule VGG16_FFTNet định nghĩa toàn bộ kiến trúc. Sử dụng hàm mất mát cross-entropy cho phân loại nhị phân, và tối ưu hoá bằng AdamW.

```
import torch, torch.nn as nn, torch.nn.functional as F
from torchvision import models
import pytorch_lightning as pl

def _build_vgg16_bn_features():
    """Load VGG16-BN pretrained và cắt đến tầng conv3_x."""
    try:
        from torchvision.models import vgg16_bn, VGG16_BN_Weights
        vgg = vgg16_bn(weights=VGG16_BN_Weights.IMAGENET1K_V1)
    except Exception:
        vgg = models.vgg16_bn(pretrained=True)
    return nn.Sequential(*list(vgg.features.children())[:24])  #
(B,256,H/8,W/8)

class VGG16Branch(nn.Module):
    """Nhánh không gian: VGG16-BN trên ảnh RGB đã chuẩn hoá."""

    def __init__(self):
        super().__init__()
        self.features = _build_vgg16_bn_features()

    def forward(self, x):
        return self.features(x)

class FFTBranch(nn.Module):
    """Nhánh tần số: FFT + lọc thông cao + VGG16-BN."""

    def __init__(self, radius: int = 8):
        super().__init__()
        self.radius = radius
        self.features = _build_vgg16_bn_features()

    def forward(self, x, original_image):
        B = x.size(0)
        with torch.no_grad():
            # chuyển ảnh gốc sang grayscale
            x_gray = (
                0.2989 * original_image[:, 0] +
                0.5870 * original_image[:, 1] +
                0.1140 * original_image[:, 2]
            ).unsqueeze(1)                              # (B,1,H,W)
            _, _, Hc, Wc = x_gray.shape
            # FFT + shift
            X = torch.fft.fft2(x_gray)
            Xs = torch.fft.fftshift(X, dim=(-2, -1))
            # mặt nạ high-pass tròn
            u = torch.arange(Hc, device=x.device) - Hc // 2
            v = torch.arange(Wc, device=x.device) - Wc // 2
            U, V = torch.meshgrid(u, v, indexing="ij")
            mask2d = (U**2 + V**2 >= self.radius**2).float()
            mask = mask2d.unsqueeze(0).unsqueeze(0).expand(B, 1, Hc, Wc)
            # áp dụng mask và IFFT
            Xhp_s = Xs * mask
            Xhp = torch.fft.ifftshift(Xhp_s, dim=(-2, -1))
            x_hp = torch.fft.ifft2(Xhp)
            mag = torch.abs(x_hp)
```

```
            # log-magnitude + min-max normalize theo từng ảnh
            hp_log = torch.log1p(mag)
            mn = hp_log.flatten(1).min(1)[0].view(B, 1, 1, 1)
            mx = hp_log.flatten(1).max(1)[0].view(B, 1, 1, 1)
            hp_norm = (hp_log - mn) / (mx - mn + 1e-8)
        hp_rgb = hp_norm.repeat(1, 3, 1, 1)         # (B,3,H,W)
        mean = torch.tensor([0.485, 0.456, 0.406],
device=hp_rgb.device).view(1, 3, 1, 1)
        std = torch.tensor([0.229, 0.224, 0.225], device=hp_rgb.device).view(1,
3, 1, 1)
        hp_rgb_norm = (hp_rgb - mean) / std
        return self.features(hp_rgb_norm)

class CrossAttentionFusion(nn.Module):
    """Cross-attention hai chiều giữa nhánh không gian và nhánh tần số."""

    def __init__(self, embed_dim=256, num_heads=16):
        super().__init__()
        self.attn1 = nn.MultiheadAttention(embed_dim, num_heads)
        self.attn2 = nn.MultiheadAttention(embed_dim, num_heads)

    def forward(self, feat1, feat2):
        B, C, H, W = feat1.shape
        f1 = feat1.view(B, C, -1).permute(2, 0, 1)  # (N,B,C)
        f2 = feat2.view(B, C, -1).permute(2, 0, 1)
        out1, _ = self.attn1(f1, f2, f2)
        out2, _ = self.attn2(f2, f1, f1)
        out1 = out1.permute(1, 2, 0).mean(dim=2)    # (B,C)
        out2 = out2.permute(1, 2, 0).mean(dim=2)    # (B,C)
        return torch.cat([out1, out2], dim=1)       # (B,512)

class VGG16_FFTNet(pl.LightningModule):
    """
    LightningModule cho bài toán phân loại nhị phân (genuine/spoof):
    - Hai nhánh VGG16 chia sẻ kiến trúc, một nhánh làm việc trên ảnh RGB,
      một nhánh trên ảnh high-frequency từ FFT.
    - Đầu ra hai nhánh được gộp bằng cross-attention và classifier MLP.
    """
    def __init__(self, num_classes: int = 2, lr: float = 1e-5, weight_decay:
float = 1e-4):
        super().__init__()
        self.save_hyperparameters()
        self.branch_spatial = VGG16Branch()
        self.branch_freq = FFTBranch()
        self.fusion = CrossAttentionFusion()
        self.classifier = nn.Sequential(
            nn.Linear(512, 64),
            nn.BatchNorm1d(64),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(64, num_classes),
        )

    def forward(self, x_resized, x_original):
        f1 = self.branch_spatial(x_resized)
        f2 = self.branch_freq(x_resized, original_image=x_original)
```

```
        f = self.fusion(f1, f2)
        return self.classifier(f)

    def training_step(self, batch, batch_idx):
        x_resized, x_original, y = batch
        logits = self(x_resized, x_original)
        loss = F.cross_entropy(logits, y)
        self.log("train_loss", loss, prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
        x_resized, x_original, y = batch
        logits = self(x_resized, x_original)
        loss = F.cross_entropy(logits, y)
        preds = logits.argmax(dim=1)
        acc = (preds == y).float().mean()
        self.log("val_loss", loss, prog_bar=True)
        self.log("val_acc", acc, prog_bar=True)

    def test_step(self, batch, batch_idx):
        x_resized, x_original, y = batch
        logits = self(x_resized, x_original)
        loss = F.cross_entropy(logits, y)
        preds = logits.argmax(dim=1)
        acc = (preds == y).float().mean()
        self.log("test_loss", loss)
        self.log("test_acc", acc)

    def configure_optimizers(self):
        return torch.optim.AdamW(self.parameters(), lr=self.hparams.lr,
weight_decay=self.hparams.weight_decay)
```

6.2. Tạo dataseet, trainer, huấn luyện và đánh giá.

Tiến hành tạo FourierImageDataset đọc ảnh đầu vào từ tập dữ liệu đã xây dựng, sau đó áp dụng biến đổi (chuẩn hoá) cho nhánh không gian, còn nhánh tần số giữ ở dạng tensor thô. Cuối cùng, PyTorch Lightning Trainer được cấu hình với EarlyStopping và TensorBoardLogger, gọi trainer.fit để huấn luyện và trainer.test để đánh giá mô hình VGG16_FFTNet.

```
import os
from pathlib import Path

import torch
from torch.utils.data import Dataset, DataLoader
from PIL import Image
import torchvision.transforms as transforms
```

```
import torchvision.transforms.functional as TF

import pytorch_lightning as pl
from pytorch_lightning.callbacks import EarlyStopping
from pytorch_lightning.loggers import TensorBoardLogger

# Giả sử lớp VGG16_FFTNet đã được định nghĩa ở phần "Kiến trúc mạng"

# Biến đổi cho nhánh không gian (giống ImageNet)
train_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225]),
])
val_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225]),
])

class FourierImageDataset(Dataset):
    """
    Dataset đọc cặp (ảnh RGB đã resize & normalize, ảnh crop gốc) và nhãn:
    - root_dir: thư mục images/<split>/<label>
    - cropped_root: thư mục crops/<split>/<label> có cấu trúc tương ứng
    """
    def __init__(self, root_dir, transform=None, cropped_root=None):
        self.root_dir = root_dir
        self.cropped_root = cropped_root
        self.transform = transform
        self.samples = self._make_dataset(root_dir)
    def _make_dataset(self, directory):
        samples = []
        for label_name in sorted(os.listdir(directory)):
            class_dir = os.path.join(directory, label_name)
            if not os.path.isdir(class_dir):
                continue
            # nhãn 0: genuine, 1: spoof
            class_idx = 0 if label_name == "genuine" else 1
            for root, _, fnames in os.walk(class_dir):
                for fname in fnames:
                    if fname.lower().endswith((".png", ".jpg", ".jpeg")):
                        path = os.path.join(root, fname)
                        samples.append((path, class_idx))
        return samples
    def __len__(self):
        return len(self.samples)
    def __getitem__(self, idx):
        image_path, label = self.samples[idx]
        img = Image.open(image_path).convert("RGB")
        x_resized = self.transform(img) if self.transform is not None else
TF.to_tensor(img)
        if self.cropped_root is None:
            raise ValueError("cropped_root must be provided")
        rel_path = os.path.relpath(image_path, self.root_dir)
        orig_path = os.path.join(self.cropped_root, rel_path)
```

```
        img_orig = Image.open(orig_path).convert("RGB")
        x_original = TF.to_tensor(img_orig)   # giữ dạng tensor (0-1), không
normalize
        return x_resized, x_original, torch.tensor(label, dtype=torch.long)

# Đường dẫn dữ liệu đã tiền xử lý (theo script split trước đó)
base_dir = "/content/dlc-2021-split"
train_img_dir = os.path.join(base_dir, "images", "train")
val_img_dir   = os.path.join(base_dir, "images", "val")
test_img_dir  = os.path.join(base_dir, "images", "test")
train_crop_dir = os.path.join(base_dir, "crops", "train")
val_crop_dir   = os.path.join(base_dir, "crops", "val")
test_crop_dir  = os.path.join(base_dir, "crops", "test")

train_dataset = FourierImageDataset(train_img_dir, transform=train_transform,
cropped_root=train_crop_dir)
val_dataset   =
FourierImageDataset(val_img_dir,   transform=val_transform,   cropped_root=val_
crop_dir)
test_dataset  =
FourierImageDataset(test_img_dir,  transform=val_transform,   cropped_root=test
_crop_dir)

train_loader = DataLoader(train_dataset, batch_size=64,
shuffle=True,  num_workers=4)
val_loader   = DataLoader(val_dataset,   batch_size=64, shuffle=False,
num_workers=4)
test_loader  = DataLoader(test_dataset,  batch_size=64, shuffle=False,
num_workers=4)

print(f"Train dataset size: {len(train_dataset)}")
print(f"Validation dataset size: {len(val_dataset)}")
print(f"Test dataset size: {len(test_dataset)}")

# Callback và logger cho Lightning
early_stopping = EarlyStopping(monitor="val_loss", patience=2, mode="min")
logger = TensorBoardLogger("logs", name="vgg16_fftnet")

# Khởi tạo mô hình và trainer
model = VGG16_FFTNet(num_classes=2, lr=1e-5, weight_decay=1e-4)

trainer = pl.Trainer(
    max_epochs=5,
    accelerator="gpu",
    devices=1,
    logger=logger,
    callbacks=[early_stopping],
    log_every_n_steps=10,
)

# Huấn luyện và đánh giá
trainer.fit(model, train_loader, val_loader)
trainer.test(model, dataloaders=test_loader)
```

7. Đo tốc độ thực thi trung bình

Đoạn mã dưới đây đánh giá tốc độ thực thi của mô hình VGG16+FFT. Cụ thể, đọc lần lượt các ảnh từ hai thư mục đại diện cho trường hợp ảnh thật và ảnh giả. Sau đó thực hiện các bước: (i) tiền xử lý và (ii) suy luận qua mô hình VGG16+FFT. Trước khi ghi nhận thời gian, chương trình thực hiện một số bước warm-up (mặc định 10 ảnh) nhằm loại bỏ ảnh hưởng của việc khởi tạo mô hình ban đầu. Để tương đồng với điều kiện CPU trong bài báo gốc, việc suy luận được ép chạy CPU-only, giới hạn số luồng (NUM_THREADS=12) và vô hiệu hóa một số tối ưu như MKL-DNN.

```
# measure_vgg16_fft_cpu.py
# Đo thời gian end-to-end trên CPU:
#   decode ảnh -> resize 224x224 -> chuẩn hoá -> suy luận với VGG16+FFT
(Lightning)

import os, glob, time
import numpy as np

# ================== CONFIG ==================
dlc_2021_path = "/kaggle/input/dlc-dataset"
OR_DIR = dlc_2021_path + "/DLC-2021/or/images/grc_passport/01.or0004"
RE_DIR = dlc_2021_path + "/DLC-2021/re/images/grc_passport/01.re0001"

TARGET_SIZE = (224, 224)
WARMUP_STEPS = 10
NUM_THREADS = 12
# ============================================

# CPU only & giới hạn số thread trước khi import torch
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
os.environ["OMP_NUM_THREADS"] = str(NUM_THREADS)
os.environ["MKL_NUM_THREADS"] = str(NUM_THREADS)

import torch
from PIL import Image, __version__ as PIL_VERSION

torch.set_num_threads(NUM_THREADS)
try:
    torch.set_num_interop_threads(1)
except Exception:
    pass
try:
    torch.backends.mkldnn.enabled = False
except Exception:
    pass
```

```
# TODO: sửa import này cho đúng file chứa LightningModule của bạn
# Ví dụ: from model_vgg16_fft import VGG16_FFTNet
from model_vgg16_fft import VGG16_FFTNet   # <-- chỉnh lại cho khớp với code
của bạn


def list_images(folder):
    exts = ("*.jpg", "*.jpeg", "*.png", "*.bmp", "*.tif", "*.tiff",
            "*.JPG", "*.JPEG", "*.PNG", "*.BMP", "*.TIF", "*.TIFF")
    fs = []
    for e in exts:
        fs.extend(glob.glob(os.path.join(folder, e)))
    fs.sort()
    if not fs:
        raise FileNotFoundError(f"Không thấy ảnh trong: {folder}")
    return fs


def preprocess_pil_for_vggfft(path, size):
    """
    Trả về:
        x_resized  : ảnh đã resize + normalize ImageNet (nhánh không gian)
        x_original : ảnh 0..1, không normalize (nhánh FFT)
    """
    with Image.open(path) as im:
        im = im.convert("RGB")
        im = im.resize(size, Image.Resampling.LANCZOS)
        arr = np.asarray(im, dtype=np.float32) / 255.0   # [H,W,3], 0..1

    x_original = torch.from_numpy(arr).permute(2, 0, 1).unsqueeze(0)  #
[1,3,H,W]

    mean = torch.tensor([0.485, 0.456, 0.406]).view(1, 3, 1, 1)
    std  = torch.tensor([0.229, 0.224, 0.225]).view(1, 3, 1, 1)
    x_resized = (x_original - mean) / std

    return x_resized, x_original


@torch.inference_mode()
def measure_folder_end2end_vggfft(lightning_model, folder, warmup_steps=10):
    """
    Đo thời gian trung bình cho mỗi ảnh:
        decode + resize(224) + normalize + forward VGG16+FFT
    """
    device = torch.device("cpu")
    lightning_model.eval()
    lightning_model.to(device)

    paths = list_images(folder)
    n = len(paths)

    # warm-up (không tính thời gian)
    for i in range(min(warmup_steps, n)):
        xr, xo = preprocess_pil_for_vggfft(paths[i], TARGET_SIZE)
```

```
        xr, xo = xr.to(device), xo.to(device)
        _ = lightning_model(xr, xo)

    # chạy đo thời gian
    times = []
    for p in paths:
        t0 = time.perf_counter()
        xr, xo = preprocess_pil_for_vggfft(p, TARGET_SIZE)
        xr, xo = xr.to(device), xo.to(device)
        y = lightning_model(xr, xo)
        _ = y.cpu()   # đồng bộ CPU
        t1 = time.perf_counter()
        times.append(t1 - t0)

    mean_sec = float(np.mean(times))
    return mean_sec, n


def report(model_name, folder, mean_sec, n):
    print(
        f"[{model_name}] [{os.path.basename(folder)}] "
        f"mean_sec_per_img={mean_sec:.6f} | n={n} "
        f"| device=CPU | threads={NUM_THREADS} "
        f"| torch={torch.__version__} | pillow={PIL_VERSION} "
        f"| mkldnn={'on' if torch.backends.mkldnn.enabled else 'off'}"
    )


if __name__ == "__main__":
    print("========== Benchmark: VGG16+FFT (Lightning) ==========")

    # Khởi tạo model (hoặc load checkpoint đã train)
    model = VGG16_FFTNet(num_classes=2)
    # Ví dụ nếu dùng checkpoint:
    # model =
VGG16_FFTNet.load_from_checkpoint("path/to/checkpoint.ckpt").eval()

    or_mean, or_n = measure_folder_end2end_vggfft(
        model, OR_DIR, warmup_steps=WARMUP_STEPS
    )
    re_mean, re_n = measure_folder_end2end_vggfft(
        model, RE_DIR, warmup_steps=WARMUP_STEPS
    )

    report("VGG16_FFTNet", OR_DIR, or_mean, or_n)
    report("VGG16_FFTNet", RE_DIR, re_mean, re_n)
```