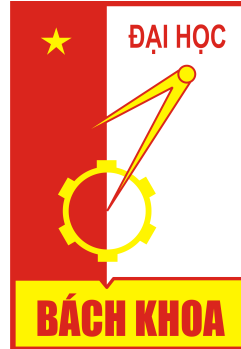


ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN



**PHÂN ĐOẠN ẢNH Y HỌC VÀ ỨNG
DỤNG TRONG PHÁT HIỆN KHỐI U VÚ**

ĐỒ ÁN II

Chuyên ngành: Toán Tin

Chuyên sâu: Toán ứng dụng

Giảng viên hướng dẫn: TS. Ngô Quốc Hoàn

Sinh viên thực hiện: Đỗ Trung Quân

Mã sinh viên: 20216873

Lớp: Toán Tin 02 K66

HÀ NỘI – 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đề án

(a) Mục tiêu:

(b) Nội dung:

2. Kết quả đạt được

3. Ý thức làm việc của sinh viên:

Hà Nội, ngày ... tháng ... năm 2025

Giảng viên hướng dẫn

Ngô Quốc Hoàn

Danh sách đánh giá đồ án

Danh sách tài liệu

Tìm tài liệu

Sắp xếp theo:

Tên tài liệu

Loại tài liệuNgày tạo

Trống

Lời cảm ơn

Trước tiên, em xin gửi lời tri ân sâu sắc đến TS. Ngô Quốc Hoàn – người đã tận tình hướng dẫn, đọc và đưa ra những ý kiến đóng góp quý báu cho em trong quá trình thực hiện và hoàn thiện đề án này. Sự tận tâm, nhiệt huyết và những lời khuyên thiết thực của thầy đã giúp em vượt qua nhiều khó khăn và có được những kiến thức chuyên sâu trong lĩnh vực nghiên cứu. Không chỉ là một người hướng dẫn, thầy còn là nguồn động lực lớn để em cố gắng hoàn thành tốt nhiệm vụ của mình.

Em cũng xin bày tỏ lòng biết ơn chân thành đến các thầy cô trong khoa Toán - Tin, Đại học Bách khoa Hà Nội. Những kiến thức nền tảng mà thầy cô đã giảng dạy suốt những năm học qua là hành trang vững chắc, giúp em có đủ tự tin và khả năng để tiến hành nghiên cứu cũng như hoàn thành đề án này. Những bài giảng tâm huyết của các thầy cô đã truyền cảm hứng và thôi thúc em không ngừng học hỏi, khám phá tri thức mới.

Trong quá trình nghiên cứu đề án và trình bày báo cáo, có thể em sẽ không thể tránh khỏi những sai sót và hạn chế về mặt kiến thức chuyên môn, cách diễn đạt, ... Em rất mong nhận được những nhận xét và đóng góp của các thầy cô và ý kiến của các bạn đọc để em có thể hoàn thiện đề án và bản báo cáo hơn.

Hà Nội, tháng 1 năm 2025

Tác giả đề án

Đỗ Trung Quân

Tóm tắt nội dung Đề án

Đề án "Phân đoạn ảnh y học và ứng dụng trong phát hiện khối u vú" trình bày việc thiết kế và triển khai một hệ thống sử dụng các kỹ thuật xử lý ảnh tiên tiến để hỗ trợ phát hiện khối u vú trong y học. Đề án phân tích bài toán phân đoạn ảnh y học và ứng dụng trong phát hiện khối u vú, bao gồm các bước chuẩn bị dữ liệu, xây dựng mô hình U-Net, và áp dụng thuật toán Otsu trong giai đoạn hậu xử lý để cải thiện chất lượng phân đoạn. Tiếp đó, đề án mô tả quá trình xây dựng hệ thống, bao gồm cài đặt môi trường phát triển, thu thập và chuẩn bị dữ liệu, huấn luyện mô hình, triển khai thuật toán Otsu, và phân tích kết quả kiểm thử để đánh giá hiệu suất hệ thống. Cuối cùng, đề án đưa ra các đề xuất cải thiện nhằm nâng cao chất lượng phân đoạn và hiệu quả ứng dụng trong thực tế.

Mục lục

Bảng ký hiệu và chữ viết tắt	1
Danh sách hình vẽ	2
Chương 1 Giới thiệu	3
1.1 Tổng quan về đề tài nghiên cứu	3
1.1.1 Đặt vấn đề	3
1.1.2 Lý do chọn đề tài	4
1.1.3 Phát biểu bài toán	5
1.1.4 Xây dựng mục tiêu	5
1.1.5 Đối tượng nghiên cứu và phương pháp nghiên cứu	6
1.1.6 Cấu trúc đề tài	6
1.2 Tổng quan về xử lý ảnh	7
1.2.1 Xử lý ảnh và vai trò của xử lý ảnh	7
1.2.2 Lịch sử phát triển của xử lý ảnh	8
1.2.3 Quá trình xử lý ảnh	8
1.3 Tổng quan về phân đoạn ảnh và ứng dụng trong lĩnh vực y học	10
1.3.1 Những khó khăn chính trong phân đoạn ảnh	10
1.3.2 Các phương pháp phân đoạn ảnh	11
Chương 2 Cơ sở lý thuyết	15
2.1 Một số khái niệm cơ bản	15
2.1.1 Ảnh và điểm ảnh	15
2.1.2 Đặc trưng của điểm ảnh	15
2.1.3 Quan hệ giữa các điểm ảnh	16
2.1.4 Lược đồ mức xám	18

2.1.5	Ngưỡng của ảnh	19
2.1.6	Độ phân giải	19
2.2	Tích chập	19
2.2.1	Khái niệm	20
2.2.2	Thành phần của phép tính tích chập	20
2.2.3	Quy trình tính tích chập	23
2.2.4	Ví dụ	25
2.2.5	Các kỹ thuật trích xuất đặc trưng hỗ trợ tích chập	27
2.3	Tích chập chuyển vị	31
2.3.1	Khái niệm	31
2.3.2	Thành phần của phép tính tích chập chuyển vị	31
2.3.3	Quy trình tính tích chập chuyển vị	31
2.3.4	Ví dụ	34
2.4	Mạng U-Net	34
2.4.1	Giới thiệu về mạng U-Net	34
2.4.2	Các lớp xử lý trong mạng U-Net	35
2.4.3	Kiến trúc mạng U-Net	39
2.5	Thuật toán Otsu	41
2.5.1	Đầu vào	42
2.5.2	Thuật toán	42
2.5.3	Đầu ra	43
Chương 3 Cài đặt, kiểm thử và đánh giá hệ thống		44
3.1	Thiết lập môi trường phát triển	44
3.1.1	Môi trường chạy	44
3.1.2	Thư viện sử dụng	44
3.2	Chuẩn bị dữ liệu	46
3.2.1	Mô tả bộ dữ liệu	46
3.2.2	Hàm đọc ảnh X-quang và ảnh phân đoạn	46
3.2.3	Hàm hiển thị 5 mẫu dữ liệu ngẫu nhiên	48

3.2.4	Chia dữ liệu thành tập huấn luyện và tập kiểm tra	50
3.3	Xây dựng mô hình kiến trúc U-Net	51
3.4	Huấn luyện mô hình kiến trúc U-Net	53
3.4.1	Biên dịch mô hình	53
3.4.2	Thiết lập Callback	54
3.4.3	Quá trình huấn luyện mô hình	54
3.5	Hậu xử lý ảnh đầu ra của mô hình U-Net bằng thuật toán Otsu .	57
3.6	Các tiêu chí đánh giá	58
3.6.1	Chất lượng phân đoạn	58
3.6.2	Thời gian phân đoạn	60
3.7	Kiểm thử	62
3.7.1	Thực hiện phân đoạn ảnh theo mô hình U-Net	62
3.7.2	Kết quả kiểm thử	65
3.7.3	Đánh giá kết quả	67
	Kết luận	68
	Chỉ mục	70
	Tài liệu tham khảo	72

Bảng ký hiệu và chữ viết tắt

4K 4 Kilopixels

8K 8 Kilopixels

acc Accuracy

CT Computed Tomography

DL Deep Learning

EM Expectation-Maximization

HD High Definition

IoU Intersection over Union

ISBI International Symposium on Biomedical Imaging

MRF Markov Random Field

MRI Magnetic Resonance Imaging

mse Mean Squared Error

ReLU Rectified Linear Unit

RGB Red, Green, Blue

Danh sách hình vẽ

1.1	Quá trình xử lý ảnh [3]	9
1.2	Các phương pháp phân đoạn ảnh [2]	11
2.1	Các lân cận của điểm ảnh có tọa độ (x, y) [3]	16
2.2	Lược đồ mức xám	18
2.3	Ma trận ảnh đầu vào	21
2.4	Max Pooling	30
2.5	Tích chập chuyển vị	34
2.6	Kiến trúc mạng U-Net [7]	39
3.1	Kết quả huấn luyện mô hình	56
3.2	Kết quả phân đoạn mỗi ảnh	65
3.3	Kết quả tổng hợp	66

Chương 1

Giới thiệu

1.1 Tổng quan về đề tài nghiên cứu

1.1.1 Đặt vấn đề

Ung thư vú hiện đang là căn bệnh ung thư phổ biến nhất ở phụ nữ trên toàn cầu, chiếm khoảng 24% tổng số ca ung thư và là nguyên nhân chính gây tử vong do ung thư ở nữ giới. Theo Tổ chức Y tế Thế giới, ước tính có hơn 2,3 triệu ca mắc ung thư vú mới và gần 685.000 ca tử vong mỗi năm do căn bệnh này [9]. Tỷ lệ mắc ung thư vú đang gia tăng, đặc biệt ở các quốc gia có nền y tế phát triển, nơi tỷ lệ sống sót cao nhờ vào việc phát hiện sớm và điều trị hiệu quả. Tuy nhiên, một vấn đề quan trọng vẫn tồn tại: việc phát hiện sớm khối u vú chưa được thực hiện đồng đều ở tất cả các quốc gia và đối tượng bệnh nhân. Các yếu tố như sự thiếu thốn trong cơ sở hạ tầng y tế, thiếu kiến thức về tầm quan trọng của việc tầm soát, và sự khó khăn trong việc phát hiện những khối u nhỏ, không có triệu chứng rõ ràng vẫn khiến tỷ lệ phát hiện ung thư vú ở giai đoạn muộn cao, ảnh hưởng đến khả năng điều trị và tiên lượng bệnh.

Một nghiên cứu từ The Lancet chỉ ra rằng tỷ lệ sống sót của bệnh nhân ung thư vú đã tăng đáng kể ở các quốc gia có chương trình tầm soát ung thư vú hiệu quả. Tuy nhiên, ở các nước có hệ thống y tế kém phát triển, tỷ lệ phát hiện ung thư vú ở giai đoạn muộn vẫn ở mức cao, điều này làm giảm cơ hội điều trị và tăng chi phí y tế. Việc phát hiện sớm ung thư vú không chỉ giúp cứu sống bệnh

nhân mà còn giúp giảm chi phí điều trị, khi các phương pháp phẫu thuật và hóa trị có thể được thực hiện ở giai đoạn bệnh nhẹ, tránh được các phương pháp điều trị tốn kém và phức tạp hơn [10].

1.1.2 Lý do chọn đề tài

Để cải thiện tình hình này, công nghệ phân đoạn ảnh y học đã chứng tỏ vai trò quan trọng trong việc hỗ trợ bác sĩ phát hiện khối u vú từ các hình ảnh y tế như mammography, siêu âm và MRI. Quá trình phân đoạn ảnh y học là một bước quan trọng trong việc xác định và phân tích các khu vực bất thường trên hình ảnh y tế. Các công nghệ này giúp tự động phát hiện các dấu hiệu của ung thư vú và phân tích thông tin một cách nhanh chóng, chính xác, đồng thời giảm thiểu sự phụ thuộc vào các phương pháp thủ công tốn thời gian và dễ xảy ra sai sót.

Một trong những bước tiến nổi bật trong lĩnh vực này là sự áp dụng của trí tuệ nhân tạo và học sâu vào phân đoạn ảnh. Các mô hình học sâu như mạng nơ-ron tích chập và mô hình U-Net đã được sử dụng để tự động phân đoạn và phát hiện các khối u vú từ hình ảnh y tế. Điều này không chỉ giúp tăng độ chính xác mà còn giảm thời gian cần thiết để phân tích và chẩn đoán. Một nghiên cứu quan trọng từ Journal of the National Cancer Institute cho thấy việc sử dụng các mô hình học sâu để phát hiện ung thư vú từ mammography có thể giúp cải thiện độ chính xác trong chẩn đoán lên đến 7% so với phương pháp truyền thống, giúp phát hiện các khối u nhỏ, không dễ nhận thấy bằng mắt thường [11].

Ngoài ra, việc áp dụng các phương pháp phân đoạn ảnh cũng giúp giảm thiểu các sai sót do yếu tố con người, đồng thời tăng cường khả năng phát hiện sớm ung thư vú ở các giai đoạn ban đầu. Các công nghệ này còn giúp bác sĩ trong việc đưa ra quyết định điều trị nhanh chóng và chính xác, từ đó cải thiện tỷ lệ sống sót cho bệnh nhân.

Vì những lý do trên, việc nghiên cứu và ứng dụng phân đoạn ảnh y học trong phát hiện khối u vú không chỉ mang lại lợi ích thiết thực cho bệnh nhân mà còn

đóng góp quan trọng vào việc nâng cao chất lượng chăm sóc sức khỏe. Điều này đặc biệt quan trọng trong việc phòng ngừa và điều trị ung thư vú, một căn bệnh ung thư phổ biến và nguy hiểm. Chính vì vậy, em chọn đề tài này nhằm nghiên cứu và phát triển các phương pháp tiên tiến giúp cải thiện hiệu quả chẩn đoán và điều trị ung thư vú, từ đó góp phần nâng cao năng lực của hệ thống y tế toàn cầu trong việc đối phó với căn bệnh này.

1.1.3 Phát biểu bài toán

Bài toán nghiên cứu trong đề tài này là phát hiện khối u vú từ ảnh siêu âm thông qua việc phân đoạn tự động các vùng khối u trên ảnh. Cụ thể, hệ thống sẽ nhận diện và tách biệt các khu vực có khối u vú, giúp hỗ trợ quá trình chẩn đoán sớm ung thư vú.

Chi tiết bài toán như sau:

- *Đầu vào*: Ảnh siêu âm vú chứa các vùng có khối u cần phát hiện. Các ảnh này có thể có độ phân giải và chất lượng khác nhau, và chứa các thông tin về kích thước, hình dạng, và vị trí của khối u.
- *Đầu ra*: Kết quả phân đoạn nhị phân, trong đó các vùng khối u vú sẽ được đánh dấu rõ ràng bằng màu trắng trên nền đen. Các khu vực không chứa khối u sẽ được giữ lại dưới nền đen, giúp phân biệt rõ ràng giữa khối u và các vùng khác xung quanh.

1.1.4 Xây dựng mục tiêu

Mục tiêu của đề tài là xây dựng và phát triển một mô hình học sâu dựa trên kiến trúc U-Net, nhằm tự động phân đoạn và phát hiện khối u vú từ ảnh siêu âm. Các mục tiêu chính của đề tài được trình bày như sau:

- *Nghiên cứu phương pháp và thuật toán phân đoạn ảnh y học*: Tìm hiểu các phương pháp phân đoạn ảnh hiện đại, đặc biệt là các thuật toán học sâu, để lựa chọn và đánh giá các kỹ thuật phù hợp với việc phân đoạn và phát

hiện khối u vú từ ảnh siêu âm. Nghiên cứu sâu về các mô hình học sâu như U-Net, các kiến trúc mạng nơ-ron tích chập,... và cách chúng có thể được áp dụng hiệu quả trong việc phân đoạn các khối u trong ảnh y học.

- *Xây dựng và huấn luyện mô hình U-Net phân đoạn ảnh vú*: Thiết kế và triển khai mô hình học sâu U-Net để phân đoạn ảnh siêu âm, giúp phát hiện khối u vú với độ chính xác cao. Mô hình sẽ được tối ưu để phân đoạn chi tiết nhỏ và giảm thời gian xử lý, đảm bảo hiệu suất cao và khả năng ứng dụng nhanh chóng trong môi trường lâm sàng, đồng thời giảm thiểu tài nguyên tính toán và thời gian huấn luyện.

1.1.5 Đối tượng nghiên cứu và phương pháp nghiên cứu

Đối tượng nghiên cứu: Đối tượng nghiên cứu chính của đề án là các phương pháp phân đoạn ảnh y học, đặc biệt là việc phân đoạn khối u vú từ ảnh siêu âm. Mô hình U-Net sẽ là đối tượng nghiên cứu trọng tâm, cùng với bộ dữ liệu ảnh siêu âm vú có chứa khối u đã được đánh dấu.

Phương pháp nghiên cứu: Phương pháp nghiên cứu chủ yếu là sử dụng kiến trúc U-Net để phân đoạn ảnh khối u vú từ ảnh siêu âm. Mô hình sẽ được huấn luyện trên bộ dữ liệu ảnh siêu âm vú, và đánh giá kết quả thông qua các chỉ số như IoU để đo độ chính xác của phân đoạn. Các bước nghiên cứu bao gồm:

- Thu thập và chuẩn bị bộ dữ liệu ảnh siêu âm vú.
- Xây dựng mô hình U-Net và huấn luyện trên bộ dữ liệu.
- Đánh giá mô hình qua chỉ số IoU.

1.1.6 Cấu trúc đề tài

Đề tài này được tổ chức thành 3 chương chính như sau:

- *Chương 1 - Giới thiệu*: Chương này giới thiệu tổng quan về xử lý ảnh và phân đoạn ảnh trong y học, cũng như các vấn đề chính và phương pháp phân đoạn ảnh.

- *Chương 2 - Cơ sở lý thuyết*: Trình bày các khái niệm cơ bản về xử lý ảnh, các kỹ thuật như tích chập, mạng nơ-ron U-Net, và thuật toán Otsu, cung cấp nền tảng lý thuyết cho việc giải quyết bài toán phân đoạn ảnh.
- *Chương 3 - Cài đặt, kiểm thử và đánh giá hệ thống*: Mô tả quy trình cài đặt hệ thống, từ việc chuẩn bị dữ liệu, xây dựng mô hình U-Net, đến huấn luyện và triển khai thuật toán Otsu trong giai đoạn hậu xử lý ảnh. Sau đó tiến hành kiểm thử, đánh giá kết quả thử nghiệm của hệ thống, phân tích chất lượng phân đoạn và hiệu quả của hệ thống trong môi trường thực tế.

1.2 Tổng quan về xử lý ảnh

1.2.1 Xử lý ảnh và vai trò của xử lý ảnh

Xử lý ảnh là một lĩnh vực quan trọng trong khoa học máy tính, tập trung vào việc phân tích, cải thiện và trích xuất thông tin từ hình ảnh kỹ thuật số. Nó đóng vai trò then chốt trong nhiều ngành công nghiệp, bao gồm y tế, an ninh, và giải trí. Với sự phát triển mạnh mẽ của công nghệ, các phương pháp xử lý ảnh ngày càng trở nên tinh vi hơn, giúp giải quyết nhiều vấn đề phức tạp và tạo ra những ứng dụng mang lại giá trị thực tiễn cao.

Cụ thể, xử lý ảnh là quá trình thay đổi hoặc phân tích hình ảnh để cải thiện chất lượng, trích xuất thông tin hoặc tạo ra các hiệu ứng hình ảnh. Quá trình này có thể bao gồm các bước như làm sạch hình ảnh, điều chỉnh độ sáng, độ tương phản, phân đoạn hình ảnh (chia hình ảnh thành các khu vực có ý nghĩa), và nhận diện các đối tượng trong ảnh. Các kỹ thuật xử lý ảnh hiện đại còn ứng dụng trí tuệ nhân tạo và học máy để tự động hóa việc phân tích và nhận diện thông tin trong hình ảnh.

Xử lý ảnh có vai trò quan trọng trong nhiều lĩnh vực, mỗi lĩnh vực đều có những ứng dụng cụ thể. Trong y tế, xử lý ảnh giúp bác sĩ phát hiện các bất thường trong cơ thể như khối u, tổn thương mô, và các bệnh lý khác qua hình ảnh y học như X-quang, MRI hay CT. Trong an ninh, xử lý ảnh giúp nhận diện

khuôn mặt, phân tích hành vi và nhận diện vật thể. Trong ngành công nghiệp ô tô, xử lý ảnh hỗ trợ các hệ thống lái tự động nhận diện vật cản và người đi bộ. Ngoài ra, trong giải trí, xử lý ảnh còn giúp tạo ra các hiệu ứng hình ảnh ấn tượng trong phim và trò chơi điện tử,... Như vậy, xử lý ảnh không chỉ giúp cải thiện chất lượng hình ảnh mà còn là công cụ quan trọng trong việc phân tích và đưa ra quyết định trong các ngành nghề khác nhau.

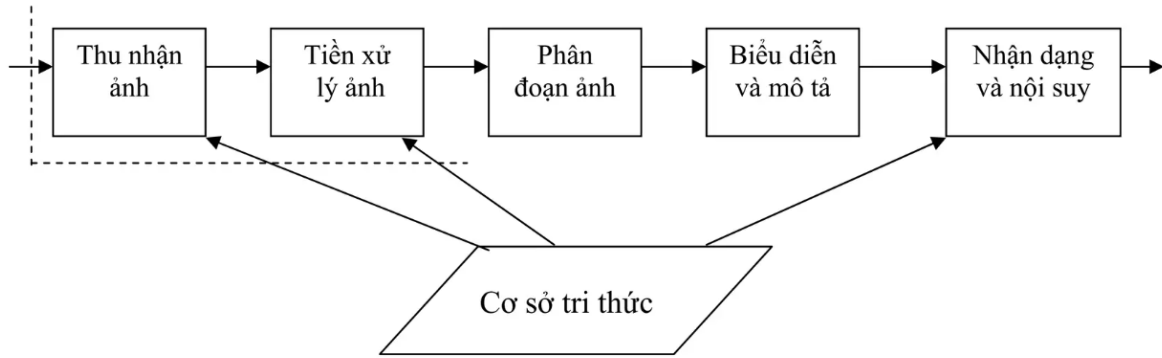
1.2.2 Lịch sử phát triển của xử lý ảnh

Các phương pháp xử lý ảnh phát triển từ những ứng dụng cơ bản như nâng cao chất lượng ảnh, phân đoạn ảnh và phân tích ảnh. Một trong những ứng dụng đầu tiên được ghi nhận là việc cải thiện chất lượng ảnh được truyền qua cáp từ Luân Đôn đến New York vào những năm 1920. Vấn đề nâng cao chất lượng ảnh liên quan đến việc điều chỉnh mức sáng và độ phân giải của ảnh. Quá trình nâng cao chất lượng ảnh bắt đầu phát triển mạnh mẽ vào những năm 1955, đặc biệt là sau Thế chiến thứ hai, khi sự phát triển nhanh chóng của máy tính đã tạo ra điều kiện thuận lợi cho việc xử lý ảnh số. Đến năm 1964, máy tính đã có khả năng xử lý và cải thiện chất lượng ảnh từ mặt trăng và vệ tinh Ranger 7 của Mỹ, bao gồm các kỹ thuật như làm nổi đường biên và lưu trữ ảnh. Kể từ năm 1964 cho đến nay, các phương tiện và kỹ thuật xử lý ảnh, từ nâng cao chất lượng, phân đoạn đến nhận dạng ảnh, đã không ngừng phát triển và tiến bộ [1].

1.2.3 Quá trình xử lý ảnh

Quá trình xử lý ảnh được hiểu là quá trình thực hiện các thao tác trên ảnh đầu vào để đạt được kết quả mong muốn. Kết quả đầu ra của quá trình này có thể là một bức ảnh với chất lượng được cải thiện hoặc một kết luận cụ thể từ ảnh đã xử lý.

Theo tài liệu [1], quá trình xử lý ảnh bao gồm các bước chính dưới đây.



Hình 1.1: Quá trình xử lý ảnh [3]

1. *Thu nhận ảnh*: Đây là bước đầu tiên quyết định đến chất lượng của toàn bộ quá trình xử lý ảnh. Ảnh đầu vào được thu nhận thông qua các thiết bị như camera, cảm biến, máy quét, v.v. Sau đó, các tín hiệu thu được từ các thiết bị này sẽ được số hóa. Các yếu tố quan trọng trong giai đoạn này bao gồm độ phân giải, chất lượng màu, dung lượng bộ nhớ và tốc độ thu nhận ảnh của các thiết bị.
2. *Tiền xử lý*: Bước này tập trung vào việc cải thiện chất lượng ảnh bằng cách nâng cao độ tương phản, khử nhiễu, giảm bóng, chỉnh sửa độ lệch, v.v. Mục tiêu của quá trình tiền xử lý là cải thiện chất lượng hình ảnh để phục vụ các bước xử lý tiếp theo. Các bộ lọc thường được sử dụng để thực hiện những thao tác này.
3. *Phân đoạn ảnh*: Phân đoạn ảnh là một bước then chốt trong quá trình xử lý ảnh. Trong bước này, ảnh được phân chia thành các thành phần riêng biệt có tính chất giống nhau, dựa trên biên hoặc các vùng liên thông. Tiêu chí để phân vùng có thể là màu sắc, mức xám hoặc độ nhám của ảnh. Quá trình phân đoạn ảnh giúp giảm lượng thông tin không cần thiết chứa trong ảnh, chỉ giữ lại các đặc tính quan trọng. Đây là giai đoạn quan trọng để giảm độ phức tạp của ảnh và dễ dàng xử lý trong các ứng dụng.
4. *Biểu diễn và mô tả ảnh*: Sau khi phân đoạn, kết quả thường ở dạng các điểm ảnh thô, thể hiện biên hoặc tập hợp các điểm ảnh trong một vùng cụ

thể. Việc chuyển đổi dữ liệu này thành một dạng thích hợp hơn cho việc xử lý máy tính là rất cần thiết. Câu hỏi quan trọng cần giải quyết là liệu ảnh nên được biểu diễn dưới dạng biên hoặc vùng hoàn chỉnh, tùy vào mục đích sử dụng. Việc chọn cách biểu diễn phù hợp sẽ quyết định đến việc xử lý và phân tích sau này.

5. *Nhận dạng và nội suy*: Đây là bước cuối cùng trong quá trình xử lý ảnh. Nhận dạng ảnh liên quan đến việc gán nhãn cho các đối tượng trong ảnh, giúp xác định và phân loại các thành phần trong ảnh. Nội suy là bước gán nghĩa cho các đối tượng đã được nhận diện, làm cho chúng có thể được sử dụng trong các ứng dụng thực tiễn.

Trong những bước trên, phân đoạn ảnh đóng vai trò then chốt, là cầu nối giữa các giai đoạn tiền xử lý và các bước phân tích nâng cao như nhận dạng và nội suy. Kết quả của phân đoạn ảnh không chỉ quyết định chất lượng của việc biểu diễn, mô tả mà còn ảnh hưởng trực tiếp đến độ chính xác trong quá trình nhận dạng. Vì vậy, việc hiểu rõ và áp dụng đúng các kỹ thuật phân đoạn phù hợp là điều kiện tiên quyết để đảm bảo hiệu quả của toàn bộ quá trình xử lý ảnh.

1.3 Tổng quan về phân đoạn ảnh và ứng dụng trong lĩnh vực y học

1.3.1 Những khó khăn chính trong phân đoạn ảnh

Như đã đề cập ở trên, phân đoạn ảnh có nhiệm vụ chia nhỏ hình ảnh thành các vùng khác biệt, mỗi vùng biểu thị một đặc điểm riêng và trích xuất các vùng quan tâm trong ảnh. Các vùng này thường có ý nghĩa rõ ràng và không bị chồng lấn lên nhau. Tuy nhiên, theo bài báo [2], phân đoạn ảnh phải đối mặt với hai khó khăn chính:

- *Xác định vùng có ý nghĩa*: Đây là việc xác định khu vực trong ảnh mà ta coi là quan trọng, như các vật thể, đường viền, hoặc vùng màu sắc cụ thể. Tuy

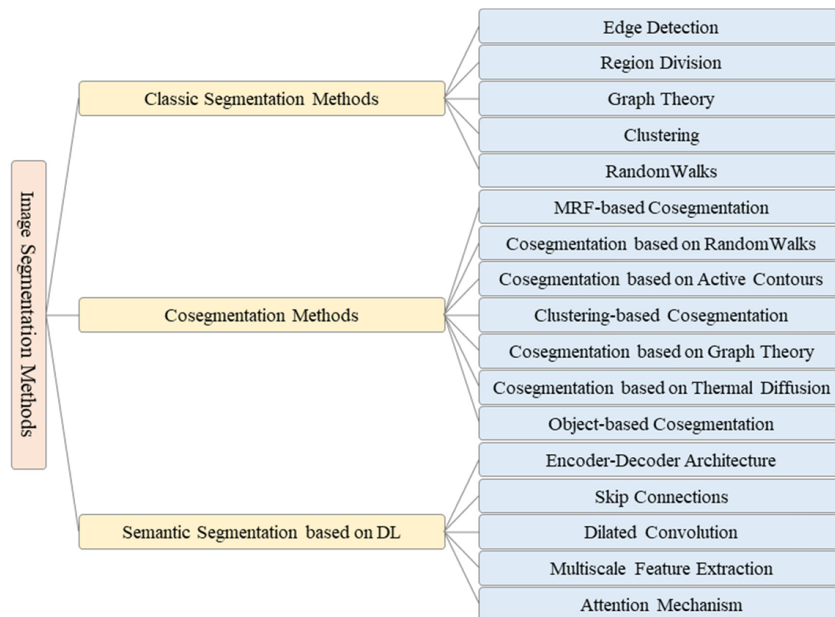
nhien, mỗi người có cách nhìn nhận khác nhau, nên không có tiêu chuẩn rõ ràng về việc vùng nào thực sự có ý nghĩa.

- *Biểu diễn các đối tượng trong ảnh*: Hình ảnh có thể được phân đoạn thành các vùng dựa trên các đặc điểm như màu sắc hoặc kết cấu. Tuy nhiên, các đặc điểm này thường chỉ cung cấp thông tin chi tiết ở mức độ nhỏ lẻ, chưa thể hiện được toàn bộ hình dạng hay vị trí tổng thể của các đối tượng trong ảnh.

Để khắc phục những khó khăn này, các nhà nghiên cứu đã phát triển nhiều phương pháp phân đoạn ảnh khác nhau. Phần tiếp theo ta sẽ đề cập về các phương pháp phân đoạn ảnh, bao gồm các cách tiếp cận cổ điển và các phương pháp tiên tiến hiện nay.

1.3.2 Các phương pháp phân đoạn ảnh

Từ những năm 1970, phân đoạn ảnh đã trở thành chủ đề thu hút sự chú ý đặc biệt của các nhà nghiên cứu trong lĩnh vực thị giác máy tính. Qua thời gian, nhiều phương pháp đã được đề xuất và phát triển nhằm giải quyết các bài toán phân đoạn với độ phức tạp ngày càng tăng [2].



Hình 1.2: Các phương pháp phân đoạn ảnh [2]

1. Phương pháp phân đoạn ảnh cổ điển (*Classic Segmentation Methods*)

- *Edge Detection*: Phát hiện biên là kỹ thuật tìm kiếm các đường ranh giới phân cách các vùng trong ảnh, chẳng hạn như đường viền của một vật thể. Đây là bước cơ bản để nhận dạng các đối tượng trong ảnh.
- *Region Division*: Phân chia vùng là kỹ thuật chia ảnh thành các vùng dựa trên các đặc điểm giống nhau như màu sắc hoặc kết cấu, giúp dễ dàng nhận diện các đối tượng trong ảnh.
- *Graph Theory*: Lý thuyết đồ thị sử dụng mô hình mạng lưới để xác định các phần tử trong ảnh có liên kết với nhau, từ đó phân loại và nhóm các vùng ảnh tương tự.
- *Clustering*: Phân cụm là kỹ thuật gom các phần của ảnh thành nhóm dựa trên các đặc điểm giống nhau như màu sắc hoặc độ sáng, giúp xác định các đối tượng trong ảnh.
- *Random Walks*: Bước ngẫu nhiên sử dụng mô phỏng chuyển động ngẫu nhiên để phân loại các vùng trong ảnh, giúp xác định các đối tượng hoặc vùng quan trọng.

2. Phương pháp đồng phân đoạn (*Cosegmentation Methods*)

- *MRF-based Cosegmentation*: Phương pháp này sử dụng mô hình toán học ngẫu nhiên để dự đoán các vùng chung trong nhiều ảnh dựa trên quy luật thống kê.
- *Cosegmentation based on Random Walks*: Dựa trên bước ngẫu nhiên để tìm các vùng tương đồng trong nhiều ảnh, nhằm phân loại và nhóm các đối tượng giống nhau.
- *Cosegmentation based on Active Contours*: Phương pháp này sử dụng các đường biên có thể điều chỉnh để tìm ra các vùng quan trọng trong ảnh, thường dùng trong phân đoạn các vật thể có biên rõ ràng.
- *Clustering-based Cosegmentation*: Phân cụm đồng phân đoạn tìm các

vùng giống nhau trong nhiều ảnh và gom nhóm chúng lại thành các phần tử quan trọng.

- *Cosegmentation based on Graph Theory*: Sử dụng lý thuyết đồ thị để xác định các vùng tương đồng giữa các ảnh thông qua các liên kết và mối quan hệ giữa các phần tử trong ảnh.
- *Cosegmentation based on Thermal Diffusion*: Phương pháp này mô phỏng quá trình khuếch tán nhiệt trong hình ảnh để xác định các vùng có đặc điểm chung.
- *Object-based Cosegmentation*: Tập trung vào việc xác định các vùng tương ứng với các đối tượng cụ thể xuất hiện trong nhiều hình ảnh.

3. Phương pháp phân đoạn dựa trên học sâu (*Semantic Segmentation Based on DL*)

- *Encoder-Decoder Architecture*: Kiến trúc mã hóa - giải mã giúp xử lý hình ảnh một cách hiệu quả, nén thông tin từ ảnh và tái tạo lại hình ảnh phân đoạn với độ chính xác cao.
- *Skip Connections*: Giúp bảo toàn thông tin chi tiết từ các tầng đầu vào của mạng học sâu, giúp mô hình phân đoạn chính xác hơn.
- *Dilated Convolution*: Tích chập giãn nở mở rộng phạm vi quan sát của mạng, giúp mô hình phân tích ảnh ở nhiều mức độ chi tiết mà không làm mất thông tin.
- *Multiscale Feature Extraction*: Trích xuất đặc trưng đa tỉ lệ giúp mạng học sâu nhận diện các chi tiết từ nhiều mức độ khác nhau, tăng cường khả năng phân đoạn.
- *Attention Mechanism*: Cơ chế tập trung giúp mô hình tập trung vào các phần quan trọng của hình ảnh, từ đó phân đoạn chính xác hơn và giảm thiểu sai sót.

Như vậy qua thời gian, nhiều phương pháp đã được đề xuất và phát triển nhằm giải quyết các bài toán phân đoạn với độ phức tạp ngày càng tăng.

Hình dưới đây cung cấp cái nhìn tổng quan về hệ thống các phương pháp phân đoạn ảnh từ truyền thống đến hiện đại, cho thấy sự tiến bộ vượt bậc trong việc tiếp cận bài toán này.

Chương 2

Cơ sở lý thuyết

2.1 Một số khái niệm cơ bản

2.1.1 Ảnh và điểm ảnh

Ảnh tự nhiên là ảnh liên tục về không gian và độ sáng. *Số hóa ảnh* là sự biến đổi gần đúng một ảnh liên tục thành một tập điểm phù hợp với ảnh thật về vị trí và độ sáng. Tập hợp các điểm này được gọi là *ảnh số*. Mỗi điểm này được gọi là *điểm ảnh (pixel)* [3].

2.1.2 Đặc trưng của điểm ảnh

Một điểm ảnh có hai đặc trưng cơ bản là *vị trí* và *mức xám* [3].

1. *Vị trí*: Trong ảnh hai chiều, vị trí của một điểm ảnh được xác định bởi một cặp tọa độ (x, y) .
2. *Mức xám*: Mức xám là giá trị biểu thị cường độ sáng của một điểm ảnh, từ 0 đến 255. Cụ thể, mức xám trong các loại ảnh số như sau:
 - Ảnh xám n -bit: Mỗi điểm ảnh được mã hóa bởi n bit và có mức xám nhận giá trị từ 0 đến $2^n - 1$. Một ví dụ thông dụng là ảnh xám 8-bit, khi đó các điểm ảnh sẽ có 256 mức xám, từ 0 (đen) đến 255 (trắng). Đặc biệt, khi $n=1$, ảnh được gọi là *ảnh nhị phân*. Mỗi điểm ảnh của ảnh nhị phân chỉ có 2 mức xám: 0 hoặc 1. Trên máy tính, ảnh nhị phân thường được biểu diễn dưới dạng ảnh xám 8-bit, trong đó các điểm ảnh

có giá trị 0 được giữ nguyên ở mức 0 (đen), các điểm ảnh có giá trị 1 được ánh xạ sang giá trị 255 (trắng).

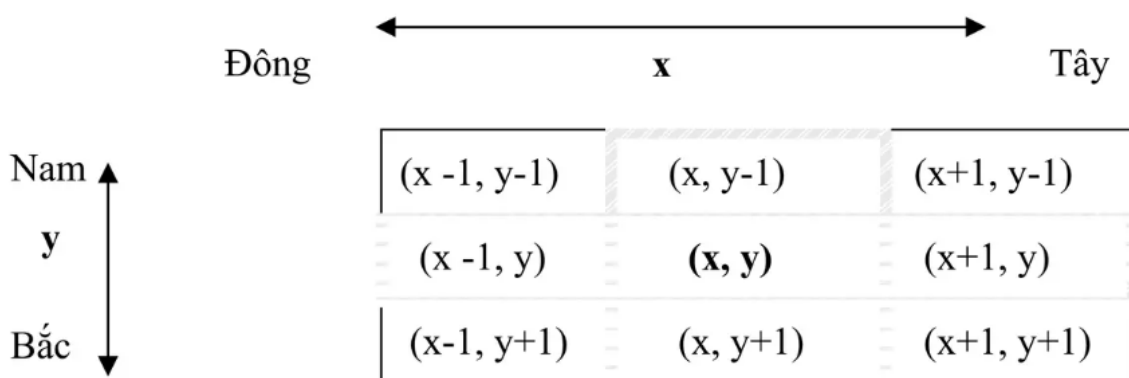
- **Ảnh màu:** Mỗi điểm ảnh được mã hóa bởi ba thành phần màu (RGB: Đỏ, Lục, Lam), mỗi thành phần được mã hóa bởi n bit để biểu diễn mức độ sáng màu tương ứng, từ 0 đến $2^n - 1$. Một ví dụ thông dụng là ảnh màu 24-bit (RGB 8 bit mỗi kênh), trong đó mỗi điểm ảnh có thể biểu diễn $2^{24} = 16777216$ màu, với giá trị của từng thành phần màu nằm trong khoảng từ 0 (không sáng) đến 255 (sáng tối đa). Chẳng hạn, điểm ảnh màu đỏ tươi sẽ được biểu diễn bởi 3 thành phần (255, 0, 0), màu xanh dương là (0, 0, 255), màu xanh lá cây là (0, 255, 0)... Khi đó, theo chuẩn BT.601 được trình bày trong tài liệu [4], mức xám của một điểm ảnh thường được tính bằng công thức:

$$Y = \frac{77}{256}R + \frac{150}{256}G + \frac{29}{256}B.$$

2.1.3 Quan hệ giữa các điểm ảnh

Quan hệ giữa các điểm ảnh bao gồm các khía cạnh như lân cận của điểm ảnh, các mối liên kết giữa các điểm ảnh và khoảng cách giữa chúng [3].

1. Các lân cận của điểm ảnh



Hình 2.1: Các lân cận của điểm ảnh có tọa độ (x, y) [3]

Xét điểm ảnh p có tọa độ (x, y) . Khi đó tập 4 điểm lân cận theo chiều đứng

và ngang của p là:

$$N_4(p) = \{(x-1, y); (x, y-1); (x, y+1); (x+1, y)\}.$$

Tập 4 điểm lân cận chéo của p là:

$$N_P(p) = \{(x+1, y+1); (x+1, y-1); (x-1, y+1); (x-1, y-1)\}.$$

Tập 8 điểm lân cận của điểm ảnh p là:

$$N_8(p) = N_4(p) \cup N_P(p).$$

Chú ý: Nếu điểm ảnh (x, y) nằm ở mép ảnh, một số điểm lân cận sẽ nằm ngoài ảnh.

2. Các mối liên kết điểm ảnh

Một *mối liên kết* giữa các điểm ảnh được đặc trưng bởi tính liên kề giữa các điểm và mức xám của chúng. Các mối liên kết giữa các điểm ảnh được sử dụng để xác định giới hạn của đối tượng hoặc vùng trong ảnh.

Giả sử V là tập các giá trị mức xám. Xét 2 điểm ảnh p và q có cùng mức xám trong V , khi đó giữa chúng có 3 loại mối liên kết:

- Liên kết 4: p và q được gọi là có liên kết 4 với giá trị cường độ sáng V nếu $q \in N_4(p)$.
- Liên kết 8: p và q được gọi là có liên kết 8 với giá trị cường độ sáng V nếu $q \in N_8(p)$.
- Liên kết m (liên kết hỗn hợp): p và q được gọi là có liên kết m với giá trị cường độ sáng V nếu $q \in N_4(p)$ hoặc $q \in N_P(p)$.

3. Khoảng cách giữa các điểm ảnh

Hàm $D(p, q)$ được gọi là *khoảng cách* giữa 2 điểm ảnh p có tọa độ (x, y) và q có tọa độ (s, t) nếu thỏa mãn:

- $D(p, q) \geq 0$. $D(p, q) = 0 \Leftrightarrow p = q$ (p, q trùng nhau).
- $D(p, q) = D(q, p)$.

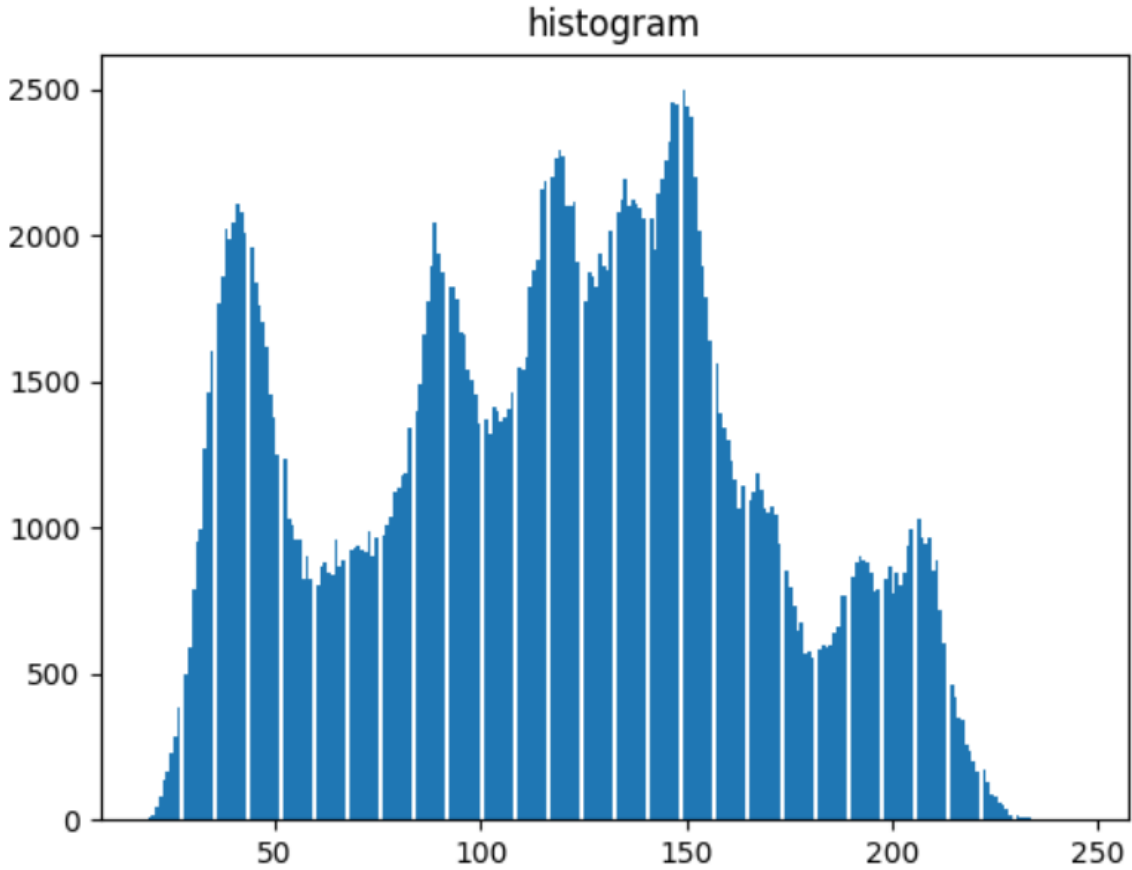
- $D(p, z) \leq D(p, q) + D(q, z)$. Trong đó z là điểm ảnh bất kì.

Ta thường quan tâm đến các loại khoảng cách sau:

- Khoảng cách Euclide: $D_e(p, q) = \sqrt{(x - s)^2 + (y - t)^2}$.
- Khoảng cách khối: $D_4(p, q) = |x - s| + |y - t|$.
- Khoảng cách bàn cờ: $D_\delta(p, q) = \max\{|x - s|, |y - t|\}$.

2.1.4 Lược đồ mức xám

Lược đồ mức xám là một biểu đồ biểu diễn phân bố tần suất các mức xám trong một ảnh xám. Mỗi mức xám (từ 0 đến $2^n - 1$ với n là số bit biểu diễn mức xám) được biểu diễn trên trục hoành, và số lượng điểm ảnh có mức xám tương ứng được biểu diễn trên trục tung.



Hình 2.2: Lược đồ mức xám

2.1.5 Ngưỡng của ảnh

Ngưỡng của ảnh là một giá trị cố định (hoặc thay đổi theo vùng), được sử dụng để so sánh với giá trị mức xám của các điểm ảnh trong ảnh. Dựa trên kết quả so sánh này, các điểm ảnh sẽ được phân thành hai hoặc nhiều nhóm khác nhau. Quá trình phân nhóm này chính là phân đoạn ảnh dựa trên ngưỡng - một kỹ thuật phân đoạn ảnh trong nhóm phương pháp phân vùng ảnh (Region Division) [1].

Giả sử rằng trong một ảnh xám 8-bit, chúng ta quan tâm các đối tượng sáng trên nền tối. Cụ thể, ta muốn sau khi phân đoạn ảnh, đối tượng sẽ có màu trắng (mức xám 255), nền sẽ có màu đen (mức xám 0). Gọi $f(x, y)$ là mức xám của điểm ảnh có tọa độ (x, y) trước khi phân đoạn, $g(x, y)$ là mức xám của điểm ảnh có tọa độ (x, y) sau khi phân đoạn. T là ngưỡng của ảnh dùng để phân đoạn. Khi đó:

$$g(x, y) = \begin{cases} 255 & \text{nếu } f(x, y) \geq T \text{ (điểm ảnh thuộc đối tượng sáng)} \\ 0 & \text{nếu } f(x, y) < T \text{ (điểm ảnh thuộc nền tối)} \end{cases}$$

2.1.6 Độ phân giải

Độ phân giải là thước đo độ chi tiết của một hình ảnh, thể hiện số lượng pixel (các điểm ảnh nhỏ nhất) mà một hình ảnh có thể chứa được. Độ phân giải thường được biểu thị bằng chiều rộng x chiều cao (ví dụ: 1920 x 1080 pixel) hoặc bằng thuật ngữ như HD, Full HD, 4K, 8K [3].

Cụ thể, một bức ảnh có độ phân giải 640 x 480 pixel nghĩa là:

- Có 640 pixel theo chiều ngang và 480 pixel theo chiều dọc.
- Tổng cộng có $640 \times 480 = 307.200$ pixel trong bức ảnh.

2.2 Tích chập

Trong xử lý ảnh và thị giác máy tính, việc phân tích và hiểu nội dung của ảnh là một nhiệm vụ rất quan trọng. Tuy nhiên, ảnh dưới dạng dữ liệu thô thường

chứa rất nhiều thông tin dư thừa hoặc không cần thiết, khiến máy tính khó có thể phân tích trực tiếp một cách hiệu quả. Do đó, thay vì xử lý toàn bộ dữ liệu thô của ảnh, cần tập trung vào các chi tiết quan trọng, được gọi là *đặc trưng của ảnh*. Những đặc trưng này phản ánh các yếu tố cốt lõi như cạnh, góc, họa tiết, hay hình dạng trong ảnh, giúp hệ thống phân tích hiệu quả và chính xác hơn. Để trích xuất các đặc trưng này, một phương pháp phổ biến và hiệu quả là phép *tích chập*. Trong nội dung này, chúng ta sẽ tập trung vào trường hợp áp dụng tích chập trên ảnh xám 8-bit (mức xám của các điểm ảnh từ 0 đến 255).

2.2.1 Khái niệm

Tích chập (convolution) là một phép toán dùng để trích xuất đặc trưng quan trọng của ảnh đầu vào như cạnh, góc, hoặc các mẫu hình học trong ảnh. Cụ thể, phép tích chập được thực hiện trên ảnh xám bằng cách áp dụng một *bộ lọc (kernel)* lên *ma trận ảnh đầu vào (image matrix)* để tạo ra *ma trận đặc trưng đầu ra (feature map)* [5].

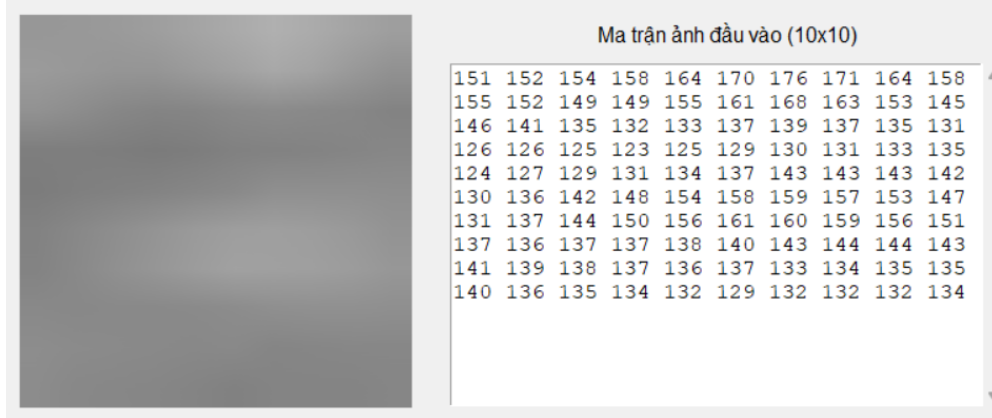
2.2.2 Thành phần của phép tính tích chập

1. Ma trận ảnh đầu vào

Ma trận ảnh đầu vào là ma trận I biểu diễn các điểm ảnh trong ảnh gốc, được xác định như sau:

- **Kích thước ma trận ảnh:** Ma trận ảnh đầu vào có kích thước $M \times N$, trong đó M là số hàng, đại diện cho số điểm ảnh theo chiều dọc của ảnh xám, và N là số cột, đại diện cho số điểm ảnh theo chiều ngang của ảnh xám. Hay nói cách khác, nếu ảnh xám có độ phân giải $N \times M$ thì ma trận ảnh đầu vào của nó có kích thước $M \times N$. Để thuận tiện cho các thao tác tiếp theo, ta đánh số các hàng và cột bắt đầu từ chỉ số 0.
- **Giá trị mỗi phần tử trong ma trận:** Mỗi phần tử của ma trận, ký hiệu là $I(i, j)$, với $i = \overline{0, M - 1}$ và $j = \overline{0, N - 1}$, biểu thị mức xám của điểm ảnh có tọa độ (i, j) .

Dưới đây là ma trận ảnh đầu vào của một ảnh xám có độ phân giải 10×10 (ảnh rất bé nên ở đây ta phóng đại lên để tiện theo dõi).



151	152	154	158	164	170	176	171	164	158
155	152	149	149	155	161	168	163	153	145
146	141	135	132	133	137	139	137	135	131
126	126	125	123	125	129	130	131	133	135
124	127	129	131	134	137	143	143	143	142
130	136	142	148	154	158	159	157	153	147
131	137	144	150	156	161	160	159	156	151
137	136	137	137	138	140	143	144	144	143
141	139	138	137	136	137	133	134	135	135
140	136	135	134	132	129	132	132	132	134

Hình 2.3: Ma trận ảnh đầu vào

2. Ma trận bộ lọc

Ma trận bộ lọc thường là một ma trận vuông và có kích thước nhỏ hơn ma trận ảnh đầu vào, phổ biến là 3×3 , 5×5 được sử dụng để thực hiện phép tích chập với ma trận ảnh đầu vào. Để thuận tiện cho các thao tác tiếp theo, ta đánh số các hàng và cột của ma trận bộ lọc bắt đầu từ chỉ số 0.

Dưới đây là các ma trận bộ lọc phổ biến với kích thước 3×3 :

- *Identity Kernel*: Không thay đổi ảnh gốc, hoạt động như một bộ lọc trung lập. Ảnh đầu ra giống hệt ảnh đầu vào:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- *Edge Detection Kernel (Sobel-X)*: Phát hiện các cạnh trong ảnh bằng cách làm nổi bật sự thay đổi cường độ pixel theo trục x :

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- *Edge Detection Kernel (Sobel-Y)*: Phát hiện các cạnh trong ảnh bằng cách làm nổi bật sự thay đổi cường độ pixel theo trục y :

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- *Edge Detection Kernel (Laplacian)*: Phát hiện các cạnh trong ảnh bằng cách làm nổi bật sự thay đổi cường độ pixel theo mọi hướng:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- *Sharpen Kernel*: Làm sắc nét ảnh bằng cách tăng cường sự khác biệt giữa các pixel:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- *Box Blur Kernel*: Làm mờ ảnh bằng cách lấy trung bình giá trị các pixel lân cận:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- *Gaussian Blur Kernel*: Làm mờ ảnh nhưng với độ mờ tăng dần theo khoảng cách từ tâm kernel, tuân theo phân bố Gaussian:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3. *Bước nhảy (Stride)* (S_h, S_w): S_h là số ô di chuyển theo hàng(xuống dưới) và S_w là số ô di chuyển theo cột (sang phải). Sau mỗi lần áp dụng phép tích chập tại một vị trí trên ma trận đầu vào, bộ lọc sẽ di chuyển sang phải thêm

$S_w - 1$ cột để đến vị trí tiếp theo trên cùng hàng. Khi kết thúc một hàng, bộ lọc sẽ di chuyển xuống đầu hàng tiếp theo, cách hàng hiện tại tại $S_h - 1$ hàng. Quá trình này lặp lại cho đến khi bộ lọc quét qua toàn bộ ma trận đầu vào.

4. *Padding*: Là các hàng và cột chứa toàn giá trị 0 được thêm vào một số vị trí xung quanh ma trận đầu vào (có thể thêm vào cạnh trên, dưới, trái, phải) để tạo ra một ma trận đầu vào mới mở rộng trước khi thực hiện phép tích chập.

2.2.3 Quy trình tính tích chập

1. Đầu vào

- Ma trận ảnh đầu vào I kích thước $I_h \times I_w$.
- Ma trận bộ lọc K kích thước $K_h \times K_w$.
- Bước nhảy (S_h, S_w) .
- *Padding* (sử dụng hoặc không sử dụng).
- Hàm kích hoạt.

2. *Quy trình tính toán*: Kết quả của phép tính tích chập là ma trận đặc trưng đầu ra O .

- Trường hợp *không sử dụng padding*:
 - *Bước 1: Đặt ma trận bộ lọc vào góc trên bên trái của ma trận ảnh đầu vào*: Bộ lọc được căn chỉnh sao cho phần tử đầu tiên (góc trên bên trái) của nó trùng với phần tử đầu tiên (góc trên bên trái) của ma trận ảnh đầu vào. Các phần tử còn lại của bộ lọc sẽ phủ lên các phần tử tương ứng trong ma trận ảnh đầu vào trong phạm vi từ hàng 0 đến $K_h - 1$ và cột 0 đến $K_w - 1$, với $K_h \times K_w$ là kích thước của bộ lọc.
 - *Bước 2: Thực hiện phép toán tích chập*: Tại mỗi vị trí của bộ lọc, một phần của ảnh đầu vào có kích thước $K_h \times K_w$ (bằng kích thước

của bộ lọc) sẽ được chọn để tính toán. Phép toán tích chập được thực hiện bằng cách nhân từng phần tử của bộ lọc với phần tử tương ứng trong phần ảnh đầu vào này, sau đó cộng tất cả các tích lại để tạo thành một giá trị duy nhất cho pixel tại vị trí tương ứng trong ảnh đầu ra. Công thức tính như sau:

$$O[i, j] = \sum_{m=0}^{K_h-1} \sum_{n=0}^{K_w-1} I[i+m, j+n] \cdot K[m, n]$$

Trong đó:

- * i là chỉ số hàng trong ảnh đầu ra, bắt đầu từ 0.
- * j là chỉ số cột trong ảnh đầu ra, bắt đầu từ 0.
- * m và n là chỉ số hàng và cột trong bộ lọc, bắt đầu từ 0.
- *Bước 3: Di chuyển bộ lọc theo bước nhảy (S_h, S_w)* : Sau khi hoàn tất việc tính toán tại vị trí (i, j) , bộ lọc sẽ đi sang phải, qua $S_w - 1$ cột tới vị trí tiếp theo trên hàng. Quá trình này tiếp tục cho đến khi bộ lọc đã đi hết hàng (tức là nếu tiếp tục di chuyển thì mép phải của bộ lọc sẽ vượt ra ngoài mép phải của ma trận ảnh đầu vào). Khi đó, bộ lọc sẽ dịch chuyển xuống đầu hàng tiếp theo, cách hàng hiện tại tại $S_h - 1$ hàng, và lặp lại quy trình tương tự.
- *Bước 4: Lặp lại lần lượt bước 2 và bước 3 cho đến khi bộ lọc quét hết ma trận ảnh đầu vào.*

• Trường hợp có sử dụng *padding*:

Tiến hành thêm $\lfloor \frac{x}{2} \rfloor$ hàng giá trị 0 vào phía trên, $\lceil \frac{x}{2} \rceil$ hàng giá trị 0 vào phía dưới và $\lfloor \frac{y}{2} \rfloor$ cột giá trị 0 vào bên trái, $\lceil \frac{y}{2} \rceil$ cột giá trị 0 vào bên phải của ma trận đầu vào, thu được ma trận đầu vào mới được mở rộng.

Trong đó:

$$\begin{aligned} x &= \left(\left\lceil \frac{I_h}{S_h} \right\rceil - 1 \right) S_h + K_h - I_h, \\ y &= \left(\left\lceil \frac{I_w}{S_w} \right\rceil - 1 \right) S_w + K_w - I_w. \end{aligned}$$

Quy trình tính toán tiếp theo tương tự như các bước trong trường hợp *không sử dụng padding*.

3. Đầu ra

Ma trận ảnh đặc trưng đầu ra có kích thước $O_h \times O_w$ được xác định như sau:

- Trường hợp *không sử dụng padding*:

$$O_h = \left\lfloor \frac{I_h - K_h}{S_h} \right\rfloor + 1, \quad O_w = \left\lfloor \frac{I_w - K_w}{S_w} \right\rfloor + 1$$

- Trường hợp *có sử dụng padding*:

$$O_h = \left\lceil \frac{I_h}{S_h} \right\rceil, \quad O_w = \left\lceil \frac{I_w}{S_w} \right\rceil$$

2.2.4 Ví dụ

Giả sử ta có ảnh được biểu diễn bởi ma trận ảnh đầu vào như sau:

$$I = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Ma trận bộ lọc sử dụng là:

$$K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Ở đây, ta không sử dụng padding và $(S_h, S_w) = (1, 1)$. Gọi ma trận A là phần của ma trận ảnh đầu vào mà bộ lọc K phủ lên khi di chuyển qua từng bước. Khi đó, phần tử của ma trận đặc trưng được tính bằng tổng các tích giữa các phần tử có vị trí tương ứng trong ma trận A và bộ lọc K .

Dưới đây là quy trình tính toán tích chập để tạo ra ma trận đặc trưng đầu ra. Trong mỗi bước, từng phần tử của ma trận đầu vào A (được ký hiệu là a) sẽ

nhân với phần tử tương ứng trong ma trận kernel K (được ký hiệu là b). Ta sẽ biểu diễn trong ma trận A dưới dạng $a_{\times b}$ để dễ dàng theo dõi. Phần tử tại mỗi vị trí trong ma trận đặc trưng đầu ra được tính bằng tổng các tích $a_{\times b}$.

• *Bước 1:*

$$A = \begin{bmatrix} 1_{\times 1} & 1_{\times 0} & 1_{\times 1} \\ 0_{\times 0} & 1_{\times 1} & 1_{\times 0} \\ 0_{\times 1} & 0_{\times 0} & 1_{\times 1} \end{bmatrix}$$

$$O[0][0] = 4$$

• *Bước 2:*

$$A = \begin{bmatrix} 1_{\times 1} & 1_{\times 0} & 0_{\times 1} \\ 1_{\times 0} & 1_{\times 1} & 1_{\times 0} \\ 0_{\times 1} & 1_{\times 0} & 1_{\times 1} \end{bmatrix}$$

$$O[0][1] = 3$$

• *Bước 3:*

$$A = \begin{bmatrix} 1_{\times 1} & 0_{\times 0} & 0_{\times 1} \\ 1_{\times 0} & 1_{\times 1} & 0_{\times 0} \\ 1_{\times 1} & 1_{\times 0} & 1_{\times 1} \end{bmatrix}$$

$$O[0][2] = 4$$

• *Bước 4:*

$$A = \begin{bmatrix} 0_{\times 1} & 1_{\times 0} & 1_{\times 1} \\ 0_{\times 0} & 0_{\times 1} & 1_{\times 0} \\ 0_{\times 1} & 0_{\times 0} & 1_{\times 1} \end{bmatrix}$$

$$O[1][0] = 2$$

• *Bước 5:*

$$A = \begin{bmatrix} 1_{\times 1} & 1_{\times 0} & 1_{\times 1} \\ 0_{\times 0} & 1_{\times 1} & 1_{\times 0} \\ 0_{\times 1} & 1_{\times 0} & 1_{\times 1} \end{bmatrix}$$

$$O[1][1] = 4$$

• *Bước 6:*

$$A = \begin{bmatrix} 1_{\times 1} & 1_{\times 0} & 0_{\times 1} \\ 1_{\times 0} & 1_{\times 1} & 1_{\times 0} \\ 1_{\times 1} & 1_{\times 0} & 0_{\times 1} \end{bmatrix}$$

$$O[1][2] = 3$$

• *Bước 7:*

$$A = \begin{bmatrix} 0_{\times 1} & 0_{\times 0} & 1_{\times 1} \\ 0_{\times 0} & 0_{\times 1} & 1_{\times 0} \\ 0_{\times 1} & 1_{\times 0} & 1_{\times 1} \end{bmatrix}$$

$$O[2][0] = 2$$

• *Bước 8:*

$$A = \begin{bmatrix} 0_{\times 1} & 1_{\times 0} & 1_{\times 1} \\ 0_{\times 0} & 1_{\times 1} & 1_{\times 0} \\ 1_{\times 1} & 1_{\times 0} & 0_{\times 1} \end{bmatrix}$$

$$O[2][1] = 3$$

• *Bước 9:*

$$A = \begin{bmatrix} 1_{\times 1} & 1_{\times 0} & 1_{\times 1} \\ 1_{\times 0} & 1_{\times 1} & 0_{\times 0} \\ 1_{\times 1} & 0_{\times 0} & 0_{\times 1} \end{bmatrix}$$

$$O[2][2] = 4$$

• *Ma trận ảnh đặc trưng đầu ra:*

$$O = \begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix}$$

2.2.5 Các kỹ thuật trích xuất đặc trưng hỗ trợ tích chập

Hàm phi tuyến

Sau khi thực hiện phép tích chập giữa ma trận ảnh đầu vào và bộ lọc, kết quả thu được là một ma trận ảnh đặc trưng đầu ra. Tuy nhiên, kết quả này mới chỉ là kết quả của các phép toán *tuyến tính*, bởi nó chỉ bao gồm các phép nhân và cộng giữa các mức xám pixel và trọng số trong bộ lọc. Các phép toán tuyến tính này chỉ biểu diễn được các đặc trưng đơn giản như cạnh, đường viền hoặc làm mờ ảnh, mà chưa thể biểu diễn được các đặc trưng phức tạp hơn.

Để khắc phục điều này, người ta áp dụng các *hàm phi tuyến* - là các hàm số biến đổi áp dụng trên từng phần tử của ma trận đầu ra, giúp mô hình học được các quan hệ phi tuyến giữa các pixel và biểu diễn tốt hơn các đặc trưng phức tạp của ảnh đầu vào. Dưới đây là thông tin về một số hàm phi tuyến thông dụng [6].

1. Hàm ReLU

• *Công thức:*

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

- *Mô tả:* Hàm ReLU giữ nguyên giá trị đầu vào nếu nó không âm và đưa các giá trị âm về 0.
- *Ưu điểm:* Tính toán đơn giản, nhanh chóng và hiệu quả.
- *Nhược điểm:* Nếu trong ma trận đặc trưng đầu ra có quá nhiều giá trị âm, sau khi áp dụng hàm ReLU, chúng sẽ chuyển thành 0. Điều này khiến ma trận không còn phản ánh được đầy đủ đặc trưng của ảnh đầu

vào, vì các phần tử có giá trị 0 không đóng góp vào việc trích xuất các đặc trưng quan trọng từ ảnh.

2. Hàm Sigmoid

- *Công thức:*

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

- *Mô tả:* Hàm sigmoid nén các giá trị đầu vào về khoảng từ 0 đến 1.
- *Ưu điểm:* Đầu ra có tính chất xác suất, nằm trong khoảng $(0, 1)$; phù hợp cho các bài toán phân loại nhị phân.
- *Nhược điểm:* Tính toán của hàm này phức tạp hơn ReLU, làm tăng thời gian và chi phí tính toán.

3. Hàm Tanh

- *Công thức:*

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

- *Mô tả:* Hàm tanh nén giá trị đầu vào về khoảng từ -1 đến 1 , cân bằng quanh 0.
- *Ưu điểm:* Đầu ra cân bằng quanh 0, phù hợp khi cần xử lý các giá trị có tính đối xứng hoặc khi cả giá trị âm và dương đều có ý nghĩa trong việc trích xuất đặc trưng.
- *Nhược điểm:* Tính toán của hàm này phức tạp hơn ReLU, làm tăng thời gian và chi phí tính toán.

Max Pooling

Max Pooling là một kỹ thuật được sử dụng để giảm kích thước của các ma trận đầu vào, trích xuất các đặc trưng quan trọng. Trong quá trình này, mỗi vùng con trong ma trận đầu vào sẽ được tính toán và biến đổi vùng con đó thành một giá trị duy nhất - là giá trị lớn nhất trong vùng con [6].

1. Đầu vào

- Ma trận đầu vào I kích thước $I_h \times I_w$.
- Vùng con có kích thước $K_h \times K_w$.
- Bước nhảy (S_h, S_w) .
- Padding (sử dụng hoặc không sử dụng).

2. Quy trình tính toán

- Trường hợp *không sử dụng padding*

Đầu tiên, xét vùng con kích thước $K_h \times K_w$ nằm ở góc trên bên trái của ma trận đầu vào.

- *Bước 1:* Lấy phần tử lớn nhất của vùng con đang xét làm phần tử của ma trận đầu ra (vị trí của phần tử này trong ma trận đầu ra tương ứng với vị trí của vùng con trong ma trận đầu vào, ví dụ nếu vùng con ở góc trên bên trái của ma trận đầu vào thì phần tử sẽ nằm ở góc trên bên trái của ma trận đầu ra - tức là nằm ở hàng đầu, cột đầu của ma trận đầu ra).
- *Bước 2:* Nếu vùng con ở vị trí cuối ma trận đầu vào (tức là nếu di chuyển tiếp theo bước nhảy sang phải và xuống dưới thì mép ngoài của vùng con sẽ vượt qua mép ngoài của ma trận đầu vào) thì kết thúc. Ngược lại chuyển xuống bước 3.
- *Bước 3:* Nếu vùng con ở vị trí cuối dãy (tức là nếu di chuyển tiếp theo bước nhảy thì mép ngoài của vùng con sẽ vượt quá mép ngoài của ma trận đầu vào), thì vùng con sẽ di chuyển xuống đầu dãy tiếp theo, sao cho hàng đầu của vùng con tại vị trí mới tiếp theo cách hàng đầu của vùng con tại vị trí hiện tại $S_h - 1$ hàng. Ngược lại, nếu vùng con không ở vị trí cuối dãy, thì vùng con sẽ di chuyển sang phải sao cho cột đầu của vùng con tại vị trí mới tiếp theo cách cột đầu của vùng con tại vị trí hiện tại $S_w - 1$ cột.

- Trường hợp *có sử dụng padding*:

Tiến hành thêm $\lfloor \frac{x}{2} \rfloor$ hàng giá trị 0 vào phía trên, $\lceil \frac{x}{2} \rceil$ hàng giá trị vào phía dưới và $\lfloor \frac{y}{2} \rfloor$ cột giá trị 0 vào bên trái, $\lceil \frac{y}{2} \rceil$ cột giá trị 0 vào bên phải của ma trận đầu vào, thu được ma trận đầu vào mới được mở rộng.

Trong đó:

$$x = \left(\left\lceil \frac{I_h}{S_h} \right\rceil - 1 \right) S_h + K_h - I_h,$$

$$y = \left(\left\lceil \frac{I_w}{S_w} \right\rceil - 1 \right) S_w + K_w - I_w.$$

Sau đó ta thực hiện tính toán trên ma trận đầu vào mới. Quy trình tính toán tương tự như trường hợp *không sử dụng padding*.

3. Đầu ra

Ma trận đầu ra có kích thước $O_h \times O_w$ được xác định như sau:

- Trường hợp *không sử dụng padding*:

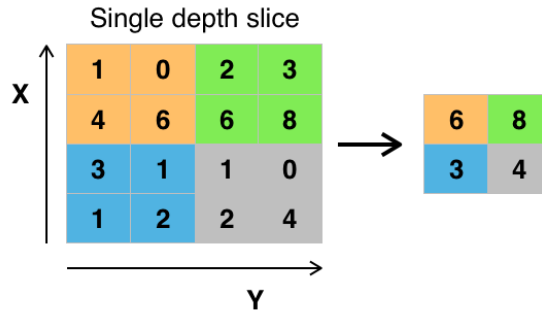
$$O_h = \left\lfloor \frac{I_h - K_h}{S_h} \right\rfloor + 1, \quad O_w = \left\lfloor \frac{I_w - K_w}{S_w} \right\rfloor + 1$$

- Trường hợp *có sử dụng padding*:

$$O_h = \left\lceil \frac{I_h}{S_h} \right\rceil, \quad O_w = \left\lceil \frac{I_w}{S_w} \right\rceil$$

4. Ví dụ

Dưới đây là ví dụ minh họa về kỹ thuật Max Pooling trên ma trận đầu vào kích thước 4×4 , với vùng con kích thước 2×2 , bước nhảy $(2, 2)$ và không sử dụng padding.



Hình 2.4: Max Pooling

2.3 Tích chập chuyển vị

2.3.1 Khái niệm

Tích chập chuyển vị (transposed convolution) là một phép toán dùng để tái tạo các chi tiết của ảnh đầu vào. Cụ thể, phép tích chập chuyển vị được thực hiện bằng cách áp dụng một *bộ lọc (kernel)* lên từng phần tử của *ma trận ảnh đầu vào (image matrix)* để tạo ra *ma trận đặc trưng đầu ra (upscaled feature map)* [5].

2.3.2 Thành phần của phép tính tích chập chuyển vị

Phép tính tích chập chuyển vị cũng bao gồm các thành phần tương tự như phép tính tích chập, bao gồm ma trận đầu vào, ma trận bộ lọc, bước nhảy (stride) và padding.

2.3.3 Quy trình tính tích chập chuyển vị

1. Đầu vào

- *Ma trận ảnh đầu vào I* kích thước $I_h \times I_w$.
- *Ma trận bộ lọc K* kích thước $K_h \times K_w$.
- *Bước nhảy (S_h, S_w)*.
- *Padding* (sử dụng hoặc không sử dụng).
- *Hàm kích hoạt*.

2. Quy trình tính toán

Kết quả của phép tính tích chập chuyển vị là ma trận mở rộng O . Mỗi phần tử trong ma trận đầu vào sẽ được nhân với bộ lọc và "mở rộng" thành một ma trận. Như vậy 1 hàng của ma trận đầu vào sẽ được "mở rộng" thành 1 dãy các ma trận (các ma trận trong dãy có thể tách biệt, có thể chồng lấn lên nhau) tập hợp các dãy này tạo thành ma trận O .

- Trường hợp *không sử dụng padding*

Xét phần tử góc trên bên trái của ma trận đầu vào. Nhân phần tử này với ma trận bộ lọc, thu được ma trận kết quả B_1 . Chuyển sang xét phần tử kế bên phải nó. Khởi tạo biến $i = 2$.

- *Bước 1:* Nhân phần tử hiện tại đang xét của ma trận đầu vào với ma trận bộ lọc, thu được ma trận kết quả B_i .
- *Bước 2:* Xếp B_i nối tiếp B_{i-1} . Cụ thể, nếu B_{i-1} là tích của 1 phần tử cuối hàng của ma trận đầu vào với ma trận bộ lọc thì B_i sẽ được xếp xuống đầu dãy tiếp theo, sao cho hàng đầu tiên của B_i cách hàng đầu tiên của các ma trận trong dãy hiện tại $(S_h - 1)$ hàng. Còn nếu B_{i-1} là tích của phần tử ở trong hàng của ma trận đầu vào (không đứng cuối hàng) với bộ lọc thì B_i sẽ được xếp vào cùng dãy với B_{i-1} sao cho cột đầu tiên của B_i cách cột đầu tiên cùng của B_{i-1} $(S_w - 1)$ cột. Tiếp theo, nếu khi xếp, B_i không chồng lấn lên các ma trận đã xếp trước đó thì chuyển xuống bước 3, ngược lại cập nhật mỗi giá trị trong các hàng/ cột mà B_i xếp chồng lên bằng tổng của giá trị ban đầu với giá trị tương ứng trong B_i khi B_i xếp chồng lên, rồi chuyển xuống bước 3.
- *Bước 3:* Tăng i lên 1. Nếu $i > I_h I_w$ thì chuyển sang bước 4. Ngược lại, chuyển sang xét *phần tử tiếp theo* của ma trận đầu vào, quay lại bước 1 (nếu phần tử hiện tại không phải phần tử cuối hàng, thì phần tử tiếp theo là phần tử kế bên phải nó trong hàng. Còn nếu phần tử hiện tại là phần tử cuối hàng, thì phần tử tiếp theo là phần tử đầu hàng tiếp theo).
- *Bước 4:* Nếu $K_h \geq S_h$ và $K_w \geq S_w$ thì kết thúc. Ngược lại, nếu $K_h < S_h$ thì giữa các dãy ma trận sẽ có các hàng trống, tương tự nếu $K_w < S_w$ thì giữa 2 ma trận xếp liền kề trong các dãy cũng sẽ có các cột trống. Thực hiện lấp đầy các vùng trống này bằng các giá trị 0. Sau đó thêm tiếp $S_h - K_h$ hàng 0 vào phía dưới và $S_w - K_w$

cột 0 vào bên phải ma trận hiện tại, thu được ma trận đầu ra O .

- Trường hợp có sử dụng padding

Khi sử dụng padding, vùng ma trận đầu ra sẽ có kích thước cố định là $(I_h S_h) \times (I_w S_w)$. Tùy vào các giá trị S_h, S_w, K_h, K_w mà ta sẽ tiến hành thêm các hàng 0, cột 0 vào các vị trí xung quanh (phía trên, phía dưới, bên trái, bên phải) vùng ma trận đầu ra để thực hiện tính toán.

– *Bước 1: Thêm các hàng 0 vào phía trên/ dưới của vùng ma trận đầu ra*

Nếu $K_h < S_h$ thì không thêm hàng 0 nào vào phía trên và dưới vùng đầu ra. Ngược lại nếu $K_h \geq S_h$, thêm $\lfloor \frac{K_h - S_h}{2} \rfloor$ hàng 0 vào phía trên, $\lceil \frac{K_h - S_h}{2} \rceil$ hàng 0 vào phía dưới của vùng đầu ra.

– *Bước 2: Thêm các cột 0 vào bên trái/ phải của vùng ma trận đầu ra*

Nếu $K_w < S_w$ thì không thêm cột 0 nào vào bên trái và phải vùng đầu ra. Ngược lại nếu $K_w \geq S_w$, thêm $\lfloor \frac{K_w - S_w}{2} \rfloor$ cột 0 vào bên trái, $\lceil \frac{K_w - S_w}{2} \rceil$ cột 0 vào bên phải của vùng đầu ra.

– *Bước 3: Tính toán tích chập chuyển vị*

Thực hiện tính toán tích chập chuyển vị và xếp các ma trận mở rộng (ma trận thu được bằng cách nhân mỗi phần tử trên ma trận đầu vào với bộ lọc) tương tự như trường hợp không sử dụng padding lên vùng đầu ra.

– *Bước 4: Loại bỏ đi các hàng/ cột mà trước đó đã thêm vào vùng ma trận đầu ra khi sử dụng padding*

Ta thu được ma trận đầu ra O .

3. Đầu ra

Ma trận ảnh đầu ra có kích thước $O_h \times O_w$ được xác định như sau:

- Trường hợp *không sử dụng padding*

$$O_h = \begin{cases} I_h S_h, & \text{nếu } S_h \geq K_h, \\ (I_h - 1)S_h + K_h, & \text{nếu } S_h < K_h. \end{cases}$$

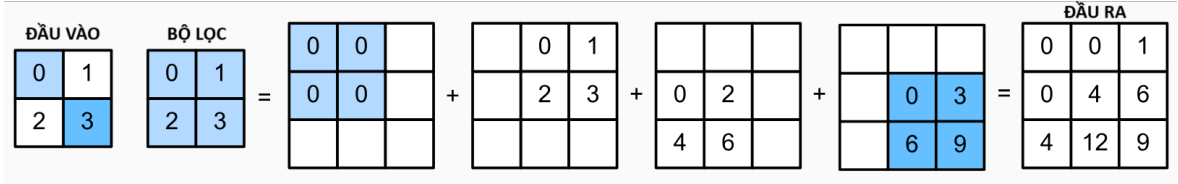
$$O_w = \begin{cases} I_w S_w, & \text{nếu } S_w \geq K_w, \\ (I_w - 1)S_w + K_w, & \text{nếu } S_w < K_w. \end{cases}$$

- Trường hợp *có sử dụng padding*:

$$O_h = I_h S_h, \quad O_w = I_w S_w$$

2.3.4 Ví dụ

Dưới đây là ví dụ minh họa về phép tính tích chập giữa ma trận đầu vào kích thước 2×2 và bộ lọc kích thước 2×2 , bước nhảy $(1, 1)$, không sử dụng padding và hàm kích hoạt.



Hình 2.5: Tích chập chuyển vị

2.4 Mạng U-Net

2.4.1 Giới thiệu về mạng U-Net

Mạng U-Net ra đời trong bối cảnh cần một giải pháp hiệu quả cho bài toán phân đoạn hình ảnh, đặc biệt trong lĩnh vực y học, nơi dữ liệu huấn luyện thường rất hạn chế. Các phương pháp trước đó (như Sliding Window, Fully Convolutional Networks,...) đã được sử dụng nhưng gặp hạn chế về tốc độ và độ

chính xác. Với kiến trúc *mã hóa - giải mã (encoder-decoder)*, U-Net tối ưu hóa việc kết hợp các đặc trưng chi tiết và ngữ cảnh, giúp phân đoạn chính xác từng pixel ngay cả khi số lượng dữ liệu huấn luyện ít.

Thuộc nhóm các phương pháp phân đoạn ảnh dựa trên *học sâu (deep learning)*, U-Net sử dụng phần *encoder* để trích xuất đặc trưng quan trọng từ ảnh đầu vào thông qua các phép tích chập và giảm kích thước, trong khi phần *decoder* dần phục hồi kích thước ảnh và tái tạo chi tiết thông qua tích chập chuyển vị (transposed convolution). Ngoài ra, các kết nối skip connection (kết nối theo chiều kênh đặc trưng) giữa *encoder* và *decoder* đóng vai trò quan trọng trong việc kết hợp thông tin từ các lớp sâu và nông, cải thiện hiệu quả phân đoạn.

Mạng này đã đạt thành tích xuất sắc trong các thử thách như phân đoạn EM tại ISBI 2012 và ISBI Cell Tracking Challenge 2015, trở thành công cụ hàng đầu trong phân đoạn hình ảnh y học [7].

2.4.2 Các lớp xử lý trong mạng U-Net

Các *lớp xử lý* là các thành phần cơ bản trong kiến trúc của mạng nơ-ron, thực hiện các phép tính toán cần thiết để tiến hành phân tích và học tập từ dữ liệu. Cụ thể trong mạng U-Net, các lớp xử lý trích xuất đặc trưng quan trọng hoặc khôi phục các chi tiết cần thiết.

1. Lớp tích chập

Lớp tích chập thực hiện phép toán tích chập trên khối ma trận ảnh đầu vào để trích xuất các đặc trưng quan trọng từ ảnh.

Thành phần cơ bản của lớp tích chập:

- *Số lượng bộ lọc*: Tương ứng với số lượng *kênh đặc trưng* đầu ra - cũng chính là số ma trận đặc trưng đầu ra. Mỗi kênh đặc trưng (ma trận đặc trưng) biểu diễn một loại đặc trưng cụ thể được trích xuất từ khối đầu vào. Ở trong lớp tích chập, mỗi bộ lọc là một *khối các ma trận bộ lọc* (là tập hợp nhiều ma trận bộ lọc, chi tiết sẽ trình bày ở phần hoạt động

của lớp tính chập dưới đây).

- *Ma trận bộ lọc kích thước $K_h \times K_w$* : thường là ma trận vuông với kích thước lẻ.
- *Hàm kích hoạt*.
- *Bước nhảy (Stride) (S_h, S_w)* .
- *Padding* (có sử dụng hoặc không sử dụng).

Hoạt động của lớp tích chập:

- *Đầu vào*: Một khối ma trận đầu vào kích thước $H \times W \times C$ (H là số hàng, W là số cột của mỗi ma trận, C là số lượng ma trận - số lượng kênh đặc trưng), số lượng bộ lọc C' , kích thước ma trận bộ lọc $K_h \times K_w$, bước nhảy (S_h, S_w) , hàm kích hoạt, padding (có sử dụng hoặc không sử dụng).
- *Quy trình tính tích chập*: Đầu vào là một khối ma trận kích thước $H \times W \times C$, mỗi bộ lọc (là một khối các ma trận bộ lọc) kích thước $K_h \times K_w \times C$. Quy trình tính toán khi 1 bộ lọc quét qua khối đầu vào gồm các bước sau:
 - *Bước 1*: Trên C cặp ma trận đầu vào - ma trận bộ lọc tương ứng trong khối đầu vào và bộ lọc, thực hiện phép tích chập (dựa trên stride, padding) cho từng cặp, thu được C ma trận đầu ra.
 - *Bước 2*: Tính tổng của C ma trận đầu ra ở bước 1, thu được 1 ma trận. Áp dụng hàm kích hoạt lên ma trận này ta thu được 1 ma trận đầu ra duy nhất.

C' bộ lọc quét qua khối ma trận đầu vào và cho ra C' ma trận đầu ra.

- *Đầu ra*:

Một khối ma trận đầu ra có kích thước $H' \times W' \times C'$. Trong đó:

- Trường hợp *sử dụng padding*

$$H' = \left\lceil \frac{H}{S_h} \right\rceil, \quad W' = \left\lceil \frac{W}{S_w} \right\rceil$$

– Trường hợp *không sử dụng padding*

$$H' = \left\lfloor \frac{H}{S_h} \right\rfloor, \quad W' = \left\lfloor \frac{W}{S_w} \right\rfloor$$

2. Lớp Max Pooling

Lớp Max Pooling là một lớp giảm kích thước của khối ma trận đầu vào bằng cách chọn giá trị lớn nhất trong mỗi *vùng gộp* - là một khu vực con trong mỗi ma trận của khối đầu vào. Lớp này giúp giảm kích thước giữ lại các đặc trưng quan trọng nhất.

Hoạt động của lớp Max Pooling:

- *Đầu vào*: Một khối ma trận đặc trưng kích thước $H \times W \times C$ (H là chiều cao, W là chiều rộng, C là số lượng kênh đặc trưng), kích thước vùng con $K_h \times K_w$, bước nhảy (S_h, S_w) , padding (không sử dụng hoặc có sử dụng).
- *Quy trình tính toán*: Thực hiện tính toán max pooling trên mỗi ma trận trong khối đầu vào (có C ma trận đầu vào), thu được C ma trận đầu ra.
- *Đầu ra*:

Khối ma trận đầu ra có kích thước $H' \times W' \times C$ được xác định như sau:

– Trường hợp *không sử dụng padding*:

$$H' = \left\lfloor \frac{H - K_h}{S_h} \right\rfloor + 1, \quad W' = \left\lfloor \frac{W - K_w}{S_w} \right\rfloor + 1$$

– Trường hợp *có sử dụng padding*:

$$H' = \left\lceil \frac{H}{S_h} \right\rceil, \quad W' = \left\lceil \frac{W}{S_w} \right\rceil$$

3. Lớp tích chập chuyển vị

Lớp tích chập chuyển vị thực hiện phép toán tích chập chuyển vị trên khối ma trận ảnh đầu vào để khôi phục các chi tiết trong ảnh.

Thành phần cơ bản của lớp này bao gồm:

- *Số lượng bộ lọc*: Tương ứng với số lượng kênh đặc trưng tái tạo đầu ra - cũng chính là số ma trận đặc trưng tái tạo đầu ra. Mỗi kênh đặc trưng

(ma trận đầu ra) biểu diễn một loại đặc trưng cụ thể được tái tạo từ khối đầu vào. Trong lớp tích chập chuyển vị, mỗi bộ lọc là một khối các ma trận bộ lọc (là tập hợp nhiều ma trận bộ lọc).

- *Ma trận bộ lọc kích thước $K_h \times K_w$* : thường là ma trận vuông với kích thước lẻ.
- *Bước nhảy (Stride) (S_h, S_w)* .
- *Padding* (có sử dụng hoặc không sử dụng).

Hoạt động của lớp tích chập chuyển vị:

- *Đầu vào*: Một khối ma trận đầu vào kích thước $H \times W \times C$ (H là số hàng, W là số cột của mỗi ma trận, C là số lượng ma trận - số lượng kênh đặc trưng), số lượng bộ lọc C' , kích thước ma trận bộ lọc $K_h \times K_w$, bước nhảy (S_h, S_w) , hàm kích hoạt, padding (có sử dụng hoặc không sử dụng).
- *Quy trình tính tích chập chuyển vị*: Đầu vào là một khối ma trận kích thước $H \times W \times C$, mỗi bộ lọc (là một khối các ma trận bộ lọc) kích thước $K_h \times K_w \times C$. Quy trình tính toán khi 1 bộ lọc quét qua khối đầu vào gồm các bước sau:

- *Bước 1*: Trên C cặp ma trận đầu vào - ma trận bộ lọc tương ứng trong khối đầu vào và bộ lọc, thực hiện phép tích chập chuyển vị (dựa trên stride, padding) cho từng cặp, thu được C ma trận đầu ra.
- *Bước 2*: Tính tổng của C ma trận đầu ra ở bước 1, thu được 1 ma trận. Áp dụng hàm kích hoạt lên ma trận này ta thu được 1 ma trận đầu ra duy nhất.

C' bộ lọc sẽ quét qua khối ma trận đầu vào và cho ra C' ma trận đầu ra.

- *Đầu ra*:
 - Trường hợp không sử dụng padding:
 - Trường hợp sử dụng padding: Một khối ma trận đầu ra có kích thước

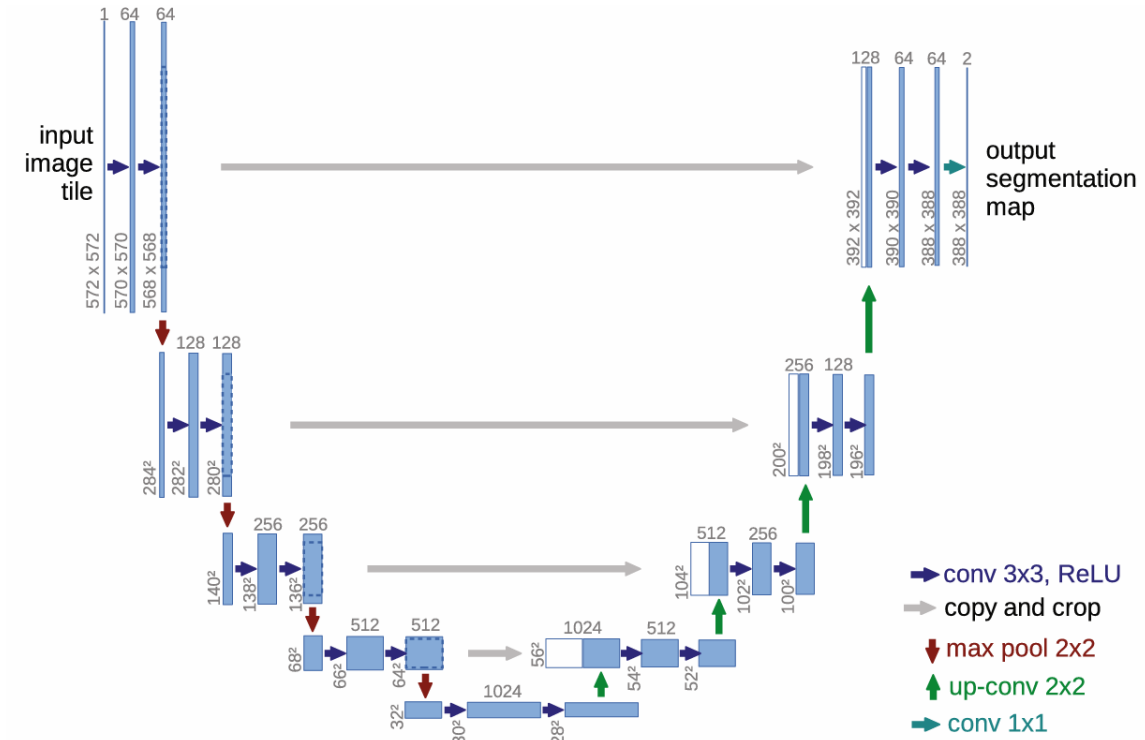
$H' \times W' \times C'$. Trong đó:

$$H' = HS_h$$

$$W' = WS_w$$

2.4.3 Kiến trúc mạng U-Net

Mạng có cấu trúc hình chữ U, gồm hai thành phần chính: *nhánh thu hẹp* (*contracting path*) và *nhánh mở rộng* (*expansive path*) [7].



Hình 2.6: Kiến trúc mạng U-Net [7]

Dưới đây là mô tả chi tiết từng phần của kiến trúc:

1. Nhánh thu hẹp

Nhánh thu hẹp là quá trình trích xuất các đặc trưng quan trọng từ ảnh đầu vào (ảnh xám) thông qua việc áp dụng các lớp tích chập và lớp max pooling để giảm kích thước của ma trận đầu ra, tăng tính khái quát hóa dữ liệu (encoder). Quá trình diễn ra như sau:

- *Bước 1:* Thực hiện áp dụng hai lớp tích chập liên tiếp lên ảnh đầu vào. Thành phần cơ bản của mỗi lớp tích chập gồm: số lượng bộ lọc (cũng chính là số kênh đặc trưng) được tăng lên (ví dụ trong kiến trúc U-Net ở trên là 64 kênh), kích thước ma trận bộ lọc 3×3 , bước nhảy (1, 1), hàm kích hoạt *Relu*, không sử dụng padding. Chuyển đến bước 3.
- *Bước 2:* Thực hiện áp dụng hai lớp tích chập liên tiếp lên khối ma trận đầu vào. Thành phần cơ bản của mỗi lớp tích chập gồm: số lượng bộ lọc (cũng chính là số kênh đặc trưng) được tăng lên gấp đôi so với bước trước, kích thước ma trận bộ lọc 3×3 , bước nhảy (1, 1), hàm kích hoạt *ReLU*, không sử dụng padding. Chuyển đến bước 3.
- *Bước 3:* Áp dụng lớp max pooling với kích thước vùng con 2×2 và bước nhảy (2, 2). Nếu kích thước của mỗi ma trận trong khối đầu ra thu được đủ nhỏ (so với yêu cầu cho trước) thì kết thúc. Ngược lại, chuyển đến bước 2.

2. Nhánh mở rộng

Nhánh mở rộng là quá trình giúp tái tạo lại chi tiết ảnh đầu vào thông qua các lớp tích chập và tích chập chuyển vị để tăng kích thước của ma trận đầu ra, tăng tính cụ thể, chi tiết của dữ liệu (decoder). Quá trình diễn ra như sau:

- *Bước 1:* Thực hiện áp dụng lớp tích chập chuyển vị lên khối ma trận đầu vào. Thành phần của lớp này gồm: số lượng bộ lọc (cũng chính là số kênh đặc trưng) giảm một nửa, kích thước ma trận bộ lọc 2×2 , bước nhảy (2, 2), không sử dụng hàm kích hoạt, không sử dụng padding.
- *Bước 2:* Kết hợp khối ma trận đầu ra thu được từ bước 1 trên *nhánh mở rộng* với khối ma trận đầu ra tương ứng trên *nhánh thu hẹp* theo *chiều kênh đặc trưng*. Đầu tiên, *sao chép (copy)* ma trận đầu ra từ *nhánh thu hẹp*. Tiếp theo, *cắt tỉa (crop)* khối ma trận này bằng cách loại bỏ các hàng và cột ở phần biên của mỗi ma trận trong khối, nhằm đảm bảo

rằng kích thước của các ma trận đầu ra trên *nhánh thu hẹp* khớp chính xác với kích thước của các ma trận đầu ra trên *nhánh mở rộng*. Cuối cùng, thực hiện phép nối (*concatenate*) theo chiều kênh đặc trưng - tức là tạo ra một khối ma trận hợp nhất với số lượng kênh đặc trưng bằng tổng số kênh của khối ma trận đầu ra trên *nhánh thu hẹp* và *nhánh mở rộng*.

- *Bước 3*: Thực hiện áp dụng hai lớp tích chập liên tiếp lên khối ma trận đầu vào. Thành phần cơ bản của mỗi lớp tích chập gồm: số lượng bộ lọc (cũng chính là số kênh đặc trưng) được tăng lên gấp đôi so với bước trước, kích thước ma trận bộ lọc 3×3 , bước nhảy (1, 1), hàm kích hoạt *ReLU*, không sử dụng padding. Nếu kích thước của mỗi ma trận trong khối đầu ra đạt yêu cầu cho trước, chuyển đến bước 4. Ngược lại, chuyển đến bước 1.
- *Bước 4*: Thực hiện áp dụng lớp tích chập cuối cùng lên khối ma trận đầu vào. Thành phần cơ bản của lớp tích chập gồm: số lượng bộ lọc (cũng chính là số kênh đặc trưng) được giảm xuống (thường bằng số lớp phân đoạn quan tâm, ví dụ trong mạng U-Net là 2), kích thước ma trận bộ lọc 1×1 , bước nhảy (1, 1), sử dụng hàm kích hoạt phù hợp với yêu cầu phân đoạn, không sử dụng padding. Cuối cùng ta thu được khối ma trận đầu ra, chính là kết quả phân đoạn ảnh.

2.5 Thuật toán Otsu

Thuật toán Otsu là một thuật toán tìm ngưỡng phân đoạn ảnh, thuộc nhóm *Region Division* nằm trong nhóm các phương pháp *phân đoạn ảnh cổ điển* (*Classic Segmentation Methods*). Mục tiêu của thuật toán là tìm ngưỡng tối ưu để phân đoạn ảnh thành hai nhóm: *đối tượng* và *nền* (ở đây ta xét đối tượng là tập hợp những pixel có giá trị xám lớn hơn hoặc bằng ngưỡng tối ưu và nền là tập hợp những pixel có giá trị xám nhỏ hơn ngưỡng tối ưu), sao cho phương sai giữa hai nhóm đạt giá trị lớn nhất.

2.5.1 Đầu vào

Đầu vào là một ảnh xám 8-bit (mức xám của các pixel từ 0 đến 255) có độ phân giải $N \times M$ được biểu diễn qua ma trận ảnh đầu vào I có kích thước $M \times N$ (chỉ số hàng $i = \overline{0, M-1}$, chỉ số cột $j = \overline{0, N-1}$).

2.5.2 Thuật toán

1. Tính tổng số điểm ảnh T và giá trị trung bình mức xám của ảnh m_G .

$$T = MN, \quad m_G = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} I[i, j]$$

2. Lập vector $Histogram[k]$ biểu diễn số lượng điểm ảnh có mức xám k và vector $T_{gray}[k] = k \cdot Histogram[k]$ với $k = 0, 1, \dots, 255$.

3. Lập qua tất cả các giá trị ngưỡng t từ 0 đến 255

- Chia ảnh thành hai vùng: A với các giá trị xám lớn hơn hoặc bằng t và B với các giá trị xám nhỏ hơn t .
- Tính tổng số pixel và tổng giá trị xám cho từng vùng:

$$T_A(t) = \sum_{k=t}^{255} histogram[k], \quad T_B(t) = T - T_A(t)$$

$$T_{gray}^A(t) = \sum_{k=t}^{255} T_{gray}[k], \quad T_{gray}^B(t) = \sum_{k=0}^{t-1} T_{gray}[k]$$

- Tính trung bình mức xám của mỗi vùng:

$$m_A(t) = \frac{T_{gray}^A(t)}{T_A(t)}, \quad m_B(t) = \frac{T_{gray}^B(t)}{T_B(t)}$$

- Tính phương sai giữa các nhóm:

$$\sigma^2(t) = P_A(t) (m_A(t) - m_G)^2 + P_B(t) (m_B(t) - m_G)^2$$

Trong đó $P_A(t)$ và $P_B(t)$ lần lượt là xác suất 1 pixel thuộc vùng A và B :

$$P_A(t) = \frac{T_A(t)}{T}, \quad P_B(t) = \frac{T_B(t)}{T}$$

4. Chọn ngưỡng t sao cho phương sai giữa các nhóm $\sigma^2(t)$ là lớn nhất:

$$t^* = \arg \max_t \sigma^2(t)$$

Ngưỡng tối ưu t^* là giá trị ngưỡng mà tại đó phương sai giữa các nhóm đạt giá trị lớn nhất.

2.5.3 Đầu ra

Thuật toán Otsu cho đầu ra là giá trị ngưỡng tối ưu t^* giúp phân đoạn ảnh thành vùng nền và vùng đối tượng. Ở đây, ta xét đối tượng là tập hợp những pixel có giá trị xám lớn hơn hoặc bằng t^* và nền là tập hợp những pixel có giá trị xám nhỏ hơn t^* .

Chương 3

Cài đặt, kiểm thử và đánh giá hệ thống

3.1 Thiết lập môi trường phát triển

Để đảm bảo mô hình được triển khai và đánh giá hiệu quả, việc thiết lập một môi trường phát triển phù hợp là yếu tố then chốt. Phần này cung cấp cái nhìn tổng quan về môi trường chạy và các thư viện đã được sử dụng, nhằm hỗ trợ quá trình xây dựng, huấn luyện, và kiểm tra mô hình một cách hiệu quả nhất.

3.1.1 Môi trường chạy

Việc thực hiện và triển khai mô hình được tiến hành trên nền tảng *Google Colab*. Đây là một môi trường phát triển trực tuyến do Google cung cấp, hỗ trợ mạnh mẽ cho các tác vụ tính toán chuyên sâu. Đặc biệt, để tăng tốc độ xử lý và huấn luyện mô hình, môi trường đã được thiết lập để sử dụng *GPU (Graphics Processing Unit)*. GPU giúp tăng tốc độ huấn luyện mô hình học sâu, đặc biệt đối với các bài toán xử lý ảnh phức tạp như phân đoạn ảnh y học.

3.1.2 Thư viện sử dụng

Trong quá trình phát triển mô hình và xử lý dữ liệu, các thư viện Python sau đã được sử dụng:

1. *Thư viện cơ bản:*

- `os`: Thư viện hỗ trợ thao tác với hệ thống tệp và thư mục.
- `time`: Được sử dụng để đo thời gian xử lý của các tác vụ.
- `numpy`: Hỗ trợ các tính toán đại số tuyến tính, mảng đa chiều, và các phép toán số học.

2. *Xử lý ảnh:*

- `cv2` (OpenCV): Dùng để đọc, xử lý, và hiển thị ảnh.
- `matplotlib.pyplot`: Sử dụng để trực quan hóa ảnh và biểu đồ.
- `sklearn.model_selection.train_test_split`: Hỗ trợ chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

3. *Xây dựng và huấn luyện mô hình:*

- `tensorflow`: Thư viện học sâu được sử dụng để xây dựng, huấn luyện và triển khai mô hình.
- `tensorflow.keras.models.Model`, `load_model`: Hỗ trợ xây dựng và tải mô hình.
- `tensorflow.keras.layers`: Bao gồm các lớp chính như `Input`, `Conv2D`, `MaxPooling2D`, `Conv2DTranspose`, và `concatenate` để xây dựng kiến trúc mô hình.
- `tensorflow.keras.callbacks`: Bao gồm `EarlyStopping` và `ReduceLROnPlateau` để kiểm soát quá trình huấn luyện, ngăn chặn hiện tượng overfitting và điều chỉnh tốc độ học.
- `tensorflow.keras.preprocessing.image`: Hỗ trợ xử lý và tạo dữ liệu hình ảnh.

Môi trường chạy Google Colab kết hợp với GPU và các thư viện mạnh mẽ như TensorFlow, OpenCV và các công cụ trực quan hóa dữ liệu giúp đảm bảo tính ổn định và hiệu quả trong việc phát triển và đánh giá mô hình phân đoạn ảnh y học. Việc sử dụng các thư viện này cũng giúp tối ưu hóa quy trình xử lý dữ liệu và xây dựng mô hình học sâu.

3.2 Chuẩn bị dữ liệu

3.2.1 Mô tả bộ dữ liệu

Bộ dữ liệu được sử dụng trong đề án này được trích xuất từ bài báo của Al-Dhabyani và cộng sự [8]. Bộ dữ liệu bao gồm các ảnh siêu âm vú thu thập từ năm 2018, với đối tượng là 600 bệnh nhân nữ trong độ tuổi từ 25 đến 75.

Bộ dữ liệu gồm tổng cộng 780 ảnh với kích thước trung bình là 500×500 pixel. Các ảnh được lưu trữ dưới định dạng PNG, bao gồm các ảnh siêu âm và ảnh phân đoạn khối u tương ứng, cho phép sử dụng trong các bài toán phân đoạn ảnh. Ở đây ta sử dụng 503 ảnh siêu âm có khối u và ảnh phân đoạn khối u tương ứng để huấn luyện mô hình và 144 ảnh còn lại để kiểm thử mô hình, so sánh ảnh phân đoạn dự đoán của mô hình với ảnh phân đoạn thực tế.

3.2.2 Hàm đọc ảnh X-quang và ảnh phân đoạn

Cài đặt

```
# Load X-ray images and segmentation masks
def load_data(xray_dir, mask_dir, image_size=(128, 128)):
    """
    Function to load X-ray images and corresponding segmentation masks.
    The images and masks are resized, normalized, and returned as numpy arrays.

    Args:
        xray_dir (str): Path to the directory containing X-ray images.
        mask_dir (str): Path to the directory containing segmentation masks.
        image_size (tuple): Target size for resizing images and masks (width, height).

    Returns:
        tuple: Two numpy arrays, one for images and one for masks.
    """
    x_images = [] # List to store preprocessed X-ray images
    y_masks = [] # List to store preprocessed masks

    # 1. Load and sort file names for X-rays and masks to ensure correct pairing
    xray_files = sorted(os.listdir(xray_dir))
    mask_files = sorted(os.listdir(mask_dir))

    # 2. Process each X-ray and its corresponding mask
    for xray_file, mask_file in zip(xray_files, mask_files):
        # Load and preprocess the X-ray image
```

```

img_path = os.path.join(xray_dir, xray_file)
img = image.load_img(img_path, target_size=image_size, color_mode='grayscale')
img_array = image.img_to_array(img) / 255.0
x_images.append(img_array)

# Load and preprocess the segmentation mask
mask_path = os.path.join(mask_dir, mask_file)
mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
mask = cv2.resize(mask, image_size)
mask = np.expand_dims(mask / 255.0, axis=-1)
y_masks.append(mask)

# 3. Convert the lists of images and masks to numpy arrays
x_images = np.array(x_images)
y_masks = np.array(y_masks)

# 4. Return the processed data
return x_images, y_masks

```

Mô tả

Hàm `load_data` có nhiệm vụ đọc các ảnh X-quang và ảnh phân đoạn từ các thư mục đã cho, xử lý chúng và trả về dưới dạng các mảng numpy.

1. Đầu vào

Hàm `load_data` nhận vào các tham số sau:

- `xray_dir`: Đường dẫn tới thư mục chứa các ảnh X-quang.
- `mask_dir`: Đường dẫn tới thư mục chứa các ảnh phân đoạn.
- `image_size`: Kích thước mục tiêu để thay đổi kích thước các ảnh và ảnh (mặc định là (128, 128)).

2. Quy trình thực hiện

Hàm thực hiện các bước sau:

- *Tải và sắp xếp các tệp tin*: Các tệp tin trong thư mục ảnh X-quang và ảnh phân đoạn được sắp xếp theo thứ tự để đảm bảo các cặp ảnh siêu âm và ảnh phân đoạn tương ứng được ghép đúng.

- *Tiền xử lý ảnh X-quang và ảnh phân đoạn*: - Mỗi ảnh X-quang được tải về, thay đổi kích thước theo `image_size` và chuẩn hóa các giá trị pixel về khoảng $[0, 1]$. - Mỗi ảnh phân đoạn cũng được tải về, thay đổi kích thước tương tự và chuẩn hóa. Ngoài ra, ảnh phân đoạn còn được mở rộng để có thêm một chiều cho phù hợp với định dạng ảnh.
- *Chuyển đổi thành mảng numpy*: Các danh sách ảnh siêu âm và ảnh phân đoạn đã tiền xử lý được chuyển thành mảng numpy để sử dụng trong các mô hình học sâu.

3. Đầu ra

Hàm trả về hai mảng numpy:

- `x_images`: Mảng numpy chứa các ảnh X-quang đã được xử lý.
- `y_masks`: Mảng numpy chứa các ảnh phân đoạn đã được xử lý.

3.2.3 Hàm hiển thị 5 mẫu dữ liệu ngẫu nhiên

Cài đặt

```
# Visualize random samples from the dataset
def visualize_samples(x_data, y_data, num_samples=5):
    """
    Function to display random samples from the dataset.
    This function will show the original image and its corresponding segmentation mask
    side by side.

    Args:
        x_data (numpy array): Preprocessed X-ray images.
        y_data (numpy array): Preprocessed segmentation masks.
        num_samples (int): Number of samples to display (default is 5).

    Returns:
        None
    """
    plt.figure(figsize=(15, num_samples * 3))

    # 1. Loop through the selected number of samples
    for i in range(num_samples):
        # 2. Display original image
        plt.subplot(num_samples, 2, 2 * i + 1)
```



```
plt.imshow(x_data[i].squeeze(), cmap='gray')
plt.title("Original Image")
plt.axis("off")

# 3. Display segmentation mask
plt.subplot(num_samples, 2, 2 * i + 2)
plt.imshow(y_data[i].squeeze(), cmap='gray')
plt.title("Segmentation Mask")
plt.axis("off")

# 4. Adjust layout for better display
plt.tight_layout()
plt.show()
```

Mô tả

Hàm `visualize_samples` có nhiệm vụ hiển thị các mẫu dữ liệu ngẫu nhiên từ tập dữ liệu, bao gồm cả ảnh gốc và ảnh phân đoạn tương ứng.

1. Đầu vào

Hàm `visualize_samples` nhận vào các tham số sau:

- **x_data**: Mảng numpy chứa các ảnh X-quang đã được xử lý.
- **y_data**: Mảng numpy chứa các ảnh phân đoạn đã được xử lý.
- **num_samples**: Số lượng mẫu muốn hiển thị (mặc định là 5).

2. Quy trình thực hiện

Hàm thực hiện các bước sau:

- *Lặp qua các mẫu dữ liệu*: Hàm sẽ lặp qua số mẫu yêu cầu (số mẫu mặc định là 5) để hiển thị.
- *Hiển thị ảnh gốc*: Mỗi ảnh X-quang trong **x_data** được hiển thị với màu sắc xám (gray) và không hiển thị trục.
- *Hiển thị ảnh phân đoạn*: Mỗi ảnh phân đoạn trong **y_data** được hiển thị song song với ảnh gốc, cũng với màu sắc xám và không hiển thị trục.
- *Điều chỉnh layout*: Đảm bảo các mẫu được hiển thị một cách rõ ràng và không bị chồng lấp.

3. Đầu ra

Hàm không trả về giá trị gì. Nó chỉ thực hiện việc hiển thị các mẫu dữ liệu dưới dạng hình ảnh. Các mẫu ảnh gốc và ảnh phân đoạn sẽ được hiển thị trực quan trong cửa sổ đồ họa.

3.2.4 Chia dữ liệu thành tập huấn luyện và tập kiểm tra

Cài đặt

```
# Split the data into training and validation sets (80% training, 20% validation)
x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size=0.2,
                                                  random_state=42)
```

Mô tả

Hàm `train_test_split` được sử dụng để chia tập dữ liệu thành hai phần: tập huấn luyện và tập kiểm tra. Trong trường hợp này, dữ liệu được chia theo tỷ lệ 80% cho huấn luyện và 20% cho kiểm tra.

1. Đầu vào

Hàm `train_test_split` nhận vào các tham số sau:

- **x_data**: Mảng numpy chứa các ảnh X-quang đã được xử lý (tập dữ liệu đầu vào).
- **y_data**: Mảng numpy chứa các ảnh phân đoạn đã được xử lý (nhãn tương ứng với dữ liệu).
- **test_size**: Tỷ lệ phần trăm của dữ liệu sẽ được sử dụng cho tập kiểm tra (mặc định là 0.2, tức là 20% cho kiểm tra).
- **random_state**: Hằng số ngẫu nhiên được sử dụng để đảm bảo tính lặp lại của quá trình chia dữ liệu.

2. Quy trình thực hiện

Hàm thực hiện các bước sau:

- *Chia dữ liệu ngẫu nhiên*: Dữ liệu được chia thành hai tập: một tập huấn luyện chiếm 80% dữ liệu và một tập kiểm tra chiếm 20% dữ liệu.
- *Đảm bảo tính lặp lại*: Để đảm bảo rằng quá trình chia dữ liệu luôn tạo ra kết quả giống nhau khi chạy lại, tham số `random_state` được thiết lập một giá trị cố định (ở đây là 42).

3. Dữ liệu

Hàm trả về bốn mảng numpy:

- `x_train`: Mảng numpy chứa các ảnh X-quang được sử dụng cho việc huấn luyện.
- `x_val`: Mảng numpy chứa các ảnh X-quang được sử dụng cho việc kiểm tra.
- `y_train`: Mảng numpy chứa các ảnh phân đoạn tương ứng với các ảnh huấn luyện.
- `y_val`: Mảng numpy chứa các ảnh phân đoạn tương ứng với các ảnh kiểm tra.

3.3 Xây dựng mô hình kiến trúc U-Net

Cài đặt

```
# Build the U-Net model with an input size of 128x128
def build_unet_model(input_size=(128, 128, 1)):
    inputs = Input(shape=input_size)

    # 1. Encoder
    conv1 = Conv2D(32, kernel_size=3, activation='relu', padding='same')(inputs)
    conv1 = Conv2D(32, kernel_size=3, activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(64, kernel_size=3, activation='relu', padding='same')(pool1)
    conv2 = Conv2D(64, kernel_size=3, activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(128, kernel_size=3, activation='relu', padding='same')(pool2)
    conv3 = Conv2D(128, kernel_size=3, activation='relu', padding='same')(conv3)
```

```

pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

conv4 = Conv2D(256, kernel_size=3, activation='relu', padding='same')(pool3)
conv4 = Conv2D(256, kernel_size=3, activation='relu', padding='same')(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

conv5 = Conv2D(512, kernel_size=3, activation='relu', padding='same')(pool4)
conv5 = Conv2D(512, kernel_size=3, activation='relu', padding='same')(conv5)

# 2. Decoder
up6 = Conv2DTranspose(256, kernel_size=2, strides=(2, 2), padding='same')(conv5)
merge6 = concatenate([up6, conv4], axis=3)
conv6 = Conv2D(256, kernel_size=3, activation='relu', padding='same')(merge6)
conv6 = Conv2D(256, kernel_size=3, activation='relu', padding='same')(conv6)

up7 = Conv2DTranspose(128, kernel_size=2, strides=(2, 2), padding='same')(conv6)
merge7 = concatenate([up7, conv3], axis=3)
conv7 = Conv2D(128, kernel_size=3, activation='relu', padding='same')(merge7)
conv7 = Conv2D(128, kernel_size=3, activation='relu', padding='same')(conv7)

up8 = Conv2DTranspose(64, kernel_size=2, strides=(2, 2), padding='same')(conv7)
merge8 = concatenate([up8, conv2], axis=3)
conv8 = Conv2D(64, kernel_size=3, activation='relu', padding='same')(merge8)
conv8 = Conv2D(64, kernel_size=3, activation='relu', padding='same')(conv8)

up9 = Conv2DTranspose(32, kernel_size=2, strides=(2, 2), padding='same')(conv8)
merge9 = concatenate([up9, conv1], axis=3)
conv9 = Conv2D(32, kernel_size=3, activation='relu', padding='same')(merge9)
conv9 = Conv2D(32, kernel_size=3, activation='relu', padding='same')(conv9)

outputs = Conv2D(1, kernel_size=1, activation='sigmoid')(conv9)

model = Model(inputs=inputs, outputs=outputs)
return model

# Build the model
model = build_unet_model()

```

Mô tả

Mô hình U-Net được xây dựng để thực hiện phân đoạn ảnh, đặc biệt là đối với các tác vụ phân đoạn y tế như phân đoạn các bộ phận cơ thể từ ảnh X-quang. Mô hình sử dụng kiến trúc *encoder-decoder* với các lớp convolution và pooling ở phần encoder để rút trích đặc trưng, và các lớp convolution transpose kết hợp với các lớp convolution ở phần decoder để tái tạo lại ảnh đầu ra.

1. Đầu vào

Mô hình nhận đầu vào là các ảnh có kích thước 128x128 với 1 kênh (ảnh xám).

- `input_size`: Kích thước của ảnh đầu vào, mặc định là (128, 128, 1).

2. Quy trình thực hiện

Mô hình bao gồm các bước sau:

- *Encoder*: Các lớp convolution và max-pooling được sử dụng để giảm độ phân giải của ảnh và rút trích các đặc trưng quan trọng. Các lớp convolution được sử dụng với kích thước kernel là 3x3 và hàm kích hoạt ReLU.
- *Decoder*: Các lớp convolution transpose được sử dụng để tăng độ phân giải của ảnh, đồng thời kết hợp với các đặc trưng từ phần encoder để cải thiện độ chính xác của phân đoạn. Các lớp convolution tiếp theo tiếp tục sử dụng ReLU để tạo ra đầu ra cuối cùng.
- Sử dụng một lớp tích chập 1x1 để giảm số lượng kênh đầu ra và áp dụng hàm kích hoạt sigmoid để đưa ra xác suất cho từng pixel thuộc về lớp mục tiêu.

3. Đầu ra

Đầu ra của hàm `build_unet_model` là mô hình U-Net đã được xây dựng hoàn chỉnh, sẵn sàng để huấn luyện.

3.4 Huấn luyện mô hình kiến trúc U-Net

3.4.1 Biên dịch mô hình

Cài đặt

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc', 'mse'])
```

Mô tả

Sau khi xây dựng xong mô hình U-Net, ta tiến hành biên dịch mô hình với các tham số sau:

- *Optimizer*: Sử dụng thuật toán tối ưu Adam, một trong những thuật toán tối ưu phổ biến cho các mô hình học sâu, với khả năng điều chỉnh tốc độ học động thích ứng.
- *Loss function*: Sử dụng hàm mất mát `binary_crossentropy`, phù hợp với các bài toán phân loại nhị phân, đặc biệt là phân đoạn ảnh.
- *Metrics*: Đo lường độ chính xác (`acc`) và trung bình bình phương sai số (`mse`) để đánh giá hiệu suất mô hình trong quá trình huấn luyện.

3.4.2 Thiết lập Callback

Cài đặt

```
early_stop = EarlyStopping(monitor='val_loss', patience=80, verbose=1)
```

Mô tả

Để cải thiện hiệu quả huấn luyện và tránh việc overfitting, ta sử dụng các callback *EarlyStopping*: Dừng quá trình huấn luyện sớm nếu hàm mất mát trên tập validation không cải thiện sau một số epoch nhất định (tại đây là 80 epoch). Điều này giúp tiết kiệm thời gian và tài nguyên, đồng thời tránh việc mô hình học quá mức.

3.4.3 Quá trình huấn luyện mô hình

Tham số huấn luyện

- *Epochs*: Huấn luyện trong 300 epoch, mỗi epoch là một lần lặp qua toàn bộ dữ liệu huấn luyện. Việc huấn luyện trong nhiều epoch giúp mô hình có đủ thời gian học và cải thiện khả năng dự đoán.

- *Batch size*: Sử dụng kích thước batch là 16, tức là mỗi lần huấn luyện, mô hình sẽ cập nhật trọng số dựa trên trung bình của 16 mẫu dữ liệu. Điều này giúp tiết kiệm bộ nhớ và tăng tốc quá trình huấn luyện.
- *Validation data*: Sử dụng tập validation (`x_val`, `y_val`) để đánh giá mô hình trong suốt quá trình huấn luyện. Tập validation giúp kiểm tra xem mô hình có đang học đúng và không bị overfitting hay không.
- *ReduceLROnPlateau*: Điều chỉnh tốc độ học động giảm xuống nếu hàm mất mát không cải thiện trong một số epoch nhất định (tại đây là 5 epoch). Điều này giúp mô hình tối ưu tốt hơn ở các bước huấn luyện tiếp theo.

Quá trình huấn luyện mô hình sẽ được ghi lại thời gian và thực hiện qua đoạn mã dưới đây:

```
start_time = time.time()
history = model.fit(
    x_train, y_train,
    validation_data=(x_val, y_val),
    epochs=300,
    batch_size=16,
    callbacks=[ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, verbose
    =1, min_lr=1e-5),
                early_stop]
)
```

Cuối cùng, tổng thời gian huấn luyện sẽ được tính toán và in ra để có cái nhìn tổng quan về quá trình huấn luyện. Thời gian huấn luyện được tính bằng cách ghi nhận thời gian bắt đầu và kết thúc huấn luyện, sau đó tính toán tổng thời gian đã sử dụng:

```
end_time = time.time()
total_time_minutes = (end_time - start_time) / 60.0
print(f"Total training time is: {total_time_minutes:.2f} minutes")
```

Kết quả

Thông số kết quả

- *acc (Accuracy)*: Độ chính xác trên tập huấn luyện, biểu thị tỷ lệ dự đoán đúng. Ví dụ: `acc: 0.9795 (97.95%)`.

```

Epoch 114/300
26/26 — 3s 83ms/step - acc: 0.9814 - loss: 0.0423 - mse: 0.0123 - val_acc: 0.9610 - val_loss: 0.1759 - val_mse: 0.0302 - learning_rate: 1.0000e-05
Epoch 115/300
26/26 — 3s 82ms/step - acc: 0.9803 - loss: 0.0444 - mse: 0.0130 - val_acc: 0.9609 - val_loss: 0.1765 - val_mse: 0.0303 - learning_rate: 1.0000e-05
Epoch 116/300
26/26 — 2s 81ms/step - acc: 0.9800 - loss: 0.0451 - mse: 0.0132 - val_acc: 0.9610 - val_loss: 0.1760 - val_mse: 0.0303 - learning_rate: 1.0000e-05
Epoch 117/300
26/26 — 2s 81ms/step - acc: 0.9806 - loss: 0.0443 - mse: 0.0129 - val_acc: 0.9610 - val_loss: 0.1780 - val_mse: 0.0303 - learning_rate: 1.0000e-05
Epoch 118/300
26/26 — 2s 81ms/step - acc: 0.9795 - loss: 0.0463 - mse: 0.0135 - val_acc: 0.9609 - val_loss: 0.1783 - val_mse: 0.0303 - learning_rate: 1.0000e-05
Epoch 118: early stopping
Total training time is: 5.17 minutes

```

Hình 3.1: Kết quả huấn luyện mô hình

- *loss*: Hàm mất mát trên tập huấn luyện, biểu thị mức độ khác biệt giữa dự đoán và nhãn thực tế. Ví dụ: `loss: 0.0463`.
- *mse* (*Mean Square Error*): Sai số bình phương trung bình trên tập huấn luyện, dùng để đo lường mức độ sai lệch. Ví dụ: `mse: 0.0135`.
- *val_acc* (*Validation Accuracy*): Độ chính xác trên tập validation. Ví dụ: `val_acc: 0.9609` (96.09%).
- *val_loss* (*Validation Loss*): Hàm mất mát trên tập validation, biểu thị độ chính xác của mô hình trên dữ liệu chưa thấy. Ví dụ: `val_loss: 0.1783`.
- *val_mse* (*Validation Mean Squared Error*): Sai số bình phương trung bình trên tập validation. Ví dụ: `val_mse: 0.0303`.
- *learning_rate*: Tốc độ học, biểu thị mức độ điều chỉnh của trọng số mô hình. Ví dụ: `1.0000e-05` (0.00001).
- *Total training time*: Tổng thời gian huấn luyện là **5.17 minutes**.

Đánh giá kết quả

- Mô hình có độ chính xác cao trên tập huấn luyện (`acc: 0.9795`) và tập validation (`val_acc: 0.9609`), cho thấy khả năng dự đoán tốt.
- Sự chênh lệch giữa `loss` (0.0463) và `val_loss` (0.1783) cho thấy mô hình có thể đang bắt đầu **overfitting**, tức là hiệu suất trên tập validation kém hơn trên tập huấn luyện.
- Giá trị `val_mse` (0.0303) phù hợp với `mse` trên tập huấn luyện (0.0135), chứng tỏ mô hình có khả năng dự đoán tốt nhưng có thể cải thiện thêm để

giảm độ sai lệch.

- Kỹ thuật *Early Stopping* đã giúp giảm thời gian huấn luyện và ngăn chặn *overfitting*.
- Tốc độ học (`learning_rate: 1.0000e-05`) nhỏ, đảm bảo cập nhật trọng số ổn định nhưng cũng có thể dẫn đến thời gian hội tụ lâu hơn.
- Tổng thời gian huấn luyện hợp lý (5.17 phút) với 118 epoch, nhờ tối ưu cấu hình và tính toán.

3.5 Hậu xử lý ảnh đầu ra của mô hình U-Net bằng thuật toán Otsu

Sau khi mô hình U-Net thực hiện phân đoạn, ảnh đầu ra thường vẫn còn mờ hoặc nhòe, do giá trị xác suất cho từng pixel chưa rõ ràng, dẫn đến việc phân biệt đối tượng và nền không hoàn toàn chính xác. Để cải thiện kết quả phân đoạn, ta áp dụng thuật toán Otsu. Thuật toán Otsu giúp tự động tìm ngưỡng tối ưu để phân tách ảnh thành hai lớp rõ ràng: nền và đối tượng. Việc sử dụng Otsu sau khi phân đoạn không chỉ giúp làm sắc nét kết quả, mà còn tối ưu hóa việc phân tách các đối tượng khỏi nền, mang lại ảnh nhị phân rõ ràng và chính xác hơn.

Cài đặt

```
def otsu_threshold(img):
    """
    Apply Otsu's thresholding to binarize the image.
    """
    # Blur the image to reduce noise
    blurred_img = cv2.GaussianBlur(img, (5, 5), 0)

    # Apply Otsu's thresholding to binarize the image
    _, otsu_img = cv2.threshold(blurred_img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    return otsu_img
```

Mô tả

Hàm `otsu_threshold` sử dụng thuật toán Otsu để phân ngưỡng và chuyển ảnh đầu vào thành ảnh nhị phân, phân tách đối tượng và nền.

1. Đầu vào

Hàm `otsu_threshold` nhận ảnh đầu vào (ảnh xám) cần phân ngưỡng.

2. Quy trình thực hiện

Hàm thực hiện các bước sau:

- *Làm mờ ảnh*: Áp dụng bộ lọc Gaussian với kích thước kernel (5, 5) để làm mờ ảnh và giảm nhiễu, giúp cải thiện kết quả phân ngưỡng.
- *Áp dụng phân ngưỡng Otsu*: Sử dụng hàm `cv2.threshold()` với cờ `cv2.THRESH_OTSU` để tự động tính toán ngưỡng tối ưu và phân ngưỡng ảnh.

3. Đầu ra

Hàm trả về ảnh nhị phân với các mức xám pixel của đối tượng được đặt thành 255 và mức xám pixel của nền được đặt thành 0.

3.6 Các tiêu chí đánh giá

3.6.1 Chất lượng phân đoạn

Chất lượng phân đoạn được thể hiện qua hai khía cạnh: hiệu suất phân đoạn của chương trình và chất lượng phân đoạn của từng ảnh. Hiệu suất phân đoạn được định nghĩa là tỷ lệ giữa số lượng ảnh được phân đoạn tốt và tổng số ảnh được phân đoạn. Chất lượng của từng ảnh được đánh giá thông qua chỉ số IoU , với quy định rằng ảnh có IoU lớn hơn hoặc bằng 0.5 được coi là phân đoạn tốt.

Chất lượng phân đoạn của từng ảnh

Chỉ số *IoU* được sử dụng để đánh giá mức độ trùng khớp giữa ảnh phân đoạn thực tế và ảnh phân đoạn dự đoán. IoU được định nghĩa như sau:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

Trong đó:

- A : Vùng đối tượng của ảnh phân đoạn thực tế.
- B : Vùng đối tượng của ảnh phân đoạn dự đoán.
- $|A \cap B|$: Diện tích vùng giao nhau giữa A và B .
- $|A \cup B|$: Diện tích vùng hợp giữa A và B .

Một ảnh được coi là "phân đoạn tốt" nếu chỉ số IoU của nó lớn hơn hoặc bằng 0.5. Chỉ số này phản ánh trực tiếp mức độ chính xác của phân đoạn từng ảnh trong tập kiểm thử.

Ta xây dựng đoạn mã thực hiện tính toán chỉ số IoU như sau:

```
def calculate_iou(true_mask, pred_mask):
    """
    Calculate the Intersection over Union (IoU) score.
    """
    true_mask_resized = cv2.resize(true_mask, (pred_mask.shape[1], pred_mask.shape[0]))
    )

    true_mask_bin = (true_mask_resized > 0.5).astype(np.uint8)
    pred_mask_bin = (pred_mask > 0.5).astype(np.uint8)

    # Calculate IoU
    intersection = np.sum(true_mask_bin * pred_mask_bin)
    union = np.sum(true_mask_bin) + np.sum(pred_mask_bin) - intersection
    iou = intersection / union if union != 0 else 0

    return iou
```

Hàm `calculate_iou` được sử dụng để tính toán chỉ số IoU cho các ảnh phân đoạn. Các bước thực hiện bao gồm:

1. Đầu vào

Hàm nhận vào hai tham số:

- `true_mask`: Ảnh phân đoạn thực tế, chứa thông tin về các đối tượng trong ảnh gốc.

- `pred_mask`: Ảnh phân đoạn dự đoán từ mô hình.

2. Quy trình thực hiện

- *Kích thước lại ảnh phân đoạn thực tế*: Đảm bảo kích thước của ảnh phân đoạn thực tế trùng khớp với kích thước của ảnh phân đoạn dự đoán.
- *Nhị phân hóa ảnh phân đoạn*: Chuyển đổi cả ảnh phân đoạn thực tế và ảnh phân đoạn dự đoán thành dạng nhị phân (0 hoặc 1) với ngưỡng 0.5.
- *Tính toán IoU*.

3. Đầu ra

Chỉ số IoU đánh giá sự tương đồng giữa hai ảnh phân đoạn.

Hiệu suất phân đoạn

Hiệu suất phân đoạn thể hiện khả năng tổng thể của chương trình trong việc phân đoạn các ảnh X-quang. Hiệu suất H được tính theo công thức:

$$H = \frac{\text{Số lượng ảnh phân đoạn tốt}}{\text{Tổng số ảnh được phân đoạn}} \times 100(\%)$$

Trong đó:

- Số lượng ảnh phân đoạn tốt: Số ảnh có IoU lớn hơn hoặc bằng 0.5.
- Tổng số ảnh được phân đoạn: Tổng số ảnh trong tập kiểm thử.

Chỉ số hiệu suất phân đoạn giúp đánh giá mức độ hiệu quả của chương trình khi áp dụng trên toàn bộ tập dữ liệu.

3.6.2 Thời gian phân đoạn

Thời gian phân đoạn là một tiêu chí quan trọng để đánh giá hiệu suất của chương trình. Tiêu chí này được chia thành hai khía cạnh: thời gian phân đoạn của từng ảnh và thời gian phân đoạn tổng thể.

Thời gian phân đoạn từng ảnh

Thời gian phân đoạn từng ảnh phản ánh thời gian mà chương trình cần để xử lý và phân đoạn một ảnh X-quang cụ thể. Ta sẽ đánh giá:

- *Thời gian phân đoạn nhỏ nhất*: Đây là thời gian ngắn nhất mà chương trình cần để phân đoạn một ảnh trong tập dữ liệu.
- *Thời gian phân đoạn lớn nhất*: Đây là thời gian dài nhất mà chương trình cần để phân đoạn một ảnh trong tập dữ liệu.

Các giá trị này giúp xác định khả năng xử lý tối thiểu và tối đa của chương trình, từ đó đánh giá độ ổn định của hệ thống.

Thời gian phân đoạn tổng thể

Thời gian phân đoạn tổng thể là tổng thời gian mà chương trình cần để hoàn thành phân đoạn tất cả các ảnh trong tập dữ liệu kiểm thử. Hai chỉ số quan trọng trong tiêu chí này bao gồm:

- *Tổng thời gian phân đoạn*: Được tính bằng công thức:

$$T_{\text{sum}} = \sum_{i=1}^N t_i$$

Trong đó:

- T_{sum} : Tổng thời gian phân đoạn.
- t_i : Thời gian phân đoạn của ảnh thứ i .
- N : Tổng số ảnh trong tập kiểm thử.

- *Thời gian phân đoạn trung bình*: Được tính bằng công thức:

$$T_{\text{mean}} = \frac{T_{\text{sum}}}{N}$$

Trong đó:

- T_{mean} : Thời gian phân đoạn trung bình cho từng ảnh.

- T_{sum} : Tổng thời gian phân đoạn.
- N : Tổng số ảnh trong tập kiểm thử.

Những chỉ số này cung cấp cái nhìn tổng quát về hiệu suất thời gian của chương trình, giúp đánh giá khả năng ứng dụng thực tế của hệ thống.

3.7 Kiểm thử

3.7.1 Thực hiện phân đoạn ảnh theo mô hình U-Net

Cài đặt

```
# Paths to image and mask directories
img_folder_test = '/content/drive/MyDrive/data_cancer_test/img'
mask_folder_test = '/content/drive/MyDrive/data_cancer_test/mask'

img_files_test = sorted(os.listdir(img_folder_test))

# Load the pre-trained U-Net model
model = load_model('/content/drive/MyDrive/unet.keras')
print("Model loaded successfully!")

processed_count = 0
good_images = 0
segmentation_times = [] # List to store segmentation times

# Iterate through each image file
for img_file in img_files_test:
    img_path = os.path.join(img_folder_test, img_file)
    mask_name = img_file.split('.')[0] + '_mask.png'
    mask_path = os.path.join(mask_folder_test, mask_name)

    # Load and preprocess the image
    img = image.load_img(img_path, target_size=(128, 128), color_mode='grayscale')
    img_array = image.img_to_array(img).squeeze()
    img_array_test = np.expand_dims(img_array, axis=0) / 255.0

    # Load the true mask
    true_mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
    if true_mask is None:
        print(f"Cannot read the mask file: {mask_path}. Skipping this pair.")
        continue

    true_mask = cv2.resize(true_mask, (128, 128)) / 255.0

    start_time = time.time() # Start time for segmentation
```

```

# Predict the segmentation mask using the U-Net model
pred_mask = model.predict(img_array_test)
pred_mask = np.squeeze(pred_mask, axis=0)

# Apply Otsu's thresholding on the predicted mask
pred_mask_otsu = otsu_threshold((pred_mask * 255).astype(np.uint8)) / 255.0

# Calculate IoU score
iou = calculate_iou(true_mask, pred_mask_otsu)

# Count "good" images (IoU >= 0.5)
if iou >= 0.5:
    good_images += 1

end_time = time.time() # End time for segmentation
segmentation_times.append(end_time - start_time) # Store segmentation time

# Display the images and metrics
plt.figure(figsize=(15, 5))

# Original image
plt.subplot(1, 3, 1)
plt.imshow(img_array, cmap='gray')
plt.title('Original Image')
plt.axis('off')

# True mask
plt.subplot(1, 3, 2)
plt.imshow(true_mask, cmap='gray')
plt.title(f'True Mask\nIoU: {iou:.2f}')
plt.axis('off')

# Predicted mask (Otsu)
plt.subplot(1, 3, 3)
plt.imshow(pred_mask_otsu, cmap='gray')
plt.title('Predicted Mask')
plt.axis('off')

plt.tight_layout()
plt.show()

processed_count += 1

# Summary of results
if segmentation_times:
    max_time = max(segmentation_times)
    min_time = min(segmentation_times)
    total_time = sum(segmentation_times)

```

```

mean_time = total_time / len(segmentation_times)

print(f"\nSegmentation Results:")
print(f"   Number of images processed: {processed_count}/{len(img_files_test)}")
print(f"   Number of good segmentations (IoU >= 0.5): {good_images}")
print(f"   Segmentation performance: {good_images / processed_count * 100:.2f}%")

print(f"\nSegmentation Time Statistics:")
print(f"   Maximum time: {max_time:.4f} seconds")
print(f"   Minimum time: {min_time:.4f} seconds")
print(f"   Total time: {total_time:.4f} seconds")
print(f"   Average time: {mean_time:.4f} seconds per image")
else:
    print("No images were processed.")

```

Mô tả

Trong mục này, chúng ta sử dụng mô hình học sâu U-Net để phân đoạn các ảnh X-quang. Hiệu suất phân đoạn được đánh giá thông qua chỉ số là IoU. Với mỗi cặp ảnh phân đoạn thực tế và dự đoán, nếu IoU lớn hơn hoặc bằng 0.5, thì ảnh phân đoạn được coi là tốt.

1. Đầu vào

- `img_folder_test`: Đường dẫn đến thư mục chứa các ảnh X-quang dùng để phân đoạn.
- `mask_folder_test`: Đường dẫn đến thư mục chứa các ảnh phân đoạn thực tế (True Segmentation Masks).
- `model`: Mô hình U-Net đã được huấn luyện và lưu trữ trong một tệp `.keras`.

2. Quy trình thực hiện

Quy trình thực hiện phân đoạn ảnh và tính toán các chỉ số được mô tả như sau:

- *Tải và xử lý ảnh và ảnh phân đoạn thực tế*: Mỗi ảnh X-quang và ảnh phân đoạn thực tế tương ứng được tải về và thay đổi kích thước về (128,

128). Ảnh X-quang được chuyển đổi thành mảng numpy và chuẩn hóa về khoảng $[0, 1]$.

- *Dự đoán ảnh phân đoạn*: Mô hình U-Net dự đoán ảnh phân đoạn từ ảnh X-quang đã được chuẩn hóa.
- *Áp dụng phân đoạn Otsu*: Ảnh phân đoạn dự đoán được áp dụng phương pháp phân đoạn Otsu để tạo ảnh phân đoạn rõ ràng hơn.
- *Tính toán chỉ số IoU*: Chỉ số IoU được tính toán cho từng cặp ảnh phân đoạn thực tế và dự đoán. Ảnh phân đoạn được đánh giá là "tốt" nếu chỉ số IoU lớn hơn hoặc bằng 0.5.

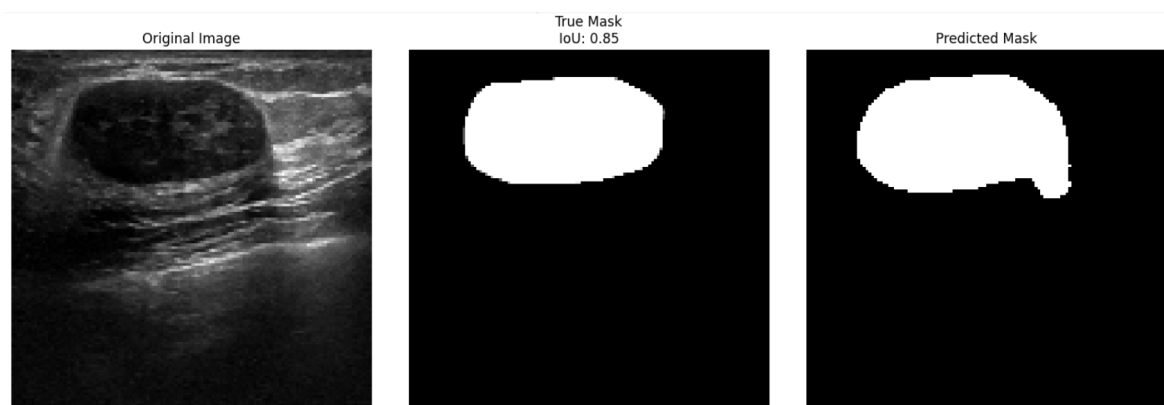
3. Đầu ra

Hàm trả về các kết quả sau:

- `processed_count`: Tổng số ảnh đã được xử lý.
- `good_images`: Số lượng ảnh phân đoạn "tốt" ($\text{IoU} \geq 0.5$).
- Các hình ảnh và ảnh phân đoạn thực tế và dự đoán được hiển thị kèm theo chỉ số IoU
- Các thống kê về chất lượng phân đoạn và thời gian phân đoạn.

3.7.2 Kết quả kiểm thử

Kết quả phân đoạn cho mỗi ảnh thu được như sau:



Hình 3.2: Kết quả phân đoạn mỗi ảnh

Kết quả tổng hợp thu được như sau:

```
Segmentation Results:
Number of images processed: 144/144
Number of good segmentations (IoU >= 0.5): 91
Segmentation performance: 63.19%

Segmentation Time Statistics:
Maximum time: 0.9031 seconds
Minimum time: 0.0569 seconds
Total time: 13.5479 seconds
Average time: 0.0941 seconds per image
```

Hình 3.3: Kết quả tổng hợp

Chất lượng phân đoạn

Kết quả phân đoạn được đánh giá thông qua chỉ số IoU. Các ảnh được phân đoạn tốt khi đạt chỉ số này lớn hơn hoặc bằng 0.5.

- Tổng số ảnh đã được phân đoạn: 144/144.
- Số ảnh phân đoạn tốt ($\text{IoU} \geq 0.5$): 91 ảnh.
- Hiệu suất phân đoạn: 63.19%.

Thời gian phân đoạn

Thời gian phân đoạn trung bình và các chỉ số liên quan đến thời gian giúp đánh giá hiệu suất của quá trình phân đoạn ảnh. Dưới đây là các thông số liên quan đến thời gian phân đoạn:

- Thời gian phân đoạn lớn nhất: 0.9031 giây.
- Thời gian phân đoạn nhỏ nhất: 0.0569 giây.
- Tổng thời gian phân đoạn: 13.5479 giây.
- Thời gian phân đoạn trung bình: 0.0941 giây/ảnh.

3.7.3 Đánh giá kết quả

Chất lượng phân đoạn

Kết quả phân đoạn được đánh giá dựa trên chỉ số IoU với ngưỡng 0.5. Tổng cộng, có 144 ảnh đã được xử lý, trong đó 91 ảnh đạt chất lượng phân đoạn tốt ($\text{IoU} \geq 0.5$). Điều này cho thấy hiệu suất phân đoạn là 63.19 %, với một tỷ lệ đáng kể các phân đoạn đạt chất lượng tốt. Mặc dù vậy, vẫn còn một số ảnh có chất lượng phân đoạn thấp, điều này có thể liên quan đến các yếu tố như độ phức tạp của ảnh, chất lượng dữ liệu huấn luyện hoặc các yếu tố bên ngoài ảnh hưởng đến kết quả phân đoạn.

Thời gian phân đoạn

Về mặt thời gian, quá trình phân đoạn trên mỗi ảnh có sự biến động rõ rệt. Thời gian phân đoạn tối đa là 0.9031 giây, trong khi thời gian phân đoạn tối thiểu là 0.0569 giây. Tổng thời gian phân đoạn cho 144 ảnh là 13.5479 giây, với thời gian trung bình khoảng 0.0941 giây mỗi ảnh. Điều này cho thấy mô hình có thể xử lý nhanh chóng hầu hết các ảnh, nhưng cũng có một số trường hợp thời gian xử lý lâu hơn, có thể do đặc điểm của ảnh hoặc sự phức tạp của phân đoạn.

Kết luận

Đồ án đã đạt được mục tiêu đề ra

Đồ án của em đã đạt được mục tiêu đề ra là xây dựng thành công mô hình U-Net để phân đoạn khối u vú trong ảnh siêu âm. Bên cạnh đó, em cũng đã nghiên cứu và tìm hiểu được tổng quan về xử lý ảnh, phân đoạn ảnh, cùng với các kỹ thuật và phương pháp ứng dụng trong lĩnh vực này.

Kết quả của đồ án

1. Xây dựng mô hình U-Net sử dụng cho phân đoạn khối u vú trong ảnh siêu âm.
2. Đạt kết quả phân đoạn tốt với tỷ lệ 63,19% trên tập ảnh siêu âm đã thử nghiệm.
3. Tìm hiểu các kỹ thuật xử lý ảnh và phân đoạn ảnh, đồng thời nắm bắt được các xu hướng nghiên cứu trong lĩnh vực này.

Kỹ năng đạt được

1. Kỹ năng nghiên cứu và đọc tài liệu, tìm kiếm các phương pháp, thuật toán mới trong lĩnh vực xử lý ảnh y học.
2. Kỹ năng trình bày và viết báo cáo khoa học, cũng như khả năng truyền đạt các kết quả nghiên cứu một cách rõ ràng, dễ hiểu.
3. Kỹ năng lập trình và sử dụng các thư viện như TensorFlow, Keras để xây dựng mô hình học sâu cho phân đoạn ảnh.
4. Kỹ năng phân tích và đánh giá hiệu suất của mô hình thông qua các chỉ số

như độ chính xác, độ nhạy, độ đặc hiệu.

5. Kỹ năng làm việc với dữ liệu y tế, đặc biệt là xử lý và phân tích ảnh siêu âm.

Hướng phát triển của đề án trong tương lai

1. Cải tiến mô hình bằng cách thêm phần tiền xử lý ảnh, hiện tại phần này chưa được triển khai trong đề án.
2. Kết hợp thêm các mô hình học sâu khác như FCN (Fully Convolutional Network), DeepLab, hoặc Mask R-CNN để nâng cao độ chính xác của kết quả phân đoạn.
3. Triển khai mô hình lên một ứng dụng di động hoặc ứng dụng web phục vụ trong lĩnh vực y tế, giúp bác sĩ dễ dàng sử dụng để phát hiện và chẩn đoán khối u vú từ ảnh siêu âm.
4. Tiến hành thử nghiệm trên một bộ dữ liệu lớn hơn và đa dạng hơn để cải thiện khả năng tổng quát của mô hình.

Chỉ mục

- Attention Mechanism, 13
- Batch size, 55
- Biểu diễn và mô tả ảnh, 9
- Bước nhảy, 22
- Clustering, 12
- Clustering-based Cosegmentation, 12
- Cosegmentation based on Active Contours, 12
- Cosegmentation based on Graph Theory, 13
- Cosegmentation based on Random Walks, 12
- Cosegmentation based on Thermal Diffusion, 13
- Dilated Convolution, 13
- Edge Detection, 12
- Encoder-Decoder Architecture, 13
- Epochs, 54
- Graph Theory, 12
- Hàm phi tuyến, 27
- Hàm ReLU, 27
- Hàm Sigmoid, 28
- Hàm Tanh, 28
- Khoảng cách giữa các điểm ảnh, 17
- Lân cận của điểm ảnh, 16
- Lược đồ mức xám, 18
- Lớp Max Pooling, 37
- Lớp tích chập, 35
- Lớp tích chập chuyển vị, 37
- Ma trận bộ lọc, 21
- Ma trận ảnh đầu vào, 20
- Ma trận ảnh đặc trưng đầu ra, 25
- Max Pooling, 28
- MRF-based Cosegmentation, 12
- Multiscale Feature Extraction, 13
- Mạng U-Net, 34
- Mối liên kết của các điểm ảnh, 17
- Mức xám, 15
- Ngưỡng của ảnh, 19
- Nhánh mở rộng, 40
- Nhánh thu hẹp, 39
- Nhận dạng và nội suy, 10
- Object-based Cosegmentation, 13
- Padding, 23

Phân đoạn ảnh, 9

Quá trình xử lý ảnh, 8

Random Walks, 12

Region Division, 12

Skip Connections, 13

Số hóa ảnh, 15

Thu nhận ảnh, 9

Thuật toán Otsu, 41

Tiền xử lý, 9

Tích chập, 20

Tích chập chuyển vị, 31

Vị trí điểm ảnh, 15

Xử lý ảnh, 7

Điểm lân cận chéo, 17

Điểm lân cận theo chiều đứng và ngang,

17

Điểm ảnh (pixel), 15

Độ phân giải, 19

Ảnh màu, 16

Ảnh nhị phân, 15

Ảnh số, 15

Ảnh tự nhiên, 15

Ảnh xám n -bit, 15

Tài liệu tham khảo

- [1] Ngô Huy Chương, Đồ án tốt nghiệp *Tìm hiểu phương pháp phân đoạn ảnh y học*, Trường Đại học Dân lập Hải Phòng, 2012.
- [2] Ying Yu, Chunping Wang, Qiang Fu, Renke Kou, Fuyu Huang, Boxiong Yang, Tingting Yang, and Mingliang Gao, *Techniques and Challenges of Image Segmentation: A Review*, Electronics, 2023.
- [3] Nguyễn Quang Hoan, Bài giảng *Xử lý ảnh*, Học viện Công nghệ Bưu chính Viễn thông, 2006.
- [4] Keith Jack, *YCbCr to RGB Considerations*, 1997.
- [5] Vincent Dumoulin and Francesco Visin, *A Guide to Convolution Arithmetic for Deep Learning*, MILA, Université de Montréal and AIRLab, Politecnico di Milano, Mar. 24, 2016.
- [6] Yang Liu, Jianpeng Zhang, Chao Gao, Jinghua Qu, and Lixin Ji, *Natural-Logarithm-Rectified Activation Function in Convolutional Neural Networks*, arXiv preprint, 2019.
- [7] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany, 2015.
- [8] Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A., *Dataset of breast ultrasound images*, Data in Brief, 2020 Feb; 28:104863, DOI: 10.1016/j.dib.2019.104863.

- [9] World Health Organization. *Breast Cancer*. 2025.
- [10] The Lancet. *Global trends in breast cancer incidence and mortality*. 2025.
- [11] DeepMind. *International evaluation of an AI system for breast cancer screening*. 2020.