

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



NGUYỄN DUY HƯNG

ỨNG DỤNG CÁC PHƯƠNG PHÁP HỌC MÁY ĐỂ PHÂN VÙNG  
ẢNH VÀ XÂY DỰNG ỨNG DỤNG XÁC ĐỊNH BỌT KHÍ TRONG  
CHẤT LỎNG

ĐỒ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT  
(*Theo định hướng ứng dụng*)

HÀ NỘI – 2025

# HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



NGUYỄN DUY HƯNG

ỨNG DỤNG CÁC PHƯƠNG PHÁP HỌC MÁY ĐỂ PHÂN VÙNG  
ẢNH VÀ XÂY DỰNG ỨNG DỤNG XÁC ĐỊNH BỌT KHÍ TRONG  
CHẤT LỎNG

Chuyên ngành: HỆ THỐNG THÔNG TIN  
Mã số: 8.52.02.08

ĐỒ 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 xin cam đoan rằng nội dung được trình bày trong luận văn này hoàn toàn là kết quả của quá trình nghiên cứu và tìm hiểu nghiêm túc của cá nhân tôi, dưới sự hướng dẫn tận tình của thầy hướng dẫn. Trong quá trình thực hiện, tôi đã cố gắng tuân thủ đầy đủ các quy định về đạo đức nghiên cứu khoa học, đảm bảo tính trung thực và chính xác trong việc sử dụng tài liệu tham khảo.

Các ý tưởng, dữ liệu, và kết quả nghiên cứu không phải của tôi đều đã được trích dẫn rõ ràng và đầy đủ theo đúng quy định. Luận văn này không sao chép từ bất kỳ công trình nghiên cứu hay luận văn nào khác.

Tôi cũng xin khẳng định rằng luận văn này chưa từng được trình bày hoặc bảo vệ tại bất kỳ hội đồng nào, ở trong nước hay quốc tế. Đồng thời, đến thời điểm hiện tại, nội dung luận văn chưa được công bố dưới bất kỳ hình thức nào trên các phương tiện thông tin đại chúng hoặc tạp chí khoa học.

Tôi xin hoàn toàn chịu trách nhiệm trước pháp luật và quy định của nhà trường nếu phát hiện có bất kỳ vi phạm nào liên quan đến nội dung cam đoan này.

Tác giả luận văn

**NGUYỄN DUY HƯNG**

## LỜI CẢM ƠN

Trước tiên, tôi xin gửi lời cảm ơn sâu sắc và chân thành nhất tới thày TS. Nguyễn Tất Thắng, người đã tận tình hướng dẫn, định hướng và chia sẻ những kiến thức quý báu, giúp tôi hoàn thành tốt luận văn này. Sự hỗ trợ và chỉ dẫn tận tâm của thày là nguồn động viên to lớn trong suốt quá trình nghiên cứu và viết luận văn.

Tôi xin chân thành cảm ơn Ban lãnh đạo Học viện Công nghệ Bưu chính Viễn thông, Khoa Đào tạo Sau Đại học cùng toàn thể các thày, cô đã luôn tạo điều kiện tốt nhất và nhiệt tình hỗ trợ tôi trong suốt quá trình học tập và nghiên cứu tại trường. Sự tận tụy và kiến thức chuyên môn của quý thày, cô đã giúp tôi có được nền tảng vững chắc để hoàn thành luận văn này.

Bên cạnh đó, tôi cũng xin bày tỏ lòng biết ơn tới các bạn học viên đã luôn đồng hành, hỗ trợ và chia sẻ những kinh nghiệm quý giá trong suốt chặng đường học tập.

Không thể không nhắc đến gia đình, bạn bè và đồng nghiệp - những người đã luôn bên cạnh động viên, khích lệ và tạo mọi điều kiện thuận lợi nhất để tôi tập trung hoàn thành tốt công việc nghiên cứu. Sự ủng hộ và tình cảm quý báu từ họ là nguồn động lực lớn lao để tôi vượt qua những khó khăn và thử thách.

Cuối cùng, dù đã nỗ lực hết mình trong quá trình thực hiện luận văn, tôi nhận thức rằng không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được sự thông cảm và những góp ý quý báu từ các thày, cô và bạn đọc để hoàn thiện hơn nữa nội dung nghiên cứu của mình.

Xin chân thành cảm ơn!

Hà Nội, ngày tháng năm 2025

Tác giả luận văn

**NGUYỄN DUY HƯNG**

## MỤC LỤC

LỜI CAM ĐOAN .....	i
LỜI CẢM ƠN .....	ii
DANH MỤC CÁC TỪ VIẾT TẮT .....	vi
DANH MỤC HÌNH VẼ .....	vii
MỞ ĐẦU .....	9
1. Lý do chọn đề tài.....	9
2. Tổng quan về vấn đề nghiên cứu .....	10
3. Mục tiêu nghiên cứu của đề tài.....	11
4. Đối tượng và phạm vi nghiên cứu của đề tài.....	11
5. Phương pháp nghiên cứu của đề tài .....	12
CHƯƠNG 1 CƠ SỞ LÝ LUẬN .....	13
1.1. Khái quát về hiện tượng bọt khí trong chất lỏng .....	13
1.2. Khái quát về phân vùng ảnh .....	14
1.2.1. Định nghĩa phân vùng ảnh.....	14
1.2.2. Lịch sử phát triển của phân vùng ảnh .....	15
1.3. Kết luận Chương 1 .....	17
CHƯƠNG 2 NGHIÊN CỨU VÀ PHÂN TÍCH MỘT SỐ PHƯƠNG PHÁP PHÂN ĐOẠN ẢNH .....	18
2.1. Phân vùng dựa trên phân ngưỡng cường độ sáng (Thresholding).....	21
2.1.1. Lý thuyết cơ sở của ngưỡng cường độ.....	21
2.1.2. Ngưỡng toàn cục (Global Thresholding) .....	23
2.1.2.1. Lý thuyết .....	23
2.1.2.2. Tìm ngưỡng toàn cục tối ưu bằng phương pháp Otsu.....	23
2.2. Phân vùng dựa trên cạnh (Edge-based segmentation) .....	27
2.2.1. Lý thuyết cơ sở .....	28
2.2.2. Phương pháp Laplacian of Gaussian (LoG) .....	31
2.2.2.1. Cách hoạt động của phương pháp Laplacian of Gaussian (LoG) .....	31
2.2.2.2. Công thức toán học của Laplacian of Gaussian (LoG) .....	34
2.2.3. Phương pháp Canny .....	36
2.3. Phương pháp phân vùng ảnh dựa trên HOG (Histogram of Oriented Gradients) .....	43
2.3.1. Các thuật ngữ.....	44
2.3.2. Tính toán gradient .....	45
2.3.3. Các bước tính HOG .....	46

2.4. Phương pháp phân vùng ảnh dựa trên mạng nơ-ron tích chập (CNN) .....	52
2.4.1. Tổng quan về mạng nơ-ron tích chập (CNN).....	52
2.4.2. Giới thiệu về mạng nơ-ron tích chập (CNN).....	53
2.4.3. Cách hoạt động của mạng nơ-ron tích chập (CNN) .....	55
2.4.3.1. Lớp tích chập (Convolutional Layer) .....	55
2.4.3.2 Lớp gộp (Pooling Layer) .....	56
2.4.3.3. Lớp kết nối đầy đủ (Fully Connected Layer) .....	56
2.4.3.4 Hàm kích hoạt (Activation) .....	57
2.4.4. Cách huấn luyện mạng nơ-ron tích chập (CNN) .....	59
2.4.4.1. Trọng số và hằng số hiệu chỉnh.....	59
2.4.4.2. Lan truyền xuôi và lan truyền ngược .....	60
2.4.5. Overfitting và Underfitting.....	61
2.5. Nghiên cứu, phân tích và ứng dụng mô hình học máy Stardist.....	62
2.5.1. Đối tượng nhận dạng mà mô hình Stardist hướng đến .....	62
2.5.2. Kiến trúc của mô hình U-net .....	63
2.5.3. Kiến trúc của mô hình Stardist .....	65
2.5.3.1. Xây dựng bản đồ xác suất .....	69
2.5.3.2. Xây dựng đa giác lồi .....	70
2.5.3.3. Giải quyết vấn đề chồng lấn vật thể và loại bỏ các dự đoán dư thừa .....	71
2.5.3.4. Các tham số khi huấn luyện mô hình Stardist .....	73
2.6. Kết luận Chương 2 .....	75
<b>CHƯƠNG 3 XÂY DỰNG ỨNG DỤNG PHÂN VÙNG ẢNH DỰA TRÊN MẠNG NO-RON TÍCH CHẬP VÀ MÔ HÌNH STARDIST.....</b>	<b>76</b>
3.1. Kiến trúc hệ thống .....	76
3.1.1. Lựa chọn công nghệ .....	77
3.1.2. Phương thức giao tiếp .....	77
3.2. Xây dựng logic .....	77
3.2.1. Xây dựng logic phân vùng ảnh cho các thuật toán cổ điển.....	77
3.2.1.1. Xây dựng logic cho thuật toán Otsu.....	77
3.2.1.2. Xây dựng logic cho thuật toán Canny .....	79
3.2.1.3. Xây dựng logic cho thuật toán Hog .....	80
3.2.2. Xây dựng logic phân vùng ảnh dựa trên mô hình Stardist.....	81
3.2.2.1. Chuẩn hóa hình ảnh .....	81
3.2.2.2. Tạo ra ảnh mask.....	83
3.2.2.3. Huấn luyện mô hình (training) .....	85
3.2.2.4. Phân vùng ảnh (predict) .....	87

3.3. Xây dựng giao diện .....	88
3.3.1. Xây dựng giao diện menu .....	88
3.3.2. Xây dựng giao diện sử dụng phương pháp Otsu .....	88
3.3.3. Xây dựng giao diện sử dụng phương pháp Canny .....	89
3.3.4. Xây dựng giao diện sử dụng phương pháp Canny .....	89
3.3.5. Xây dựng giao diện sử dụng mô hình Stardist .....	89
3.4. Thủ nghiệm .....	91
3.4.1. Huấn luyện mô hình .....	91
3.4.2. Thủ nghiệm phân vùng hình ảnh .....	94
3.5. Kết luận Chương 3 .....	94
KẾT LUẬN.....	95
TÀI LIỆU THAM KHẢO .....	97
BẢN CAM ĐOAN.....	98

## **DANH MỤC CÁC TỪ VIẾT TẮT**

<b>STT</b>	<b>Từ viết tắt</b>	<b>Thuật ngữ tiếng Việt</b>
1	HOG	Histogram of oriented gradients
2	MRDM	Multi-radial distance maps
3	CNN	Convolutional neural network
4	AI	Artificial intelligence
5	ML	Machine learning
6	DL	Deep learning
7		
8		
10		
11		
12		
13		
14		

## DANH MỤC HÌNH VẼ

Hình 1.1	Hình ảnh minh họa về bọt khí .....	13
Hình 1.2	Hình ảnh minh họa về phân vùng ảnh .....	16
Hình 2.1	Các phương pháp phân đoạn hình ảnh truyền thống .....	18
Hình 2.2	Ví dụ về Object Detection và Image Segmentation .....	19
Hình 2.3	Ba loại tác vụ phân đoạn hình ảnh .....	20
Hình 2.4	Biểu đồ cường độ (Histogram).....	22
Hình 2.5	Ví dụ sử dụng ngưỡng toàn cục .....	23
Hình 2.6	Ví dụ phân ngưỡng bằng phương pháp Otsu .....	24
Hình 2.7	Một số đoạn cạnh thông dụng .....	28
Hình 2.8	Hình ảnh ví dụ .....	30
Hình 2.9	Biểu đồ cường độ ngang đi qua trung tâm hình, bao gồm cả điểm nhiễu đơn lẻ .....	30
Hình 2.10	Biểu đồ đơn giản hóa (các điểm được nối bằng nét gạch cho rõ ràng hơn). Đại hình ảnh này tương ứng với biểu đồ cường độ, và các số trong ô biểu thị giá trị cường độ của các điểm trên biểu đồ.....	30
Hình 2.11	Bộ lọc kernel 3x3 tổng quát .....	31
Hình 2.12	Ví dụ hình ảnh 10x1 pixel.....	32
Hình 2.13	Đồ thị độ dốc cường độ điểm ảnh .....	32
Hình 2.14	Đồ thị độ dốc cường độ điểm ảnh .....	33
Hình 2.15	Hình ảnh mèo có nhiều nhiễu (là lông của mèo).....	33
Hình 2.16	Hai bộ lọc kernel phổ biến trong xử lý ảnh với LoG .....	34
Hình 2.17	Ví dụ bộ lọc kernel trong xử lý ảnh với LoG .....	35
Hình 2.18	Hình ảnh gốc và sau khi áp dụng LoG.....	35
Hình 2.19	Hình ảnh gốc (trái) – Hình ảnh đã được chuyển sang thang độ xám .....	36
Hình 2.20	Hình ảnh bị làm mờ với bộ lọc Gaussian (sigma = 1.4 và kích thước kernel là 5×5) .....	37
Hình 2.21	Hình ảnh cường độ gradient .....	38
Hình 2.22	Ví dụ duyệt qua cạnh trên ảnh .....	39
Hình 2.23	Tập trung vào khu vực pixel trong hộp màu đỏ góc trên bên trái .....	39
Hình 2.24	Một ví dụ khác .....	40
Hình 2.25	Hình ảnh sau khi áp dụng thuật toán non-maximum suppression.....	41
Hình 2.26	Pixel yếu có màu xám và pixel mạnh có màu trắng .....	42
Hình 2.27	Không có pixel mạnh xung quanh .....	42
Hình 2.28	Có pixel mạnh ở cạnh .....	43
Hình 2.29	Hình ảnh kết quả cuối cùng sau khi thực hiện phương pháp Canny .....	43
Hình 2.30	Hình ảnh vận động viên được chia thành các lưới ô vuông.....	47
Hình 2.31	Mapping độ lớn gradients với các bins. ....	48
Hình 2.32	Ví dụ chia bins .....	49
Hình 2.33	Biểu đồ Histogram of Gradient gồm 9 bins tương ứng với một ô vuông trong lưới ô vuông. ....	49

Hình 2.34 Hình ảnh được phân chia thành lưới các ô vuông con, mỗi ô kích thước 8x8 pixel.....	50
Hình 2.35 Biểu diễn nhóm véc tơ histogram trên các lưới ô vuông của hình ảnh gốc. Các phương véc tơ phổ biến là chiều dọc trùng với chiều bức ảnh. ....	51
Hình 2.36 Mối quan hệ giữa AI, ML và DL.....	52
Hình 2.37 Sơ đồ tổng quát mạng nơ-ron tích chập .....	54
Hình 2.38 Mô tả phương pháp tích chập .....	55
Hình 2.39 Phương pháp max pooling.....	56
Hình 2.40 Phương pháp average pooling .....	56
Hình 2.41 Mô tả tầng kết nối đầy đủ.....	57
Hình 2.42 Quy trình huấn luyện mạng CNN, lan truyền xuôi và lan truyền ngược. ....	60
Hình 2.43 Hình Star-Convex (bên trái) và không phải là Star-Convex (bên phải) ..	62
Hình 2.44 Hình Star-Convex 2D.....	63
Hình 2.45 Hình Star-Convex 3D.....	63
Hình 2.46 Kiến trúc mạng U-net.....	64
Hình 2.47 Phương pháp sử dụng bounding boxes .....	66
Hình 2.48 Tổng quan kiến trúc Stardist .....	67
Hình 2.49 Xây dựng lại đối tượng từ các tia từ tâm .....	71
Hình 3.1 Kiến trúc hệ thống.....	76
Hình 3.2 Giao diện sau khi đánh label.....	84
Hình 3.3 Giao diện menu và màn hình home .....	88
Hình 3.4 Giao diện sử dụng phương pháp Otsu .....	88
Hình 3.5 Giao diện sử dụng phương pháp Canny.....	89
Hình 3.6 Giao diện sử dụng phương pháp HOG.....	89
Hình 3.7 Giao diện danh sách các mô hình đã được training .....	90
Hình 3.8 Giao diện popup cấu hình training .....	90
Hình 3.9 Giao diện phân vùng ảnh .....	90
Hình 3.10 Kết quả Loss sau quá trình huấn luyện với n_rays = 32; epochs = 20 .....	91
Hình 3.11 Kết quả Loss sau quá trình huấn luyện với n_rays = 32; epochs = 100 .....	92
Hình 3.12 Kết quả Loss sau quá trình huấn luyện với n_rays = 64; epochs = 50 .....	92
Hình 3.13 Kết quả Loss sau quá trình huấn luyện với n_rays = 64; epochs = 100 .....	93
Hình 3.14 Kết quả Loss sau quá trình huấn luyện với n_rays = 64; epochs = 200 .....	93
Hình 3.15 Kết quả phân vùng thử nghiệm.....	94

## MỞ ĐẦU

### 1. Lý do chọn đề tài

Bọt khí là một hiện tượng phổ biến trong lĩnh vực công nghiệp và các ngành khác. Trong sản xuất công nghiệp, bọt khí làm giảm chất lượng sản phẩm, cản trở quá trình truyền nhiệt, truyền khói, gây mài mòn thiết bị và tăng chi phí sản xuất do yêu cầu sử dụng thiết bị khử bọt. Trong y học và sinh học, bọt khí trong mạch máu có thể gây tắc mạch, đặc biệt nguy hiểm trong các ca phẫu thuật và quá trình sử dụng thiết bị y tế như máy lọc máu. Tương tự, trong thực phẩm và đồ uống, bọt khí ảnh hưởng đến hương vị, độ ổn định và hình thức của các sản phẩm như bia, nước giải khát và sữa; vì vậy, việc kiểm soát bọt là cần thiết để duy trì chất lượng và trải nghiệm của người tiêu dùng. Trong hóa học và dược phẩm, bọt khí có thể làm thay đổi tốc độ phản ứng và ảnh hưởng đến độ tinh khiết của dược phẩm, trong khi trong ngành khoa học vật liệu, chúng làm giảm độ bền, gây rỗ bè mặt và giảm tuổi thọ của sản phẩm như cao su, nhựa và kính. Cuối cùng, trong xử lý môi trường và nước thải, bọt khí gây cản trở cho các quy trình sinh học và hóa học, làm giảm hiệu suất xử lý nước thải.

Phân vùng ảnh là một kỹ thuật quan trọng trong xử lý ảnh, có vai trò phân chia ảnh thành các vùng có đặc trưng thống nhất. Đây là bước tiền xử lý cần thiết cho nhiều ứng dụng xử lý ảnh khác, chẳng hạn như nhận dạng đối tượng, theo dõi chuyển động, phân tích hình ảnh y tế, v.v. Trong những năm qua, các phương pháp phân vùng ảnh đã được nghiên cứu và phát triển mạnh mẽ. Các phương pháp cổ điển như phân vùng dựa trên ngữ cảnh, phân vùng dựa trên tính liên tục, phân vùng dựa trên thuộc tính, ... đã được nghiên cứu và phát triển từ lâu. Tuy nhiên, các phương pháp này có một số hạn chế như: nhạy cảm với điều kiện ánh sáng và màu sắc, khả năng xử lý nhiễu chưa tốt, khả năng xử lý biên giới không rõ ràng,... Do đó, việc áp dụng các phương pháp cổ thường tỏ ra thiếu chính xác trong việc xác định bọt trong chất lỏng.

Để khắc phục các hạn chế của các phương pháp cổ điển, với đề tài “ỨNG DỤNG CÁC PHƯƠNG PHÁP HỌC MÁY ĐỂ PHÂN VÙNG ẢNH VÀ XÂY DỰNG ỨNG

DỤNG XÁC ĐỊNH BỌT KHÍ TRONG CHẤT LỎNG” em xin được nghiên cứu các phương pháp phân vùng ảnh dựa trên các phương pháp học máy để cải thiện tính chính xác trong việc xác định bọt khí trong chất lỏng.

## 2. Tổng quan về vấn đề nghiên cứu

Phân vùng ảnh là một kỹ thuật quan trọng trong xử lý ảnh, có vai trò phân chia ảnh thành các vùng có đặc trưng thống nhất. Trong những năm qua, các phương pháp phân vùng ảnh đã được nghiên cứu và phát triển mạnh mẽ, trải qua hai giai đoạn chính:

### Giai đoạn sử dụng các phương pháp truyền thống:

Trong giai đoạn này, các phương pháp phân vùng ảnh chủ yếu dựa trên các đặc trưng đơn giản của ảnh, chẳng hạn như giá trị cường độ, độ tương phản, độ đồng nhất, v.v. Các phương pháp này có một số hạn chế như:

- Độ nhạy với nhiễu: Các phương pháp này dễ bị ảnh hưởng bởi nhiễu trong ảnh.
- Không linh hoạt: Các phương pháp này chỉ áp dụng được cho các loại ảnh nhất định.
- Yêu cầu nhiều tham số: Các phương pháp này thường yêu cầu người dùng cung cấp nhiều tham số.

### Hiện nay – kết hợp với các phương pháp học máy:

Với sự phát triển của học máy, các phương pháp phân vùng ảnh mới dựa trên trí tuệ nhân tạo đã được phát triển. Các phương pháp này sử dụng các mô hình học máy để học các đặc trưng phức tạp của ảnh, từ đó phân vùng ảnh một cách hiệu quả hơn.

Mạng nơ-ron tích chập (CNN) là một phương pháp phân vùng ảnh dựa trên học máy được sử dụng phổ biến nhất. CNN hoạt động bằng cách sử dụng các bộ lọc tích

chập để trích xuất các đặc trưng cục bộ của ảnh. Các đặc trưng này sau đó được sử dụng để phân loại các pixel của ảnh thành các vùng.

CNN có những ưu điểm vượt trội so với các phương pháp phân vùng ảnh cổ điển, chẳng hạn như:

- **Khả năng chống nhiễu tốt:** CNN có thể sử dụng các đặc trưng phức tạp của ảnh để chống lại nhiễu.
- **Linh hoạt hơn:** CNN có thể áp dụng được cho nhiều loại ảnh khác nhau.
- **Yêu cầu ít tham số hơn:** CNN thường chỉ yêu cầu một số tham số cơ bản.

### **3. Mục tiêu nghiên cứu của đề tài**

Mục đích của nghiên cứu này là tìm hiểu về các phương pháp phân vùng ảnh cổ điển và hiện đại (dựa trên mạng nơ ron tích chập), xây dựng ứng dụng phân vùng ảnh sử dụng học sâu, áp dụng cho bài toán xác định bọt khí trong chất lỏng để nâng cao độ chính xác phân vùng ảnh trong công nghiệp và có thể công bố kết quả nghiên cứu.

Cụ thể, nghiên cứu sẽ thực hiện các nội dung sau:

- Tìm hiểu một số phương pháp phân vùng ảnh cổ điển như: Phân vùng dựa trên ngưỡng (Threshold), phân vùng dựa trên cạnh (Edge-based segmentation).
- Tìm hiểu phương pháp phân vùng ảnh dựa trên mạng nơ ron tích chập (CNN) và mô hình Stardist.
- Xây dựng ứng dụng phân vùng ảnh dựa trên mạng nơ ron tích chập và mô hình Stardist, áp dụng cho việc xác định bọt trong chất lỏng.
- Định hướng công bố kết quả nghiên cứu.

### **4. Đối tượng và phạm vi nghiên cứu của đề tài**

#### **Đối tượng nghiên cứu**

Đối tượng nghiên cứu của đề tài là các phương pháp phân vùng ảnh, bao gồm các phương pháp phân vùng ảnh cổ điển và hiện đại (dựa trên mạng nơ ron tích chập).

Và áp dụng phương pháp phân vùng ảnh dựa trên mạng nơ ron tích chập và mô hình Stardist để xác định bọt trong chất lỏng.

### **Phạm vi nghiên cứu**

Phạm vi nghiên cứu của đề tài này bao gồm các nội dung sau:

- Nghiên cứu lý thuyết về các phương pháp phân vùng ảnh cổ điển và hiện đại.
- Xây dựng ứng dụng phân vùng ảnh dựa trên phương pháp hiện đại (dựa trên mạng nơ ron tích chập và mô hình học máy startdist).
- Đánh giá hiệu quả của ứng dụng phân vùng ảnh trên các tập dữ liệu thực tế.
- Viết bài báo đăng lên tạp chí có uy tín.

## **5. Phương pháp nghiên cứu của đề tài**

Phương pháp nghiên cứu được sử dụng trong đề tài này bao gồm các phương pháp sau:

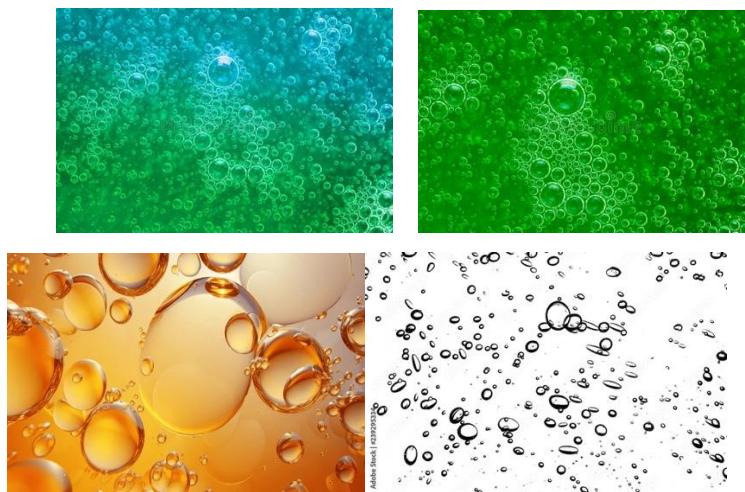
- Nghiên cứu lý thuyết: Nghiên cứu và đánh giá ưu điểm và nhược điểm của các phương pháp phân vùng ảnh cổ điển và hiện đại. Nghiên cứu mô hình học máy Stardist, từ đó áp dụng cho bài toán xác định bọt khí trong chất lỏng.
- Nghiên cứu thực nghiệm: Nghiên cứu thực nghiệm sẽ sử dụng các tiêu chí đánh giá sau:
  - Độ chính xác: Tỷ lệ số bọt khí được phân vùng chính xác trên tổng số bọt khí thực thể đã cho trong ảnh.
  - Tốc độ: Thời gian cần thiết để thực hiện phân vùng ảnh.
  - Khả năng mở rộng: Khả năng áp dụng phương pháp cho các hình ảnh có độ phân giải cao hoặc các hình ảnh có nhiều cao.

## CHƯƠNG 1 CƠ SỞ LÝ LUẬN

### 1.1. Khái quát về hiện tượng bọt khí trong chất lỏng

Bọt khí là hiện tượng vật lý xảy ra khi một hoặc nhiều bọt khí nhỏ được hình thành trong một chất lỏng hoặc chất rắn. Những bọt khí này thường có hình cầu hoặc hình dạng gần giống cầu và chứa không khí hoặc một loại khí khác bên trong. Bọt khí có thể được hình thành qua nhiều quá trình khác nhau, bao gồm phản ứng hóa học, sự khuấy trộn, hoặc sự gia tăng nhiệt độ.

Bọt khí thường được hình thành từ các bọt nhỏ, ngăn cách nhau bằng lớp màng mỏng. Kích thước và hình dạng của bọt khí có thể thay đổi tùy thuộc vào điều kiện môi trường (như áp suất, nhiệt độ, và độ nhớt của chất lỏng). Bọt khí có khả năng nổi lên bề mặt chất lỏng do sự chênh lệch mật độ giữa khí và chất lỏng. Chúng cũng có thể di chuyển và tương tác với nhau, tạo thành các cấu trúc bọt lớn hơn.



**Hình 1.1 Hình ảnh minh họa về bọt khí**

Bọt khí có khả năng nổi lên bề mặt chất lỏng do sự chênh lệch mật độ giữa khí và chất lỏng. Chúng cũng có thể di chuyển và tương tác với nhau, tạo thành các cấu trúc bọt lớn hơn. Đặc điểm của hầu hết các bọt khí khi xuất hiện trong chất lỏng là chúng thường có hình sao lồi, tức là hình dạng có xu hướng

phình to ra bên ngoài (chúng ta sẽ cùng nói kỹ hơn về hình sao lồi - star-convex trong chương 2).

## **1.2. Khái quát về phân vùng ảnh**

### **1.2.1. Định nghĩa phân vùng ảnh**

Theo IBM: *Image segmentation is a computer vision technique that partitions a digital image into discrete groups of pixels—image segments—to inform object detection and related tasks. By parsing an image's complex visual data into specifically shaped segments, image segmentation enables faster, more advanced image processing.*

Tạm dịch là: Phân vùng ảnh là một kỹ thuật thị giác máy tính phân chia một hình ảnh kỹ thuật số thành các nhóm pixel riêng biệt — các vùng ảnh — để hỗ trợ phát hiện đối tượng và các nhiệm vụ liên quan. Bằng cách phân tích dữ liệu hình ảnh phức tạp thành các vùng có hình dạng cụ thể, phân vùng ảnh giúp tăng tốc độ và cải thiện xử lý hình ảnh tiên tiến.

Theo Wikipedia: *In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.*

Tạm dịch là: Trong xử lý ảnh kỹ thuật số và thị giác máy tính, phân vùng ảnh là quá trình phân chia một hình ảnh kỹ thuật số thành nhiều vùng ảnh, còn được gọi là các vùng ảnh hoặc đối tượng ảnh (tập hợp các pixel). Mục tiêu của phân vùng là đơn giản hóa và/hoặc thay đổi cách biểu diễn hình ảnh để trở nên có ý nghĩa hơn và dễ phân tích hơn. Phân vùng ảnh thường được sử dụng để xác định vị trí của các đối tượng và biên giới (đường thẳng, đường cong, v.v.) trong ảnh. Cụ thể hơn, phân vùng

ảnh là quá trình gán nhãn cho mỗi pixel trong một hình ảnh sao cho các pixel cùng nhãn chia sẻ các đặc điểm nhất định.

Theo Bách khoa Toàn thư Việt Nam: *Phân vùng ảnh* (hay *Phân đoạn ảnh*, *tiếng Anh Image segmentation*) là quá trình phân chia ảnh thành các vùng hoặc đối tượng có tính chất thỏa mãn một tiêu chí xác định (có sự tương đồng về mức xám, kết cấu, màu sắc, v.v.). Mức độ chi tiết của việc phân chia phụ thuộc vào từng bài toán cần giải quyết. *phân vùng ảnh* là một bài toán căn bản nhưng cũng rất phức tạp trong chuỗi xử lý và phân tích ảnh nói chung bởi sự đa dạng trong định nghĩa cũng như tính chất của vùng hoặc đối tượng quan tâm trong ảnh.

### **1.2.2. Lịch sử phát triển của phân vùng ảnh**

Phân vùng ảnh là bài toán được đề cập và giải quyết từ những năm 1970 trong các công bố của Brice và Fenema. Năm 1974 Watanabe đề xuất kỹ thuật phân vùng ảnh dựa trên *lấy ngưỡng*. Năm 1978 Jack Sklandsky đề xuất kỹ thuật phân vùng ảnh dựa trên *phát hiện biên*. Kỹ thuật *lan vùng* xác định trực tiếp vùng bằng cách lan vùng từ một vị trí trong ảnh cho đến khi nào tiêu chí vùng vẫn còn thỏa mãn do Brice và Fenema đề xuất năm 1970, sau đó được cải tiến bởi Pavlidis và các cộng sự năm 1990, R. Adams and L. Bischof – 1994, Zugaj và cộng sự năm 1998; Hojjatoleslami, S. A. và Kittler, J – 1998.

Năm 1979, Coleman và Andrews giới thiệu kỹ thuật phân vùng ảnh dựa trên *phân cụm*. Kỹ thuật *Watershed* coi ảnh là một bề mặt topo, khi đó việc phân vùng được xem là thực hiện phép biến đổi watershed để tìm ra các vùng ảnh quan tâm. Kỹ thuật này lần đầu tiên được giới thiệu bởi Digabel và Lantu'ejoul vào năm 1978, tiếp tục được cải tiến bởi S Beucher 1992, V Grau và các cộng sự năm 2004.

Năm 2004, Rother và cộng sự đề xuất kỹ thuật *Graph cuts*. Năm 2006, Kato và Pong đề xuất kỹ thuật *Trường ngẫu nhiên markov có điều kiện*. Kỹ thuật này coi bài toán phân vùng là bài toán gán nhãn và đi tìm lời giải theo hướng tiếp cận xác suất. Hiện nay, các giải thuật học sâu đã cho những kết quả ánh tượng trên rất nhiều bài toán liên quan đến phân tích và hiểu ảnh. Các giải thuật học sâu coi bài toán phân vùng

ảnh là bài toán gán nhãn mức điểm ảnh và đưa ra các kết quả phân vùng thực thể ngữ nghĩa phục vụ cho nhiều bài toán sau đó. Năm 2015, Long và cộng sự đề xuất *Mạng tích chập đầy đủ* (*Fully Convolutional Networks*) cho bài toán phân vùng ảnh bằng cách thay đổi kiến trúc mạng VGG-16 và GoogleNet để xử lý các đầu vào và đầu ra có kích thước không cố định. Cũng vào năm này, No và cộng sự đề xuất Mô hình *Tự mã hóa – giải mã* cho phân vùng ảnh. Ren và cộng sự đề xuất mạng theo *Mô hình mạng tích chập vùng* (*R-CNN*) cho phân vùng thực thể.



**Hình 1.2 Hình ảnh minh họa về phân vùng ảnh**

Năm 2016, Visin và cộng sự đề xuất mạng theo *Mô hình hồi qui* (Recurrent Neural Network - RNN) cho phép mô hình hóa quan hệ phục thuộc ngắn/dài hạn giữa các điểm ảnh để cải thiện chất lượng phân vùng ảnh. Năm 2017, Lin và cộng sự đề xuất mạng FPN (Feature Pyramid Network) theo *Mô hình đa phân giải và cấu trúc kim tự tháp*, trong đó, cấu trúc phân cấp đa tầng các mạng CNN được lồng ghép để tạo các đặc trưng đa phân giải nhằm phát hiện các đối tượng ở kích thước khác nhau trong ảnh.

Năm 2018, Marcos và cộng sự đã đề xuất *Mô hình mạng CNN kết hợp với mô hình biên động* (active contour).

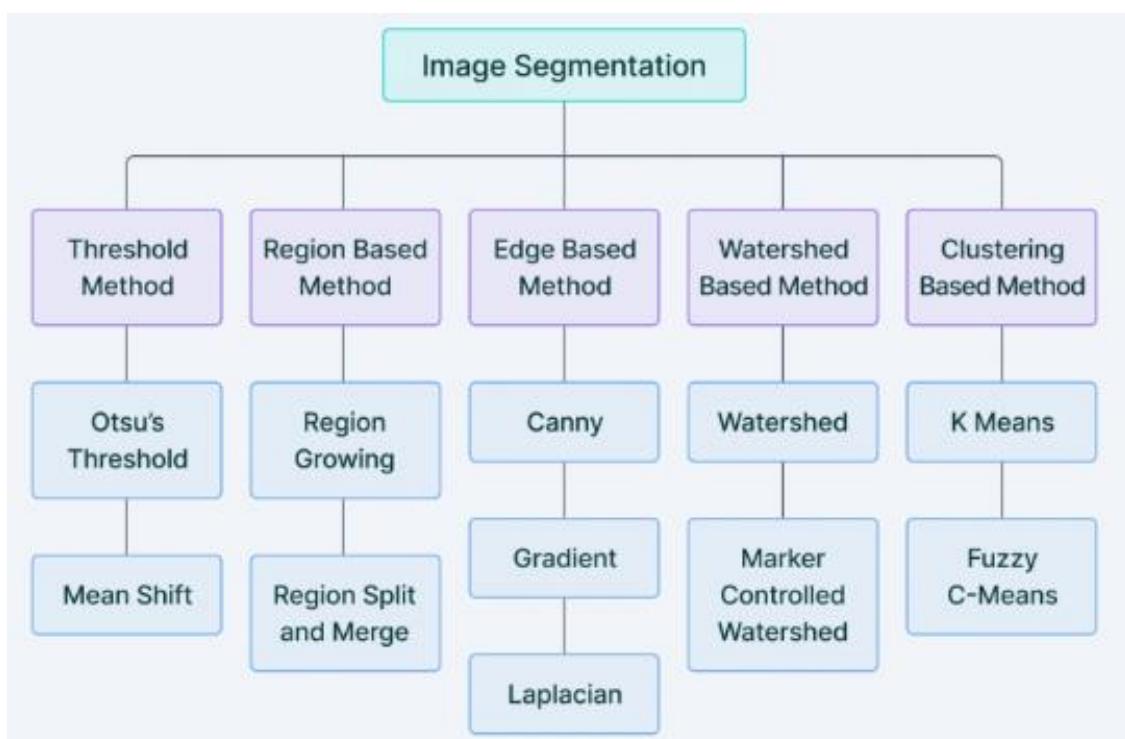
Hiện nay, tại Việt Nam, có nhiều phòng thí nghiệm, viện nghiên cứu, bộ môn hoặc các nhà nghiên cứu độc lập, các công ty đang theo đuổi và giải quyết các bài toán liên quan đến phân vùng ảnh phục vụ cho các ứng dụng khác nhau như phân vùng ảnh nội soi, ảnh X quang phục vụ các ứng dụng y tế (Viện MICA, Viện Điện tử Y Sinh – ĐHBK Hà Nội); phân vùng ảnh văn bản phục vụ tự động nhận dạng ký tự (Viện công nghệ thông tin thuộc Viện Hàn lâm KHCNVN); phân vùng ảnh vệ tinh (Viện công nghệ thông tin thuộc Viện Hàn lâm KHCNVN, Học viện Kỹ thuật Quân sự, Đại học Quốc gia Hà Nội); phân vùng ảnh giao thông trong điều hướng phân luồng (Công ty Biển Bạc), phân vùng ảnh người, tay trong ứng dụng giám sát hoặc tương tác người máy (Viện MICA, ĐHBK Hà Nội, Đại học Quốc gia HCM), v.v.

### **1.3. Kết luận Chương 1**

Chương 1 đã trình bày sơ qua về hiện tượng bọt khí trong chất lỏng, chúng ta đã biết được hiện tượng bọt khí là gì và đối tượng nghiên cứu cần phân vùng trong nghiên cứu này. Chúng ta cũng đã biết được lịch sử hình thành và phát triển của phân vùng hình ảnh (image segmentation). Những điều trên sẽ là nên tăng quan trọng và sẽ được nghiên cứu sâu hơn trong chương 2.

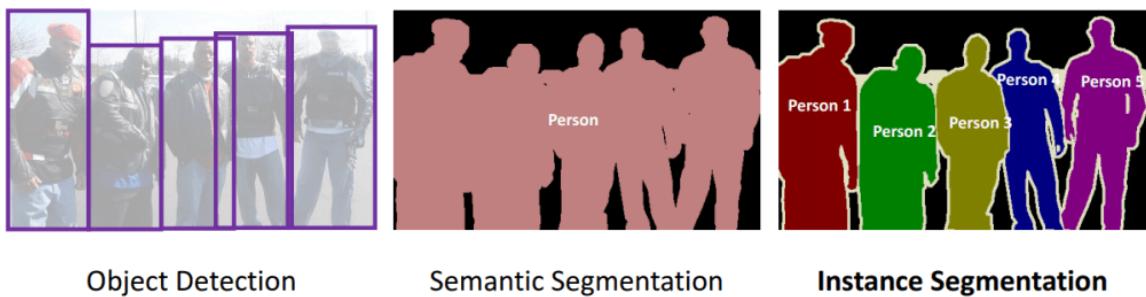
## CHƯƠNG 2 NGHIÊN CỨU VÀ PHÂN TÍCH MỘT SỐ PHƯƠNG PHÁP PHÂN ĐOẠN ẢNH

Phân đoạn ảnh hay thuật ngữ trong tiếng Anh là Segmentation là một phần quan trọng trong lĩnh vực thị giác máy tính (Computer Vision). Nó là quá trình chia nhỏ một bức ảnh thành nhiều phần, với mục tiêu đơn giản hóa hoặc thay đổi biểu diễn của bức ảnh để dễ dàng phân tích. Mức độ chi tiết của quá trình chia nhỏ phụ thuộc vào vấn đề đang được giải quyết. Nghĩa là, quá trình phân vùng nên dừng lại khi các đối tượng hoặc vùng quan tâm trong ứng dụng đã được xác định. Ví dụ, trong việc kiểm tra tự động các sản phẩm điện tử, mục tiêu là phân tích hình ảnh của sản phẩm để xác định sự có mặt hay thiếu vắng của các lỗi cụ thể, như các thành phần bị thiếu hoặc các đường kết nối bị hỏng. Việc phân vùng chi tiết hơn mức cần thiết để xác định các yếu tố này là không cần thiết. Phân vùng hình ảnh là một trong những nhiệm vụ khó khăn nhất trong xử lý ảnh. Độ chính xác của phân vùng quyết định sự thành công hay thất bại của các quy trình phân tích hình ảnh bằng máy tính.



*Hình 2.1 Các phương pháp phân đoạn hình ảnh truyền thống*

Phân vùng ảnh cũng có một mục tiêu chung với phát hiện vật thể (Object Detection) là phát hiện ra vùng ảnh chứa vật thể và gán nhãn phù hợp cho chúng. Tuy nhiên tiêu chuẩn về độ chính xác của Image Segmentation ở mức cao hơn so với Object Detection khi nó yêu cầu nhãn dự báo đúng tới từng pixel.



**Hình 2.2 Ví dụ về Object Detection và Image Segmentation**

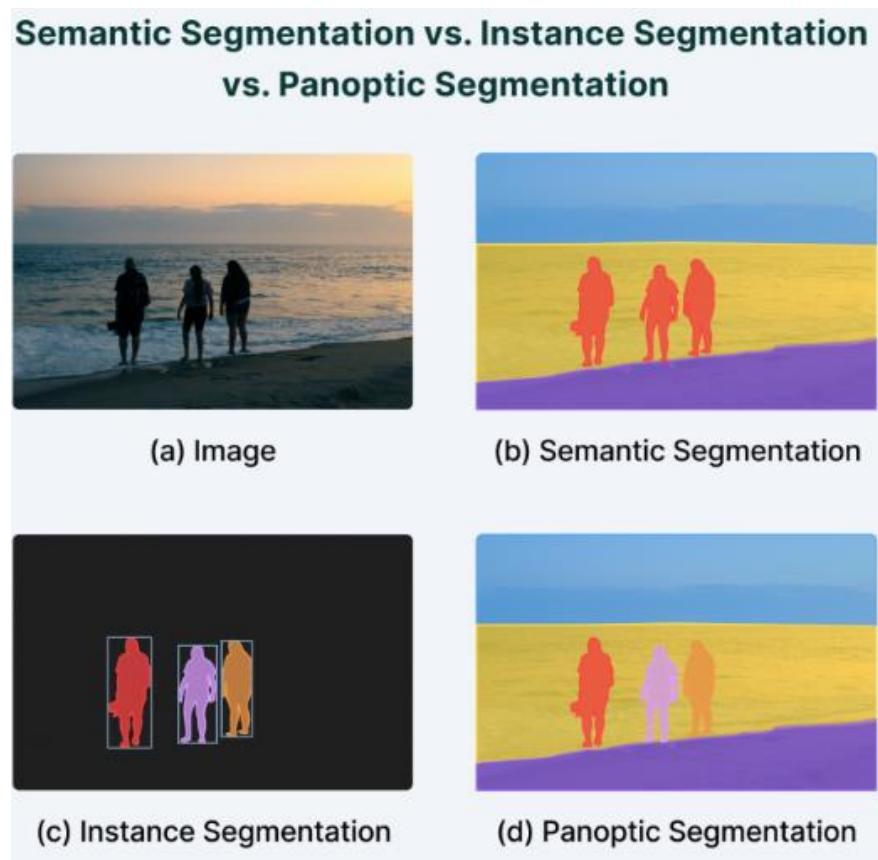
Trong quá trình này, mỗi pixel trong hình ảnh được liên kết với một loại đối tượng. Phân vùng ảnh có thể được chia thành ba nhóm tác vụ dựa trên số lượng và loại thông tin mà chúng truyền tải: Phân vùng ngữ nghĩa (semantic segmentation), phân vùng cá thể (instance segmentation) và phân vùng toàn cảnh (panoptic segmentation).

Semantic segmentation đề cập đến việc phân loại các pixel trong một hình ảnh thành các lớp ngữ nghĩa. Các pixel thuộc về một lớp cụ thể được phân loại một cách đơn giản vào lớp đó mà không cần xem xét thông tin hoặc bối cảnh nào khác. Tác vụ này không phù hợp khi có nhiều đối tượng được nhóm chặt chẽ trên cùng một lớp trong hình ảnh. Ví dụ: Hình ảnh đám đông trên đường phố sẽ có mô hình semantic segmentation dự đoán toàn bộ khu vực đám đông thuộc về lớp “người đi bộ”, do đó cung cấp rất ít chi tiết hoặc thông tin chuyên sâu về hình ảnh.

Các mô hình instance segmentation phân loại các pixel thành các danh mục trên cơ sở “các đối tượng” (instances) với nhãn cụ thể. Ví dụ: Nếu một hình ảnh về đám đông được đưa vào mô hình instance segmentation, thì mô hình đó sẽ có thể

tách biệt từng người khỏi đám đông cũng như các đối tượng xung quanh (lý tưởng nhất).

Panoptic segmentation, tác vụ phân vùng được phát triển gần đây nhất, là sự kết hợp giữa semantic segmentation và instance segmentation, trong đó ranh giới của mỗi đối tượng trong ảnh được tách biệt và danh tính của đối tượng được dự đoán. Các thuật toán panoptic segmentation có khả năng ứng dụng quy mô lớn trong các tác vụ phổ biến như ô tô tự lái, khi một lượng lớn thông tin về môi trường xung quanh phải được ghi lại ngay lập tức với sự trợ giúp của một luồng hình ảnh.



**Hình 2.3 Ba loại tác vụ phân đoạn hình ảnh**

Trong chương này sẽ giới thiệu về một số phương pháp phân đoạn truyền thống như là phân đoạn dựa trên phân ngưỡng cường độ sáng (Thresholding), phân đoạn dựa trên cạnh (Edge-based segmentation). Tiếp theo đó là các phương pháp phân

đoạn hình ảnh hiện đại dựa trên các phương pháp học máy mà phổ biến nhất hiện nay là sử dụng mạng nơ-ron tích chập – Convolutional neural network – hay còn được biết đến là CNN. Trong đề án này, tôi sẽ tập chung vào nghiên cứu một mô hình cụ thể là Stardish, dựa vào kết quả nghiên cứu đó để xây dựng lên ứng dụng phân đoạn ảnh.

## **2.1. Phân vùng dựa trên phân ngưỡng cường độ sáng (Thresholding)**

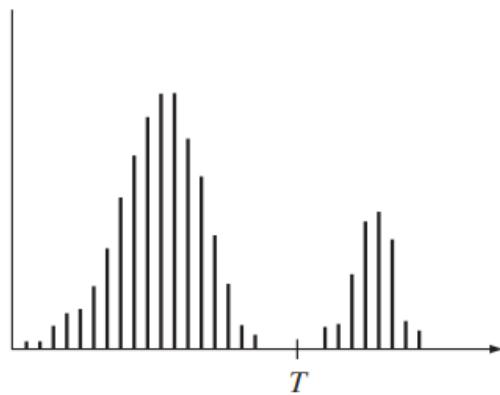
Trong xử lý hình ảnh kỹ thuật số, ngưỡng là phương pháp phân vùng hình ảnh đơn giản nhất. Nó đóng vai trò quan trọng trong xử lý hình ảnh vì nó cho phép phân vùng và trích xuất thông tin quan trọng từ hình ảnh. Bằng cách chia hình ảnh thành các vùng riêng biệt dựa trên cường độ điểm ảnh hoặc giá trị điểm ảnh, ngưỡng giúp phân biệt các đối tượng hoặc đặc điểm quan tâm với nền. Kỹ thuật này được sử dụng rộng rãi trong nhiều ứng dụng khác nhau như phát hiện đối tượng, phân vùng hình ảnh và nhận dạng ký tự, cho phép phân tích và diễn giải hiệu quả các hình ảnh kỹ thuật số. Ngoài ra, ngưỡng hình ảnh có thể nâng cao chất lượng hình ảnh bằng cách giảm nhiễu và cải thiện độ rõ nét hình ảnh tổng thể.

### **2.1.1. Lý thuyết cơ sở của ngưỡng cường độ**

Ngưỡng hình ảnh liên quan đến việc chia một ảnh thành hai hoặc nhiều vùng dựa trên mức cường độ, cho phép phân tích và trích xuất dễ dàng các đặc điểm mong muốn. Bằng cách đặt giá trị ngưỡng, các pixel có cường độ trên hoặc dưới ngưỡng có thể được phân loại theo đó. Kỹ thuật này hỗ trợ các tác vụ như phát hiện đối tượng, phân vùng và nâng cao hình ảnh.

Ngưỡng ảnh là một kỹ thuật đơn giản hóa ảnh thang độ xám thành ảnh nhị phân bằng cách phân loại từng giá trị pixel thành đen hoặc trắng dựa trên mức cường độ hoặc mức độ xám so với giá trị ngưỡng. Kỹ thuật này giảm ảnh xuống chỉ còn hai mức cường độ, giúp dễ dàng xác định và cô lập các đối tượng quan tâm. Chuyển đổi ảnh nhị phân cho phép xử lý và phân tích ảnh hiệu quả, cho phép nhiều ứng dụng thị giác máy tính như phát hiện cạnh và nhận dạng mẫu.

Trong các thuật toán xử lý hình ảnh, nguyên tắc phân loại pixel dựa trên ngưỡng cường độ được sử dụng rộng rãi. Bằng cách thiết lập một giá trị ngưỡng cụ thể, các pixel có mức cường độ trên ngưỡng được phân loại là màu trắng, trong khi các pixel dưới ngưỡng được phân loại là màu đen. Nguyên tắc này tạo thành nền tảng cho nhiều kỹ thuật tăng cường hình ảnh khác nhau giúp trích xuất các đặc điểm quan trọng từ hình ảnh để phân tích thêm.



**Hình 2.4 Biểu đồ cường độ (Histogram)**

Giả sử rằng biểu đồ cường độ trong Hình 2.1 tương ứng với một hình ảnh  $f(x,y)$  bao gồm các vật thể sáng trên nền tối, theo cách mà các điểm ảnh của vật thể và nền có các giá trị cường độ được nhóm thành hai vùng chính. Một cách rõ ràng để tách các vật thể ra khỏi nền là chọn một ngưỡng,  $T$ , ngưỡng này sẽ phân chia các vùng này. Sau đó, bất kỳ điểm nào  $(x,y)$  trong hình ảnh  $f(x,y) > T$  được gọi là điểm vật thể; ngược lại, điểm này được gọi là điểm nền. Nói cách khác, hình ảnh phân vùng,  $g(x,y)$ , được cho bởi:

$$g(x,y) = \begin{cases} 1 & \text{nếu } f(x,y) > T \\ 0 & \text{nếu } f(x,y) \leq T \end{cases}$$

Khi  $T$  là hằng số áp dụng cho toàn bộ hình ảnh, quá trình này được gọi là ngưỡng toàn cục. Khi giá trị của  $T$  thay đổi trong ảnh, quá trình này được gọi là ngưỡng biến. Thuật ngữ ngưỡng cục bộ hoặc ngưỡng vùng đôi khi được sử dụng để chỉ loại ngưỡng biến trong đó giá trị của  $T$  tại bất kỳ điểm nào  $(x,y)$  trong ảnh phụ thuộc vào các thuộc tính của vùng lân cận của  $(x,y)$  (ví dụ, cường độ trung bình của các điểm ảnh trong vùng lân cận đó). Nếu  $T$  phụ thuộc trực tiếp vào tọa độ không

gian (x,y), thì ngưỡng biến thường được gọi là ngưỡng động hoặc ngưỡng thích ứng. Tuy nhiên, các thuật ngữ này không được sử dụng một cách nhất quán và có thể được dùng thay thế lẫn nhau trong tài liệu xử lý hình ảnh.

### **2.1.2. Ngưỡng toàn cục (Global Thresholding)**

#### **2.1.2.1. Lý thuyết**

Global Thresholding là một trong các kỹ thuật phân ngưỡng (Threshold) đơn giản, được sử dụng rộng rãi, trong đó một giá trị ngưỡng duy nhất được áp dụng trên toàn bộ hình ảnh để phân chia các điểm ảnh thành hai nhóm: đối tượng (foreground) và nền (background). Mục tiêu của ngưỡng toàn cục là xác định ranh giới giữa các điểm ảnh thuộc đối tượng và các điểm ảnh thuộc nền, dựa trên cường độ sáng của từng điểm ảnh.

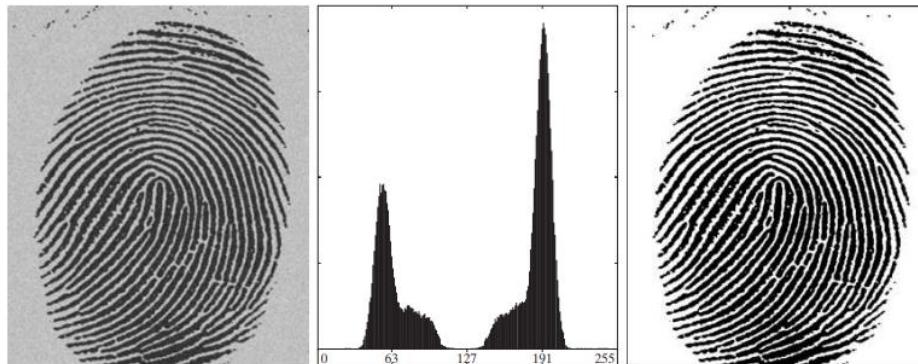


**Hình 2.5 Ví dụ sử dụng ngưỡng toàn cục**

#### **2.1.2.2. Tìm ngưỡng toàn cục tối ưu bằng phương pháp Otsu**

Phương pháp Otsu, được đặt theo tên của một nhà nghiên cứu người Nhật đã nghĩ ra ý tưởng cho việc tính ngưỡng một cách tự động *Nobuyuki Otsu* (大津展之, Ōtsu Nobuyuki). Thuật toán Otsu là một phương pháp để xác định ngưỡng phân vùng ảnh tự động trong xử lý ảnh. Nó dựa trên việc phân tích biểu đồ phân bố điểm ảnh (histogram) của ảnh nhằm tìm ra một giá trị ngưỡng tối ưu để phân tách các vùng có cường độ sáng khác nhau. Phương pháp này tìm ngưỡng sao cho sự phân biệt giữa

các vùng sáng và tối trong ảnh là lớn nhất. Dưới đây là hình ảnh ví dụ cho quá trình phân ngưỡng, từ trái qua phải là ảnh gốc, biểu đồ phân bố điểm ảnh (histogram), và hình ảnh kết quả sau khi được phân ngưỡng (hình ảnh được lấy từ Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ - National Institute of Standards and Technology):



**Hình 2.6 Ví dụ phân ngưỡng bằng phương pháp Otsu**

Phương pháp Otsu là một kỹ thuật phân ngưỡng ảnh tự động dựa trên phân tích thống kê. Phương pháp này được thiết kế để tìm ra ngưỡng tối ưu  $k^*$  nhằm phân chia các pixel của ảnh thành hai lớp sao cho phương sai giữa các lớp đạt giá trị cực đại, từ đó đảm bảo rằng các lớp này có sự phân biệt rõ ràng nhất về mức xám. Cụ thể, cho một ảnh có các mức cường độ xám rời rạc  $i = 0, 1, \dots, L - 1$  và kích thước  $M \times N$  pixel, số lượng pixel có mức xám  $i$  được ký hiệu là  $n_i$ . Tổng số pixel trong ảnh là  $MN = n_0 + n_1 + n_2 + \dots + n_{L-1}$ , và histogram chuẩn hóa  $p_i$  có thể được tính theo công thức:

$$p_i = \frac{n_i}{MN} \quad , \text{với } i = 0, 1, \dots, L - 1.$$

Sau khi xây dựng histogram chuẩn hóa, ta chọn một ngưỡng kkk để phân ảnh thành hai lớp C1 và C2. Lớp C1 bao gồm các pixel có mức xám từ 0 đến  $k$ , còn lớp C2 chứa các pixel có mức xám từ  $k+1$  đến  $L-1$ . Xác suất để một pixel thuộc về lớp C1 hoặc C2 lần lượt là:

$$P_1(k) = \sum_{i=0}^k p_i \quad \text{và} \quad P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

Giá trị trung bình cường độ của hai lớp được tính lần lượt bởi:

$$m_1(k) = \frac{\sum_{i=0}^k i p_i}{P_1(k)} \quad \text{và} \quad m_2(k) = \frac{\sum_{i=k+1}^{L-1} i p_i}{P_2(k)}$$

Trong đó, giá trị trung bình tổng thể của ảnh là:

$$m(k) = \sum_{i=0}^{L-1} i p_i$$

Cường độ trung bình của toàn bộ hình ảnh (hay còn gọi là giá trị trung bình toàn cục) được tính bằng:

$$m_G = \sum_{i=0}^{L-1} i p_i$$

Tính hợp lệ của hai phương trình sau có thể được kiểm chứng bằng cách thay thế trực tiếp vào các kết quả trước đó:

$$P_1 m_1 + P_2 m_2 = m_G$$

$$P_1 + P_2 = 1$$

Ở đây, chúng ta đã bỏ qua chỉ số k tạm thời để làm rõ ký hiệu. Để đánh giá "độ tốt" của ngưỡng tại mức k, chúng ta sử dụng chỉ số chuẩn hóa, không có đơn vị:

$$\eta = \frac{\sigma_B^2}{\sigma_G^2}$$

Trong đó,  $\sigma_G^2$  là phương sai toàn cục (tức là độ phân tán cường độ của tất cả các điểm ảnh trong hình):

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$$

và  $\sigma_B^2$  là phương sai giữa các lớp, được xác định như sau:

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2$$

Phương sai giữa các lớp (giữa hai lớp C1 và C2) được biểu diễn dưới dạng:

$$\begin{aligned} \sigma_B^2(k) &= P_1 P_2 [m_1 - m_2]^2 \\ &= \frac{(m_G P_1 - m)^2}{P_1 (1 - P_1)} \end{aligned}$$

Cách viết này hiệu quả hơn một chút về mặt tính toán vì giá trị tổng thể  $m_G$  chỉ cần tính một lần, do đó chỉ cần tính hai tham số  $m$  và  $P_1$  cho bất kỳ giá trị nào của  $k$ . Ta thấy rằng khoảng cách giữa hai giá trị trung bình  $m_1$  và  $m_2$  càng lớn thì phương sai giữa các lớp  $\sigma_B^2$  càng cao, thể hiện rằng phương sai giữa các lớp là thước đo khả năng phân tách giữa các lớp. Vì  $\sigma_G^2$  là một hằng số, nên  $\eta$  cũng là một thước đo cho khả năng phân tách, và việc tối ưu hóa thước đo này tương đương với việc tối ưu hóa  $\sigma_B^2$ . Mục tiêu là tìm ngưỡng  $k$  sao cho phương sai giữa các lớp được tối đa hóa, như đã đề cập ở phần đầu. Cần lưu ý rằng phương sai này chỉ bằng 0 khi tất cả các mức cường độ trong ảnh đều giống nhau, tức là chỉ có một lớp điểm ảnh duy nhất. Điều này đồng nghĩa với việc  $\eta=0$  đối với một bức ảnh đồng nhất, vì không có khả năng phân tách nào giữa các điểm trong cùng một lớp.

Đưa chỉ số  $k$  vào, ta có kết quả là:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$$

Trong đó:

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}$$

Do đó, giá trị ngưỡng tối ưu có thể được tìm thấy là  $k^*$  với giá trị của  $\sigma_B^2(k^*)$  là lớn nhất:

$$\sigma_B^2(k^*) = \max_{0 \leq k \leq L-1} \sigma_B^2(k)$$

Nói cách khác, để tìm  $k^*$ , chúng ta chỉ cần đánh giá phương trình cho tất cả các giá trị nguyên của  $k$  (với điều kiện  $0 < P_1(k) < 1$  thỏa mãn) và chọn giá trị  $k$  cho kết quả  $\sigma_B^2(k)$  lớn nhất. Nếu có nhiều hơn một giá trị  $k$  cho kết quả tối đa, thường sẽ tính trung bình các giá trị kkk này để tìm giá trị tối ưu. Có thể chứng minh rằng luôn tồn tại giá trị tối đa với điều kiện  $0 < P_1(k) < 1$ . Việc đánh giá các phương trình cho tất cả các giá trị  $k$  là một quá trình tính toán tương đối đơn giản vì số lượng giá trị nguyên lớn nhất mà  $k$  có thể có là  $L$ .

Khi đã tìm được  $k^*$ , ảnh đầu vào  $f(x,y)$  được phân vùng như công thức đã nhắc đến ở phần trước:

$$g(x, y) = \begin{cases} 1 & \text{nếu } f(x, y) > k^* \\ 0 & \text{nếu } f(x, y) \leq k^* \end{cases}$$

Với  $x = 0, 1, 2, \dots, M - 1$  và  $y = 0, 1, 2, \dots, N - 1$ .

Ngoài ngưỡng tối ưu, các thông tin khác về hình ảnh phân vùng cũng có thể được rút ra từ histogram. Ví dụ,  $P_1(k^*)$  và  $P_2(k^*)$ , là xác suất của các lớp tại ngưỡng tối ưu, biểu thị tỷ lệ diện tích mà các lớp (nhóm điểm ảnh) chiếm trong hình ảnh sau khi phân ngưỡng. Tương tự, các giá trị trung bình  $m_1(k^*)$  và  $m_2(k^*)$  là các ước lượng của cường độ trung bình của các lớp trong hình ảnh gốc.

Hệ số chuẩn hóa, được đánh giá tại giá trị ngưỡng tối ưu  $\eta(k^*)$ , có thể được sử dụng để đánh giá định lượng về độ tách biệt giữa các lớp, qua đó đưa ra ý tưởng về độ dễ phân ngưỡng của một hình ảnh cụ thể. Giá trị này nằm trong khoảng từ:

$$0 < \eta(k^*) \leq 1$$

Sau khi xác định ngưỡng tối ưu  $k^*$ , ảnh sẽ được phân ngưỡng: tất cả các pixel có giá trị nhỏ hơn hoặc bằng  $k^*$  sẽ được gán vào lớp nền (0), trong khi các pixel lớn hơn  $k^*$  sẽ thuộc lớp đối tượng (1). Phương pháp này hiệu quả vì chỉ cần tính toán dựa trên histogram của ảnh, giúp giảm thiểu thời gian và tài nguyên tính toán.

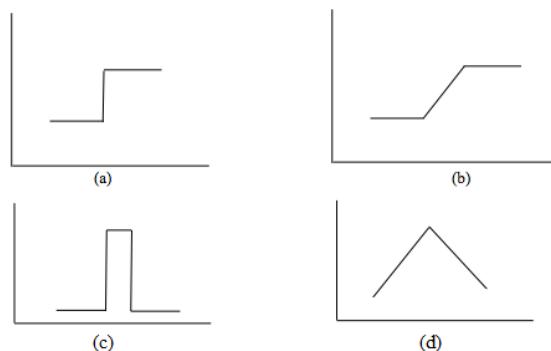
## 2.2. Phân vùng dựa trên cạnh (Edge-based segmentation)

Phân đoạn dựa trên cạnh hay tên tiếng Anh, để cho là Edge-based segmentation là một phương pháp phân vùng ảnh trong lĩnh vực thị giác máy tính, dựa trên việc phát hiện các cạnh trong hình ảnh để chia ảnh thành các vùng riêng biệt. Mỗi cạnh đại diện cho ranh giới giữa các đối tượng hoặc khu vực có sự thay đổi đột ngột về cường độ hoặc màu sắc. Phương pháp này giúp nhận diện và phân biệt các đối tượng trong ảnh, chẳng hạn như tách nền và đối tượng, hoặc xác định các vùng có ý nghĩa đặc biệt trong phân tích hình ảnh.

Cách tiếp cận này thường được sử dụng trong các ứng dụng mà các ranh giới rõ ràng giữa các vùng hoặc đối tượng có thể được phát hiện, như nhận dạng khuôn mặt, nhận diện vật thể, và phân tích y tế.

Phần này tập trung vào các phương pháp phân vùng dựa trên việc phát hiện các thay đổi cục bộ đột ngột trong cường độ. Ba loại đặc trưng hình ảnh mà chúng ta quan tâm là các điểm rời rạc, đường thẳng và cạnh. Pixel cạnh là những pixel mà tại đó cường độ của hàm hình ảnh thay đổi đột ngột, và cạnh (hay đoạn cạnh) là tập hợp các pixel cạnh được kết nối với nhau. Bộ dò cạnh là các phương pháp xử lý ảnh cục bộ được thiết kế để phát hiện các pixel cạnh. Một đường thẳng có thể được xem như một đoạn cạnh, trong đó cường độ nền ở hai phía của đường thẳng cao hơn hoặc thấp hơn nhiều so với cường độ của các pixel trên đường thẳng.

Một số kiểu đoạn cạnh thông dụng:



**Hình 2.7 Một số đoạn cạnh thông dụng**

(a) Cạnh dạng nhảy bậc      (b) Cạnh dốc      (c) Cạnh dạng xung vuông      (d) Cạnh dạng hình nón

### 2.2.1. Lý thuyết cơ sở

Phần này tập trung vào việc sử dụng các phương pháp làm mịn ảnh qua trung bình cục bộ. Vì trung bình hóa tương tự như tích phân, nên cũng không có gì bất ngờ khi có thể phát hiện các thay đổi đột ngột, cục bộ trong cường độ ảnh bằng cách sử dụng đạo hàm. Đặc biệt, các đạo hàm bậc nhất và bậc hai rất phù hợp cho mục đích này.

Đạo hàm của hàm số kỹ thuật số được xác định bằng cách dựa trên sự chênh lệch. Có nhiều cách để xấp xỉ chênh lệch này, việc xấp xỉ đạo hàm bậc nhất cần đáp ứng các điều kiện sau: (1) bằng 0 ở những vùng có cường độ không đổi; (2) khác 0 ở điểm bắt đầu của bước hoặc đoạn thay đổi cường độ; và (3) khác 0 tại các điểm trên đoạn thay đổi cường độ. Tương tự, để xấp xỉ đạo hàm bậc hai, ta yêu cầu rằng: (1) nó phải bằng 0 ở các vùng có cường độ không đổi; (2) khác 0 ở điểm bắt đầu và kết thúc của bước hoặc đoạn thay đổi cường độ; và (3) phải bằng 0 đọc theo các đoạn thay đổi cường độ. Do các giá trị kỹ thuật số có giới hạn, sự thay đổi cường độ lớn nhất cũng bị giới hạn, và khoảng cách ngắn nhất mà sự thay đổi có thể xảy ra là giữa các pixel liền kề.

Ta có thể tính xấp xỉ đạo hàm bậc nhất tại điểm  $x$  của một hàm một chiều  $f(x)$  bằng cách mở rộng hàm  $f(x+\Delta x)$  thành chuỗi Taylor xung quanh  $x$ , cho  $\Delta x=1$ , và giữ lại các thành phần tuyến tính. Kết quả thu được là:

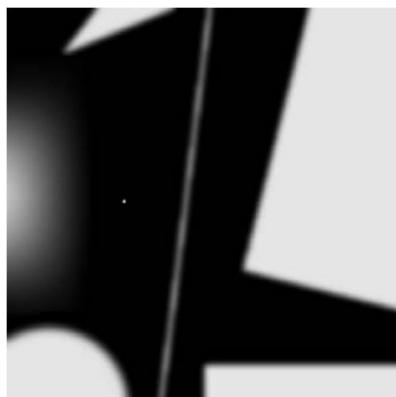
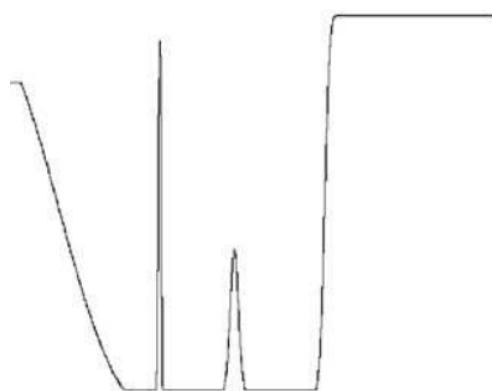
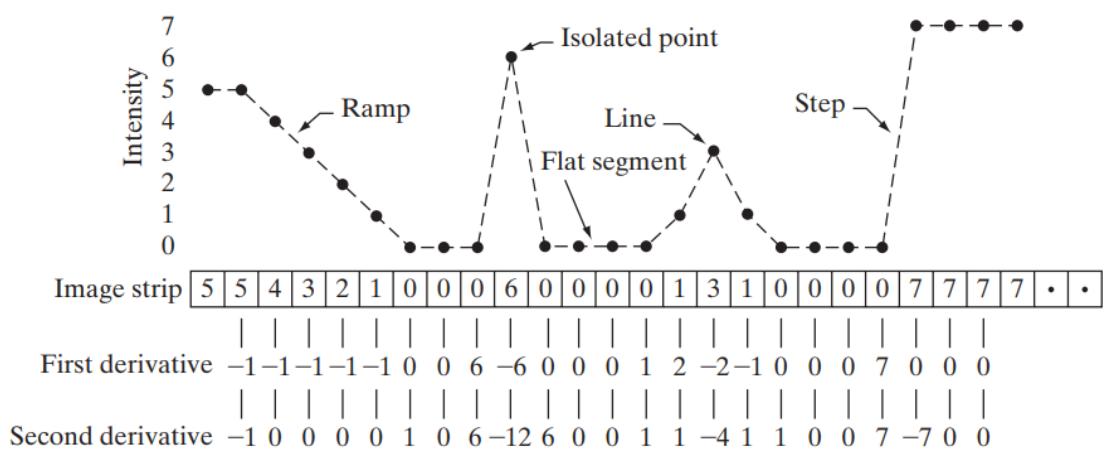
$$\frac{\partial^2 f}{\partial x^2} = f'(x) = f(x+1) - f(x)$$

Ký hiệu đạo hàm riêng được dùng ở đây để giữ tính nhất quán khi xét đến hàm ảnh hai biến  $f(x, y)$ , lúc này ta sẽ cần các đạo hàm riêng đọc theo hai trục không gian. Rõ ràng,  $\frac{\partial f}{\partial x} = \frac{df}{dx}$  chỉ là hàm của một biến duy nhất. Để tính đạo hàm bậc hai, ta lấy đạo hàm của phương trình trên theo  $x$ :

$$\frac{\partial^2 f}{\partial x^2} = f(x+2) - 2f(x+1) + f(x)$$

Các phương trình trên đáp ứng các điều kiện đã nêu về đạo hàm bậc nhất và bậc hai.

Hình 2.8 cho thấy một ảnh có chứa các vật thể rắn, một đường thẳng và một điểm nhiễu đơn lẻ. Hình 2.9 là một biểu đồ cường độ ngang của ảnh, bao gồm cả điểm rời rạc. Các chuyển tiếp cường độ giữa các vật thể rắn và nền đọc theo đường quét cho thấy hai loại cạnh: cạnh dạng dốc (trái) và cạnh dạng bậc (phải).

**Hình 2.8 Hình ảnh ví dụ****Hình 2.9 Biểu đồ cường độ ngang đi qua trung tâm hình, bao gồm cả điểm nhiễu đơn lẻ****Hình 2.10 Biểu đồ đơn giản hóa (các điểm được nối bằng nét gạch cho rõ ràng hơn). Dải hình ảnh này tương ứng với biểu đồ cường độ, và các số trong ô biểu thị giá trị cường độ của các điểm trên biểu đồ.**

Hình 2.10 là phiên bản đơn giản của biểu đồ, với các điểm đú để phân tích cách đạo hàm bậc nhất và bậc hai phản ứng với điểm nhiễu, đường thẳng và cạnh của các vật thể. Qua các điểm trên, ta rút ra các kết luận sau:

- Đạo hàm bậc nhất thường tạo ra cạnh dày hơn trong ảnh.
- Đạo hàm bậc hai có phản ứng mạnh hơn với các chi tiết mảnh như đường mảnh, điểm riêng lẻ và nhiễu.

- Đạo hàm bậc hai tạo ra phản ứng kép ở các cạnh dạng dốc và dạng bậc.
- Dấu của đạo hàm bậc hai có thể dùng để xác định hướng chuyển tiếp từ sáng sang tối hoặc từ tối sang sáng ở các cạnh.

Phương pháp phổ biến để tính đạo hàm bậc nhất và bậc hai tại mỗi vị trí pixel trong ảnh là sử dụng các bộ lọc không gian. Với mặt nạ lọc (mask) 3 x 3 trong hình 2.11, quy trình bao gồm việc tính tổng tích giữa các hệ số của mặt nạ và giá trị cường độ của các pixel trong vùng mà mặt nạ bao phủ. Cụ thể, đáp ứng của mặt nạ tại điểm trung tâm của vùng được tính bằng:

$$R = w_1z_1 + w_2z_2 + \dots + w_9z_9 = \sum_{k=1}^9 w_k z_k$$

Trong đó,  $z$  là cường độ của pixel tại vị trí tương ứng với hệ số thứ  $k$  trong mặt nạ. Nói cách khác, việc tính đạo hàm dựa trên mặt nạ không gian thực chất là quá trình lọc không gian của ảnh bằng các mặt nạ này.

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

**Hình 2.11 Bộ lọc kernel 3x3 tổng quát**

## 2.2.2. Phương pháp Laplacian of Gaussian (LoG)

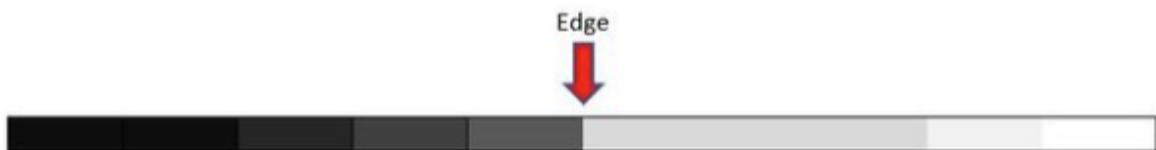
### 2.2.2.1. Cách hoạt động của phương pháp Laplacian of Gaussian (LoG)

Laplacian of Gaussian (LoG) kết hợp hai phép toán: **Laplacian** (đạo hàm bậc 2) và **Gaussian** (làm mờ ảnh). Phương pháp này giúp phát hiện biên trong hình ảnh và

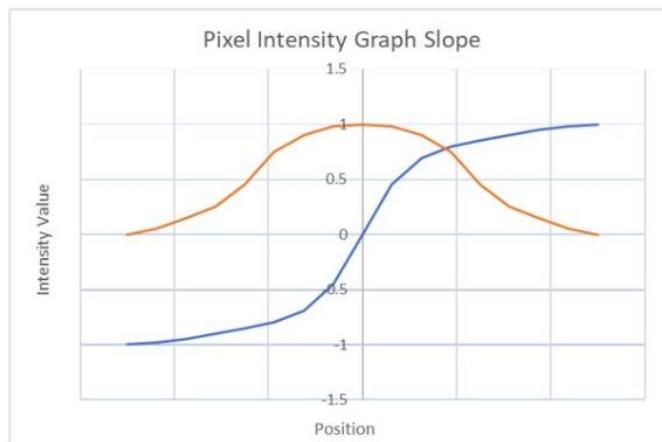
đồng thời giảm nhiễu, cải thiện độ chính xác trong quá trình phát hiện các biên quan trọng.

Các thuật toán phát hiện cạnh như Toán tử Sobel hoạt động trên đạo hàm bậc nhất của một hình ảnh. Nói cách khác, nếu chúng ta có một biểu đồ về các giá trị cường độ cho mỗi pixel trong một hình ảnh, Toán tử Sobel sẽ xem xét nơi độ dốc của biểu đồ cường độ đạt đến đỉnh và đỉnh đó được đánh dấu là một cạnh.

Hãy thử xét ví dụ sau, đối với hình ảnh  $10 \times 1$  pixel được cho ở hình 2.12, đường cong màu xanh bên dưới là biểu đồ các giá trị cường độ và đường cong màu cam là biểu đồ đạo hàm bậc nhất của đường cong màu xanh. Nói một cách dễ hiểu, đường cong màu cam là biểu đồ độ dốc.



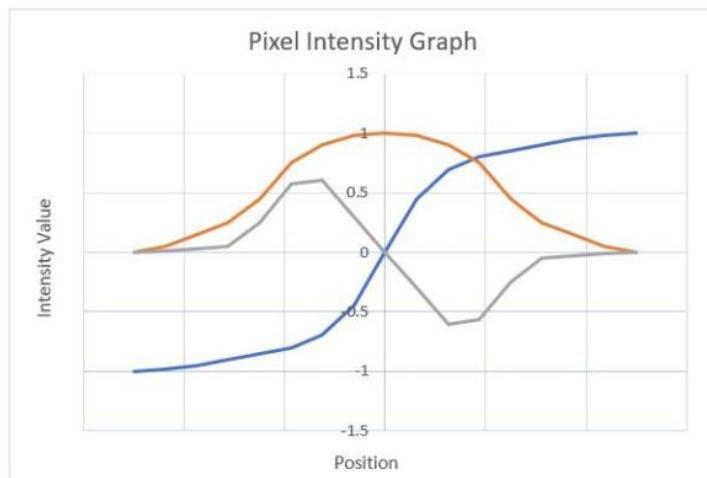
**Hình 2.12 Ví dụ hình ảnh  $10 \times 1$  pixel**



**Hình 2.13 Đồ thị độ dốc cường độ điểm ảnh**

Đường cong màu cam đạt đỉnh ở giữa, vì vậy chúng ta biết rằng có khả năng đó là một cạnh. Khi chúng ta nhìn vào hình ảnh gốc, chúng ta xác nhận rằng đúng, đó là một cạnh. Một hạn chế với cách tiếp cận trên là đạo hàm đầu tiên của hình ảnh có thể bị nhiễu nhiều. Các đỉnh cục bộ trong độ dốc của các giá trị cường độ có thể là do bóng hoặc các thay đổi màu nhỏ không phải là cạnh.

Một giải pháp thay thế cho việc sử dụng đạo hàm bậc nhất của một hình ảnh là sử dụng đạo hàm bậc hai, là độ dốc của đường cong đạo hàm bậc nhất (tức là đường cong màu cam ở trên). Đường cong như vậy trông giống như thế này (xem đường cong màu xám bên dưới):



**Hình 2.14 Đồ thị độ dốc cường độ điểm ảnh**

Một cảnh xuất hiện khi đồ thị của đạo hàm bậc hai cắt 0. Phương pháp dựa trên đạo hàm bậc hai này được gọi là thuật toán Laplacian. Thuật toán Laplacian cũng bị nhiễu. Ví dụ, hãy cùng xem xét ảnh một con mèo.



**Hình 2.15 Hình ảnh mèo có nhiễu (là lông của mèo)**

Một sợi lông hoặc ria mèo có thể được coi là một cạnh vì đó là một vùng có cường độ thay đổi đột ngột. Tuy nhiên, đó không phải là một cạnh. Nó chỉ là nhiễu. Để giải quyết vấn đề này, bộ lọc làm mịn Gaussian thường được áp dụng cho một hình ảnh để giảm nhiễu trước khi áp dụng Laplacian. Phương pháp này được gọi là Laplacian của Gaussian (LoG) .

Chúng ta cũng có thể đặt giá trị ngưỡng để phân biệt nhiễu với các cạnh. Nếu độ lớn đạo hàm bậc hai tại một điểm ảnh vượt quá ngưỡng này, điểm ảnh đó là một phần của cạnh.

#### 2.2.2.2. Công thức toán học của Laplacian of Gaussian (LoG)

Trên thực tế, với một pixel (x,y), Laplacian L(x,y) của một hình ảnh có giá trị cường độ  $I_i$  có thể được biểu diễn bằng công thức toán học sau:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Giống như trong trường hợp của Toán tử Sobel, chúng ta không thể tính đạo hàm bậc hai trực tiếp vì các pixel trong ảnh là rời rạc. Chúng ta cần phải xấp xỉ nó bằng cách sử dụng toán tử tích chập. Hai bộ lọc kernel phổ biến nhất là:

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

**Hình 2.16 Hai bộ lọc kernel phổ biến trong xử lý ảnh với LoG**

Chỉ tính toán Laplacian sẽ tạo ra rất nhiều nhiễu, vì vậy chúng ta cần tích chập bộ lọc làm mịn Gaussian với bộ lọc Laplacian để giảm nhiễu trước khi tính đạo hàm bậc hai. Phương trình kết hợp cả hai bộ lọc này được gọi là Laplacian của Gaussian và như sau:

$$LoG = -\frac{1}{\pi\sigma^4} \left[ 1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Phương trình trên là liên tục, do đó chúng ta cần phải rời rạc hóa nó để có thể sử dụng trên các điểm ảnh rời rạc trong hình ảnh.

Dưới đây là một ví dụ về bộ lọc kernel xấp xỉ LoG trong đó  $\sigma = 1,4$ . Đây chỉ là một ví dụ về một bộ lọc kernel tích chập có thể được sử dụng. Có những bộ lọc kernel khác cũng có thể hoạt động tốt, thậm chí tốt hơn.

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

**Hình 2.17 Ví dụ bộ lọc kernel trong xử lý ảnh với LoG**

Bộ lọc kernel LoG này được áp dụng chập với hình ảnh đầu vào đen trắng để phát hiện các điểm giao cắt với giá trị bằng không của đạo hàm bậc hai. Chúng ta thiết lập một ngưỡng cho các điểm giao cắt này và chỉ giữ lại những điểm giao cắt vượt qua ngưỡng đó. Các điểm giao cắt mạnh là những điểm có sự chênh lệch lớn giữa giá trị cực đại dương và cực tiểu âm ở hai phía của điểm giao cắt. Các điểm giao cắt yếu thường là nhiều, vì vậy chúng sẽ bị loại bỏ nhờ vào việc áp dụng ngưỡng.



**Hình 2.18 Hình ảnh gốc và sau khi áp dụng LoG**

### 2.2.3. Phương pháp Canny

Phát hiện cạnh Canny (Canny edge detector) là một phương pháp bao gồm nhiều giai đoạn để phát hiện một loạt các cạnh trong hình ảnh. Nó được phát triển bởi John F. Canny vào năm 1986. Canny cũng đưa ra một lý thuyết tính toán về phát hiện cạnh giải thích tại sao kỹ thuật này hoạt động.

Một điều quan trọng, đó là thuật toán dựa trên ảnh xám. Do đó, điều kiện tiên quyết là phải chuyển hình ảnh sang thang độ xám trước khi thực hiện các bước của phương pháp Canny.

Dưới đây là ví dụ về cách thức mà phương pháp Canny hoạt động. Nhưng trước khi thực hiện, tôi sẽ chuyển ảnh gốc sang ảnh thang độ xám như hình bên dưới.



**Hình 2.19 Hình ảnh gốc (trái) – Hình ảnh đã được chuyển sang thang độ xám**

#### Bước 1: Giảm nhiễu

Việc loại bỏ các yếu tố nhiễu là đặc biệt quan trọng đối với kết quả phát hiện cạnh. Một cách để loại bỏ nhiễu là áp dụng Gaussian Filter giúp làm mịn ảnh. Để làm như vậy, kỹ thuật tích chập hình ảnh được áp dụng với Gaussian Kernel (kích thước  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$ , v.v.). Kích thước kernel phụ thuộc vào hiệu ứng làm mờ mong muốn.

Về cơ bản, kernel càng nhỏ, hiệu ứng mờ càng ít. Ví dụ dưới đây sử dụng  $5 \times 5$  Gaussian kernel.

Phương trình cho một bộ lọc Gaussian kernel có kích thước  $(2k + 1) \times (2k + 1)$ :

$$H_{ij} = \frac{1}{2\pi\sigma^2} \left( -\frac{(i - (k + 1))^2 + (j - (k + 1))^2}{2\sigma^2} \right) ; \quad 1 \leq i, j \leq (2k + 1)$$

Sau khi áp dụng hiệu ứng mờ Gaussian, ta nhận được kết quả sau:



**Hình 2.20 Hình ảnh bị làm mờ với bộ lọc Gaussian ( $\sigma = 1.4$  và kích thước kernel là  $5 \times 5$ )**

#### Bước 2: Tính toán Gradient độ xám của ảnh

Bước tính toán Gradient độ xám phát hiện các cạnh thông qua cường độ và hướng của gradient độ xám. Các cạnh tương ứng với sự thay đổi cường độ sáng của pixel. Để phát hiện nó, cách dễ nhất là áp dụng các Filter làm nổi bật sự thay đổi cường độ này theo cả hai hướng: ngang (x) và dọc (y). Sau Khi hình ảnh được làm mịn, các đạo hàm  $I_x$  và  $I_y$  w.r.t. x và y được tính. Việc này có thể được thực hiện bằng cách nhân chập I với các Sobel kernel  $K_x$  và  $K_y$ , tương ứng:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \quad K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

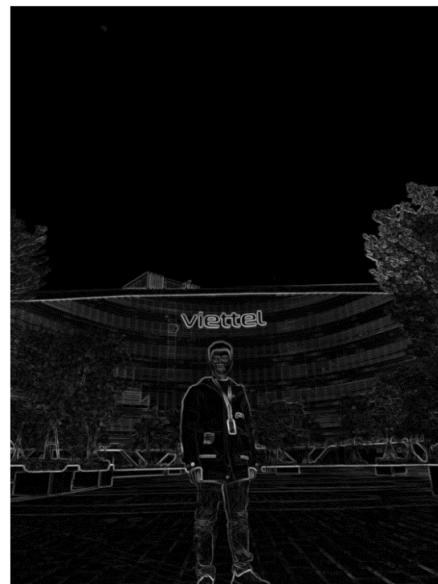
Sobel Filter cho cả hai chiều ngang và dọc.

Sau đó, độ lớn G và góc  $\theta$  của gradient được tính như sau:

$$|G| = \sqrt{I_x^2 + I_y^2}$$

$$\theta(x, y) = \arctan\left(\frac{I_y}{I_x}\right)$$

Dưới đây là hình ảnh kết quả sau khi áp dụng bước 2:



**Hình 2.21 Hình ảnh cường độ gradient**

Kết quả gần như là mong đợi, nhưng chúng ta có thể thấy trong bức ảnh kết quả vẫn còn tồn tại một số cạnh dày và một số cạnh mỏng khác nhau. Áp dụng bước 3, bước non-maximum suppression sẽ giúp ta giảm thiểu các cạnh dày.

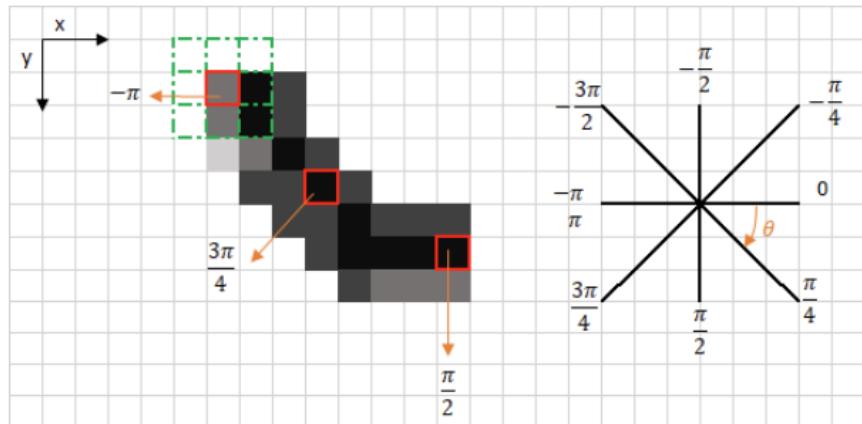
Hơn nữa, mức cường độ gradient nằm trong khoảng từ 0 đến 255 nên độ sáng của các cạnh không đồng nhất. Các cạnh trên kết quả cuối cùng sau khi áp dụng phương pháp Canny phải có cùng cường với độ sáng là 255, hay nói cách khác thì bức ảnh kết quả cuối cùng phải là ảnh nhị phân.

### Bước 3: Áp dụng Non-maximum suppression

Hình ảnh cuối cùng nên có các cạnh mỏng. Vì vậy, ta phải thực hiện thuật toán Non-maximum suppression để làm mỏng chúng.

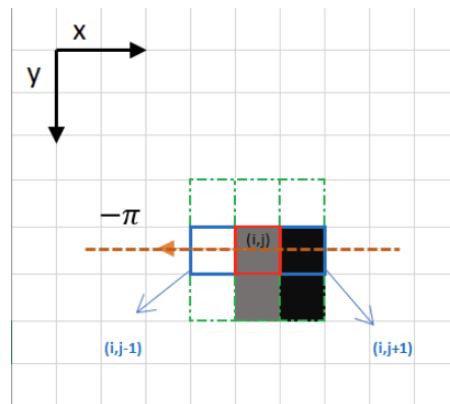
Nguyên tắc rất đơn giản: thuật toán đi qua tất cả các điểm trên ma trận cường độ gradient và tìm các pixel có giá trị lớn nhất theo các hướng cạnh.

Hãy thử một ví dụ dễ hiểu như sau:



**Hình 2.22 Ví dụ duyệt qua cạnh trên ảnh**

Hộp màu đỏ ở góc trên bên trái đại diện cho một pixel cường độ của ma trận Cường độ Gradient đang được xử lý. Hướng gradient tương ứng được biểu diễn bằng mũi tên màu cam với góc  $-\pi$  radian ( $\pm 180$  độ).

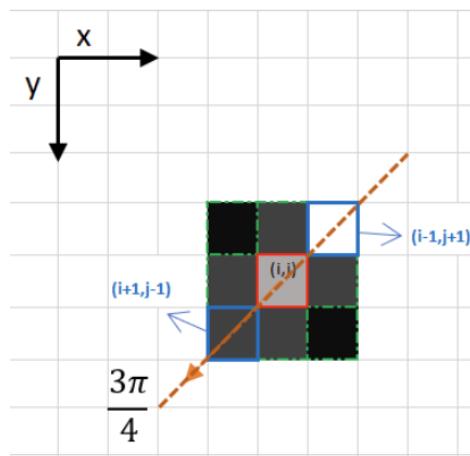


**Hình 2.23 Tập trung vào khu vực pixel trong hộp màu đỏ góc trên bên trái**

Hướng gradient là đường chấm màu cam (nằm ngang từ trái sang phải). Mục đích của thuật toán là để kiểm tra xem các pixel trên cùng một hướng có cường độ cao

hơn hay thấp hơn các pixel đang được xử lý. Trong ví dụ trên, pixel  $(i, j)$  đang được xử lý và các pixel trên cùng một hướng được đánh dấu bằng màu xanh lam  $(i, j-1)$  và  $(i, j+1)$ . Nếu một trong hai pixel đó có cường độ cao hơn pixel đang được xử lý, thì chỉ có một pixel có cường độ cao hơn được giữ lại. Pixel  $(i, j-1)$  có vẻ sáng hơn, vì nó có màu trắng (giá trị 255). Do đó, giá trị cường độ của pixel hiện tại  $(i, j)$  được đặt thành 0. Nếu không có pixel nào ở hướng cạnh có giá trị cường độ cao hơn thì giá trị của pixel hiện tại được giữ nguyên.

Bây giờ chúng ta hãy thử xem một ví dụ khác:



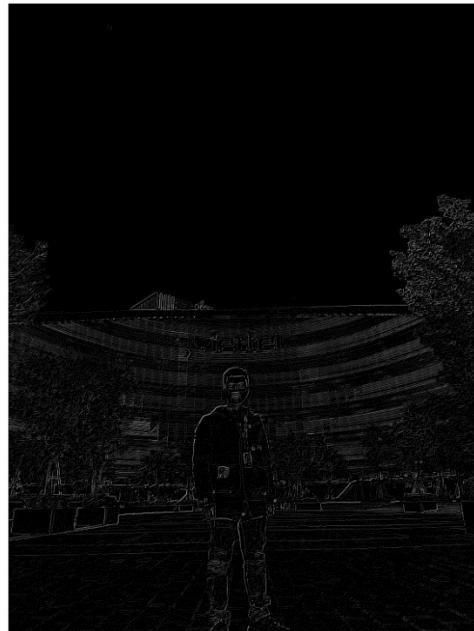
**Hình 2.24 Một ví dụ khác**

Trong trường hợp này, hướng là đường chéo chấm màu cam. Do đó, pixel cường độ cao nhất theo hướng này là pixel  $(i-1, j+1)$ .

Tóm gọn lại, mỗi pixel được đánh giá bằng 2 tiêu chí chính (hướng cạnh tính bằng radian và cường độ pixel (từ 0–255). Dựa trên các đầu vào này, các bước non-maximum suppression là:

- Tạo một ma trận được khởi tạo bằng 0 có cùng kích thước của ma trận cường độ gradient ban đầu.
- Xác định hướng của gradient dựa trên giá trị góc từ ma trận góc.
- Kiểm tra xem pixel ở cùng một hướng có cường độ cao hơn pixel hiện đang được xử lý hay không.
- Trả lại hình ảnh được xử lý bằng thuật toán non-maximum suppression.

Kết quả là cùng một hình ảnh với các cạnh mỏng hơn. Tuy nhiên, ta vẫn có thể nhận thấy sự không đồng đều trong cường độ của các cạnh: một số pixel có vẻ sáng hơn những pixel khác và điều này sẽ được khắc phục bằng hai bước cuối cùng.



**Hình 2.25 Hình ảnh sau khi áp dụng thuật toán non-maximum suppression**

#### Bước 4: Ngưỡng kép (Double Threshold)

Bước này nhằm mục đích xác định 3 loại pixel: mạnh, yếu và ngoại lai:

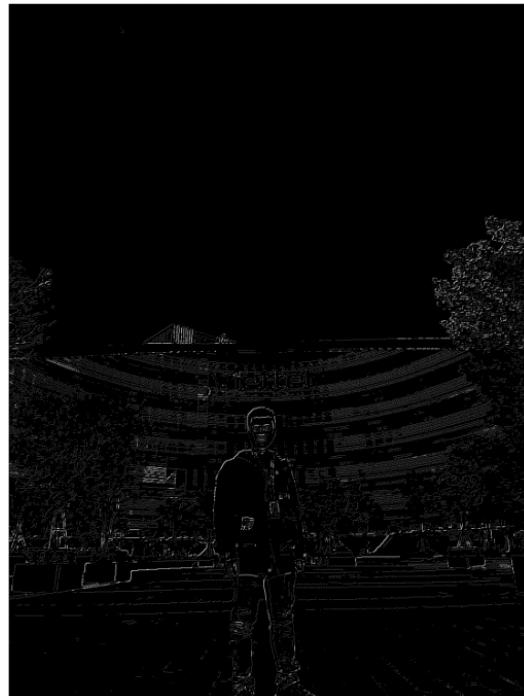
- Pixel mạnh là pixel có cường độ cao và trực tiếp góp phần vào sự hình thành cạnh.
- Pixel yếu là pixel có giá trị cường độ không đủ để được coi là mạnh nhưng chưa đủ nhỏ để được coi là ngoại lai.
- Các pixel khác được coi là ngoại lai.

Bây giờ chúng ta có thể thấy vai trò của ngưỡng kép như sau:

- Ngưỡng cao được sử dụng để xác định các pixel mạnh (cường độ cao hơn ngưỡng cao)
- Ngưỡng thấp được sử dụng để xác định các pixel ngoại lai (cường độ thấp hơn ngưỡng thấp)

- Tất cả các điểm ảnh có cường độ giữa cả hai ngưỡng đều được gắn là yếu. Cơ chế Độ trễ (bước tiếp theo) sẽ giúp ta xác định xem các Pixel yếu sẽ được giữ lại như các pixel mạnh hay bị loại bỏ như pixel ngoại lai.

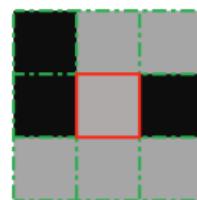
Kết quả của bước này là một hình ảnh chỉ có 2 giá trị cường độ pixel (mạnh và yếu):



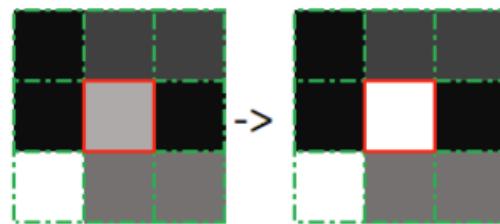
**Hình 2.26 Pixel yếu có màu xám và pixel mạnh có màu trắng**

*Bước 5: Theo dõi cạnh bằng độ trễ (Edge Tracking by Hysteresis)*

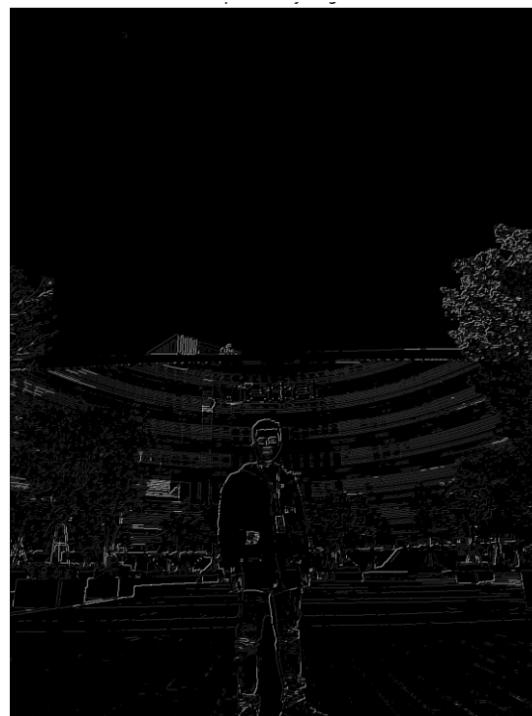
Dựa trên kết quả sau khi áp dụng ngưỡng, Hysteresis Tracking chuyển đổi các pixel yếu thành mạnh, khi và chỉ khi có ít nhất một trong các pixel xung quanh pixel đang xét là pixel mạnh, các pixel yếu còn lại bị loại bỏ.



**Hình 2.27 Không có pixel mạnh xung quanh**



**Hình 2.28** Có pixel mạnh ở cạnh



**Hình 2.29** Hình ảnh kết quả cuối cùng sau khi thực hiện phương pháp Canny

### 2.3. Phương pháp phân vùng ảnh dựa trên HOG (Histogram of Oriented Gradients)

Histogram of Oriented Gradients (HOG) là một kỹ thuật trích xuất đặc trưng từ hình ảnh phổ biến trong lĩnh vực thị giác máy tính và xử lý ảnh. Phương pháp này phân tích sự phân bố của hướng biên trong một đối tượng nhằm mô tả hình dạng và diện mạo của nó. Kỹ thuật HOG bao gồm việc tính toán độ lớn và hướng của gradient tại mỗi pixel trong hình ảnh, sau đó chia hình ảnh thành các ô nhỏ để xử lý.

HOG tập trung vào cấu trúc hoặc hình dạng của một đối tượng. Điều này khác biệt so với các đặc trưng biên (edge features) được trích xuất từ hình ảnh. Với đặc trưng biên, chúng ta chỉ xác định liệu một pixel có phải là biên hay không. Trong khi đó, HOG còn cung cấp thông tin về hướng của biên. Điều này được thực hiện bằng cách trích xuất gradient và hướng (cũng có thể hiểu là độ lớn và phương hướng) của các biên.

Hơn nữa, các hướng này được tính toán trên các vùng cục bộ. Nghĩa là, hình ảnh được chia nhỏ thành các vùng và gradient cũng như hướng được tính toán riêng biệt cho từng vùng. Cuối cùng, HOG tạo ra một histogram riêng biệt cho từng vùng. Các histogram này được xây dựng dựa trên gradient và hướng của các giá trị pixel, từ đó hình thành tên gọi “Histogram of Oriented Gradients”.

### **2.3.1. Các thuật ngữ**

- Feature descriptor: Bộ mô tả đặc trưng, là phép chuyển đổi dữ liệu thành các đặc trưng có thể được sử dụng để hỗ trợ phân loại hoặc nhận diện vật thể. Các phương pháp tiêu biểu bao gồm HOG, SURF, và SIFT.
- Histogram: Biểu đồ phân phối, thể hiện sự phân bố của các giá trị cường độ màu sắc trong một khoảng giá trị xác định.
- Gradient: Đạo hàm của véc tơ cường độ màu sắc, được sử dụng để phát hiện hướng thay đổi cường độ và định hướng di chuyển của các đối tượng trong ảnh.
- Local cell: Ô cục bộ. Trong thuật toán HOG, hình ảnh được chia thành các ô vuông nhỏ (cells) trên một lưới. Mỗi ô được gọi là một ô cục bộ, là đơn vị cơ bản để tính toán histogram.
- Local portion: Vùng cục bộ, là một khu vực được trích xuất từ các ô vuông trong ảnh. Trong ngữ cảnh thuật toán HOG, vùng cục bộ thường được gọi là block, bao gồm nhiều ô cục bộ liền kề.

- Local normalization: Phép chuẩn hóa được áp dụng trên một vùng cục bộ, thường sử dụng các norm như norm bậc 1 hoặc norm bậc 2. Quá trình này giúp đồng nhất hóa giá trị cường độ màu sắc, đưa chúng về cùng một phân phối. Điều này sẽ được giải thích chi tiết trong phần trình bày thuật toán.
- Gradient direction: Phương gradient, là góc của véc tơ gradient trong không gian 2 chiều, xác định hướng thay đổi cường độ màu sắc. Để tính phương gradient, giả sử  $G_x$  và  $G_y$  lần lượt là các giá trị gradient theo phương x và phương y của ảnh. Khi đó, phương gradient được tính bằng công thức:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

- Gradient magnitude: Độ lớn gradient, là chiều dài của véc tơ gradient trong không gian hai chiều, bao gồm các thành phần gradient theo phương x và phương y. Khi biểu diễn phân phối histogram của véc tơ này dựa trên phương gradient, ta có thể thu được véc tơ mô tả đặc trưng HOG. Độ lớn gradient được tính theo công thức sau:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

### 2.3.2. Tính toán gradient

Trong hầu hết các thuật toán xử lý ảnh, bước đầu tiên thường là tiền xử lý dữ liệu ảnh (pre-processing image), bao gồm các công việc như chuẩn hóa màu sắc và giá trị gamma. Tuy nhiên, trong quá trình tính toán bộ mô tả HOG, bước tiền xử lý này có thể được bỏ qua, vì việc chuẩn hóa bộ mô tả ở các bước sau sẽ đạt được kết quả tương tự. Thay vào đó, tại bước đầu tiên, ta sẽ tiến hành tính toán các giá trị gradient của hình ảnh.

Một trong những phương pháp phổ biến để tính gradient là sử dụng mặt nạ đạo hàm rời rạc (discrete derivative mask) cho cả hai chiều ngang và dọc. Cụ thể, phương pháp này áp dụng các bộ lọc như Sobel hoặc Scharr lên ma trận cường độ của hình ảnh.

Để tính bộ lọc Sobel, phép tích chập giữa kernel có kích thước nhất định và hình ảnh gốc sẽ được thực hiện. Giả sử  $I$  là ma trận ảnh gốc và  $G_x, G_y$  lần lượt là hai ma trận đại diện cho đạo hàm của ảnh theo trục  $x$  và trục  $y$ . Ta có thể tính toán các kernel tương ứng cho mỗi trục như sau:

Theo chiều ngang:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} * I$$

Theo chiều dọc:

$$G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} * I$$

Kí hiệu  $*$  tương tự như phép tích chập giữa bộ lọc bên trái và ảnh đầu vào bên phải.

Gía trị độ lớn gradient (gradient magnitude) và phương gradient (gradient direction) có thể được tạo ra từ 2 đạo hàm  $G_x$  và  $G_y$  theo công thức bên dưới:

- Độ lớn gradient:

$$G = \sqrt{G_x^2 + G_y^2}$$

- Độ lớn gradient:

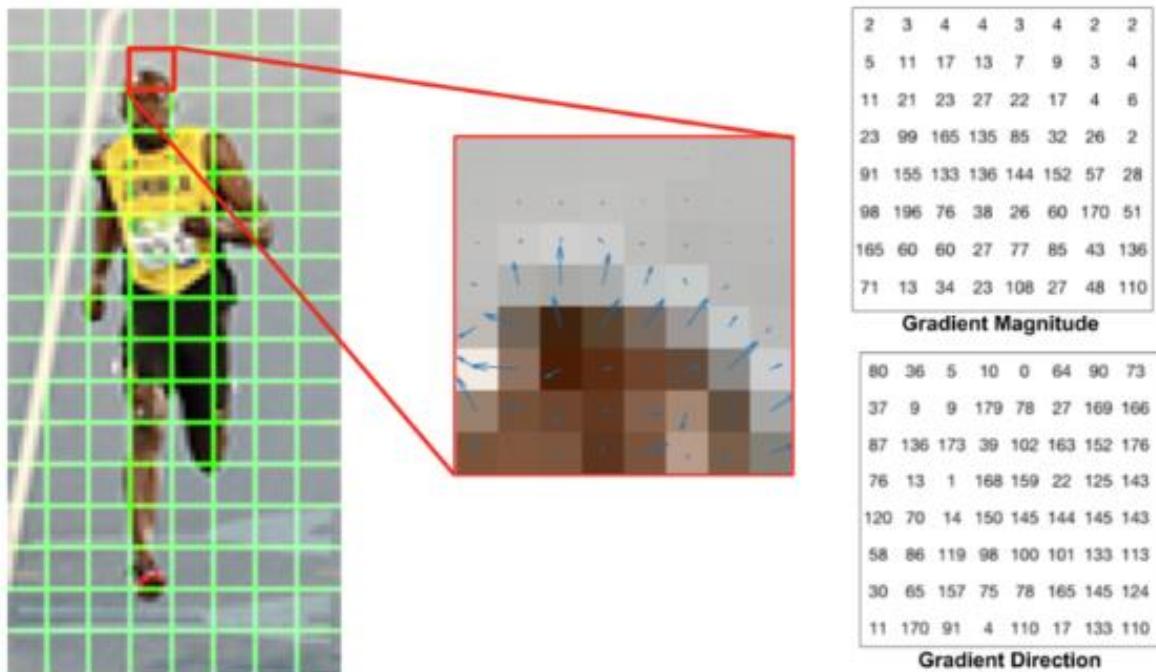
$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

### 2.3.3. Các bước tính HOG

Đặc trưng của mỗi bức ảnh có thể được biểu diễn thông qua hai thông số chính: mức độ thay đổi cường độ màu sắc (ma trận gradient magnitude) và hướng thay đổi cường độ màu sắc (ma trận gradient direction). Do đó, để tạo ra một bộ mô tả (feature descriptor) cho hình ảnh, chúng ta cần chuyển đổi bức ảnh thành một véc tơ sao cho có thể thể hiện đầy đủ cả hai thông tin này.

Để thực hiện điều này, hình ảnh được chia thành một lưới các ô vuông, mỗi ô có kích thước 8x8 pixels, tạo thành tổng cộng 64 ô pixels. Mỗi ô trong số 64 ô này sẽ

được tính toán hai tham số: độ lớn gradient (gradient magnitude) và phương gradient (gradient direction). Tổng cộng, chúng ta cần tính 128 giá trị, bao gồm 64 giá trị gradient magnitude và 64 giá trị gradient direction, như minh họa trong ma trận hình dưới đây:



**Hình 2.30 Hình ảnh vận động viên được chia thành các lưới ô vuông**

Tiếp theo, véc tơ histogram được xây dựng theo các bước sau:

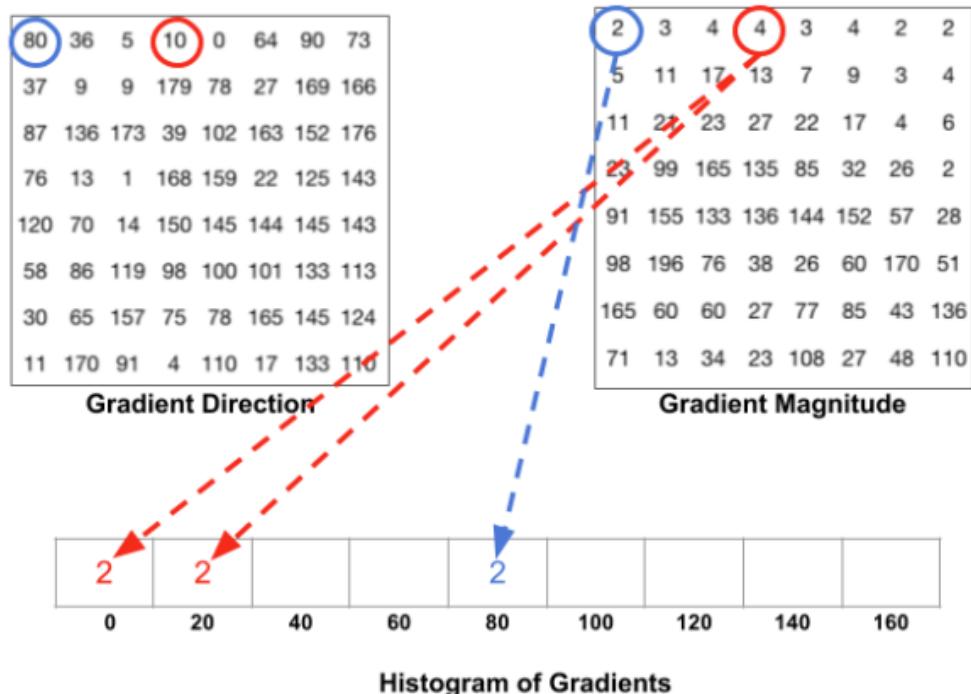
*Bước 1: Mapping độ lớn gradient vào các bins tương ứng của phương gradient.*

Các giá trị phương gradient sẽ được sắp xếp theo thứ tự từ nhỏ đến lớn và chia thành 9 bins, với mỗi bin có độ dài 20, do phương gradient có giá trị trong khoảng [0, 180].

Mỗi phương gradient sẽ được ghép cặp với độ lớn gradient ở cùng vị trí tọa độ.

Khi xác định được phương gradient thuộc bin nào trong véc tơ bins, ta sẽ điền giá trị độ lớn gradient vào bin tương ứng. Ví dụ, trong hình dưới đây, ô được khoanh tròn với phương gradient là 80 và độ lớn gradient là 2. Khi đó, trong véc tơ bins của HOG,

phuong gradient 80 sẽ rơi vào vị trí thứ 5, và ta sẽ điền giá trị 2 (độ lớn gradient) vào bin này.



**Hình 2.31 Mapping độ lớn gradients với các bins.**

Các giá trị đầu mứt của mỗi bin là các giá trị chia hết cho độ rộng của bin (chẳng hạn như 0, 20, 40,... là các giá trị đầu mứt của bin). Trong trường hợp độ lớn phuong gradient không rơi vào các đầu mứt, phuong pháp nội suy tuyén tính (linear interpolation) sẽ được áp dụng để phân chia độ lớn gradient vào hai bin liền kề mà giá trị phuong gradient rơi vào. Cụ thể, giả sử giá trị phuong gradient là  $x$  và độ lớn gradient là  $y$ , nếu phuong gradient rơi vào khoảng giữa bin thứ  $l - 1$  và bin thứ  $l$ , công thức nội suy tuyén tính được áp dụng như sau:

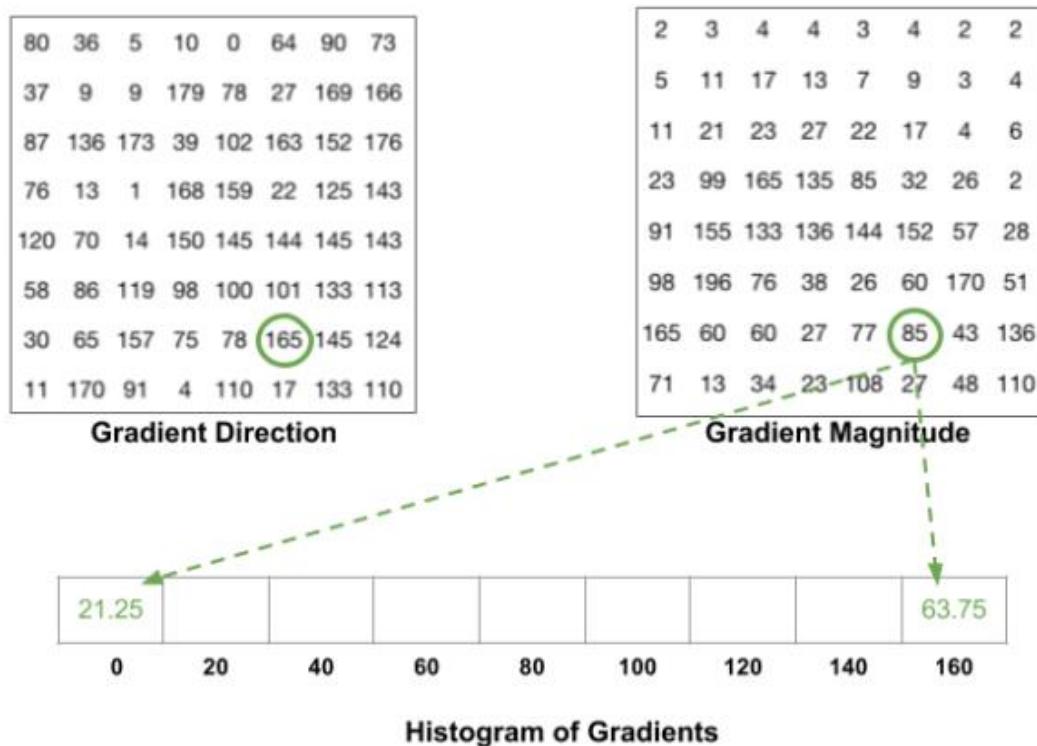
Giá trị tại bin  $l - 1$ :

$$x_{l-1} = \frac{(x_1 - x)}{(x_1 - x_0)} \cdot y$$

Giá trị tại bin  $l$ :

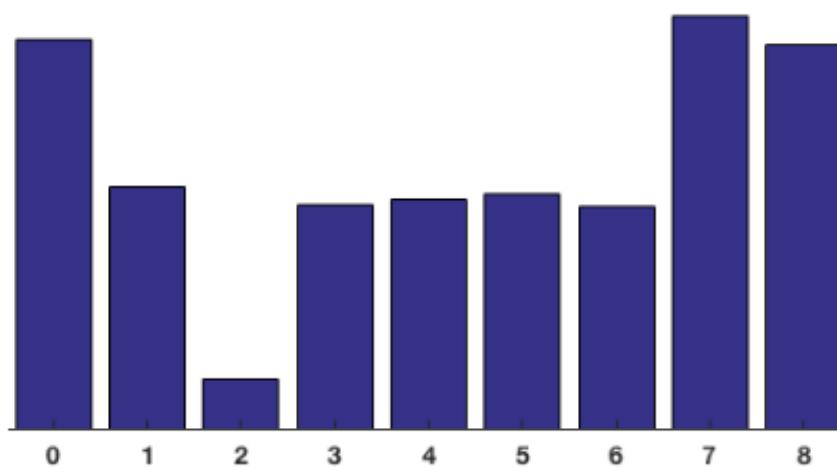
$$x_l = \frac{(x - x_0)}{(x_1 - x_0)} \cdot y$$

Công thức này cho phép phân bổ giá trị cường độ gradient vào các bin liền kề sao cho tổng giá trị độ lớn gradient được duy trì.



**Hình 2.32 Ví dụ chia bins**

Tính tổng tất cả các độ lớn gradient thuộc cùng 1 bins của véc tơ bins ta thu được biểu đồ Histogram of Gradients như bên dưới:



**Hình 2.33 Biểu đồ Histogram of Gradient gồm 9 bins tương ứng với một ô vuông trong lưới ô vuông.**

### Bước 2: Chuẩn hóa véc tơ histogram theo block 16x16

Vector histogram của một bức ảnh sẽ phụ thuộc vào cường độ các pixel trong ảnh. Đối với hai bức ảnh có nội dung tương tự nhưng một bức ảnh bị biến thể tối hơn (được tạo ra từ ma trận ảnh gốc nhân với  $1/2$ ), giá trị véc tơ histogram của ảnh gốc sẽ gấp đôi so với véc tơ histogram của ảnh biến thể. Do đó, cần thực hiện chuẩn hóa véc tơ histogram để đảm bảo rằng cả hai bức ảnh có cùng một véc tơ biểu diễn.

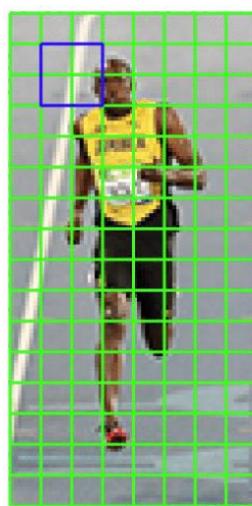
Quá trình chuẩn hóa có thể sử dụng chuẩn norm bậc 2:

$$\text{normalize}(h) = \frac{h}{\|h\|_2}$$

Trong đó,  $h$  là vector histogram của các gradient.

Ngoài ra, chuẩn norm bậc 1 cũng có thể được áp dụng.

Quá trình chuẩn hóa được thực hiện trên một block có kích thước  $2x2$  trong lưới ô vuông ban đầu (mỗi ô có kích thước  $8x8$  pixel). Vì vậy, ta sẽ có tổng cộng 4 véc tơ histogram có kích thước  $1x9$ . Sau khi kết hợp (concatenate) các véc tơ này, ta sẽ thu được một véc tơ histogram tổng hợp với kích thước  $1x36$  và tiến hành chuẩn hóa theo chuẩn norm bậc 2 trên véc tơ này. Quá trình di chuyển các cửa sổ (window) thực hiện tương tự như phép tích chập 2 chiều trong mạng CNN với bước nhảy (step size) là 8 pixel, như minh họa trong hình dưới đây:



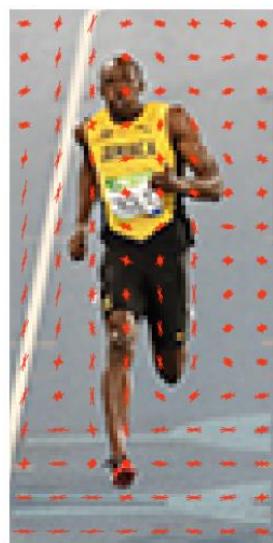
**Hình 2.34 Hình ảnh được phân chia thành lưới các ô vuông con, mỗi ô kích thước  $8x8$  pixel.**

### Bước 3: Tính toán véc tơ đặc trưng HOG.

Sau khi chuẩn hóa các véc tơ histogram, các véc tơ  $1 \times 36$  sẽ được ghép nối lại với nhau (concatenate) để tạo thành một véc tơ lớn. Đây chính là véc tơ HOG đại diện cho toàn bộ hình ảnh.

Ví dụ: Hình ảnh được chia thành lưới ô vuông kích thước  $16 \times 8$  (mỗi ô  $8 \times 8$ ). Quá trình tính toán HOG sẽ di chuyển 7 lượt theo chiều rộng và 15 lượt theo chiều cao. Do đó, tổng số patches là  $7 \times 15 = 105$ , mỗi patch sẽ tương ứng với một véc tơ histogram có kích thước 36 chiều. Vì vậy, véc tơ HOG cuối cùng sẽ có kích thước  $105 \times 36 = 3780$  chiều. Đây là một véc tơ có kích thước tương đối lớn và có thể mô phỏng được đặc trưng của ảnh một cách hiệu quả.

Đối với mỗi ô trên lưới ô vuông, phân phối HOG sẽ được biểu diễn thông qua nhóm 9 véc tơ với độ dài bằng độ lớn gradient và góc bằng phương gradient. Các véc tơ này có hình dáng tương đối giống với dáng của một vận động viên trong ảnh, đặc biệt tại các vị trí chân và tay. Cụ thể, hãy xem hình minh họa dưới đây.



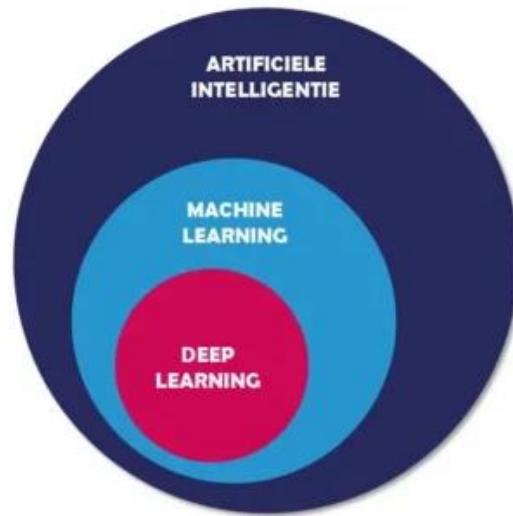
**Hình 2.35 Biểu diễn nhóm véc tơ histogram trên các lưới ô vuông của hình ảnh gốc. Các phương véc tơ phổ biến là chiều đọc trùng với chiều bức ảnh.**

Điều này chứng tỏ bộ mô tả HOG đã mã hóa được các đặc trưng của một bức ảnh khá tốt.

## 2.4. Phương pháp phân vùng ảnh dựa trên mạng nơ-ron tích chập (CNN)

### 2.4.1. Tổng quan về mạng nơ-ron tích chập (CNN)

Trong những năm gần đây, Trí tuệ nhân tạo (Artificial Intelligence - AI) đã trở thành một lĩnh vực nghiên cứu quan trọng với nhiều ứng dụng sâu rộng trong đời sống và công nghiệp. AI là tập hợp các kỹ thuật và công nghệ giúp máy tính có khả năng thực hiện các tác vụ phức tạp mà trước đây chỉ có con người mới làm được, bao gồm phân tích dữ liệu, nhận diện giọng nói và hình ảnh. Một trong những nhánh phát triển mạnh mẽ của AI là học máy (Machine Learning - ML), phương pháp cho phép các hệ thống học hỏi từ dữ liệu đầu vào mà không cần lập trình tất cả các quy tắc. Thông qua ML, các mô hình máy tính có thể phát triển khả năng dự đoán và đưa ra quyết định dựa trên dữ liệu đã học, mang lại độ chính xác cao trong các ứng dụng thực tế.



**Hình 2.36 Mối quan hệ giữa AI, ML và DL**

Học sâu (Deep Learning - DL), một nhánh chuyên sâu của Machine Learning, đặc biệt được quan tâm vì khả năng xử lý lượng dữ liệu lớn và học hỏi từ những môi liên hệ phức tạp. Deep Learning được xây dựng trên nền tảng mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN), một mô hình được lấy cảm hứng từ hệ thần kinh của con người. ANN bao gồm nhiều lớp nơ-ron liên kết với nhau, mỗi nơ-ron thực

hiện các phép tính toán học và truyền tín hiệu tới các nơ-ron khác. Cấu trúc nhiều lớp của ANN giúp mô hình có khả năng tự học hỏi và nhận diện các đặc trưng ẩn sâu trong dữ liệu, từ đó đưa ra kết quả chính xác và phức tạp hơn.

Một trong những ứng dụng nổi bật nhất của mạng nơ-ron nhân tạo là mạng nơ-ron tích chập (Convolutional Neural Network - CNN), được xem như một công cụ hiệu quả trong xử lý và phân vùng ảnh. Với các lớp tích chập (convolutional layers), CNN có khả năng tự động phát hiện và trích xuất các đặc trưng từ hình ảnh, từ các chi tiết cơ bản như cạnh và góc, đến các cấu trúc phức tạp hơn. Khả năng này giúp CNN không chỉ nhận diện đối tượng mà còn phân vùng các vùng ảnh theo cách chính xác và rõ ràng, phục vụ cho các ứng dụng đòi hỏi độ chi tiết cao như y học, giám sát an ninh và xe tự lái. CNN đã chứng minh tính ưu việt của mình trong việc phân tích hình ảnh một cách toàn diện, từ việc xác định các vùng cần chú ý đến phân tách các đối tượng trong khung cảnh, đóng vai trò quan trọng trong nhiều lĩnh vực khoa học và công nghiệp.

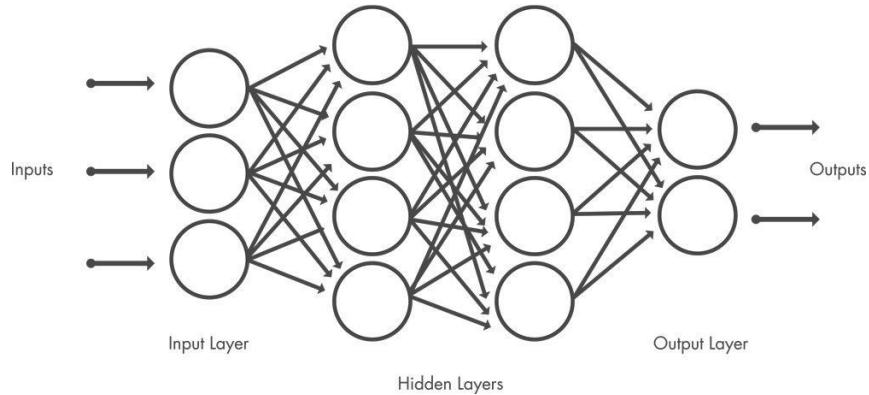
Trước CNN, các phương pháp trích xuất tính năng thủ công, tốn nhiều thời gian đã được sử dụng để xác định các đối tượng trong hình ảnh. Tuy nhiên, mạng nơ-ron tích chập hiện cung cấp một phương pháp tiếp cận có khả năng mở rộng hơn cho các tác vụ phân loại hình ảnh và nhận dạng đối tượng, tận dụng các nguyên tắc từ đại số tuyến tính, cụ thể là phép nhân ma trận, để xác định các mẫu trong hình ảnh. Tuy nhiên, chúng có thể đòi hỏi nhiều tính toán, yêu cầu các đơn vị xử lý đồ họa (GPU) để đào tạo các mô hình.

#### **2.4.2. Giới thiệu về mạng nơ-ron tích chập (CNN)**

Mạng nơ-ron tích chập thuật ngữ tiếng Anh là Convolutional Neural Network - hay còn được biết đến với cái tên là CNN, đóng vai trò quan trọng và trở thành một công cụ hiệu quả trong việc xử lý dữ liệu dạng hình ảnh và tín hiệu. CNN được thiết kế đặc biệt để tận dụng cấu trúc không gian của dữ liệu hình ảnh, giúp tối ưu hóa quá

trình trích xuất và học các đặc trưng quan trọng mà không yêu cầu các phương pháp tiền xử lý phức tạp.

CNN khác biệt với các mô hình mạng nơ-ron truyền thống ở chỗ sử dụng các phép tích chập, một phép toán quan trọng giúp phát hiện và trích xuất đặc trưng từ hình ảnh đầu vào. Thay vì kết nối toàn bộ nơ-ron giữa các lớp, CNN tận dụng các lớp tích chập (convolutional layers) để tìm kiếm và lưu trữ các mẫu đặc trưng cục bộ, như các cạnh, góc, và hình dạng, thông qua một tập hợp các bộ lọc (filters) hoặc nhân tích chập (kernels). Điều này không chỉ giúp giảm đáng kể số lượng tham số, từ đó tăng tính hiệu quả và giảm yêu cầu tính toán, mà còn giúp mô hình có khả năng tự động học các đặc trưng cấp cao từ dữ liệu hình ảnh khi số lượng lớp tăng lên.



**Hình 2.37 Sơ đồ tổng quát mạng nơ-ron tích chập**

Một CNN thường bao gồm các thành phần chính như lớp đầu vào (input layer), lớp đầu ra (output layer), lớp tích chập (convolutional layers), lớp gộp (pooling layers), lớp kích hoạt (activation layers), và lớp hoàn toàn kết nối (fully connected layers). Lớp tích chập hoạt động như một máy trích xuất đặc trưng, còn lớp gộp giúp giảm kích thước của đặc trưng, giảm số lượng tính toán và tránh hiện tượng quá khớp (overfitting). Cuối cùng, các lớp hoàn toàn kết nối đóng vai trò tổng hợp và dự đoán đầu ra từ các đặc trưng đã học. Các mô hình nổi bật dựa trên kiến trúc CNN đã đạt được nhiều thành công trong các bài toán thị giác máy tính và có ảnh hưởng lớn trong lĩnh vực học sâu có thể kể đến là: LeNet (1998), AlexNet (2012), VGGNet (2014),

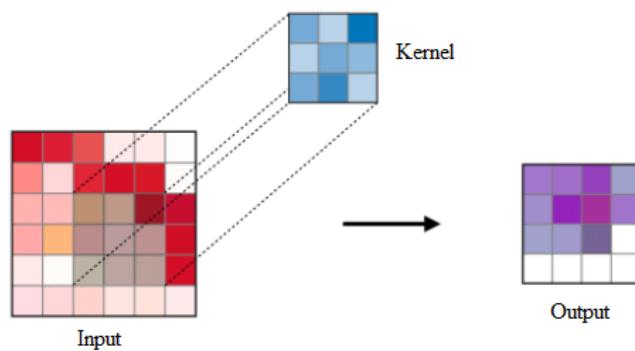
GoogLeNet/Inception (2014), ResNet (2015), U-Net (2015), DenseNet (2017), MobileNet (2017), Stardist (2018), EfficientNet (2019).

#### **2.4.3. Cách hoạt động của mạng nơ-ron tích chập (CNN)**

CNN hoạt động dựa trên việc áp dụng các bộ lọc để trích xuất các đặc trưng của vật thể trong ảnh, học cách kết hợp các đặc trưng này thông qua hàm kích hoạt và các lớp kết nối đầy đủ. Nó được tối ưu hóa qua quá trình huấn luyện để có thể đạt được độ chính xác cao nhất trong nhiệm vụ cụ thể. Như ở phần trước chúng ta đã biết rằng CNN bao gồm một lớp đầu vào, một lớp đầu ra và nhiều lớp ẩn ở giữa. Các lớp này thực hiện các hoạt động thay đổi dữ liệu với mục đích học các tính năng cụ thể của dữ liệu. Bốn lớp phổ biến nhất là lớp tích chập (Convolutional Layer), lớp kích hoạt hoặc lớp ReLU (Activation Layer), lớp gộp (Pooling Layer) và lớp kết nối đầy đủ (Fully Connected Layer).

##### **2.4.3.1. Lớp tích chập (Convolutional Layer)**

Lớp tích chập là thành phần cốt lõi của CNN, đóng vai trò trích xuất các đặc trưng từ dữ liệu đầu vào. Trong lớp tích chập, một tập hợp các bộ lọc (filters) hoặc nhân tích chập (kernels) sẽ di chuyển (quét) qua hình ảnh đầu vào, thực hiện phép tích chập trên từng vùng nhỏ (patch) của hình ảnh.



**Hình 2.38 Mô tả phương pháp tích chập**

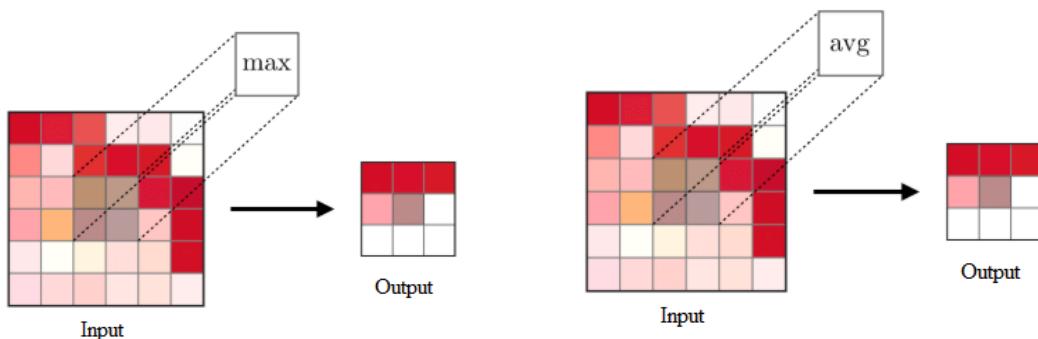
Mỗi bộ lọc có kích thước nhỏ hơn so với hình ảnh đầu vào, chẳng hạn như 3x3 hoặc 5x5. Khi bộ lọc di chuyển qua từng vùng của ảnh, nó thực hiện phép nhân điểm

giữa giá trị của bộ lọc và vùng ảnh, rồi cộng tất cả các giá trị kết quả để tạo ra một giá trị duy nhất, gọi là đầu ra tích chập. Quá trình này tạo ra một bản đồ đặc trưng (feature map), biểu diễn sự hiện diện của các đặc trưng cụ thể trong hình ảnh đầu vào.

#### 2.4.3.2 Lớp gộp (Pooling Layer)

Lớp pooling là một phép downsampling, thường được sử dụng sau tầng tích chập, giúp giảm kích thước của bản đồ đặc trưng, giảm số lượng tham số và độ phức tạp tính toán, đồng thời giảm hiện tượng quá khớp (overfitting). Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

Max pooling sẽ sử dụng 1 kernel duyệt qua tất cả các vùng trên ảnh, và ở mỗi vùng sẽ chọn ra giá trị của điểm ảnh lớn nhất lấy làm kết quả cho điểm ảnh của giá trị đầu ra (output). Sử dụng phép max pooling cho phép làm nổi bật các đặc trưng quan trọng.



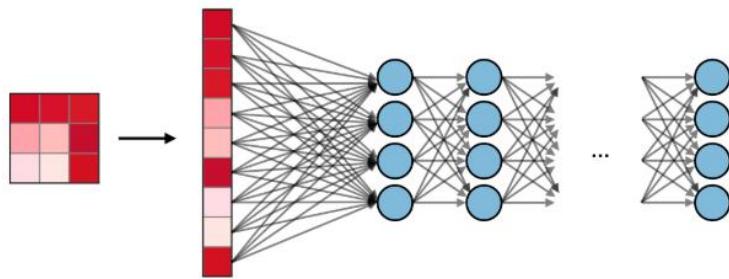
Hình 2.39 Phương pháp max pooling

Hình 2.40 Phương pháp average pooling

Average pooling sẽ sử dụng 1 kernel duyệt qua tất cả các vùng trên ảnh, và ở mỗi vùng sẽ tính trung bình giá trị của điểm ảnh và lấy làm kết quả cho điểm ảnh của giá trị đầu ra (output). Sử dụng phép average pooling cho phép bảo toàn thông tin tổng thể.

#### 2.4.3.3. Lớp kết nối đầy đủ (Fully Connected Layer)

Sau các lớp tích chập và gộp, các bản đồ đặc trưng sẽ được chuyển thành một vector phẳng và đi qua các lớp kết nối đầy đủ, giống như trong mạng nơ-ron truyền thống. Tại đây, mạng sẽ học cách kết hợp các đặc trưng đã được trích xuất để dự đoán đầu ra cuối cùng, chẳng hạn như phân loại hình ảnh. Mỗi nơ-ron trong lớp kết nối đầy đủ được kết nối với tất cả các nơ-ron trong lớp trước đó, giúp mạng nơ-ron tổng hợp thông tin từ tất cả các đặc trưng đã học. Trong mô hình mạng CNN, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



**Hình 2.41 Mô tả tầng kết nối đầy đủ**

#### 2.4.3.4 Hàm kích hoạt (Activation)

Hàm kích hoạt là một chức năng được áp dụng sau mỗi phép toán trong các lớp của mạng, bao gồm các lớp Convolutional, Fully Connected, hoặc bất kỳ lớp nào có tính toán đầu ra. Mục tiêu của hàm kích hoạt là tạo ra tính phi tuyến cho mạng, giúp mạng học được các mối quan hệ phức tạp trong dữ liệu. Trong CNNs, hàm activation thường được áp dụng sau mỗi lớp convolutional hoặc fully connected để chuyển đổi đầu ra của lớp đó, tạo ra sự phi tuyến và cải thiện khả năng học của mạng. Các hàm kích hoạt phổ biến như là:

- **Hàm Sigmoid**

Công thức:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

**Đặc điểm:** Đầu ra có giá trị trong khoảng (0, 1). Thường được dùng trong các bài toán phân loại.

Ưu điểm: Đơn giản, dễ hiểu. Biến đầu ra thành giá trị xác suất phù hợp để dự đoán.

Nhược điểm: Tính toán tốn thời gian hơn so với các hàm khác do cần tính lũy thừa.

Và có vấn đề "vanishing gradient" khi giá trị đầu vào quá lớn hoặc quá nhỏ.

- **Hàm Tanh (Hyperbolic Tangent)**

Công thức:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Đặc điểm: Biến đầu vào thành giá trị trong khoảng  $(0, 1)$ . Trung hòa hơn Sigmoid vì đầu ra có giá trị âm và dương.

Ưu điểm: Tốt hơn Sigmoid trong việc xử lý dữ liệu, vì giá trị trung bình gần 0.

Nhược điểm: Cũng gặp vấn đề "vanishing gradient" khi đầu vào lớn hoặc nhỏ.

- **Hàm ReLU (Rectified Linear Unit)**

Công thức:

$$ReLU(x) = \max(0, x)$$

Đặc điểm: Đầu ra là 0 nếu đầu vào âm; bằng chính đầu vào nếu đầu vào dương.

Ưu điểm: Đơn giản, hiệu quả. Giải quyết được vấn đề vanishing gradient trong phần lớn trường hợp.

Nhược điểm: Có vấn đề Dying ReLU, khi tất cả đầu vào âm, gradient trở thành 0, neuron "chết" và không được cập nhật.

- **Hàm Leaky ReLU**

Công thức:

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{nếu } x > 0 \\ \alpha x & \text{nếu } x \leq 0 \end{cases}$$

Trong đó  $\alpha$  là một hằng số nhỏ (thường là 0.01).

Đặc điểm: Là biến thể của ReLU, cho phép gradient khác 0 ở vùng âm.

Ưu điểm: Giải quyết được vấn đề Dying ReLU.

Nhược điểm: Giá trị của  $\alpha$  cần được lựa chọn phù hợp để đảm bảo hiệu quả.

- **Hàm Softmax**

Công thức:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Đặc điểm: Thường được sử dụng ở lớp cuối cùng trong các bài toán phân loại đa lớp. Chuyển đổi các đầu vào thành xác suất (tổng xác suất của tất cả các lớp bằng 1).

Ưu điểm: Biểu diễn trực quan, giá trị đầu ra có thể hiểu là xác suất.

Nhược điểm: Tốn thời gian tính toán. Có thể bị ảnh hưởng bởi vấn đề "exploding gradient" khi giá trị đầu vào quá lớn.

#### **2.4.4. Cách huấn luyện mạng nơ-ron tích chập (CNN)**

##### **2.4.4.1. Trọng số và hằng số hiệu chỉnh**

Mạng CNN bao gồm các lớp nơ-ron được tổ chức thành nhiều tầng khác nhau. Dữ liệu hình ảnh đầu vào được cung cấp cho mạng và lần lượt truyền qua các lớp để xử lý. Bước đầu tiên trong mạng CNN là phát hiện và phân tích các đặc trưng và cấu trúc độc nhất của các đối tượng cần phân loại. Quá trình này được thực hiện thông qua các bước tích chập. Khi một mạng CNN mới được thiết kế, các ma trận lọc này ban đầu chưa được xác định, dẫn đến việc mạng chưa có khả năng phát hiện mẫu hoặc nhận diện các đối tượng. Để đạt được mục tiêu này, chúng ta cần phải xác định các tham số và phần tử trong ma trận lọc nhằm tối ưu hóa độ chính xác của việc phát hiện đối tượng hoặc giảm thiểu hàm mất mát (loss function). Đây chính là quá trình huấn luyện mạng neuron. Trong hầu hết ứng dụng phổ biến, mạng được huấn luyện một lần trong giai đoạn phát triển và kiểm thử. Sau đó, mạng sẽ sẵn sàng để sử dụng mà không cần điều chỉnh thêm các tham số, ở đây là các trọng số weights và độ lệch bias, miễn là nhiệm vụ phân loại các đối tượng quen thuộc. Tuy nhiên, khi mạng cần phân loại các đối tượng hoàn toàn mới, quá trình huấn luyện bổ sung sẽ trở nên cần thiết. Công thức tính giá trị tổng quát của một nơ-ron có thể được biểu diễn là:

$$y = w * x + b$$

Trong đó:  $y$  là giá trị đầu ra sau khi qua nơ-ron.

$w$  (weights) là trọng số, đại diện cho mức độ quan trọng của từng đặc trưng đầu vào  $x$ .

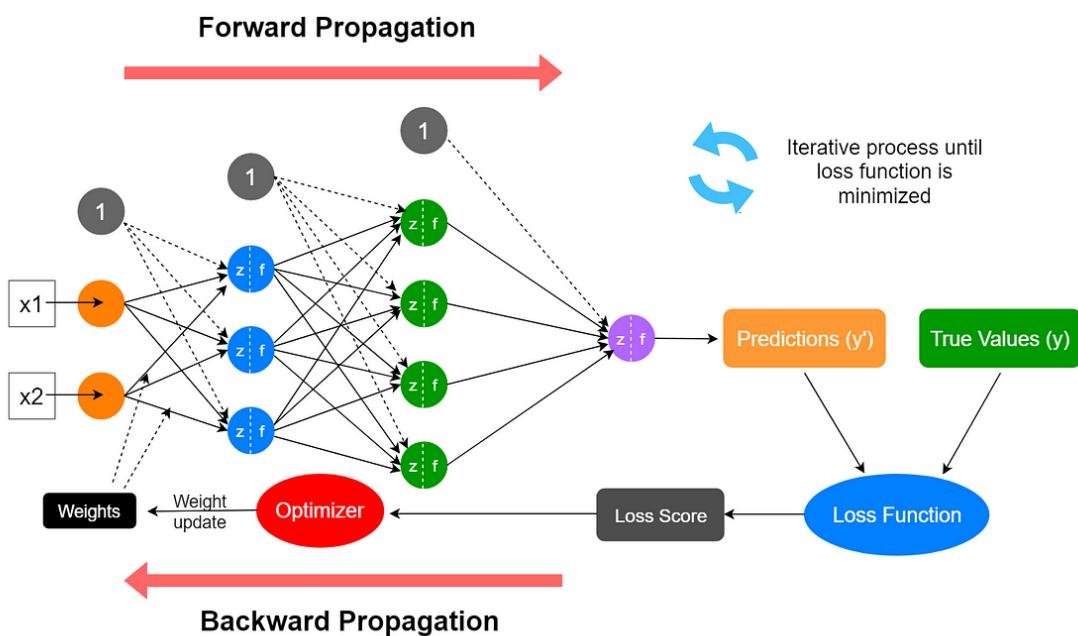
$x$  là đầu vào của nơ-ron.

$b$  (bias) là hằng số hiệu chỉnh – hay độ lệch, cho phép mô hình dịch chuyển hàm kích hoạt để phù hợp hơn với dữ liệu.

#### 2.4.4.2. Lan truyền xuôi và lan truyền ngược

Trong huấn luyện CNN, hai quá trình lan truyền xuôi (forward propagation) và lan truyền ngược (backward propagation) đóng vai trò quan trọng. Chúng hoạt động phối hợp, đảm bảo rằng mạng không chỉ dự đoán đầu ra từ dữ liệu đầu vào mà còn tự điều chỉnh để cải thiện chất lượng dự đoán qua từng chu kỳ. Trong quá trình này, mạng học cách tối ưu hóa các tham số (trọng số weight và bias) để đạt được hiệu suất cao nhất.

Lan truyền xuôi là giai đoạn đầu tiên, nơi mà dữ liệu đầu vào, chẳng hạn như một bức ảnh, được truyền qua các tầng của mạng. Tại đây, các phép tính như tích chập và hàm kích hoạt được thực hiện để trích xuất đặc trưng từ ảnh. Ví dụ, với một ảnh đầu vào, các tầng tích chập sử dụng các bộ lọc (kernel) để phát hiện các đặc trưng quan trọng như đường biên, góc hoặc họa tiết. Các kết quả này được lưu trữ dưới dạng bản đồ đặc trưng (feature map), thể hiện các thông tin cần thiết để nhận diện đối tượng. Sau đó, các tầng pooling được áp dụng để giảm kích thước không gian, giúp làm nổi bật các đặc trưng quan trọng và giảm độ phức tạp tính toán. Quá trình này tiếp tục qua nhiều tầng trong mạng, với kết quả cuối cùng là một dự đoán đầu ra.



**Hình 2.42 Quy trình huấn luyện mạng CNN, lan truyền xuôi và lan truyền ngược**

Tuy nhiên, dự đoán ban đầu từ lan truyền xuôi không phải lúc nào cũng chính xác. Đây là lúc lan truyền ngược được thực hiện để cải thiện chất lượng của mạng. Sai số giữa đầu ra dự đoán và giá trị thực tế được tính toán bằng một hàm mất mát, như Cross-entropy loss hoặc Mean squared error. Sai số này phản ánh mức độ khác biệt giữa kết quả mạng đưa ra và kết quả mong muốn. Lan truyền ngược sau đó sử dụng sai số này để điều chỉnh trọng số và bias của mạng.

Quá trình lan truyền ngược diễn ra theo hướng ngược lại so với lan truyền xuôi, từ tầng cuối cùng về tầng đầu tiên. Mỗi tầng trong mạng tính toán mức độ ảnh hưởng của các tham số đối với sai số, dựa trên quy tắc dây chuyền (chain rule). Điều này cho phép mạng xác định các tham số cần được điều chỉnh. Sau khi các gradient được tính toán, các trọng số và bias được cập nhật bằng các thuật toán tối ưu hóa như Gradient Descent. Điều này giúp mạng giảm sai số và cải thiện độ chính xác trong các chu kỳ huấn luyện tiếp theo.

Kết hợp lan truyền xuôi và lan truyền ngược tạo thành một quy trình khép kín, trong đó mạng liên tục học từ dữ liệu và cải thiện khả năng dự đoán. Lan truyền xuôi tập trung vào việc xử lý dữ liệu và tạo ra đầu ra, trong khi lan truyền ngược giúp mạng hiểu và sửa chữa sai sót. Hai quá trình này là nền tảng cho bất kỳ mô hình học sâu nào, đặc biệt là các ứng dụng sử dụng CNN trong xử lý ảnh.

#### **2.4.5. Overfitting và Underfitting**

Overfitting và underfitting là hai vấn đề quan trọng trong học máy và học sâu nói chung, phản ánh khả năng của một mô hình trong việc học và tổng quát hóa dữ liệu.

Overfitting xảy ra khi mô hình học quá tốt trên dữ liệu huấn luyện, đến mức nó bắt đầu "ghi nhớ" các chi tiết và nhiễu trong dữ liệu thay vì học các mẫu tổng quát. Nhưng lại đạt kết quả không cao với dữ liệu thử nghiệm. Hậu quả là mô hình hoạt động rất tốt trên dữ liệu huấn luyện nhưng lại kém hiệu quả khi được áp dụng lên dữ liệu mới (dữ liệu kiểm tra). Dấu hiệu của overfitting có thể kể đến như là giá trị của

hàm mất mát thấp trên dữ liệu huấn luyện nhưng cao trên dữ liệu kiểm tra. Hay là độ chính xác giảm trên dữ liệu kiểm tra mặc dù độ chính xác trên dữ liệu huấn luyện rất cao. Giải pháp chính là giảm độ phức tạp của mô hình và tăng đa dạng mẫu dữ liệu huấn luyện.

Underfitting xảy ra khi mô hình không đủ phức tạp để học đầy đủ các mẫu có trong dữ liệu huấn luyện. Điều này dẫn đến hiệu suất thấp trên cả dữ liệu huấn luyện và dữ liệu kiểm tra. Dấu hiệu của underfitting có thể kể đến như là độ chính xác trên dữ liệu huấn luyện và dữ liệu. Giải pháp chính là tăng độ phức tạp của mô hình và thêm thời gian huấn luyện.

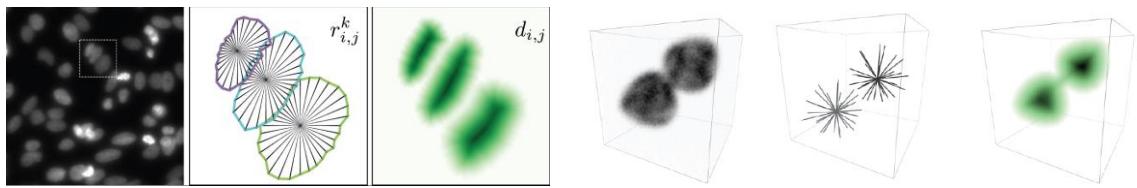
## 2.5. Nghiên cứu, phân tích và ứng dụng mô hình học máy StarDist

### 2.5.1. Đối tượng nhận dạng mà mô hình StarDist hướng đến

Mục đích ban đầu của StarDist là phát triển một mô hình phân vùng hình ảnh sinh học chính xác và hiệu quả, đặc biệt cho các đối tượng dạng star-convex trong ảnh tế bào học, nhằm xử lý các thách thức về hình dạng phức tạp và sự chồng lấn giữa các đối tượng. Với cách tiếp cận dựa trên bản đồ khoảng cách đa hướng (multi-radial distance maps), StarDist phân vùng độc lập từng đối tượng, ngay cả khi chúng chồng lấn, và mở rộng từ 2D lên 3D để hỗ trợ nghiên cứu tế bào và mô phức tạp. Mô hình này giúp tự động hóa quy trình phân vùng y sinh, nâng cao tính chính xác và hiệu suất, phục vụ đặc lực cho nghiên cứu khoa học và y học. Những hình ảnh vật thể mà mô hình này hướng đến để nhận diện thường là những vật thể có hình sao lồi, tiếng Anh gọi là Star-Convex.



**Hình 2.43 Hình Star-Convex (bên trái) và không phải là Star-Convex (bên phải)**

**Hình 2.44 Hình Star-Convex 2D****Hình 2.45 Hình Star-Convex 3D**

Star-Convex là các đối tượng luôn tồn tại một điểm tâm (center point) mà mọi điểm trên biên của hình đều có thể được nối với tâm bằng một đường thẳng, và đường thẳng này luôn nằm hoàn toàn bên trong hình. Nếu có tồn tại một đoạn thẳng nối từ tâm đến rìa mà cắt qua chính rìa của đối tượng đó thì đối tượng này không phải là hình Star-Convex. Tính chất này giúp mô hình dễ dàng mô tả và phân vùng các đối tượng sinh học với hình dạng không đều và không đối xứng. Tính chất star-convex giúp Stardist đơn giản hóa việc phân vùng các đối tượng chồng chéo nhau mà vẫn giữ được độ chính xác cao.

Stardist sử dụng phân vùng instance có nghĩa là mô hình không chỉ phân loại từng pixel thuộc về đối tượng nào mà còn phải phân biệt từng đối tượng riêng biệt trong ảnh. Điều này khác với semantic segmentation, nơi chỉ phân chia các vùng theo loại đối tượng mà không phân biệt giữa các cá thể riêng rẽ của chúng. Trong Stardist, mỗi đối tượng trong ảnh được phân vùng thành một đối tượng (instance) riêng biệt nhờ vào việc dự đoán đa giác star-convex cho từng đối tượng. Điều này rất quan trọng trong các hình ảnh sinh học, nơi các tế bào hoặc nhân tế bào có thể nằm gần nhau hoặc thậm chí chồng lấn. Với khả năng phân biệt và phân vùng từng đối tượng riêng biệt, Stardist cho phép chúng ta có thể phân tích số lượng và kích thước của các đối tượng mà chúng ta đang nghiên cứu một cách chính xác.

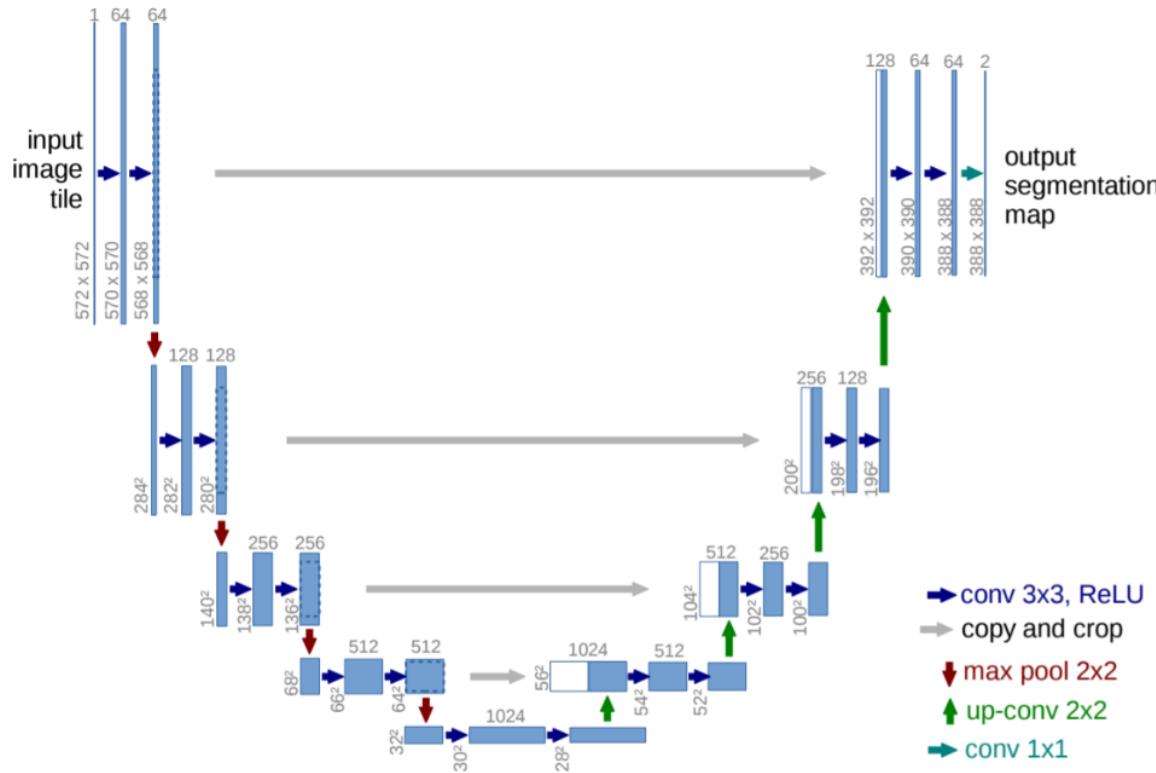
### 2.5.2. Kiến trúc của mô hình U-net

U-Net là một kiến trúc mạng nơ-ron tích chập (CNN) phổ biến, được thiết kế chủ yếu cho các bài toán phân đoạn ảnh trong thị giác máy tính. U-Net được giới

thiệu lần đầu vào năm 2015 trong bài báo "U-Net: Convolutional Networks for Biomedical Image Segmentation". Mạng này có cấu trúc hình chữ "U", bao gồm hai phần chính:

Encoder (Contraction Path): Học các đặc trưng từ ảnh đầu vào.

Decoder (Expansion Path): Tái dựng ảnh từ các đặc trưng để phân đoạn.



**Hình 2.46 Kiến trúc mạng U-net**

Nhìn vào hình 2.46 chúng ta có thể thấy Phần encoder của U-Net thực hiện nhiệm vụ giảm kích thước không gian của hình ảnh đầu vào thông qua các lớp tích chập và pooling. Cụ thể, các lớp tích chập (Conv2D) không chỉ giúp trích xuất các đặc trưng từ hình ảnh mà còn thêm sự phi tuyến tính nhờ vào các hàm kích hoạt như ReLU. Sau mỗi bước tích chập, một lớp pooling (MaxPooling) được áp dụng để giảm kích thước không gian, qua đó giúp mô hình học được các đặc trưng trừu tượng từ hình ảnh. Quá trình này giúp giảm thiểu độ phân giải của hình ảnh, tạo ra các đặc trưng có mức độ trừu tượng cao hơn.

Ngược lại, phần decoder có nhiệm vụ tái tạo lại hình ảnh với độ phân giải cao hơn, đồng thời phục hồi các chi tiết cần thiết cho phân đoạn. Để làm điều này, decoder sử dụng các lớp upsampling hoặc deconvolution (Conv2DTranspose) để phóng to các đặc trưng từ phần encoder. Tuy nhiên, một thử thách lớn đối với quá trình này là sự mất đi thông tin chi tiết trong quá trình giảm kích thước không gian tại phần encoder. Để giải quyết vấn đề này, U-Net sử dụng cơ chế copy and crop. Cụ thể, các đặc trưng từ phần encoder, sau khi được "sao chép" (copy), sẽ được "cắt" (crop) sao cho chúng có kích thước phù hợp với đặc trưng trong phần decoder. Thao tác này đảm bảo rằng thông tin từ encoder có thể được sử dụng để làm giàu thêm cho các đặc trưng được phóng to tại phần decoder, giúp cải thiện độ chính xác của phân đoạn.

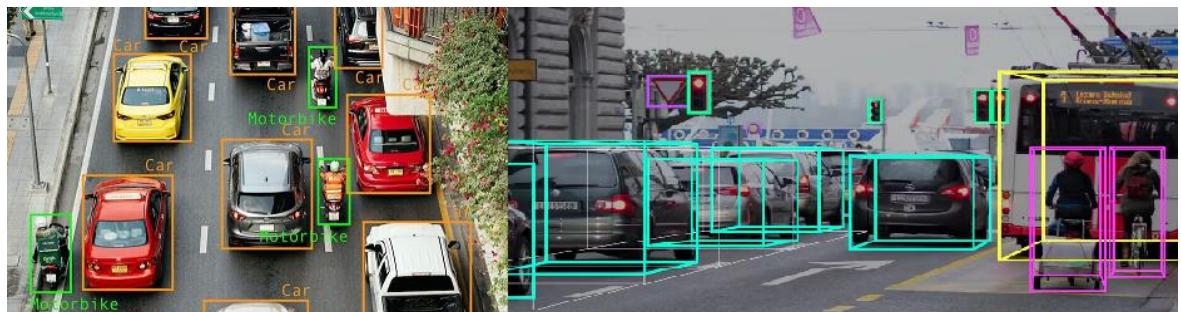
Kết nối giữa encoder và decoder thông qua cơ chế skip connection là một trong những yếu tố quan trọng giúp U-Net hoạt động hiệu quả. Các kết nối này cho phép thông tin được truyền trực tiếp từ các lớp của encoder đến các lớp tương ứng trong decoder, giúp duy trì thông tin chi tiết ở các cấp độ thấp trong quá trình học, đồng thời kết hợp với các đặc trưng trừu tượng hơn ở các cấp độ cao hơn. Điều này không chỉ giúp mạng học tốt hơn mà còn giúp cải thiện khả năng phân đoạn của mô hình, nhất là trong các trường hợp yêu cầu độ chính xác cao.

Cuối cùng, phần đầu ra của U-Net thường sử dụng một lớp tích chập cuối cùng với một số lượng kênh tương ứng với số lớp cần phân đoạn trong hình ảnh, đi kèm với một hàm kích hoạt như sigmoid hoặc softmax để chuyển đầu ra thành các xác suất cho từng lớp. Quá trình này không chỉ giúp phân loại các pixel trong hình ảnh mà còn tái tạo lại hình ảnh phân đoạn chi tiết, có thể ứng dụng trong nhiều lĩnh vực như y học, phân tích ảnh vệ tinh, hay nhận diện vật thể trong ảnh.

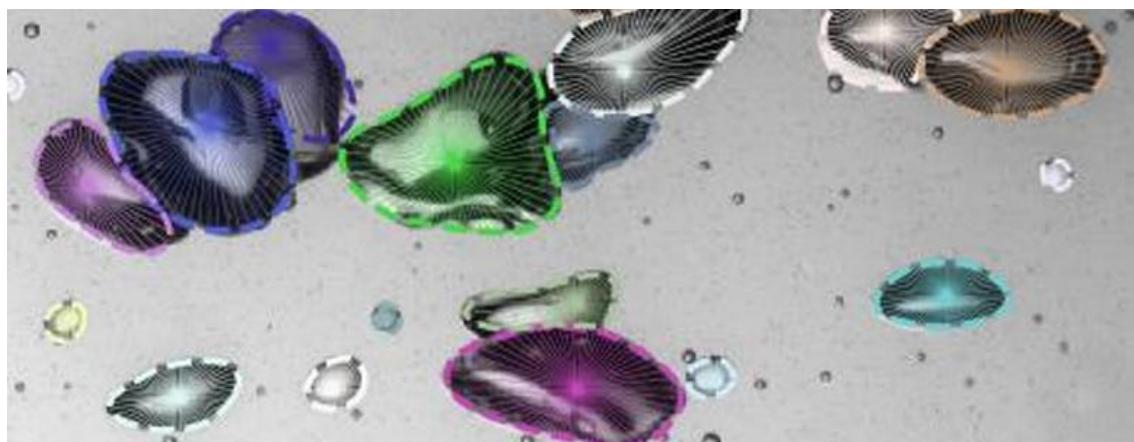
### **2.5.3. Kiến trúc của mô hình Stardist**

StarDist là một phương pháp tiên tiến được thiết kế để giải quyết bài toán phát hiện và phân đoạn các đối tượng, đặc biệt hiệu quả trong việc xử lý các đối tượng có hình dạng phức tạp hoặc xuất hiện chen chúc, như nhân tế bào trong ảnh kính hiển vi

hay các bọt khí xuất hiện trong chất lỏng. Khác với các phương pháp truyền thống sử dụng hộp giới hạn (bounding boxes), StarDist biểu diễn hình dạng đối tượng bằng các đa giác lồi sao (star-convex polygons), cung cấp một cách tiếp cận linh hoạt và chính xác hơn.



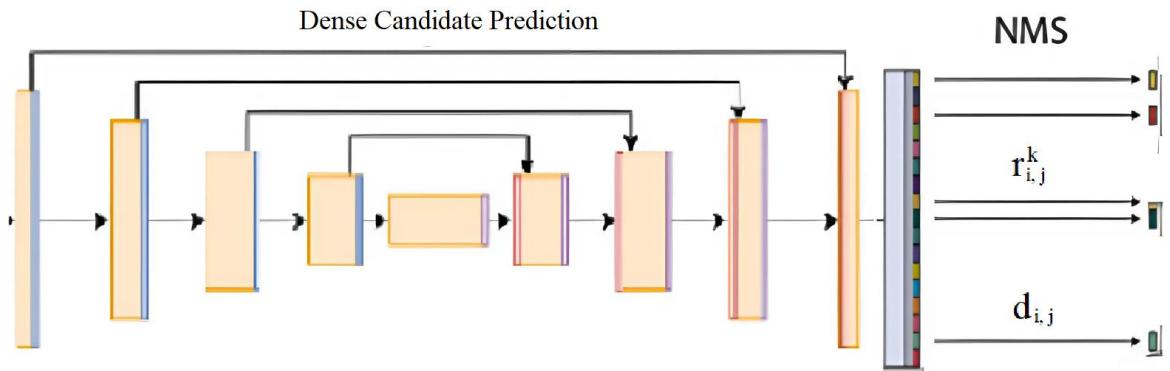
**Hình 2.47 Phương pháp sử dụng bounding boxes**



**Hình 2.48 Phương pháp sử dụng đa giác lồi**

Với các đối tượng mà mô hình Stardist hướng đến là các đối tượng có hình dạng phức tạp và thường nằm chồng lấn lên nhau trong các bức ảnh, giải pháp đưa ra là thực hiện xác định tâm hoặc các pixel gần tâm của đối tượng và từ đó phát đi các tia đèn rìa của đối tượng đó hoặc là rìa của đối tượng khác để xác định đây có phải là một đối tượng hay không. Nhưng trong một bức ảnh, không phải pixel nào trong ảnh cũng đều thuộc đối tượng cần phân vùng. Nếu chúng ta sử dụng tất cả các pixel trong ảnh để phát ra tia, việc này sẽ gây lãng phí rất nhiều hiệu năng của chương trình một

cách không cần thiết. Thay vào đó Stardist sử dụng bản đồ xác suất (probability map) và khoảng cách Euclid (radial distance map).



**Hình 2.49 Tổng quan kiến trúc Stardist**

Quá trình dự đoán trong mô hình StarDist được thực hiện dựa trên kiến trúc U-Net, với mục tiêu phân đoạn các vật thể trong ảnh đầu vào dưới dạng đa giác lồi (star-convex polygons). Giai đoạn này bao gồm các bước xử lý chính như sau:

#### Bước 1: Tiền xử lý ảnh đầu vào

Ảnh đầu vào, có thể ở định dạng grayscale hoặc RGB, được chuẩn hóa để đảm bảo tính nhất quán về cường độ sáng và kích thước. Quá trình này bao gồm việc chia tỷ lệ giá trị pixel về khoảng  $[0, 1]$  và, nếu cần, chuyển đổi ảnh RGB thành grayscale.

#### Bước 2: Trích Xuất Đặc Trung Bằng Mạng U-Net

Mạng U-Net, với cấu trúc gồm các nhánh mã hóa (encoder) và giải mã (decoder), được sử dụng để xử lý ảnh đầu vào.

- **Giai đoạn mã hóa:**

- Ảnh đầu vào đi qua các tầng tích chập (convolutional layers), kết hợp với các hàm kích hoạt phi tuyến tính như ReLU, để trích xuất các đặc trưng quan trọng.
- Các tầng pooling được sử dụng để giảm kích thước không gian, giúp tập trung vào các đặc trưng cốt lõi.

- **Giai đoạn giải mã:**

- Các đặc trưng được mở rộng trở lại kích thước không gian ban đầu thông qua các tầng giải chập (deconvolutional layers).
- Quá trình này kết hợp với các kết nối tắt (skip connections) từ nhánh mã hóa để giữ lại thông tin chi tiết không gian.

Kết quả cuối cùng là một biểu diễn đặc trưng của ảnh đầu vào, được sử dụng làm cơ sở cho hai nhánh đầu ra.

#### *Bước 3: Dự đoán Xác suất (Probability Map)*

Nhánh đầu ra thứ nhất của U-Net được thiết kế để dự đoán một bản đồ xác suất (probability map), trong đó:

- Mỗi pixel được gán một giá trị từ 0 đến 1, biểu thị xác suất thuộc về tâm của một vật thể.
- Giá trị xác suất cao nhất xuất hiện tại các pixel gần tâm vật thể, giảm dần khi khoảng cách tăng.

Bản đồ này được tính toán bằng các phép tích chập và chuẩn hóa qua hàm kích hoạt sigmoid. Nó đóng vai trò quan trọng trong việc xác định các điểm neo (anchor points) làm trung tâm của các vật thể.

#### *Bước 4: Dự đoán Khoảng cách Euclid (Radial Distance Map)*

Nhánh đầu ra thứ hai của U-Net dự đoán một bản đồ khoảng cách Euclid (radial distance map). Tại mỗi pixel thuộc về vật thể, mô hình ước lượng khoảng cách từ pixel đó đến biên của vật thể theo một số lượng hướng cố định k (thường là 32 hoặc 64 hướng). Các giá trị khoảng cách này được lưu dưới dạng mảng 3 chiều.

Các khoảng cách được tính toán thông qua các phép tích chập và giải chập trên ảnh đặc trưng, đảm bảo tính chính xác cao ngay cả đối với các vật thể có hình dạng phức tạp.

## Bước 5: Hậu Xử Lý và Tái Tạo Hình Dạng Vật Thể

Sau khi thu được bản đồ xác suất và bản đồ khoảng cách Euclid. Non-Maximum Suppression (NMS) được áp dụng lên bản đồ xác suất để loại bỏ các dự đoán dư thừa, đảm bảo mỗi vật thể chỉ có một tâm duy nhất. Sử dụng các giá trị khoảng cách Euclid, hình dạng của vật thể được tái dựng dưới dạng đa giác lồi (star-convex polygon). Các đa giác này được ánh xạ trở lại ảnh gốc, tạo ra kết quả phân đoạn trong đó mỗi vật thể được biểu diễn rõ ràng và không chồng lấp.

### 2.5.3.1. Xây dựng bản đồ xác suất

Bản đồ xác suất (probability map) là một biểu diễn không gian 2D, trong đó mỗi pixel trong ảnh được gán một giá trị xác suất nằm trong khoảng từ 0 đến 1. Giá trị này phản ánh khả năng pixel đó thuộc về một đối tượng hay nền. Bản đồ xác suất là kết quả đầu ra của mạng U-Net trong quá trình dự đoán. Nó có cùng kích thước với ảnh đầu vào và chứa giá trị xác suất  $d_{i,j}$  tại mỗi vị trí pixel  $(i, j)$ . Giá trị  $d_{i,j}$  càng cao (gần 1) thì khả năng pixel đó thuộc về một đối tượng càng lớn.  $d_{i,j}$  thấp (gần 0), pixel được coi là thuộc nền.

Mạng U-Net dự đoán bản đồ xác suất bằng cách phân tích các đặc trưng của từng pixel trong ảnh đầu vào và so sánh với nhãn thực tế (ground truth) dưới dạng mặt nạ nhị phân. Trong mặt nạ này, mỗi pixel được gán nhãn 1 nếu thuộc đối tượng và 0 nếu thuộc nền. Mục tiêu của mạng là học cách dự đoán giá trị xác suất  $d_{i,j}$  cho từng pixel sao cho gần nhất với giá trị nhãn thực tế. Quá trình tối ưu hóa được thực hiện bằng cách sử dụng hàm mất mát binary cross-entropy để đo lường sự khác biệt giữa giá trị dự đoán  $d_{i,j}$  và nhãn thực tế  $y_{i,j}$ . Qua đó, mạng được cải thiện để dự đoán chính xác hơn trong các bước tiếp theo. Hàm mất mát cross-entropy:

$$\text{Loss} = - \sum [y_{i,j} \log(d_{i,j}) + (1 - y_{i,j}) \log(1 - d_{i,j})]$$

Trong đó:  $y_{i,j}$  là nhãn thực tế (1 nếu pixel thuộc đối tượng, 0 nếu thuộc nền).

$d_{i,j}$  là xác suất dự đoán bởi mô hình.

U-Net có khả năng học cả đặc trưng cục bộ (như các đường biên và chi tiết nhỏ) và đặc trưng toàn cục (như hình dạng tổng thể của đối tượng). Với các ảnh chứa đối tượng đồng đúc như nhân tế bào hay bot khí, việc duy trì thông tin chi tiết này là rất quan trọng để phân biệt các đối tượng gần nhau.

StarDist sử dụng mạng U-Net để dự đoán bản đồ xác suất, xác định khả năng một pixel thuộc về đối tượng. Từ đó xác định được các pixel có khả năng phát ra tia để tái tạo lại đối tượng.

Sau khi xây dựng bản đồ xác suất, một ngưỡng xác định  $t$  (thường từ 0.5 đến 0.8) được sử dụng để chọn những pixel có xác suất đủ cao ( $d_{i,j} > t$ ) làm ứng viên cho việc phát hiện và phân đoạn đối tượng. Pixel không đạt ngưỡng được coi là nền và bị loại bỏ.

#### 2.5.3.2. Xây dựng đa giác lồi

Để xây dựng đa giác lồi mô hình Stardist đã sử dụng vector khoảng cách để lưu giữ các khoảng cách từ tâm của đối tượng đến điểm biên. Vector khoảng cách có dạng  $\{r_k\}_k^n$  trong đó mỗi phần tử  $r_k$  là khoảng cách từ một pixel trung tâm (được chọn làm tâm của đối tượng) đến biên của đối tượng theo hướng  $k$ . Các hướng  $k$  thường được chia đều trên vòng tròn  $360^0$ .

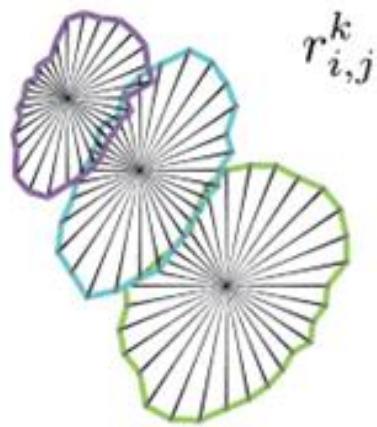
Ví dụ: mô hình được chia thành 8 hướng, chúng sẽ cách nhau:  $360^0 / 8 = 40^0$

Các hướng này là:  $0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0$ .

Lúc này vector khoảng cách sẽ có dạng:  $\{r_k\} = \{5,4.5,3,4.8,6,5.5,4,4.2\}$ .

Trong đó  $r_k$  là khoảng cách đo được từ tâm đến biên tại mỗi hướng  $k$ .

Trong mô hình Stardist các vector khoảng cách này sẽ được chuẩn hóa theo các hướng mà mô hình đã định nghĩa trước đó, điều này khiến cho mô hình chỉ cần lưu giữ các vector khoảng cách mà không cần lưu giữ số góc của vector đó, tối ưu được không gian lưu trữ và làm cho mô hình đơn giản hơn so với việc lưu giữ tọa độ của các điểm biên (x,y).



**Hình 2.50 Xây dựng lại đối tượng từ các tia từ tâm**

Mặc dù mô hình chỉ lưu giữ vector khoảng cách, nhưng khi cần tìm tọa độ các điểm trên biên thì chúng ta có thể dễ dàng chuyển đổi bằng công thức:

$$x_k = x_{center} + r_k \cos(\theta_k), \quad y_k = y_{center} + r_k \sin(\theta_k)$$

#### 2.5.3.3. Giải quyết vấn đề chồng lấn vật thể và loại bỏ các dự đoán dư thừa

Từ phương pháp dựng đa giác lồi trên ta lại thấy có một vấn đề, đó là trong bản đồ xác suất luôn tồn tại nhiều hơn một điểm thuộc về cùng một đối tượng. Trong mô hình StarDist, tất cả các pixel có xác suất cao (tức là những pixel được dự đoán là thuộc về một đối tượng) đều được phép phát ra tia để tái dựng hình dạng đối tượng. Tuy nhiên, chính việc này có thể dẫn đến hiện tượng nhiều đa giác (hình dạng dự đoán) được tạo ra cho cùng một đối tượng trong thực tế. Điều này dẫn đến tình trạng "nhiều vật thể" được dự đoán cho cùng một đối tượng thực tế.

Để đảm bảo mỗi đối tượng chỉ được đại diện bởi một đa giác duy nhất, StarDist áp dụng một bước xử lý quan trọng gọi là Non-Maximum Suppression (NMS). NMS là một phương pháp dùng để loại bỏ các dự đoán dư thừa (các đa giác hoặc đối tượng chồng lấn) và chỉ giữ lại một kết quả duy nhất, đáng tin cậy nhất cho mỗi đối tượng trong ảnh. Cách mà NMS hoạt động như sau:

*Bước 1: Xác định độ tin cậy của mỗi đa giác*

Mỗi đa giác đều có một điểm tin cậy, chính là xác suất dự đoán của pixel trung tâm phát ra đa giác đó ( $d_{i,j}$ ). Đa giác có điểm tin cậy cao hơn sẽ có ưu tiên cao hơn trong quá trình xử lý.

#### *Bước 2: Sắp xếp đa giác theo điểm tin cậy*

Tất cả các đa giác được sắp xếp theo thứ tự giảm dần của điểm tin cậy. Điều này đảm bảo rằng các đa giác có điểm tin cậy cao nhất sẽ được xử lý trước.

#### *Bước 3: Dánh giá mức độ chòng lấn (IoU)*

Đối với mỗi cặp đa giác, mức độ chòng lấn được đo bằng chỉ số Intersection over Union (IoU):

$$IoU = \frac{\text{Diện tích phần giao giữa hai đa giác}}{\text{Diện tích phần hợp giữa hai đa giác}}$$

IoU càng cao, hai đa giác càng trùng lặp. Chúng ta cần xác định một ngưỡng  $t$  sao cho phù hợp để không để sót vật thể trùng lặp và cũng không xóa nhầm vật thể bị chòng lấn:

- Nếu  $IoU > t$ , đa giác có điểm tin cậy thấp hơn sẽ bị loại bỏ.
- Nếu  $IoU \leq t$ , cả hai đa giác sẽ được giữ lại.

#### *Bước 4: Giữ lại một đa giác cho mỗi đối tượng*

Bắt đầu từ đa giác có điểm tin cậy cao nhất, NMS kiểm tra mức độ chòng lấn của đa giác này với các đa giác còn lại, nếu một đa giác bị chòng lấn quá nhiều với đa giác đang được giữ (IoU vượt ngưỡng), nó sẽ bị loại bỏ. Lặp lại quy trình cho đến khi tất cả các đa giác được xử lý.

Bằng cách sử dụng phương pháp NMS với các bước như trên thì mô hình Stardist có thể loại bỏ được các vật thể trùng lặp trong quá trình xây dựng vật thể, đồng thời cũng chính bằng phương pháp này Stardist có thể giữ lại được các vật thể bị chòng lấn lên nhau trong ảnh, điều mà các phương pháp phân vùng ảnh truyền

thống trước đây chưa làm được. Nhưng để phân vùng được chính xác cho các vật thể chồng lấn này thì việc chọn ngưỡng phù hợp là rất quan trọng. Nếu ngưỡng cao quá có thể giữ lại nhầm vật thể trùng lặp do quá trình xây dựng vật thể, ngược lại nếu ngưỡng quá cao có thể xóa bỏ nhầm đi vật thể bị trùng lặp thực tế trong ảnh. Ngoài ra, nếu như điểm tin cậy của pixel trung tâm hai vật thể tương đối cao thì mô hình cũng có thể dễ dàng nhận diện đây là hai vật thể riêng biệt. Hay hình dạng của các đa giác cũng là một tiêu chí để nhận diện, Stardist có thể dựa vào vector khoảng để xác định hình dạng của hai vật thể chồng lấn, nếu chúng khác biệt thì cũng có thể coi như đây là hai vật thể riêng biệt.

Để nâng cao hiệu quả của mô hình khi chồng lấn phức tạp chúng ta có thể điều chỉnh một số thông số sau:

- Điều chỉnh bản đồ xác suất: Tăng cường độ chính xác của bản đồ xác suất trong vùng giao nhau giữa hai vật thể bằng cách huấn luyện thêm với dữ liệu khó.
- Tăng số lượng tia phát ra: Sử dụng nhiều tia hơn (ví dụ 64 thay vì 32) để nắm bắt biên rõ ràng hơn.
- Điều chỉnh IoU: Dựa vào thực tế có thể điều chỉnh ngưỡng từ sao cho phù hợp (thường 0.3 – 0.7).

#### *2.5.3.4. Các tham số khi huấn luyện mô hình Stardist*

Khi huấn luyện Stardist, các tham số được chia làm hai nhóm chính: các tham số liên quan trực tiếp đến U-Net (cấu trúc mạng cơ bản) và các tham số đặc thù của Stardist (liên quan đến việc học các hình dạng và dự đoán).

➤ Các tham số cho mạng U-net bao gồm:

- n\_channel\_in: Số lượng kênh của hình ảnh đầu vào (ví dụ: 1 cho ảnh xám, 3 cho ảnh màu RGB).
- n\_channel\_out: Số lượng kênh đầu ra (thường là số đặc trưng cần học).
- n\_depth: Độ sâu của U-Net, tức là số lượng tầng encoder/decoder.
- kernels: Kích thước kernel trong các lớp tích chập.

- batch\_norm: Có sử dụng chuẩn hóa batch hay không.
- activation: Hàm kích hoạt trong các lớp (thường là ReLU hoặc LeakyReLU).
- pooling: Loại pooling được sử dụng trong encoder (ví dụ: max-pooling hoặc average-pooling).
- Các tham số đặc thù của Stardist gồm:
  - n\_rays: Số lượng tia hướng tâm (rays) được sử dụng để mô tả hình dạng các đối tượng. Giá trị này càng cao, mô hình càng chi tiết nhưng cũng phức tạp hơn.
  - grid: Kích thước lưới (grid) mà Stardist sử dụng để xử lý dữ liệu đầu ra của U-Net. Ví dụ, (2, 2) nghĩa là mỗi ô lưới xử lý 2x2 pixel.
  - anisotropy: Điều chỉnh tính không đồng nhất trong ảnh đầu vào, quan trọng đối với dữ liệu 3D hoặc dữ liệu có độ phân giải không đồng đều.
  - foreground\_prob: Nguồn xác suất để xác định vùng tiền cảnh (foreground), tức vùng chứa đối tượng cần phân đoạn.
  - prob\_thresh: Nguồn xác suất dự đoán để quyết định xem một pixel có thuộc về một đối tượng hay không.
  - dist\_thresh: Nguồn khoảng cách để hợp nhất các đối tượng dự đoán chồng lấn.
  - normalize: Các thông số chuẩn hóa ảnh đầu vào, thường bao gồm giá trị trung bình và độ lệch chuẩn.

Trong ngôn ngữ Python khi sử dụng với thư viện stardist và tensorflow chúng ta có thể dễ dàng tùy chỉnh một số tham số trên thông qua đoạn mã sau:

```
config = Config2D(
    n_channel_in=1,                      # Số kênh đầu vào (1 cho ảnh xám)
    n_channel_out=32,                     # Số kênh đầu ra (các đặc trưng trung gian)
    n_rays=32,                           # Số tia hướng tâm
    grid=(2, 2),                          # Kích thước lưới (grid)
    n_depth=3,                            # Độ sâu của U-Net
    train_batch_size=16,                  # Kích thước batch
    train_epochs=100,                     # Số lượng epochs huấn luyện
    train_learning_rate=0.0003,           # Tốc độ học
    train_loss_weights=(1, 0.1),          # Trọng số của loss (dist_loss,
    prob_loss)
    anisotropy=None # Độ bất đồng xứng (None nếu dữ liệu không có bất đồng
    xứng)
)
```

Ngoài ra có một số tham số sẽ được gắn liền theo mô hình và được cài đặt mặc định tham số trong thư viện như là kernels, batch\_norm, activation, pooling, anisotropy, foreground\_prob, prob\_thresh, dist\_thresh, normalize. Ví dụ như là hàm kích hoạt (activation) được định nghĩa mặc định trong cấu trúc U-Net cơ bản của Stardist, mà cụ thể thường là ReLU trong các tầng tích chập. Nếu cần thay đổi hàm kích hoạt mặc định chúng ta có thể sử dụng đoạn code sau để sửa đổi thay vì dùng mặc định như trong thư viện cung cấp sẵn:

```
from tensorflow.keras.layers import Input, Conv2D, Activation
from tensorflow.keras.models import Model
from stardist.models import StarDist2D

# Tạo U-Net tùy chỉnh với hàm kích hoạt LeakyReLU
def custom_unet(input_shape):
    inputs = Input(input_shape)
    x = Conv2D(64, (3, 3), padding='same')(inputs)
    x = Activation('leaky_relu')(x) # Sử dụng LeakyReLU
    outputs = Conv2D(1, (1, 1), activation='sigmoid')(x)
    return Model(inputs, outputs)

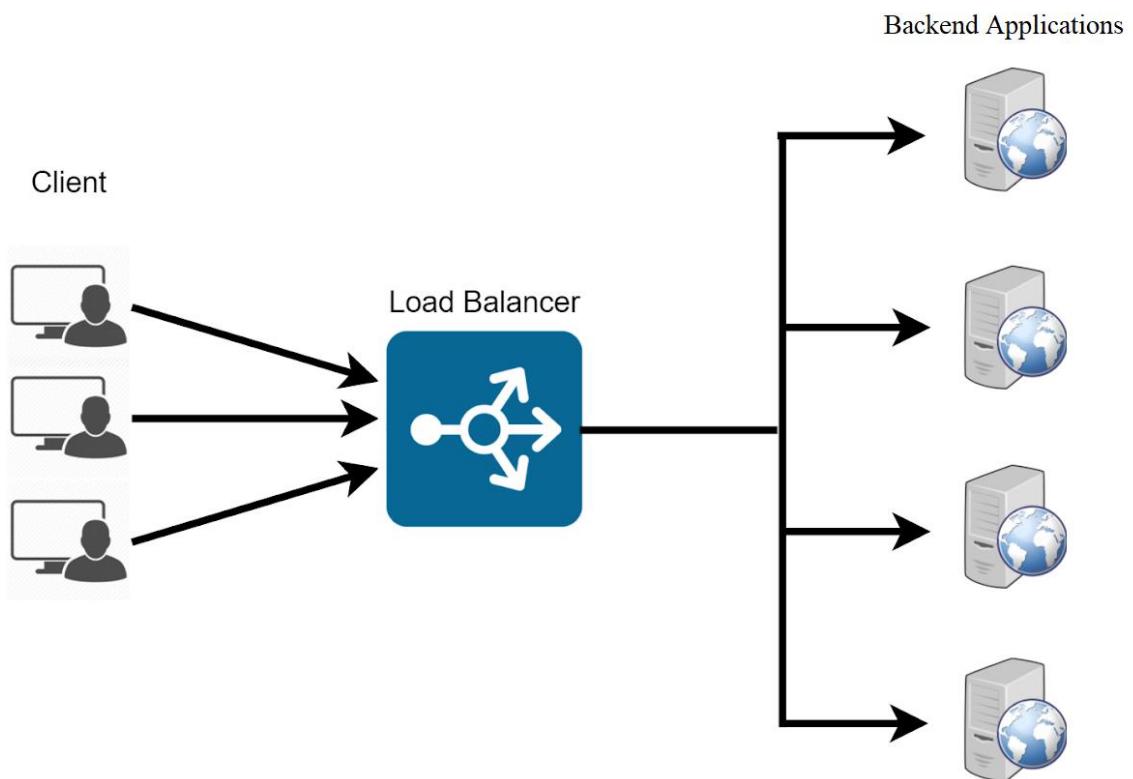
# Khởi tạo Stardist với mạng U-Net tùy chỉnh
input_shape = (None, None, 1) # (height, width, channels)
unet_model = custom_unet(input_shape)
model = StarDist2D(config=None, name="custom_stardist",
backbone=unet_model)
```

## 2.6. Kết luận Chương 2

Chương này đã thực hiện tìm hiểu về các phương pháp phân vùng ảnh từ các kỹ thuật thô sơ nhất như phân vùng dựa trên ngưỡng độ sáng (thresholding) hay phân vùng dựa trên cạnh (edge base segmentation). Cho đến các phương pháp phân vùng hiện đại ngày nay có sử dụng học máy như mạng nơ-ron tích chập (CNN). Trong chương 2 cũng đã đi sâu vào nghiên cứu mô hình phân vùng hình ảnh Stardist, đây là phương pháp hiện đại ban đầu được thiết kế để phân vùng các hình ảnh như các tế bào trong y sinh học. Vì đặc điểm của các bợt khí trong chất lỏng và các tế bào khá giống nhau cùng có dạng star-convex và thường nằm chồng chéo lên nhau trong các bức ảnh. Nên trong nghiên cứu này chúng ta sẽ sử dụng mô hình Stardist để tạo ứng dụng phân vùng ảnh xác định bợt khí trong chất lỏng ở chương 3.

## CHƯƠNG 3 XÂY DỰNG ỨNG DỤNG PHÂN VÙNG ẢNH DỰA TRÊN MẠNG NO-RON TÍCH CHẬP VÀ MÔ HÌNH STARDIST

### 3.1. Kiến trúc hệ thống



*Hình 3.1 Kiến trúc hệ thống*

Hệ thống có kiến trúc đơn giản bao gồm:

- Một UI application: Thực hiện các thao tác người dùng và gọi đến Backend application thông qua api để thực hiện các yêu cầu về phân vùng ảnh.
- Một Backend application: Thực hiện xử lý hình ảnh và quản lý dữ liệu như models, raw data,...

### **3.1.1. Lựa chọn công nghệ**

Stardist UI application: lựa chọn Vue 3 Framework là vì Vue 3 hiện đại và có hiệu năng cao, dễ sử dụng để xây dựng giao diện tương tác, trực quan, hiển thị kết quả phân vùng ảnh.

Lựa chọn công nghệ cho Backend application:

- Python: Ngôn ngữ lập trình phổ biến với nhiều thư viện hỗ trợ mạnh mẽ cho học máy và xử lý ảnh.
- TensorFlow: Framework học sâu tối ưu để triển khai mô hình Stardist, hỗ trợ huấn luyện và triển khai trên nhiều nền tảng.
- Flask: Framework nhẹ và linh hoạt, phù hợp để xây dựng API backend, dễ tích hợp với TensorFlow.
- OpenCV-Python: Thư viện xử lý ảnh mạnh mẽ, hỗ trợ tiền xử lý và hậu xử lý ảnh trước và sau khi áp dụng mô hình Stardist.

### **3.1.2. Phương thức giao tiếp**

RESTful API được chọn vì tính đơn giản, hiệu quả và khả năng mở rộng. Kiến trúc này sử dụng các phương thức HTTP trực quan, không trạng thái, giúp tối ưu hiệu suất và đảm bảo tương thích tốt với các nền tảng như Vue 3. Ngoài ra, RESTful API dễ bảo trì, dễ tích hợp với các dịch vụ khác và phù hợp cho việc mở rộng ứng dụng trong tương lai.

## **3.2. Xây dựng logic**

### **3.2.1. Xây dựng logic phân vùng ảnh cho các thuật toán cổ điển**

#### **3.2.1.1. Xây dựng logic cho thuật toán Otsu**

Dựa vào lý thuyết như đã nêu trong mục 2.1. Phân vùng dựa trên phân ngưỡng cường độ sáng (Thresholding) và 2.1.2.3. Tìm ngưỡng toàn cục tối ưu bằng phương pháp Otsu, tôi đã xây dựng api để cung cấp dịch vụ phân vùng ảnh sử dụng phương pháp Otsu như sau:

```

THRESHOLD_FOLDER = 'threshold_images'
CORS(threshold, origins=["http://localhost:8080"])

@threshold.route('/api/v1.0/threshold', methods=['POST'])
def threshold_image():
    # Kiểm tra xem có tệp được tải lên không
    if 'image' not in request.files:
        return jsonify({'error': 'No image file provided'}), 400

    file = request.files['image']

    # Kiểm tra xem tệp có phải là hình ảnh không
    if not file or not file.filename:
        return jsonify({'error': 'No image file provided'}), 400

    # Đọc hình ảnh từ tệp
    image = cv2.imdecode(np.frombuffer(file.read(), np.uint8),
cv2.IMREAD_GRAYSCALE)

    if image is None:
        return jsonify({'error': 'Could not read the image'}), 400

    # Đảm bảo thư mục THRESHOLD_FOLDER tồn tại
    if not os.path.exists('static/' + THRESHOLD_FOLDER):
        os.makedirs('static/' + THRESHOLD_FOLDER, exist_ok=True)

    # Áp dụng phương pháp Otsu để phân ngưỡng
    threshold_value, thresholded_image = cv2.threshold(image, 0, 255,
cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # Lưu hình ảnh đã phân ngưỡng
    image_file_name = generate_filename()
    histogram_file_name = generate_filename()
    thresholded_image_path = os.path.join('static/' + THRESHOLD_FOLDER,
image_file_name)
    cv2.imwrite(thresholded_image_path, thresholded_image)

    # Tính histogram của hình ảnh đầu vào
    histogram, bin_edges = np.histogram(image, bins=256, range=(0, 256))

    # Vẽ histogram và lưu dưới dạng ảnh
    plt.figure(figsize=(10, 5))
    plt.plot(bin_edges[0:-1], histogram, color="black")
    plt.axvline(threshold_value, color='red', linestyle='dashed',
linewidth=1.5, label=f'Threshold: {threshold_value}')
    plt.legend()
    plt.title("Histogram with Otsu's Threshold")
    plt.xlabel("Pixel Intensity")
    plt.ylabel("Frequency")

    # Lưu histogram vào tệp
    histogram_image_path = os.path.join('static/' + THRESHOLD_FOLDER,
histogram_file_name)
    plt.savefig(histogram_image_path)
    plt.close()

```

```

# Trả về đường dẫn URL của hình ảnh đã phân ngưỡng và histogram
return jsonify({
    'thresholded_image_url': url_for('static',
filename=f'{THRESHOLD_FOLDER}/{image_file_name}'),
    'histogram_url': url_for('static',
filename=f'{THRESHOLD_FOLDER}/{histogram_file_name}'),
    'threshold_value': threshold_value
})

```

Thêm vào đó để giải quyết vấn đề trùng tên file ảnh, tôi đã xử lý vấn đề này bằng cách tạo tên ảnh ngẫu nhiên bằng cách sử dụng hàm dưới đây:

```

def generate_filename(extension='png'):
    timestamp = datetime.now().strftime("%d%m%y-%H%M%S")
    unique_id = uuid.uuid4() # Tạo UUID ngẫu nhiên
    return f"{timestamp}-{unique_id}.{extension}"

```

### 3.2.1.2. Xây dựng logic cho thuật toán Canny

Dựa vào lý thuyết như đã nêu trong mục 2.2. Phân vùng dựa trên cạnh (Edge-based segmentation) và 2.2.3. Phương pháp Canny, tôi đã xây dựng api để cung cấp dịch vụ phân vùng ảnh sử dụng phương pháp Canny như sau:

```

EDGE_FOLDER = 'edge_segmentation_images'
CORS(edge, origins=["http://localhost:8080"])

@edge.route('/api/v1.0/canny', methods=['POST'])
def canny_edge_segmentation():
    # Kiểm tra xem có file ảnh trong request không
    if 'image' not in request.files:
        return jsonify({"error": "No image provided"}), 400

    # Đảm bảo thư mục THRESHOLD_FOLDER tồn tại
    if not os.path.exists('static/' + EDGE_FOLDER):
        os.makedirs('static/' + EDGE_FOLDER, exist_ok=True)

    # Lấy file ảnh từ request
    file = request.files['image']
    image = Image.open(file.stream).convert('L') # Chuyển ảnh thành grayscale

    # Chuyển ảnh thành mảng numpy
    image_np = np.array(image)

    # Áp dụng thuật toán Canny với ngưỡng tùy chọn
    threshold_value1, threshold_value2 = 100, 200 # Các ngưỡng tùy chọn
    edges = cv2.Canny(image_np, threshold1=threshold_value1,
threshold2=threshold_value2)

    # Lưu ảnh đã xử lý vào thư mục static
    file_name = generate_filename()
    thresholded_image_path = os.path.join('static/' + EDGE_FOLDER,

```

```

file_name)
cv2.imwrite(thresholded_image_path, edges)

return jsonify({
    'canny_image_url': url_for('static',
filename=f'{EDGE_FOLDER}/{file_name}')
}), 200

```

Thêm vào đó để giải quyết vấn đề trùng tên file ảnh, tôi đã xử lý vấn đề này bằng cách tạo tên ảnh ngẫu nhiên bằng cách sử dụng hàm dưới đây:

```

def generate_filename(extension='png'):
    timestamp = datetime.now().strftime("%d%m%y-%H%M%S")
    unique_id = uuid.uuid4() # Tạo UUID ngẫu nhiên
    return f"{timestamp}-{unique_id}.{extension}"

```

### 3.2.1.3. Xây dựng logic cho thuật toán Hog

Dựa vào lý thuyết như đã nêu trong mục 2.3. Phương pháp phân vùng ảnh dựa trên HOG (Histogram of Oriented Gradients). Tôi đã xây dựng api để cung cấp dịch vụ phân vùng ảnh sử dụng phương pháp Hog như sau:

```

if 'image' not in request.files:
    return jsonify({"error": "No image provided"}), 400

file = request.files['image']
image = Image.open(file.stream).convert('RGB')
image_np = np.array(image)

# 1. Khởi tạo tham số cho HOG
cell_size = (8, 8)
block_size = (2, 2)
bins = 9 # Số hướng của gradient

# 2. Tính toán đặc trưng HOG và trực quan hóa HOG (chỉ rõ channel_axis)
hog_features, hog_image = hog(image_np,
                                pixels_per_cell=cell_size,
                                cells_per_block=block_size,
                                orientations=bins,
                                visualize=True,
                                channel_axis=-1) # Chỉ rõ trực màu cuối cùng

# 3. Tạo ảnh HOG để trực quan hóa và lưu lại
hog_image_file_name = generate_filename()
hog_image_path = os.path.join(f'static/{HOG_FOLDER}', hog_image_file_name)
cv2.imwrite(hog_image_path, (hog_image * 255).astype('uint8'))

# 4. Áp dụng GrabCut để phân đoạn ảnh
mask = np.zeros(image_np.shape[:2], np.uint8)

```

```

bgd_model = np.zeros((1, 65), np.float64)
fgd_model = np.zeros((1, 65), np.float64)
rect = (10, 10, image_np.shape[1] - 10, image_np.shape[0] - 10)
cv2.grabCut(image_np, mask, rect, bgd_model, fgd_model, 5,
cv2.GC_INIT_WITH_RECT)
mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
segmented_image = image_np * mask2[:, :, np.newaxis]

# 5. Tìm các bounding boxes
gray_image = cv2.cvtColor(segmented_image, cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray_image, 1, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

bounding_boxes = []
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    bounding_boxes.append({'x': x, 'y': y, 'width': w, 'height': h})

segmented_image_with_boxes = cv2.cvtColor(segmented_image,
cv2.COLOR_BGR2RGB)
for box in bounding_boxes:
    x, y, w, h = box['x'], box['y'], box['width'], box['height']
    cv2.rectangle(segmented_image_with_boxes, (x, y), (x + w, y + h), (0,
0, 255), 2)

# 6. Lưu ảnh phân đoạn
segmented_image_file_name = generate_filename()
segmented_image_path = os.path.join(f'static/{HOG_FOLDER}', segmented_image_file_name)
cv2.imwrite(segmented_image_path,
cv2.cvtColor(segmented_image_with_boxes, cv2.COLOR_RGB2BGR))

return jsonify({
    'hog_image_url': url_for('static',
filename=f'{HOG_FOLDER}/{hog_image_file_name}'),
    'segmented_image_url': url_for('static',
filename=f'{HOG_FOLDER}/{segmented_image_file_name}'),
    'bounding_boxes': bounding_boxes,
    'total_bounding_boxes': len(bounding_boxes)
}), 200

```

### 3.2.2. Xây dựng logic phân vùng ảnh dựa trên mô hình Stardist

#### 3.2.2.1. Chuẩn hóa hình ảnh

##### ➤ Vấn đề định dạng ảnh

Như chúng ta biết ảnh có đuôi mở rộng .jpg có kích thước tệp nhỏ, sử dụng nén lossy, có thể làm mất chi tiết, không phù hợp cho việc training. Ảnh có đuôi mở rộng .png và .tif/.tiff đều là ảnh có độ phân giải cao, hỗ trợ đến 16 triệu màu và không làm mất chi tiết. Tuy nhiên .tif/.tiff tốt hơn cho cả ảnh lớn và hỗ trợ cho cả

ảnh có độ sâu màu 16-bit hoặc 32 bit thích hợp hơn cho bài toán phân vùng hình ảnh mà tôi đang nghiên cứu trong đề án này.

Từ đó, tôi quyết định sử dụng định dạng .tif/tiff làm định dạng chuẩn của ảnh để training model.

Trong tập dataset ‘*bubbly Computer Vision Project*’ thì tất cả ảnh đang có đuôi mở rộng là .jpg, nên tôi đã sử dụng đoạn code sau để convert ảnh từ .jpg sang .tif, đồng thời trong quá trình này tôi đã chuyển ảnh về dạng gray và giữ nguyên kích thước của ảnh gốc:

```
from PIL import Image
import os

def convert_jpg_to_tif(input_folder, output_folder):
    # Tạo thư mục đầu ra nếu chưa tồn tại
    os.makedirs(output_folder, exist_ok=True)

    for filename in os.listdir(input_folder):
        if filename.lower().endswith('.jpg'): # Chỉ xử lý file .jpg
            input_path = os.path.join(input_folder, filename)
            output_path = os.path.join(output_folder,
os.path.splitext(filename)[0] + '.tif')
            try:
                # Mở ảnh và đảm bảo không thay đổi kích thước
                with Image.open(input_path) as img:
                    # Chuyển ảnh sang chế độ Grayscale
                    img = img.convert("L") # Chế độ "L" cho ảnh grayscale

                # Lưu ảnh dưới dạng .tif với kích thước không thay đổi
                img.save(output_path, format='TIFF')

                print(f"Converted: {input_path} -> {output_path}")

            except Exception as e:
                print(f"Error processing {input_path}: {e}")

# Thư mục chứa file .jpg và nơi lưu file .tif
input_folder = "data/bubble-dataset/valid/origin"
output_folder = "data/bubble-dataset/valid/images"

# Gọi hàm chuyển đổi
convert_jpg_to_tif(input_folder, output_folder)
```

➤ Vấn đề về độ phân giải

Với mô hình Stardist, để đảm bảo mô hình hoạt động hiệu quả thì Stardist có yêu cầu tối thiểu về kích thước hình ảnh đầu vào phải lớn hơn hoặc bằng 256 pixel.

Trong dataset mà tôi đang sử dụng có 2 dạng kích thước ảnh là 320x200 và 480x270. Chúng ta có thể dễ dàng thấy được là kích thước 320x200 là không đạt yêu cầu về kích thước để training, do đó tôi cần loại bỏ chúng. Tôi đã sử dụng đoạn mã code sau đây:

```
import os
from PIL import Image

# Đường dẫn tới folder chứa ảnh
folder_path = "data/bubble-dataset/valid/masks"

# Nguồn kích thước tối thiểu (width, height)
min_width = 256
min_height = 256

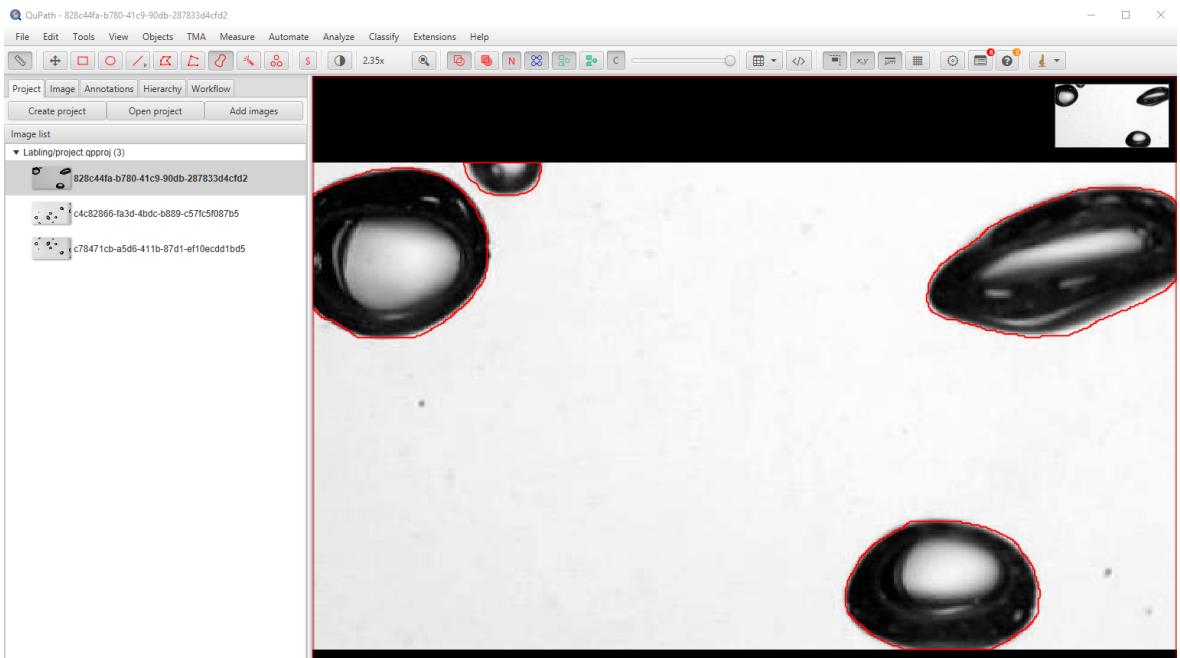
# Duyệt qua tất cả file trong folder
for filename in os.listdir(folder_path):
    file_path = os.path.join(folder_path, filename)

    # Kiểm tra nếu file là ảnh
    try:
        with Image.open(file_path) as img:
            width, height = img.size

            # Xóa ảnh nếu kích thước nhỏ hơn nguồn
            if width < min_width or height < min_height:
                print(f"Deleting {filename}: {width}x{height}")
                os.remove(file_path)
    except Exception as e:
        print(f"Skipping {filename}: {e}")
```

### 3.2.2.2. Tạo ra ảnh mask

Để training model thì chúng ta cần có ảnh gốc (images) và ảnh mặt nạ (masks). Cách truyền thông thường là tạo ra ảnh mask thủ công. Phương pháp này mang lại độ chính xác cao, đảm bảo chất lượng dữ liệu và giúp cải thiện hiệu suất mô hình. Trong đề án này tôi sử dụng phần mềm QuPath để thực hiện việc đánh label



**Hình 3.2 Giao diện sau khi đánh label**

Ngoài ra, đối với những dữ liệu public có sẵn thì thường chúng ta sẽ có tập ảnh gốc và thường là một file json lưu giữ vị trí tọa độ của các vật thể trong từng ảnh. Do đó chúng ta phải tạo ra ảnh mặt nạ từ data đã có. Tôi sử dụng đoạn mã sau:

```

import json
import numpy as np
from skimage.draw import polygon
from tifffile import imwrite
import os
from tqdm import tqdm

# Nguồn kích thước tối thiểu (width, height)
min_width = 256
min_height = 256

def create_mask_from_coco(json_file, output_mask_folder):
    # Tải dữ liệu COCO
    with open(json_file, 'r') as f:
        coco_data = json.load(f)

    # Tạo thư mục lưu mask nếu chưa tồn tại
    os.makedirs(output_mask_folder, exist_ok=True)

    # Duyệt qua từng ảnh trong dữ liệu
    for img_info in tqdm(coco_data['images']):
        # Thông tin ảnh
        img_id = img_info['id']
        width = img_info['width']

```

```

height = img_info['height']
file_name = img_info['file_name']

# Tạo mask đa lớp (ban đầu tất cả là 0 - nền)
mask = np.zeros((height, width), dtype=np.uint8)

# Lấy annotations liên quan đến ảnh này
annotations = [ann for ann in coco_data['annotations'] if
ann['image_id'] == img_id]

# Duyệt qua từng annotation
for ann in annotations:
    category_id = ann['category_id'] # Lớp của đối tượng
    for seg in ann['segmentation']:
        # Xây dựng polygon từ segmentation
        poly = np.array(seg).reshape(-1, 2)
        rr, cc = polygon(poly[:, 1], poly[:, 0], mask.shape)
        mask[rr, cc] = category_id # Gán giá trị lớp

# Lưu mask thành tệp ảnh
mask_file_name = os.path.splitext(file_name)[0] + '_mask.tif'
mask_path = os.path.join(output_mask_folder, mask_file_name)
imwrite(mask_path, mask)

# Thông tin đầu vào
json_file = 'data/bubble-dataset/train/_annotations.coco.json'
images_folder = 'data/bubble-dataset/train/origin'
output_mask_folder = 'data/bubble-dataset/train/masks'
output_format = 'tif'

# Tạo mask
create_mask_from_coco(json_file, images_folder, output_mask_folder)

```

### 3.2.2.3. Huấn luyện mô hình (training)

Sau khi chuẩn hóa được dataset thành công, bây giờ chúng ta đã có hai folder là images và masks để bắt đầu training.

#### ➤ Bước 1: Load và xác thực hình ảnh trước khi training

```

image_files = sorted(glob(os.path.join(image_folder, '*.tif')))
mask_files = sorted(glob(os.path.join(mask_folder, '*.tif')))

# Kiểm tra số lượng tệp ảnh và mặt nạ
assert len(image_files) == len(mask_files), "Số lượng ảnh và mặt nạ không khớp"

# Đọc dữ liệu
X = list(map(imread, image_files))
Y = list(map(imread, mask_files))

# Kiểm tra kích thước của các ảnh và mặt nạ
for x, y in zip(X, Y):
    assert x.shape == y.shape, "Ảnh và mặt nạ phải có cùng kích thước"

```

```
# Kiểm tra số lượng dữ liệu, phải có ít nhất 2 mẫu trong tên
assert len(X) > 1, "not enough training data"
```

- Bước 2: Chia tập dữ liệu thành phần training và phần validation. Ở đây tôi chọn 15% ảnh trong tập ảnh gốc để làm validation

```
rng = np.random.RandomState(42)
# Hoán vị chỉ số của toàn bộ dữ liệu
ind = rng.permutation(len(X))

# Chia dữ liệu thành tập train và validation, tỷ lệ validation là 15%
n_val = max(1, int(round(0.15 * len(ind))))
ind_train, ind_val = ind[:-n_val], ind[-n_val:]

# Lấy dữ liệu train và validation theo các chỉ số
X_val, Y_val = [X[i] for i in ind_val], [Y[i] for i in ind_val]
X_trn, Y_trn = [X[i] for i in ind_train], [Y[i] for i in ind_train]
```

- Bước 3: Cấu hình mô hình Stardist

```
conf = Config2D(
    n_rays=RAYS,
    grid=GRID,
    use_gpu=USE_GPU,
    n_channel_in=N_CHANNEL,
)
```

- Bước 4: Tăng cường dữ liệu bằng cách lật ảnh ngẫu nhiên và thêm nhiễu Gaussian

```
def random_fliprot(img, mask):
    assert img.ndim >= mask.ndim
    axes = tuple(range(mask.ndim))
    perm = tuple(np.random.permutation(axes))
    img = img.transpose(perm + tuple(range(mask.ndim, img.ndim)))
    mask = mask.transpose(perm)
    for ax in axes:
        if np.random.rand() > 0.5:
            img = np.flip(img, axis=ax)
            mask = np.flip(mask, axis=ax)
    return img, mask

def random_intensity_change(img):
    img = img * np.random.uniform(0.6, 2) + np.random.uniform(-0.2, 0.2)
    return img

def augmenter(x, y):
    # Áp dụng phép lật và xoay ngẫu nhiên cho ảnh và nhãn
    x, y = random_fliprot(x, y)

    # Thay đổi cường độ ảnh một cách ngẫu nhiên (tăng/giảm độ sáng)
    x = random_intensity_change(x)

    # Thêm một chút nhiễu Gaussian vào ảnh để tăng độ phong phú cho dữ
```

```

liệu
    sig = 0.02 * np.random.uniform(0, 1)      # Tạo giá trị sigma ngẫu
nhiên từ 0 đến 0.02
    x = x + sig * np.random.normal(0, 1, x.shape)    # Thêm nhiễu Gaussian
vào ảnh

# Trả về ảnh đã được tăng cường (augmented) cùng với nhãn gốc
return x, y

```

➤ Bước 5: Khởi tạo mô hình với cấu hình ở bước 3, training và tối ưu hóa ngưỡng

```
# Khởi tạo mô hình StarDist2D với cấu hình và thư mục lưu trữ
model = StarDist2D(conf, name=modelName, basedir=MODEL_FOLDER)
```

```
# Huấn luyện mô hình với dữ liệu train và validation, đồng thời sử dụng
augmenter
history = model.train(X_trn, Y_trn, epochs=EPOCHS,
validation_data=(X_val, Y_val), augmenter=augmenter)
# Tối ưu hóa ngưỡng phân đoạn (thresholds) dựa trên tập validation
model.optimize_thresholds(X_val, Y_val)
```

Sau khi training thành công, ta sẽ nhận được mô hình đã huấn luyện bao gồm folder log, file config.json, threshold.json, weights\_best.h5, weights\_last.h5. Chúng ta sẽ dùng mô hình đã huấn luyện này để phân vùng ảnh.

#### 3.2.2.4. Phân vùng ảnh (predict)

➤ Bước 1: Mở ảnh và chuẩn hóa

```
# Mở ảnh
image = io.imread(imageUpload)

# Kiểm tra xem ảnh có phải là ảnh 1D hay 2D
if image.ndim == 1:
    axis_norm = 0 # Dùng axis=0 cho mảng 1D
elif image.ndim == 2:
    axis_norm = (0, 1) # Dùng cả hai trục cho mảng 2D
else:
    axis_norm = (0, 1) # Mặc định cho các mảng nhiều chiều

# Chuẩn hóa ảnh
image = normalize(image, 1, 99.8, axis=axis_norm)
```

➤ Bước 2: Mở mô hình đã được huấn luyện

```
# Mở mô hình đã được huấn luyện
model = StarDist2D(None, name=modelName, basedir=MODEL_FOLDER)
```

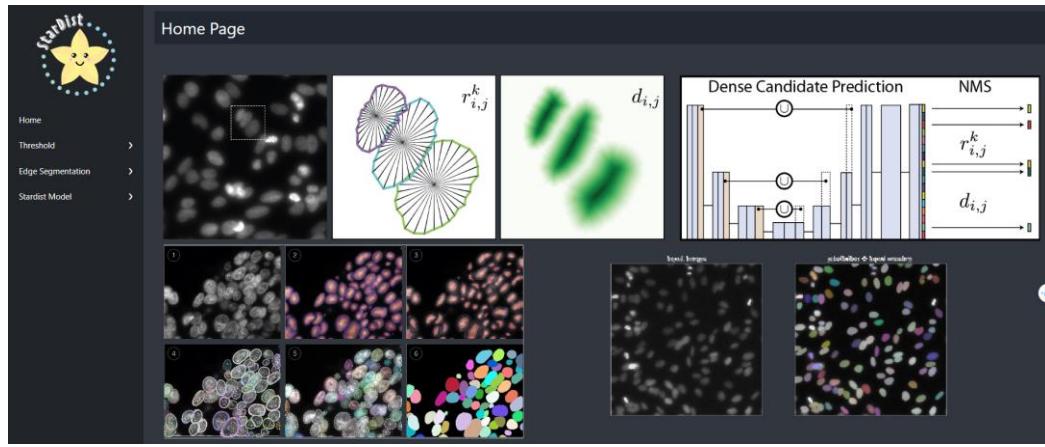
➤ Bước 3: Dự đoán ảnh

```
# Dự đoán ảnh
y_test = model.predict_instances(image,
n_tiles=model._guess_n_tiles(image), show_tile_progress=False)
```

### 3.3. Xây dựng giao diện

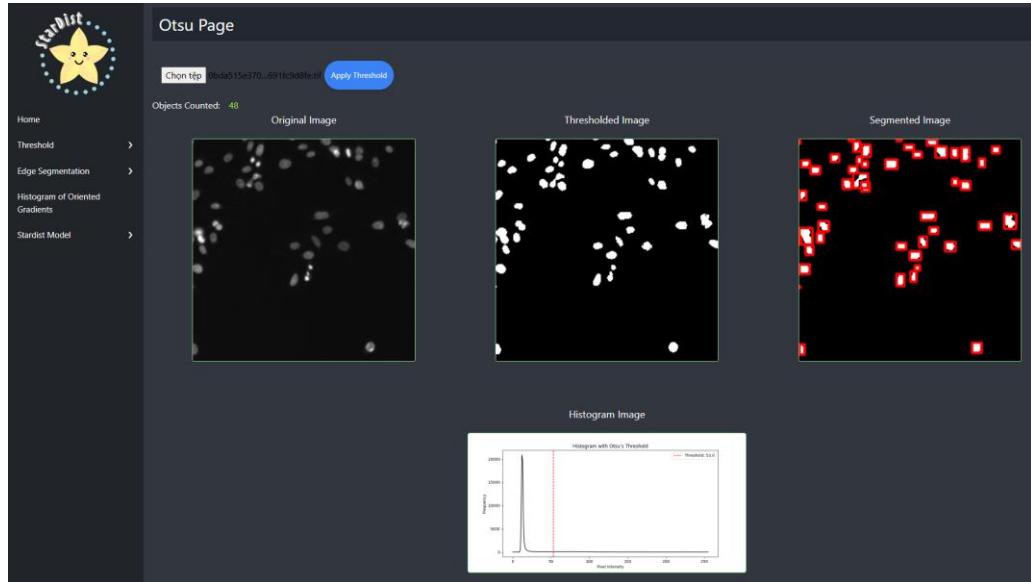
#### 3.3.1. Xây dựng giao diện menu

Xây dựng giao diện menu gồm các mục như trong hình ảnh bên dưới nhằm đáp ứng nhu cầu bài toán đưa ra.



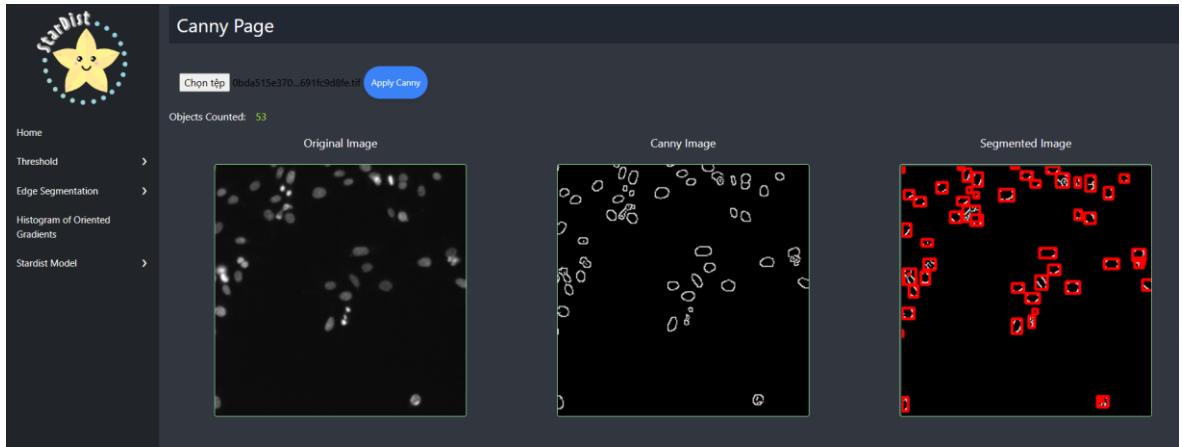
**Hình 3.3 Giao diện menu và màn hình home**

#### 3.3.2. Xây dựng giao diện sử dụng phương pháp Otsu



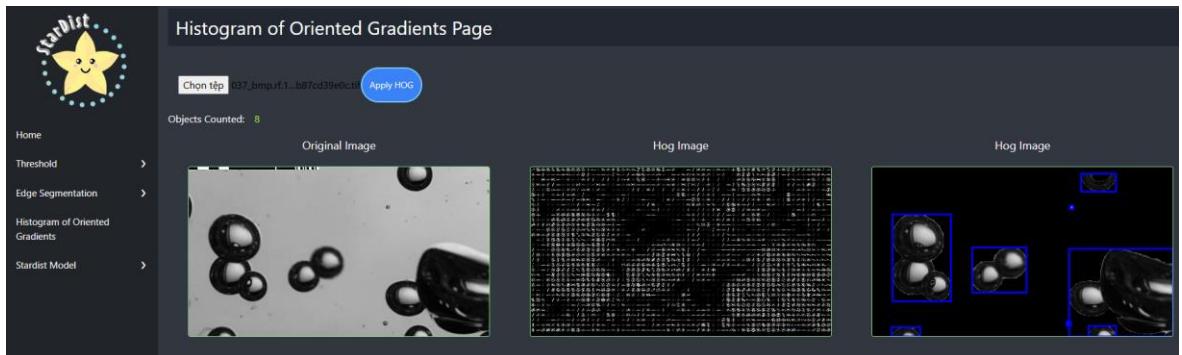
**Hình 3.4 Giao diện sử dụng phương pháp Otsu**

### 3.3.3. Xây dựng giao diện sử dụng phương pháp Canny



**Hình 3.5 Giao diện sử dụng phương pháp Canny**

### 3.3.4. Xây dựng giao diện sử dụng phương pháp Canny



**Hình 3.6 Giao diện sử dụng phương pháp HOG**

### 3.3.5. Xây dựng giao diện sử dụng mô hình Stardist

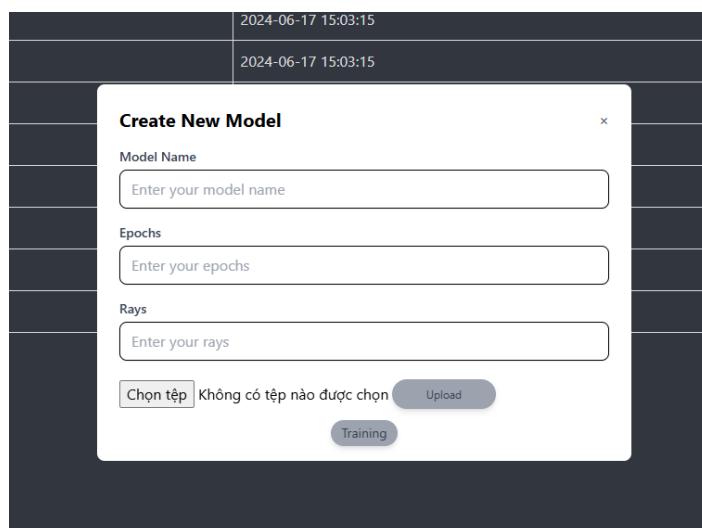
#### 3.3.5.1. Giao diện training

Tên mô hình có cấu trúc:

{tên mô hình}\_{số tia phát ra từ tâm vật thể}\_{số lần training}

Model Name	Creation Time	Action
bubble_32_10	2024-06-17 15:03:15	Delete
bubble_32_100	2024-06-17 15:03:15	Delete
bubble_32_20	2024-06-17 15:03:15	Delete
bubble_32_5	2024-06-17 15:03:15	Delete
bubble_64_10	2024-06-17 15:03:15	Delete
bubble_64_100	2024-06-17 15:03:15	Delete
bubble_64_30	2024-06-17 15:03:15	Delete
bubble_64_50	2024-06-17 15:03:15	Delete

**Hình 3.7 Giao diện danh sách các mô hình đã được training**



**Hình 3.8 Giao diện popup cấu hình training**

### 3.3.5.2. Giao diện predict

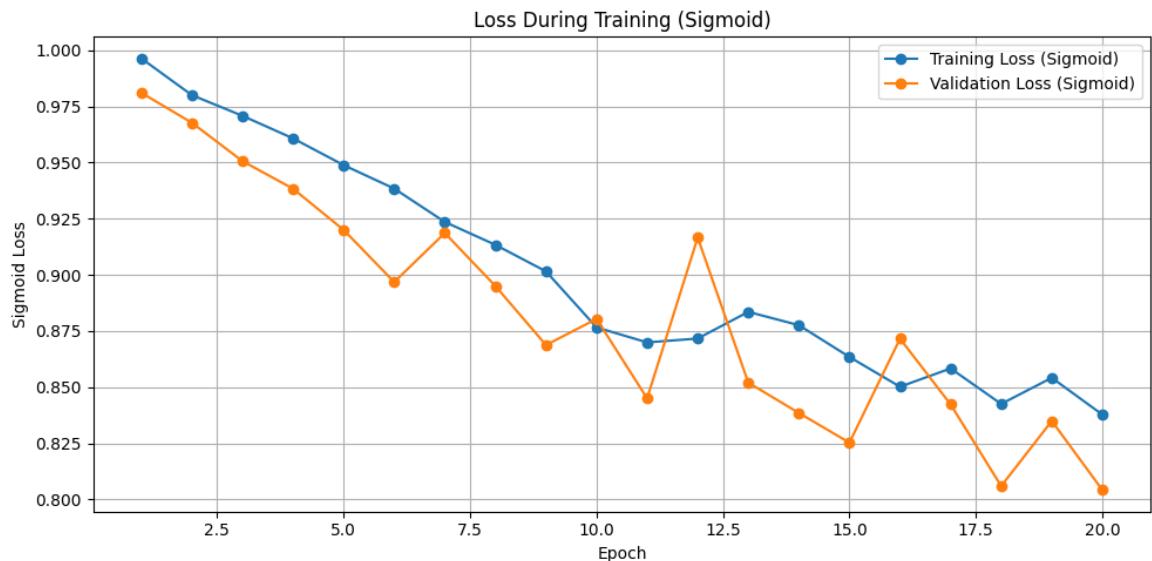
**Hình 3.9 Giao diện phân vùng ảnh**

### 3.4. Thủ nghiệm

#### 3.4.1. Huấn luyện mô hình

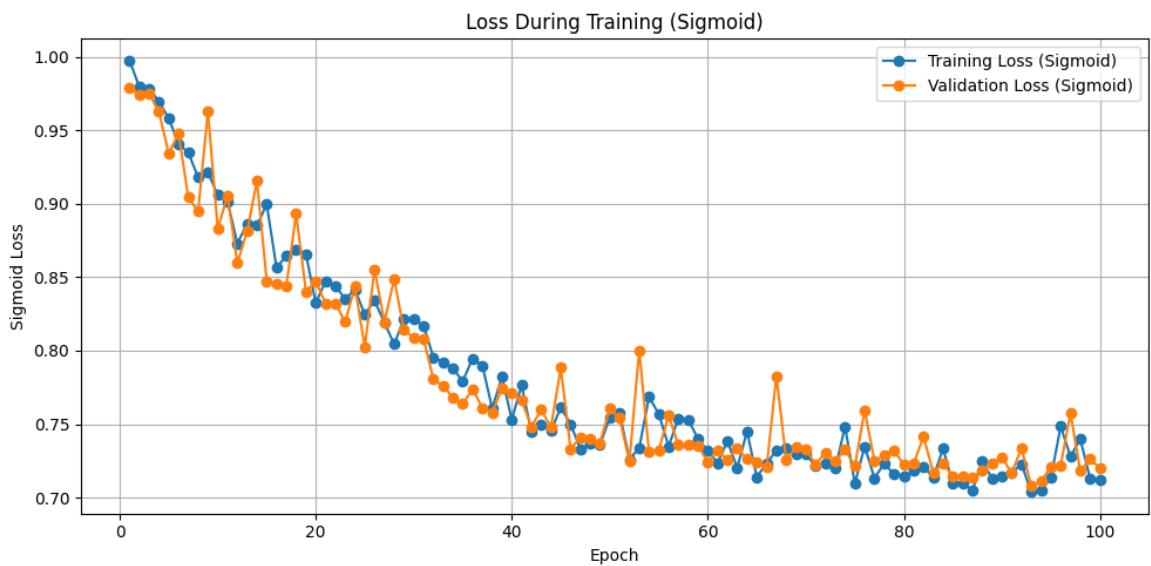
Thực hiện luyện mô hình với các điều kiện khác nhau để xem kết quả. Trong đề án này tôi sẽ thực hiện thay đổi với 2 biến là `n_rays` (số tia phát ra từ tâm) và `epochs` (số vong huấn luyện). Cả 2 biến này đều tỷ lệ thuận với độ chính xác của mô hình sau khi huấn luyện.

Trong quá trình tôi đã sử dụng hàm binary cross-entropy làm hàm loss cho quá trình huấn luyện. Sau khi huấn luyện tôi đã thống kê dữ liệu sau mỗi epoch và chuyển kết quả về hàm sigmoid để tiện quan sát. Dưới đây là kết quả



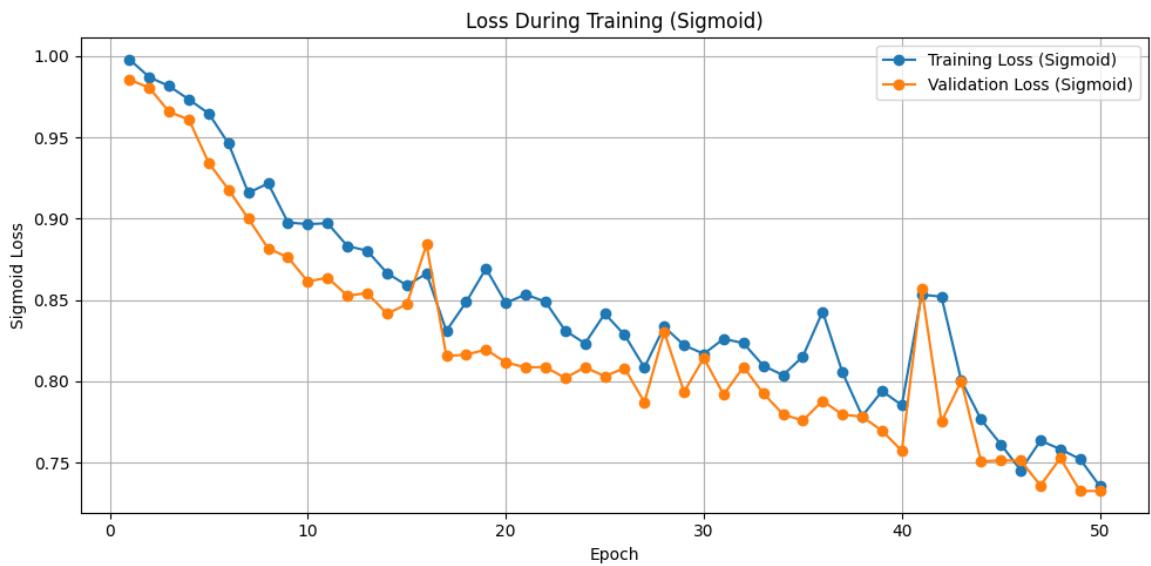
*Hình 3.10 Kết quả Loss sau quá trình huấn luyện với*

*n\_rays = 32; epochs = 20*



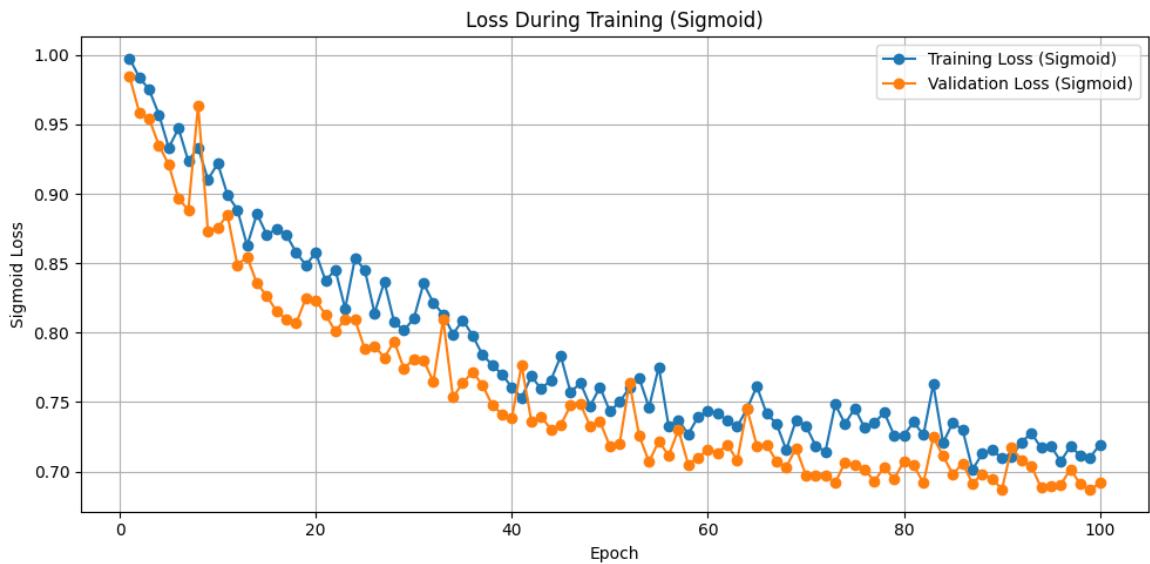
*Hình 3.11 Kết quả Loss sau quá trình huấn luyện với*

*n\_rays = 32; epochs = 100*



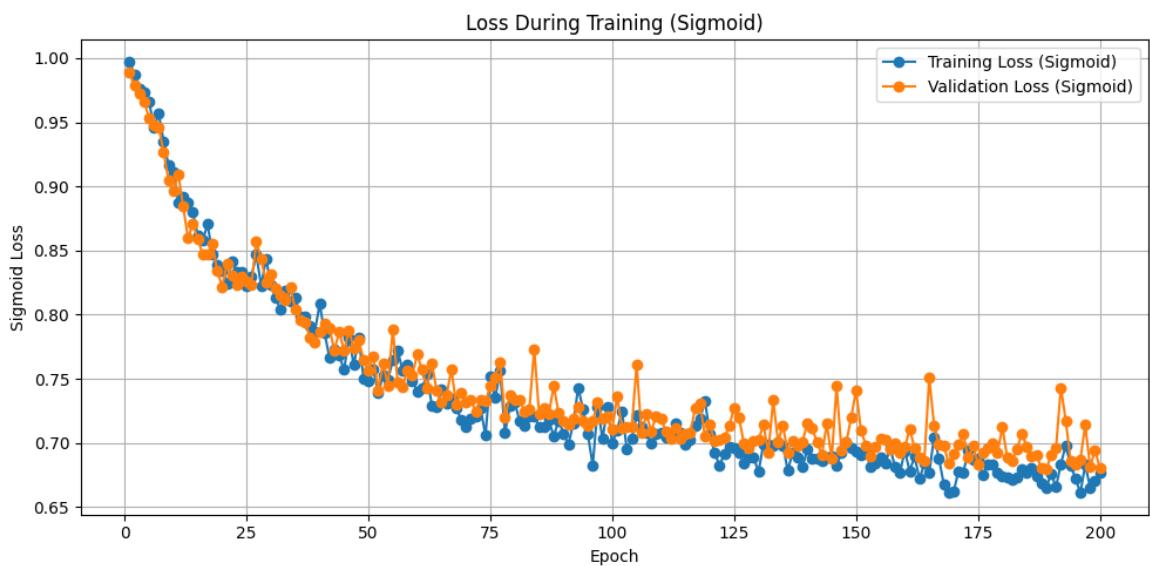
*Hình 3.12 Kết quả Loss sau quá trình huấn luyện với*

*n\_rays = 64; epochs = 50*



**Hình 3.13 Kết quả Loss sau quá trình huấn luyện với**

**$n\_rays = 64; epochs = 100$**

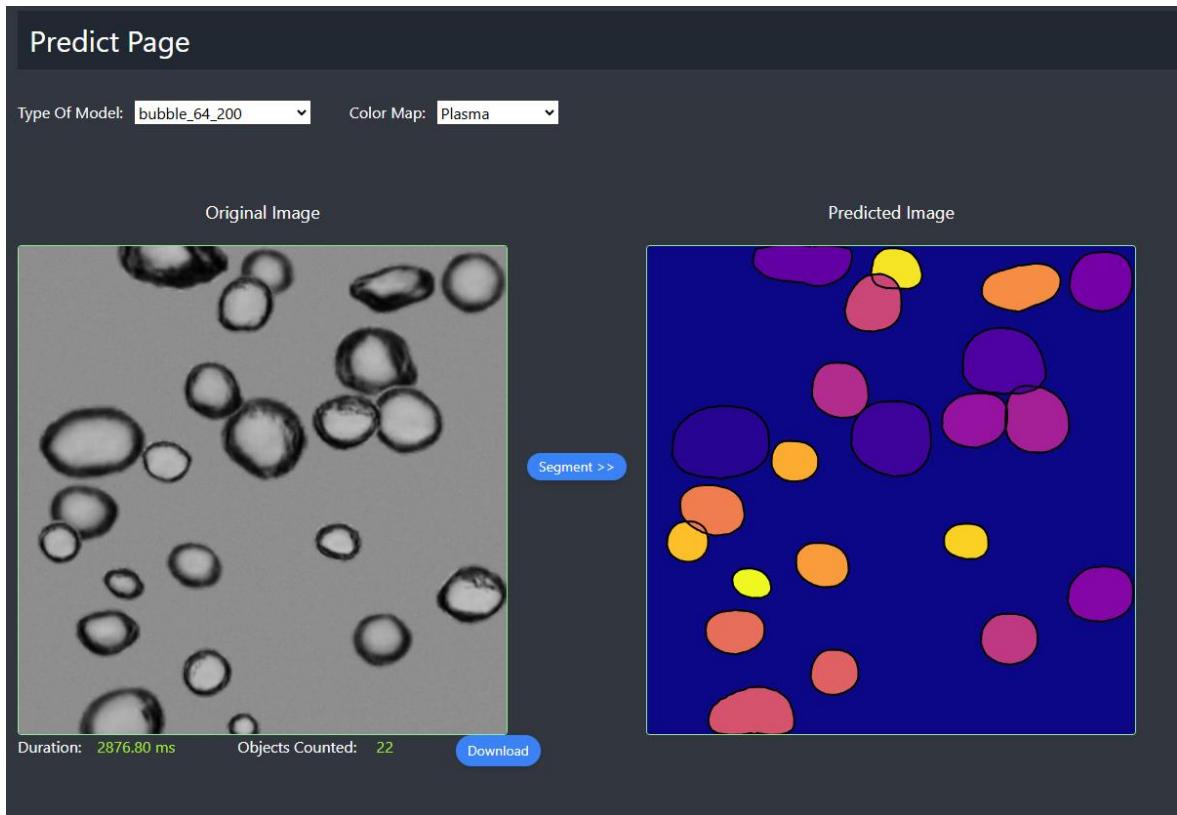


**Hình 3.14 Kết quả Loss sau quá trình huấn luyện với**

**$n\_rays = 64; epochs = 200$**

Dựa vào kết quả ta có thể thấy sau mỗi vòng huấn luyện cả *training loss* và *validation loss* đều giảm dần qua các epoch. Điều này cho thấy mô hình đang học tốt và đang tối ưu hóa được việc phân loại.

### 3.4.2. Thử nghiệm phân vùng hình ảnh



**Hình 3.15 Kết quả phân vùng thử nghiệm**

Kết quả thử nghiệm có thể thấy rằng mô hình hoạt động khá tốt kêt cả với hình ảnh có vật thể đang chồng lấp lên nhau.

## 3.5. Kết luận Chương 3

Chương này đã tập trung vào quá trình xây dựng ứng dụng cho những nghiên cứu lý thuyết ở các chương trước đó. Trong nội dung chương đã giới thiệu mô hình ứng dụng, cách chuẩn hóa dữ liệu, cách tạo ra ảnh mask từ ảnh gốc đã có trước đó, hướng dẫn viết mã code cho các phương pháp Otsu, Canny, HOG. Trong chương cũng đã hướng dẫn và viết mã code mẫu cho phần huấn luyện mô hình từ ảnh thô và ảnh mask đã có trước đó, và cũng có hướng dẫn cho phần phân vùng ảnh bằng các mô hình đã được huấn luyện. Ứng dụng không chỉ chỉ dùng lại ở việc huấn

luyện và phân vùng bọt khí trong chất lỏng mà còn có thể được dùng để phân vùng tế bào hoặc các dòng xu hoặc bất kể dữ liệu nào có tính chất tương tự.

## KẾT LUẬN

Trong đề tài "Ứng dụng các phương pháp học máy để phân vùng ảnh và xây dựng ứng dụng xác định bọt khí trong chất lỏng", chúng tôi đã nhận thức rõ tầm quan trọng của việc kết hợp các kỹ thuật học máy tiên tiến với các bài toán thực tế, đặc biệt là trong việc phân vùng ảnh và nhận diện bọt khí trong chất lỏng. Nghiên cứu này không chỉ giải quyết một vấn đề cụ thể mà còn mở rộng ứng dụng của các phương pháp học máy trong các lĩnh vực khoa học và công nghệ khác nhau. Qua việc phát triển và ứng dụng các mô hình học sâu trong phân tích ảnh, đề tài này góp phần làm rõ hiệu quả của việc ứng dụng các phương pháp học máy trong việc tối ưu hóa quá trình phát hiện và phân tích bọt khí – một yếu tố quan trọng trong nhiều quy trình công nghiệp.

Không chỉ giới hạn trong lĩnh vực nghiên cứu, kết quả của đề tài còn mang lại những ứng dụng thực tiễn có thể cải thiện đáng kể quy trình sản xuất trong ngành công nghiệp, giúp nâng cao hiệu quả và độ chính xác trong việc giám sát và kiểm soát chất lượng sản phẩm. Hơn nữa, việc kết hợp nghiên cứu và ứng dụng thực tế tạo ra cơ hội thúc đẩy sự đổi mới và hợp tác giữa các tổ chức nghiên cứu và các doanh nghiệp, mở ra các cơ hội triển khai ứng dụng thực tiễn và thương mại hóa kết quả nghiên cứu.

Ngoài ra, những kết quả và ứng dụng của đề tài này có thể lan tỏa và áp dụng trong nhiều lĩnh vực khác như thể thao, y tế và an ninh. Cụ thể, trong y tế, các phương pháp phân vùng ảnh có thể được áp dụng trong việc phân tích hình ảnh y tế, giúp phát hiện các vấn đề như khối u hoặc tổn thương mô. Trong thể thao, việc phân tích và nhận diện các hình ảnh chuyển động có thể hỗ trợ huấn luyện viên và vận động viên trong việc cải thiện kỹ năng thi đấu. Còn trong lĩnh vực an ninh, các kỹ thuật phân tích ảnh có thể giúp trong việc giám sát và bảo vệ các khu vực quan trọng. Do đó, nghiên cứu này không chỉ góp phần thúc đẩy sự phát triển của ngành công nghiệp mà

còn tạo cơ hội ứng dụng rộng rãi trong các lĩnh vực quan trọng khác, mang lại giá trị thực tiễn cho xã hội và cộng đồng.

## **TÀI LIỆU THAM KHẢO**

- [1] Cell Segmentation: 50 Years Down the Road. Erik Meijering (2012).
- [2] Bubble identification from images with machine learning methods - arXiv preprint arXiv:2202.03107 - H. Hessenkemper, S. Starke, Y. Atassi, T. Ziegenhein, D. Lucas (2022).
- [3] Cell Detection with Star-Convex Polygons - Medical Image Analysis - Uwe Schmidt, Martin Weigert, Coleman Broaddus, Gene Myers (2019)
- [4] Life Sciences Solutions Guide. Machine vision and AI-based solutions for the toughest applications – Google Cloud Platform.
- [5] Food And Beverage Solutions Guide. Safeguard Inspections, Improve Quality, and Protect Your Brand - Siemens Digital Industries Software
- [6] Image Processing with ImageJ (second edition) - Jurjen Broeke, Jose Maria Mateos Perez, and Javirer Pascau (November 2015)
- [7] Digital Image Processing (fourth edition). Rafael C.Gonzalez and Richard E. Woods (2022)
- [8] Versatile (fluorescent nuclei) and DSB 2018: Trained on a subset of the DSB 2018 nuclei segmentation challenge dataset (from StarDist 2D paper).
- [9] Versatile (H&E nuclei): Trained on images from the MoNuSeg 2018 training data and the TNBC dataset from Naylor et al. (2018).
- [10] Cheatsheet for Convolutional Neural Networks - Stanford. Available at: <https://stanford.edu/~shervine/lvi/teaching/cs-230/cheatsheet-convolutional-neural-networks>

## BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung luận văn qua phần mềm “**Kiểm Tra Tài Liệu**” một cách trung thực và đạt kết quả mức độ tương đồng không quá **11%** toàn bộ nội dung luận văn. Luận văn này sau khi đã kiểm tra qua phần mềm và bản cứng luận văn đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

Hà Nội, ngày      tháng      năm 20##

**Tác giả luận văn**

**Tên học viên**



## BÁO CÁO KIỂM TRA TRÙNG LẶP

### Thông tin tài liệu

Tên tài liệu:	Xây dựng ứng dụng quản lý sinh viên tại Viện công nghệ thông tin và Truyền thông Lào
Tác giả:	Khamstone Keovilay
Điểm trùng lặp:	11
Thời gian tải lên:	15:03 24/05/2023
Thời gian sinh báo cáo:	15:21 24/05/2023
Các trang kiểm tra:	71/71 trang



### Kết quả kiểm tra trùng lặp



### Nguồn trùng lặp tiêu biểu

123docz.net tailieu.vn

HỌC VIÊN  
(Ký và ghi rõ họ tên)

NGƯỜI HƯỚNG DẪN KHOA HỌC  
(Ký và ghi rõ họ tên)