

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



ĐỒ ÁN TỐT NGHIỆP
NGÀNH KHOA HỌC MÁY TÍNH
ĐỀ TÀI: KỸ THUẬT PHÂN LOẠI KHỐI U NÃO DỰA TRÊN
HÌNH ẢNH BẰNG INCEPTIONV3

GVHD	: TS. Nguyễn Mạnh Cường
Sinh viên	: Ngô Quý Điệp – 2021602059
Mã số sinh viên	: 2021602059

Hà Nội – 2025

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



ĐỒ ÁN TỐT NGHIỆP
NGÀNH: KHOA HỌC MÁY TÍNH

ĐỀ TÀI: KỸ THUẬT PHÂN LOẠI KHỐI U NÃO DỰA TRÊN
HÌNH ẢNH BẰNG INCEPTIONV3

GVHD	: TS. Nguyễn Mạnh Cường
Sinh viên	: Ngô Quý Điệp – 2021602059
Mã số sinh viên	: 2021602059

Hà Nội – 2025

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT	
DANH MỤC BẢNG BIỂU	
DANH MỤC HÌNH ẢNH	
LỜI CẢM ƠN	
LỜI NÓI ĐẦU	1
CHƯƠNG 1: KHẢO SÁT VÀ PHÁT BIỂU BÀI TOÁN	4
1.1. Hiện trạng y học và bệnh u não tại Việt Nam	4
1.1.1. Hiện trạng y học Việt Nam	4
1.1.2. Hiện trạng bệnh u não ở Việt Nam	5
1.2. Mục tiêu của bài toán.....	9
1.3. Phát biểu bài toán	10
1.3.1. Đầu vào (input).....	10
1.3.2. Đầu ra (output).....	11
1.3.3. Các ràng buộc của bài toán.....	11
1.4. Ý nghĩa của bài toán đối với thực tế.....	13
1.5. Cơ hội và thách thức.....	14
1.5.1. Cơ hội.....	14
1.5.2. Thách thức.....	15
CHƯƠNG 2: MỘT SỐ KỸ THUẬT GIẢI QUYẾT BÀI TOÁN.....	17
2.1. Phương hướng tiếp cận bài toán	17
2.2. Một số kỹ thuật sử dụng để giải quyết bài toán.....	18
2.3. Máy vector hỗ trợ (Support Vector Machine - SVM)	19
2.4. Mạng nơ ron tích chập (Convolution Neural Networks - CNNs)	22
2.4.1. Tổng quan	22
2.4.2. Cấu trúc	23
2.4.3. Cách chọn tham số	42

2.5. Mạng nơ ron CNN - VGG16.....	42
2.5.1. Giới thiệu	42
2.5.2. Kiến trúc mạng VGG16.....	43
2.5.3. Ưu và nhược điểm.....	44
2.6. Mạng nơ ron CNN – InceptionV3.....	45
2.6.1. Giới thiệu	45
2.6.2. Kiến trúc mạng InceptionV3.....	46
2.6.3. Ưu và nhược điểm.....	50
CHƯƠNG 3: THỰC NGHIỆM	52
3.1. Dữ liệu thực nghiệm	52
3.2. Tiền xử lý dữ liệu	54
3.3. Chuẩn bị dữ liệu cho mô hình	55
3.4. Huấn luyện các mô hình	56
3.4.1. Mô hình mạng nơ ron VGG16.....	56
3.4.2. Mô hình mạng nơ ron InceptionV3	58
3.5. Các kết quả thực nghiệm	60
CHƯƠNG 4: XÂY DỰNG SẢN PHẨM DEMO	64
4.1. Giới thiệu các framework sử dụng	64
4.1.1. ReactJS.....	64
4.1.2. FastAPI	65
4.2. Phân tích thiết kế hệ thống	66
4.2.1. Biểu đồ use case tổng quát	66
4.2.2. Mô tả chi tiết use case.....	67
4.3. Giao diện hệ thống	72
4.4. Chức năng chính của hệ thống	76
KẾT LUẬN	78
TÀI LIỆU THAM KHẢO.....	79

DANH MỤC CÁC TỪ VIẾT TẮT

AI	Artificial Intelligence
CNN	Convolutional Neural Networks
MRI	Magnetic Resonance Imaging
SVM	Support Vector Machine
FC	Fully Connected

DANH MỤC BẢNG BIỂU

Bảng 3.1 Mô tả dữ liệu.....	52
Bảng 3.2 So sánh kết quả trực quan hóa mô hình VGG16 với InceptionV3.....	61
Bảng 3.3 So sánh kết quả trên tập test.....	62

DANH MỤC HÌNH ẢNH

Hình 2.1 Minh họa thuật toán SVM.....	20
Hình 2.2 Minh họa kiến trúc mạng CNN.....	24
Hình 2.3 Minh họa độ trượt (stride).....	25
Hình 2.4 Minh họa kiểu tích chập 2 chiều.....	26
Hình 2.5: Minh họa kiểu tích chập theo chiều sâu.....	27
Hình 2.6: Minh họa kiểu tích chập theo từng điểm.....	28
Hình 2.7: Minh họa tích chập phân tách theo chiều sâu	29
Hình 2.8: Minh họa tích chập giãn.....	30
Hình 2.9: Minh họa tích chập giãn từng phần.....	31
Hình 2.10: Minh họa tích chập chuyển đổi.....	31
Hình 2.11: Minh họa Max pooling.....	33
Hình 2.12 Minh họa Average pooling.....	33
Hình 2.13: Minh họa Global average pooling.....	34
Hình 2.14: Minh họa phép làm phẳng.....	35
Hình 2.15: Minh họa phép làm phẳng với các tầng nơ ron.....	36
Hình 2.16: Minh họa hàm kích hoạt ReLU.....	37
Hình 2.17: Minh họa hàm kích hoạt Sigmoid.....	38
Hình 2.18: Minh họa hàm kích hoạt Tanh.....	38
Hình 2.19: Minh họa hàm kích hoạt GELU.....	39
Hình 2.20: Minh họa hàm kích hoạt SiLU.....	40
Hình 2.21: Minh họa hàm kích hoạt ReLU6.....	42
Hình 2.22: Minh họa kiến trúc mạng VGG16.....	43
Hình 2.23: Inception Module cơ bản.....	46
Hình 2.24: Inception Module với Convolution 1x1.....	47
Hình 2.25: Mô hình InceptionV1 với các Inception module.....	47
Hình 2.26: Mô hình InceptionV3 với các Inception module.....	49

Hình 2.27: Simple và Spatial Separable Convolution.....	49
Hình 3.1 Một số hình ảnh về u não trong bộ dữ liệu Brain Tumor MRI.....	53
Hình 3.2 Ảnh code tiền xử lý dữ liệu.....	54
Hình 3.3 Ví dụ về ảnh được huấn luyện qua từng epoch.....	55
Hình 3.4 Ảnh code chuẩn bị dữ liệu huấn luyện.....	55
Hình 3.5 Cấu trúc mô hình VGG16.....	56
Hình 3.6 Huấn luyện mô hình VGG16.....	57
Hình 3.7 Quá trình huấn luyện mô hình VGG16 qua các epoch.....	58
Hình 3.8 Cấu trúc mô hình InceptionV3.....	58
Hình 3.9 Huấn luyện mô hình InceptionV3.....	59
Hình 3.10 Quá trình huấn luyện mô hình InceptionV3 qua các epoch.....	60
Hình 3.11 Ảnh code trực quan hóa mô hình.....	60
Hình 3.12 Ảnh code xử lý dữ liệu test để phù hợp với mô hình.....	62
Hình 4.1 Giới thiệu framework ReactJS.....	64
Hình 4.2 Giới thiệu framework FastAPI.....	65
Hình 4.3 Biểu đồ use case tổng quát.....	66
Hình 4.4 Giao diện màn hình đăng nhập.....	73
Hình 4.5 Giao diện màn hình đăng ký.....	73
Hình 4.6 Giao diện trang chủ.....	74
Hình 4.7 Giao diện đăng tải chẩn đoán.....	74
Hình 4.8 Giao diện báo cáo chẩn đoán.....	75
Hình 4.9 Giao diện xem kết quả chẩn đoán.....	75
Hình 4.10 Giao diện cập nhật kết quả chẩn đoán (Chỉ với Quản trị viên).....	76
Hình 4.11 Ảnh code xử lý ảnh MRI gửi lên từ người dùng.....	76
Hình 4.12 Ảnh code dự đoán kết quả và lưu vào cơ sở dữ liệu.....	77

LỜI CẢM ƠN

Lời đầu tiên cho phép em gửi lời cảm ơn sâu sắc tới các thầy cô trường Công nghệ Thông tin và Truyền thông - Đại học Công Nghiệp Hà Nội, những người đã hết mình truyền đạt và chỉ dẫn cho em những kiến thức, những bài học quý báu và bổ ích. Đặc biệt em xin được bày tỏ sự tri ân và xin chân thành cảm ơn giảng viên Tiến Sĩ Nguyễn Mạnh Cường, người trực tiếp hướng dẫn, chỉ bảo em trong suốt quá trình học tập, nghiên cứu và hoàn thành được đồ án. Sau nữa, em xin gửi tình cảm sâu sắc tới gia đình và bạn bè vì đã luôn bên cạnh khuyến khích, động viên, giúp đỡ cả về vật chất lẫn tinh thần cho em trong suốt quy trình học tập để em hoàn thành tốt việc học tập của bản thân.

Trong quá trình nghiên cứu và làm đề tài, do năng lực, kiến thức, trình độ bản thân em còn hạn hẹp nên không tránh khỏi những thiếu sót và em mong mọi nhận được sự thông cảm và những góp ý từ quý thầy cô cũng như các bạn trong lớp.

Em xin trân trọng cảm ơn!

Sinh viên thực hiện

Ngô Quý Điệp

LỜI NÓI ĐẦU

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, trí tuệ nhân tạo (AI) và học máy đang trở thành những công cụ thiết yếu để giải quyết các bài toán phức tạp và cải thiện hiệu quả trong nhiều lĩnh vực. Từ công nghệ, y tế, giáo dục đến kinh tế, trí tuệ nhân tạo không chỉ giúp nâng cao năng suất mà còn mở ra những cơ hội mới, đồng thời cũng đặt ra những thách thức cần được giải quyết.

Y tế hay Chăm sóc sức khỏe là một trong những lĩnh vực đang được quan tâm và chú trọng bậc nhất trong bối cảnh dữ liệu lớn (Big Data) có thể tạo nên những tác động mang tính cách mạng. Việc ứng dụng AI vào phân tích và sử dụng dữ liệu y tế có thể coi là vấn đề sống còn, ảnh hưởng trực tiếp đến hàng triệu người trên toàn cầu mỗi ngày. AI hoàn toàn có thể hỗ trợ bác sĩ, y tá và nhân viên y tế khác trong công việc hằng ngày. Rộng hơn, ứng dụng trí tuệ nhân tạo (AI) có thể nâng cao dịch vụ chăm sóc phòng ngừa và chất lượng cuộc sống bằng việc đưa ra các chẩn đoán và kế hoạch điều trị chính xác hơn và hiệu quả hơn cho bệnh nhân. AI hoàn toàn có thể đóng một vai trò quan trọng trong y tế công cộng toàn cầu bằng việc dự đoán và theo dõi sự lây lan của các bệnh truyền nhiễm nguy hiểm thông qua phân tích dữ liệu từ chính phủ, cơ quan y tế, từ đó trở thành cánh tay đắc lực chống lại dịch bệnh và đại dịch.

Trong bối cảnh đó, việc ứng dụng học máy (Machine Learning) vào phân loại khối u não dựa trên hình ảnh y tế nổi lên như một giải pháp đầy tiềm năng, có khả năng khắc phục những hạn chế của phương pháp truyền thống. Việc thực hiện đề tài hy vọng sẽ góp phần thúc đẩy trào lưu nghiên cứu, sản xuất các sản phẩm phần mềm trí tuệ nhân tạo và mở ra nhiều hướng nghiên cứu, ứng dụng nhận dạng bằng hình ảnh, đặc biệt trong lĩnh vực y tế.

Để thực hiện đề tài, trước tiên tôi nghiên cứu thực hiện khảo sát và phân tích bài toán nhận dạng khối u não. Sau đó, tôi tiến hành thu thập và tiền xử lý bộ dữ liệu huấn luyện cho các module trí tuệ nhân tạo. Dữ liệu tập trung vào 3 loại u

não phổ biến: u thần kinh đệm (glioma), u màng não (meningioma), u tuyến yên (pituitary) và lớp cuối cùng là không có khối u (no tumor). Tiếp theo, tôi tập trung vào việc nghiên cứu, phát biểu bài toán nhận dạng khối u não qua hình ảnh bằng kỹ thuật học sâu Deep Learning sử dụng mạng CNNs; nghiên cứu chuyên sâu về kỹ thuật CNNs, Transfer Learning; thực nghiệm kết quả và xây dựng chương trình demo nhằm ứng dụng bài toán vào thực tế.

Nội dung quyển báo cáo sẽ bao gồm các chương như sau:

Chương 1: Khảo sát và phát biểu bài toán

Trong chương 1, tôi tiến hành khảo sát thực trạng bệnh u não tại Việt Nam, bối cảnh hiện tại của chẩn đoán u não, tổng quan các nghiên cứu liên quan về áp dụng học máy trong y tế và xử lý ảnh y khoa. Chương này nhấn mạnh vào việc phát biểu vấn đề nghiên cứu, chỉ ra hạn chế của phương pháp truyền thống và đề xuất giải pháp dựa trên học máy. Cuối cùng, xác định mục tiêu và phạm vi nghiên cứu cụ thể, tạo nền tảng cho các chương tiếp theo.

Chương 2: Các kỹ thuật giải quyết bài toán

Sau khi phát biểu và xác định rõ yêu cầu của bài toán, ở chương này sẽ trình bày các phương pháp hiện có để giải quyết vấn đề, đồng thời phân tích chi tiết những ưu điểm và hạn chế của từng phương pháp. Bên cạnh đó, tôi cũng đề cập đến những nghiên cứu nổi bật đã đạt được thành công nhất định khi áp dụng các kỹ thuật này. Dựa trên những phân tích đó, tôi đề xuất một giải pháp mới phù hợp với bài toán đã đặt ra, kỳ vọng sẽ khắc phục được những hạn chế trước đây và mang lại kết quả tốt hơn

Chương 3: Thực nghiệm

Tại chương 3, tôi tập trung trình bày về quá trình thực nghiệm cũng như các kết quả đạt được với kỹ thuật giải quyết bài toán được đề xuất ở chương 2. Tôi cũng tiến hành so sánh kết quả thực nghiệm thu được từ phương pháp tôi đề xuất giải quyết bài toán với một số phương pháp phổ biến hiện nay và đưa ra nhận xét.

Chương 4: Xây dựng sản phẩm demo

Để tận dụng kết quả thực nghiệm đã thu được, tôi tiến hành xây dựng sản phẩm demo là một trang web có chức năng chính là chẩn đoán các loại u não đã được đào tạo bằng hình ảnh. Tôi sử dụng thư viện FastAPI của Python để xây dựng backen cho chương trình, đồng thời phần front-end sẽ sử dụng framework ReactJS cùng với ngôn ngữ định kiểu CSS (Cascading Style Sheets) và ngôn ngữ lập trình JavaScript. Bên cạnh đó cho phép lưu kết quả chẩn đoán vào cơ sở dữ liệu nhằm phục vụ cho việc phát triển mô hình, nâng cao độ chính xác.

Nội dung chương bao gồm: trình bày các framework được sử dụng và lý do lựa chọn, phân tích thiết kế hệ thống, trình bày các kết quả của hệ thống.

CHƯƠNG 1: KHẢO SÁT VÀ PHÁT BIỂU BÀI TOÁN

1.1. Hiện trạng y học và bệnh u não tại Việt Nam

1.1.1. Hiện trạng y học Việt Nam

Việt Nam có truyền thống y học cổ truyền lâu đời với nhiều bài thuốc và phương pháp điều trị hiệu quả. Hiện nay, y học cổ truyền đang được kết hợp với y học hiện đại để nâng cao hiệu quả điều trị. Các bệnh viện y học cổ truyền ngày càng phát triển, góp phần bảo tồn và phát huy giá trị của nền y học dân tộc.

Cụ thể, năm 2024, Việt Nam đạt 14 bác sĩ trên 10.000 dân, tăng so với năm 2023 là 12,5 bác sĩ trên 10.000 dân. Ngành y tế đặt mục tiêu có 15 bác sĩ trên 10.000 dân vào 2025 (theo báo Người Lao Động). Tuy nhiên, vẫn còn tình trạng thiếu hụt nhân lực y tế, đặc biệt là tại các vùng sâu, vùng xa. Việc đào tạo nhân lực y tế đang được chú trọng nhưng vẫn cần thêm nhiều cải tiến để đáp ứng nhu cầu thực tế. Ngành y tế Việt Nam đã có nhiều cải tiến trong chất lượng dịch vụ, nhưng vẫn còn nhiều bất cập như tình trạng quá tải bệnh viện, thiếu thuốc, thiếu trang thiết bị y tế. Chính phủ đang thực hiện nhiều chính sách để cải thiện tình trạng này, bao gồm đẩy mạnh bảo hiểm y tế toàn dân và tăng cường đầu tư vào cơ sở hạ tầng y tế.

Y học Việt Nam đã có nhiều bước phát triển đáng kể trong những năm qua, đặc biệt là trong nghiên cứu khoa học, ứng dụng công nghệ, và chất lượng dịch vụ y tế. Công nghệ đã thúc đẩy lĩnh vực y tế số, một lĩnh vực kết hợp giữa chăm sóc sức khỏe, công nghệ thông tin và kinh doanh. Cuộc cách mạng này trước mắt đã đem lại những tiện ích đáng kể đối với đội ngũ y bác sĩ cũng như bệnh nhân. Chuyển đổi số y tế cho phép chăm sóc chất lượng cao, hạn chế tối đa các sự cố trong y tế theo cách hiệu quả nhất. Một trong những ưu điểm nổi bật của y tế số chính là dễ dàng truy cập các thông tin quan trọng cho các chuyên gia, y bác sĩ, bệnh nhân... ở những nơi xa. Bệnh án điện tử sẽ lưu trữ đầy đủ thông tin rõ ràng về bệnh sử của bệnh nhân, đưa ra các cảnh báo về tương tác thuốc, dựa vào đó

cung cấp các đơn thuốc an toàn và đáng tin cậy hơn. Người ta ước tính rằng khoảng 50% khoản chi cho các dịch vụ chăm sóc sức khỏe bị lãng phí do quy trình khám, chữa bệnh và chăm sóc không hiệu quả.

Với y học ngày càng hiện đại, các loại máy móc phục vụ cho việc chẩn đoán, xét nghiệm đều tự động hoàn toàn, điều này giúp nâng cao độ chính xác kết quả. Song song với đó, các máy móc chẩn đoán hình ảnh cũng cần được trang bị ứng dụng kỹ thuật dựng hình để phản ánh rõ tình trạng bệnh. Bên cạnh đó kỹ thuật nội soi cũng là một trong những bước tiến quan trọng giúp can thiệp và điều trị bệnh một cách hiệu quả và tiết kiệm thời gian, tiết kiệm chi phí. Ngoài ra với sự phát triển về kinh tế xã hội, y học và hỗ trợ của Công nghệ thông tin, các bệnh viện đang được vận hành các máy xét nghiệm tự động, chụp X-quang kỹ thuật số hay các máy siêu âm 3D, 4D, chụp CT, cộng hưởng từ... giúp mang lại hiệu quả cao trong việc chẩn đoán và điều trị bệnh. Đồng thời góp phần đưa danh tiếng cũng như uy tín y học Việt Nam ngày một đi lên nằm trong Top những bệnh viện chuyên khoa hô hấp hàng đầu thế giới.

1.1.2. Hiện trạng bệnh u não ở Việt Nam

U não là một tập hợp số lượng lớn các tế bào não phát triển bất thường vượt ngoài tầm kiểm soát của cơ thể. Các u não có thể bắt đầu trực tiếp từ tế bào não, tế bào đệm của hệ thần kinh trung ương, hoặc cũng có thể bắt đầu từ các bộ phận khác (ví dụ như phổi, thận...) rồi theo máu đến não, được gọi là u di căn não. Cơ chế hình thành nên u não thông thường là từ lúc sinh ra đến lúc mất đi, không có thêm tế bào thần kinh nào được sinh thêm ra nữa. Khi có đột biến không rõ nguyên nhân trong DNA khiến các tế bào phân chia mất kiểm soát thì sẽ hình thành nên u não. Tốc độ phát triển cũng như vị trí của u não quyết định mức độ nghiêm trọng và tầm ảnh hưởng của nó đến chức năng hệ thần kinh, thậm chí là đe dọa đến tính mạng nếu không được chẩn đoán, theo dõi và chữa trị kịp thời.

U não chiếm 2% trong tổng số các ca ung thư từ mọi nhóm tuổi. Trong số các trường hợp tử vong do ung thư ở nhóm trẻ em dưới 15 tuổi và nhóm từ 20-39 tuổi, bệnh u não là nguyên nhân gây tử vong cao thứ 2. Người ngoài 85 tuổi có tỉ lệ bị u não cao nhất.

Hiện nay, có 04 loại khối u não thường gặp là:

- U não Gliomas: Còn gọi là u thần kinh đệm vì đây là khối u não bắt đầu từ trong các tế bào thần kinh đệm ở não hoặc tủy sống. U não Gliomas là loại u não nguyên phát ác tính, chiếm khoảng 50,4% tổng số các ca u não và 78% các ca có khối u não nguyên phát ác tính
- U màng não: Là một khối u phát triển chậm, hình thành từ màng não hay lớp màng bao quanh tủy sống. U màng não là loại u lành tính, thường xuất hiện ở nữ giới, chiếm khoảng 20,8% tổng số các ca bị u não với tỉ lệ tái phát sau phẫu thuật thấp (ít hơn 20%).
- U tuyến yên: Là khối u xảy ra trong tuyến yên (nằm ở bề mặt dưới não) với hơn 60% các ca được chẩn đoán là lành tính, 35% là loại khối u có xâm lấn. Theo thống kê, u tuyến yên chiếm từ 10% – 25% trên tổng số các ca u não và tỷ lệ mắc bệnh có thể lên đến 17% dân số.
- Khối u thần kinh ngoại biên: Do các nguyên bào sợi tăng trưởng bao quanh bó thần kinh gây ra và chiếm khoảng 10% tổng số ca bị u não. Hầu hết các khối u thần kinh ngoại biên là lành tính (không phải ung thư). Tuy nhiên, khối u chèn ép thần kinh gây đau và có thể làm mất khả năng kiểm soát cơ bắp.

Nguyên nhân gây u não chính xác trong hầu hết mọi trường hợp là không thể xác định. Tuy nhiên, nguyên nhân tiềm ẩn thì có rất nhiều. Bất cứ điều gì làm tăng khả năng mắc bệnh u não đều là có thể được xem là một yếu tố nguy cơ của bệnh (tức nguyên nhân tiềm ẩn). Việc bạn có một trong số các nguy cơ gây u não

dưới đây không chắc chắn là bạn sẽ bị u não trong tương lai. Các yếu tố nguy cơ làm tăng rủi ro u não có thể là:

- Tuổi tác: Người càng lớn tuổi càng có nguy cơ bị u não. Hầu hết các khối u não xảy ra ở người lớn tuổi từ 85 đến 89, mặc dù vẫn có một số loại u não phổ biến hơn ở trẻ em dưới 15 tuổi.
- Tiền sử gia đình (di truyền): Theo báo cáo, chỉ có từ 5-10% tổng số ca ung thư là do di truyền. U não chỉ chiếm 2% tổng số ca ung thư trên toàn thế giới, do đó tỉ lệ khối u não được di truyền là rất thấp. Một số tình trạng di truyền được biết là làm tăng nguy cơ mắc khối u não, bao gồm: bệnh xơ cứng củ, bệnh u sợi thần kinh loại 1, loại 2, hội chứng Turner, hội chứng Li-Fraumeni, hội chứng Turcot, hội chứng Gorlin,...
- Chế độ ăn thiếu khoa học: Một số nghiên cứu đã chỉ ra rằng các hợp chất N-nitroso trong chế độ ăn uống có thể ảnh hưởng đến nguy cơ mắc các khối u não ở trẻ em và người lớn. Gần đây, Tiến sĩ. Lee Wrensch phát hiện ra rằng người mắc bệnh u thần kinh đệm có tỉ lệ lớn tiêu thụ chế độ ăn ít trái cây, ít rau quả, ít vitamin C mà chứa nhiều nitrit như phô mai, cá, thịt xông khói, thức ăn đã qua chế biến, lên men, ử muối qua đêm (cá khô), đồ đóng hộp.
- Thừa cân và béo phì: Thừa cân hoặc béo phì làm tăng nguy cơ mắc bệnh u màng não. Khoảng 2% tổng số ca được chẩn đoán u não ở Anh mỗi năm là do thừa cân hoặc béo phì. Thừa cân hoặc béo phì làm tăng nguy cơ mắc bệnh u màng não. Khoảng 2% tổng số ca được chẩn đoán u não ở Anh mỗi năm là do thừa cân hoặc béo phì.
- Không có tiền sử bệnh thủy đậu: Dựa theo một báo cáo năm 2016 được xuất bản trên tạp chí Cancer Medicine, những người chưa có tiền sử mắc bệnh thủy đậu ở thời thơ ấu nguy cơ phát triển u thần kinh đệm cao hơn 21% so với người đã nhiễm bệnh thủy đậu.

- Phơi nhiễm hóa chất: Một số ngành nghề do môi trường làm việc đặc thù cần tiếp xúc với nhiều hóa chất có thể làm tăng nguy cơ ung thư não, chẳng hạn như: Người làm nông nghiệp phải tiếp xúc nhiều với thuốc trừ sâu, Công nhân làm việc trong môi trường nhiều kim loại nặng (niken, thủy ngân),...
- Tiếp xúc với bức xạ: Bức xạ ion hóa là một loại bức xạ được sử dụng bởi một số quy trình quét y tế, chẳng hạn như chụp X-quang và chụp CT. Những người đã tiếp xúc với bức xạ ion hóa có nguy cơ mắc các khối u não cao hơn người bình thường. Do đó, nếu bạn đã có tiền sử xạ trị trước đây với các bệnh ung thư khác thì cũng có thể làm tăng nguy cơ bị u não của bạn lên một chút. Tuy nhiên, u não do tiếp xúc với bức xạ xảy ra với tỉ lệ rất hiếm (dưới 1%).

Khối u não rất nguy hiểm dù là u não lành tính hay u não ác tính. Bệnh dù được điều trị kịp thời hay không đều có thể để lại những biến chứng nhất định, ảnh hưởng nghiêm trọng đến sinh hoạt, có thể rút ngắn tuổi thọ bệnh nhân hay thậm chí đe dọa tính mạng. Những biến chứng này có mức độ nặng nhẹ khác nhau tùy vào cơ địa, trong đó bao gồm:

- Đau đầu – hoa mắt – chóng mặt: Là biến chứng phổ biến nhất sau khi điều trị u não. Tác dụng phụ này đến từ việc vết thương phẫu thuật chưa lành cũng như thuốc mê chưa hết tác dụng.
- Mệt mỏi – buồn ngủ: Tác dụng của thuốc an thần sử dụng trong suốt quá trình phẫu thuật u não có thể đem lại cho bạn cảm giác uể oải và buồn rùn tay chân sau khi điều trị.
- Đau họng: Trong khi phẫu thuật u não, bệnh nhân sẽ được đặt ống thở (thông vào cổ họng) để điều chỉnh nhịp thở và lượng oxy lên não. Do đó, khi phẫu thuật kết thúc, đau họng là biến chứng không hiếm gặp.

- Suy giảm giao tiếp: Một số bệnh nhân sau khi điều trị u não thì chậm nói, chậm đọc, chậm viết, chậm hiểu, phản xạ giao tiếp kém, nói ngọng,... ảnh hưởng đến khả năng giao tiếp hàng ngày.
- Suy giảm giác quan và khả năng vận động: Mắt có thể mờ, tai có thể bị ãng nhẹ, vị giác ăn mất ngon, mũi mất vị, tay chân bị tê, liệt một phần, bị yếu, run hay khó kiểm soát, giữ thăng bằng là những biến chứng khác nhau xảy ra tùy cơ địa.
- Giảm tuổi thọ hoặc tử vong: Với u não lành tính, tỷ lệ sống sót sau 5 năm đối với u màng não, loại u não nguyên phát lành tính phổ biến nhất, là trên 96% đối với trẻ em từ 14 tuổi trở xuống, 97% ở những người từ 15 đến 39 tuổi và trên 87% ở người lớn từ 40 tuổi trở lên. Còn với u não ác tính, tỷ lệ sống sót trung bình sau 05 năm đối với tất cả các bệnh nhân u não ác tính ở Hoa Kỳ và Anh lần lượt là 33% và 10%.

1.2. Mục tiêu của bài toán

Mục tiêu của bài toán này là xây dựng một mô hình trí tuệ nhân tạo sử dụng mạng nơ-ron tích chập (CNN), cụ thể là InceptionV3, để tự động phân loại khối u não dựa trên ảnh chụp MRI. Hình ảnh MRI là một phương pháp chẩn đoán hình ảnh phổ biến trong y học, giúp cung cấp thông tin chi tiết về cấu trúc não bộ mà không gây hại cho bệnh nhân. Tuy nhiên, việc phân tích các ảnh này thường đòi hỏi sự can thiệp của các chuyên gia y tế có kinh nghiệm, dẫn đến sự phụ thuộc vào yếu tố con người và có thể xảy ra sai sót. Do đó, việc ứng dụng trí tuệ nhân tạo nhằm tự động hóa và nâng cao độ chính xác trong chẩn đoán là một hướng nghiên cứu quan trọng.

Trong nghiên cứu này, mô hình sẽ được huấn luyện để nhận diện và phân loại các loại khối u như Glioma, Meningioma, Pituitary Tumor hoặc xác định trường hợp không có khối u (No tumor). InceptionV3 là một kiến trúc CNN tiên tiến, có khả năng trích xuất đặc trưng hiệu quả từ hình ảnh y tế nhờ vào thiết kế

sâu và tối ưu hóa việc xử lý dữ liệu. Việc sử dụng mô hình này giúp cải thiện độ chính xác trong chẩn đoán, đồng thời giảm thời gian phân tích hình ảnh so với các phương pháp truyền thống.

Bên cạnh đó, nghiên cứu cũng tập trung vào việc tối ưu hóa mô hình để đạt hiệu suất cao, tránh overfitting và đảm bảo khả năng tổng quát hóa trên tập dữ liệu mới. Điều này bao gồm các kỹ thuật như tiền xử lý dữ liệu, tăng cường dữ liệu (data augmentation), tinh chỉnh mô hình (fine-tuning) và đánh giá trên các bộ dữ liệu khác nhau. Việc đảm bảo mô hình hoạt động ổn định trên dữ liệu thực tế sẽ giúp nâng cao tính ứng dụng trong môi trường lâm sàng.

Kết quả của bài toán có thể hỗ trợ bác sĩ trong việc phát hiện sớm và phân loại khối u não, góp phần nâng cao hiệu quả chẩn đoán và điều trị cho bệnh nhân. Một hệ thống phân loại chính xác sẽ giúp giảm gánh nặng công việc cho bác sĩ, hỗ trợ đưa ra quyết định nhanh chóng và chính xác hơn, từ đó cải thiện chất lượng chăm sóc y tế.

1.3. Phát biểu bài toán

1.3.1. Đầu vào (input)

*** Ảnh đầu vào:**

- Dạng ảnh: Ảnh số chứa hình ảnh MRI của một bộ não duy nhất. Các ảnh có thể là ảnh chụp hoặc ảnh đã được xử lý.
- Độ phân giải: Ảnh phải có kích thước cố định là 224 x 224 pixels
- Màu sắc: Ảnh phải là ảnh RGB với 3 kênh màu (Đỏ, Xanh lá, Xanh dương).
- Chuẩn hóa: Ảnh đầu vào phải được chuẩn hóa về giá trị pixel. Thông thường, các giá trị pixel nằm trong khoảng $[0, 1]$ sau khi chia giá trị pixel gốc (0-255) cho 255.
- Định dạng ảnh: Có thể là các định dạng phổ biến như JPEG hoặc PNG.

*** Đặc điểm của ảnh:**

- Một đối tượng duy nhất: Mỗi ảnh chỉ chứa một bộ não duy nhất, không có vật cản lớn che khuất đối tượng hoặc các đối tượng khác gây nhiễu.
- Đủ dữ liệu mẫu: Đảm bảo có đủ ảnh đại diện cho từng loại u não để tránh mất cân bằng dữ liệu.
- Góc nhìn đa dạng: Ảnh của các loại u não cần bao gồm nhiều góc chụp khác nhau (trực diện, nghiêng) để mô hình có thể học các đặc trưng đặc thù của từng loại u não từ nhiều góc độ khác nhau.

1.3.2. Đầu ra (output)

* Dạng đầu ra:

- Vector xác suất: Đầu ra từ mô hình là một vector xác suất, mỗi phần tử của vector tương ứng với một loại u não. Số lượng phần tử của vector bằng với số loại u não cần phân loại là 4. (bao gồm cả không có khối u)
- Xác suất chuẩn hóa: Tổng các xác suất trong vector luôn bằng 1, sử dụng hàm softmax cho lớp đầu ra để chuẩn hóa xác suất này.

* Nhận dự đoán:

- Lựa chọn nhãn có xác suất cao nhất: Loại u não tương ứng với xác suất cao nhất trong vector sẽ được chọn làm nhãn dự đoán cho ảnh.
- Độ chính xác của nhãn: Để đánh giá hiệu quả mô hình, có thể yêu cầu độ chính xác trên tập kiểm tra phải đạt ngưỡng xác định, khoảng trên 90%.
- Thông tin bổ sung: Ngoài nhãn dự đoán, mô hình có thể xuất ra các thông tin bổ sung như mức độ tự tin (confidence score) để người dùng hiểu mức độ chắc chắn của mô hình với dự đoán đó.

1.3.3. Các ràng buộc của bài toán

* Ràng buộc về đầu vào:

- Độ rõ của ảnh: Ảnh đầu vào phải rõ ràng, không bị nhòe hoặc nhiễu mạnh vì điều này có thể làm giảm độ chính xác của mô hình. Các yếu tố ánh sáng phải hợp lý để không che khuất đặc điểm nhận dạng của khối u.

- Định dạng ảnh cố định: Đảm bảo rằng tất cả ảnh đầu vào được chuẩn hóa về kích thước và định dạng trước khi đưa vào mô hình. Sử dụng cùng kích thước ảnh giúp duy trì tính nhất quán và tối ưu hóa hiệu suất mô hình.
- Chất lượng dữ liệu: Cần có một lượng lớn ảnh chất lượng cao cho từng loại u não. Điều này đặc biệt quan trọng vì kết quả để phục vụ cho y học, lĩnh vực không nên có sai số.
- Tiền xử lý dữ liệu: Ngoài việc chuẩn hóa pixel, các bước tiền xử lý khác như làm mờ ảnh, xoay ảnh, cắt ảnh để tăng tính đa dạng trong dữ liệu huấn luyện (data augmentation) nhằm cải thiện khả năng tổng quát hóa của mô hình.

*** Ràng buộc về đầu ra:**

- Yêu cầu độ chính xác cao: Mô hình phải đạt độ chính xác cao trên tập kiểm tra ($>90\%$) để đáp ứng yêu cầu của bài toán phân loại.
- Khả năng phân biệt các khối u não chính xác: Mô hình cần có khả năng phân biệt giữa các khối u não hoặc là không có khối u một cách chính xác để có tính ứng dụng

*** Ràng buộc về môi trường**

- Phần cứng: Để huấn luyện mô hình CNN hiệu quả, cần có GPU hoặc TPU để tăng tốc độ tính toán, đặc biệt với mô hình lớn và dữ liệu nhiều ảnh.
- Phần mềm: Môi trường lập trình Python cùng với các thư viện học sâu như TensorFlow hoặc PyTorch là cần thiết để xây dựng và huấn luyện mô hình.
- Quản lý dữ liệu: Cần có một hệ thống quản lý và lưu trữ dữ liệu ảnh lớn, chẳng hạn như Amazon S3 hoặc Google Cloud Storage, để đảm bảo quá trình xử lý không bị gián đoạn.
- Bảo mật: Nếu dữ liệu ảnh không phải công khai, cần có các biện pháp bảo mật dữ liệu để bảo vệ tính riêng tư, đặc biệt nếu chứa thông tin nhận dạng người dùng hoặc địa điểm.

* Ràng buộc về hiệu năng

- Tốc độ dự đoán: Mô hình cần dự đoán nhanh (thời gian suy luận dưới 100ms mỗi ảnh đầu vào) để có thể tích hợp vào các ứng dụng thời gian thực hoặc các hệ thống phân loại tự động.
- Tối ưu hóa mô hình: Sử dụng các kỹ thuật giảm kích thước mô hình như chuyển đổi sang TensorRT hoặc sử dụng phương pháp pruning và quantization để giảm kích thước và tăng tốc độ mô hình mà không làm mất nhiều độ chính xác.
- Khả năng mở rộng: Đảm bảo mô hình có thể dễ dàng mở rộng để phân loại thêm loại u não mới nếu cần, mà không cần huấn luyện lại từ đầu.
- Khả năng tổng quát hóa: Mô hình cần được huấn luyện đủ lâu để tránh quá khớp (overfitting) và đảm bảo tính tổng quát khi phân loại ảnh chưa từng gặp trong tập huấn luyện.

1.4. Ý nghĩa của bài toán đối với thực tế

Bài toán “Phân loại khối u não dựa trên hình ảnh bằng mạng InceptionV3” mang ý nghĩa rất lớn trong lĩnh vực y tế và khoa học dữ liệu, đặc biệt là khi kết hợp trí tuệ nhân tạo để hỗ trợ chẩn đoán và điều trị bệnh.

Mạng InceptionV3 được thiết kế để phân tích hình ảnh y tế như MRI, từ đó phát hiện và phân loại các loại khối u não. Với khả năng nhận diện và phân loại hình ảnh phức tạp, mạng này có thể cung cấp kết quả chẩn đoán đáng tin cậy, giảm thiểu sai sót do yếu tố chủ quan trong phân tích của con người. Điều này mang lại lợi ích lớn trong việc xác định tình trạng sức khỏe của bệnh nhân một cách nhanh chóng và hiệu quả.

Việc sử dụng học sâu để phân loại khối u tự động hóa quá trình phân tích hình ảnh y tế vốn tiêu tốn rất nhiều thời gian và công sức nếu thực hiện thủ công. Các bác sĩ và chuyên gia y tế có thể tập trung nhiều hơn vào việc đưa ra các quyết định lâm sàng và điều trị phù hợp mà không cần mất thời gian vào các quy trình

phức tạp. Mỗi loại khối u não yêu cầu các phác đồ điều trị khác nhau. Phân loại chính xác loại khối u giúp bác sĩ đưa ra phương pháp điều trị phù hợp nhất với tình trạng của bệnh nhân, từ đó tăng khả năng chữa khỏi hoặc cải thiện chất lượng cuộc sống. Đây là một bước tiến lớn trong việc cá nhân hóa điều trị trong lĩnh vực y học.

InceptionV3 là một mạng học sâu tiên tiến, được phát triển bởi Google và được thiết kế để xử lý các vấn đề phức tạp liên quan đến hình ảnh. Việc ứng dụng mạng này trong bài toán y tế không chỉ là sự kết hợp giữa trí tuệ nhân tạo và sức khỏe, mà còn thể hiện sự tiến bộ vượt bậc trong việc nâng cao chất lượng chăm sóc sức khỏe bằng công nghệ hiện đại.

Bên cạnh giá trị thực tiễn, bài toán này còn tạo ra cơ hội nghiên cứu trong cả hai lĩnh vực trí tuệ nhân tạo và y học. Các nhà khoa học có thể tiếp tục cải tiến các thuật toán, nâng cấp độ chính xác và hiệu quả của mô hình, đồng thời khám phá những tiềm năng mới trong việc kết hợp công nghệ với y tế.

1.5. Cơ hội và thách thức

1.5.1. Cơ hội

Bài toán này không chỉ mang lại lợi ích thiết thực cho lĩnh vực y tế mà còn mở ra nhiều cơ hội phát triển và ứng dụng rộng rãi trong các lĩnh vực liên quan khác. Công nghệ chẩn đoán dựa trên InceptionV3 có tiềm năng trở thành các sản phẩm thương mại hóa, chẳng hạn như phần mềm hỗ trợ chẩn đoán bệnh lý. Điều này không chỉ mang lại lợi ích kinh tế mà còn góp phần cải thiện chất lượng chăm sóc sức khỏe.

Kết quả của bài toán này có thể giúp thúc đẩy sự phát triển của y tế chính xác. Phân loại khối u não bằng học sâu là một bước tiến trong việc cá nhân hóa y tế. Mô hình InceptionV3 có thể được áp dụng để phân tích các loại hình ảnh y tế khác, từ đó mở rộng khả năng chẩn đoán và hỗ trợ điều trị không chỉ cho ung thư não mà còn các bệnh lý phức tạp khác.

Với sự phát triển của trí tuệ nhân tạo, các công cụ hỗ trợ chẩn đoán như mạng InceptionV3 có thể được triển khai tại các trung tâm y tế trên toàn thế giới, kể cả những nơi có nguồn lực hạn chế. Điều này giúp giảm bất bình đẳng trong tiếp cận y tế giữa các khu vực phát triển và kém phát triển.

Một cơ hội lớn của bài toán này là hướng tới việc xây dựng hệ thống chẩn đoán tự động, kết hợp với các mô hình học sâu khác. Hệ thống này sẽ giúp tự động hóa quy trình phân tích hình ảnh y tế, giảm sự phụ thuộc vào chuyên gia và tối ưu hóa thời gian điều trị.

Hơn thế, bài toán phân loại khối u còn tạo tiền đề cho các nghiên cứu liên ngành giữa trí tuệ nhân tạo, khoa học dữ liệu và y học. Các nhà nghiên cứu có thể khám phá các kỹ thuật mới, cải tiến độ chính xác của mô hình và ứng dụng học sâu vào các lĩnh vực khác, chẳng hạn như sinh học và dược học.

Bài toán này cũng là một cơ hội để đào tạo các chuyên gia y tế và công nghệ về cách sử dụng trí tuệ nhân tạo trong thực tiễn. Từ đó, các khóa học và chương trình đào tạo liên quan có thể được phát triển, mở ra hướng đi mới cho ngành giáo dục.

1.5.2. Thách thức

Dù mang lại nhiều cơ hội và tiềm năng, bài toán phân loại khối u não cũng đối mặt với một số thách thức lớn, bao gồm:

- **Chất lượng và số lượng dữ liệu:** Để huấn luyện một mô hình học sâu như InceptionV3 đạt hiệu quả cao, cần một lượng dữ liệu lớn và chất lượng cao. Tuy nhiên, dữ liệu y tế, đặc biệt là hình ảnh khối u não, thường khó thu thập do vấn đề bảo mật, quyền riêng tư và chi phí. Hơn nữa, hình ảnh MRI có thể không đồng nhất về chất lượng do sự khác biệt về thiết bị hoặc điều kiện chụp.
- **Gán nhãn dữ liệu chính xác:** Việc gán nhãn dữ liệu y tế đòi hỏi chuyên môn cao từ các bác sĩ và chuyên gia y tế. Quá trình này vừa tốn thời gian, vừa dễ

xảy ra sai sót. Sai lệch trong gán nhãn dữ liệu có thể làm giảm độ chính xác của mô hình phân loại.

- Chi phí tính toán cao: Huấn luyện và chạy mạng InceptionV3 yêu cầu tài nguyên tính toán lớn, bao gồm GPU mạnh mẽ và dung lượng lưu trữ lớn. Điều này có thể gây khó khăn cho các tổ chức y tế hoặc nghiên cứu với ngân sách hạn chế.
- Khả năng tổng quát hóa của mô hình: Mô hình học sâu thường hoạt động tốt trên dữ liệu mà nó đã được huấn luyện, nhưng khi áp dụng vào các dữ liệu thực tế khác biệt hoặc không quen thuộc, hiệu suất có thể giảm đáng kể. Điều này đặt ra câu hỏi về khả năng tổng quát hóa của mạng InceptionV3 khi áp dụng trong môi trường y tế thực tế.
- Đối mặt với các trường hợp ngoại lệ: Trong y học, luôn có những trường hợp ngoại lệ hoặc không điển hình mà mô hình AI khó có thể nhận diện hoặc xử lý chính xác. Đối với bài toán phân loại khối u, các loại khối u hiếm hoặc đặc biệt có thể khiến mô hình gặp khó khăn trong việc phân loại.
- Tính hợp pháp và đạo đức: Việc áp dụng AI trong y tế đòi hỏi phải tuân thủ các quy định pháp luật và đảm bảo yếu tố đạo đức, đặc biệt liên quan đến quyền riêng tư dữ liệu bệnh nhân. Các vấn đề như bảo mật thông tin, quyền sử dụng dữ liệu và trách nhiệm khi xảy ra lỗi cần được giải quyết triệt để.
- Chấp nhận và tin tưởng từ cộng đồng y tế: Dù AI có tiềm năng lớn, việc chấp nhận và tin tưởng vào các công cụ AI từ phía bác sĩ và chuyên gia y tế vẫn là một thách thức. Sự nghi ngại về tính chính xác, độ tin cậy, và khả năng hỗ trợ thực tế cần được giải quyết thông qua các nghiên cứu và chứng minh hiệu quả rõ ràng.

Những thách thức trên không chỉ đặt ra những bài toán cần giải quyết mà còn mở ra cơ hội để cải tiến và nâng cao ứng dụng của trí tuệ nhân tạo trong y tế.

CHƯƠNG 2: MỘT SỐ KỸ THUẬT GIẢI QUYẾT BÀI TOÁN

2.1. Phương hướng tiếp cận bài toán

Bài toán phân loại khối u não dựa trên hình ảnh là một bài toán phức tạp, đòi hỏi sự kết hợp của các kỹ thuật xử lý ảnh vào học máy. Phương pháp tiếp cận của bài toán này tập trung vào việc xây dựng một hệ thống phân loại tự động và có hiệu quả, bao gồm các giai đoạn chính sau:

- Thu thập và tiền xử lý dữ liệu:
 - Thu thập một tập dữ liệu đa dạng các ảnh MRI não, bao gồm các loại khối u khác nhau (ví dụ: u nguyên bào thần kinh đệm, u màng não, u tuyến yên) và cả ảnh não bình thường.
 - Thực hiện các bước tiền xử lý ảnh để cải thiện chất lượng dữ liệu và giảm nhiễu. Ví dụ như: chuẩn hóa, khử nhiễu, cắt ảnh...
- Trích xuất đặc trưng: Áp dụng các phương pháp trích xuất đặc trưng để thu thập thông tin có giá trị từ các ảnh MRI đã được tiền xử lý. Các phương pháp có thể là: tính toán các đặc trưng thống kê từ khối u, đặc trưng dựa trên kết cấu, hình dạng, đặc trưng học sâu...
- Xây dựng và huấn luyện mô hình phân loại: Sử dụng các thuật toán khác nhau để xây dựng mô hình phân loại khối u, huấn luyện mô hình trên tập dữ liệu đã được gán nhãn, sử dụng các kỹ thuật như chia tập dữ liệu thành tập huấn luyện, tập kiểm chứng và tập kiểm thử, cũng như các phương pháp tối ưu hóa phù hợp
- Đánh giá hiệu suất mô hình: Đánh giá hiệu suất của mô hình trên tập dữ liệu đã huấn luyện bằng các tiêu chí đánh giá phù hợp cho bài toán phân loại đa lớp, ví dụ: độ chính xác (accuracy), precision, f1-score, hoặc ma trận nhầm lẫn. Sau đó so sánh hiệu suất của các mô hình khác nhau để lựa chọn mô hình tốt nhất

- Triển khai và ứng dụng: Bước cuối cùng (nằm ngoài phạm vi chính của bài toán nhưng cần được đề cập) là khả năng triển khai mô hình đã được huấn luyện vào thực tế để hỗ trợ các bác sĩ trong việc chẩn đoán và phân loại khối u não dựa trên ảnh MRI.

Phương pháp tiếp cận này mang tính hệ thống, cho phép khám phá và so sánh hiệu quả của nhiều kỹ thuật khác nhau trong từng giai đoạn, từ đó xây dựng được một hệ thống phân loại khối u não mạnh mẽ và đáng tin cậy.

2.2. Một số kỹ thuật sử dụng để giải quyết bài toán

Các kỹ thuật giải quyết bài toán “Phân loại khối u não dựa trên hình ảnh” có thể chia thành 3 phương pháp như sau:

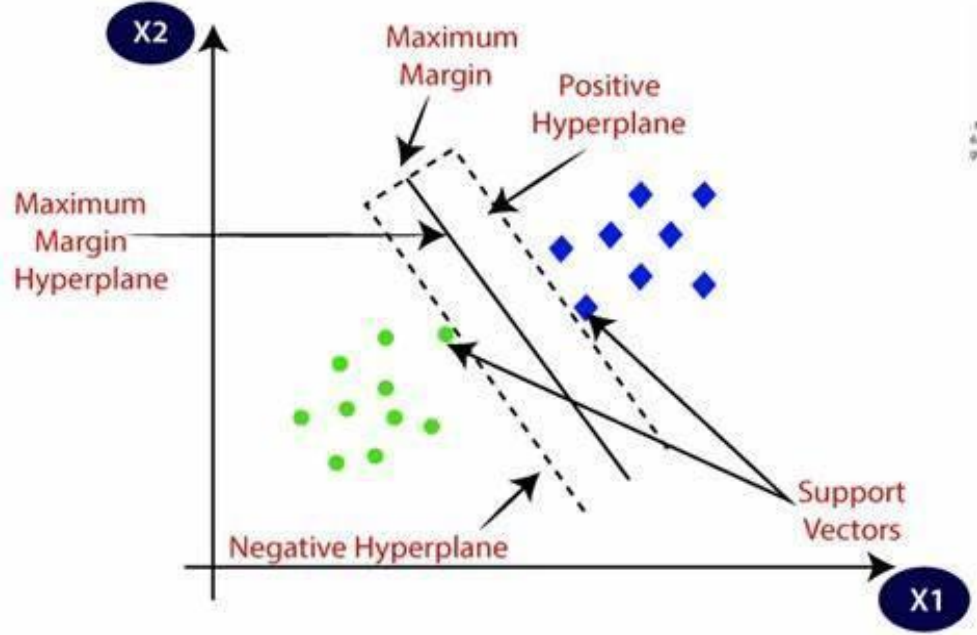
- Phương pháp dựa trên học máy truyền thống (Traditional Machine Learning): Kỹ thuật này bao gồm việc trích xuất các đặc trưng có ý nghĩa từ ảnh MRI (như hình dạng, kết cấu, cường độ pixel). Sau đó, các đặc trưng này được đưa vào các bộ phân loại học máy như Support Vector Machines (SVM), Random Forests, Logistic Regression, K-Nearest Neighbors (KNN) để phân loại khối u. Ưu điểm của phương pháp này là dễ sử dụng, có thể tận dụng các kiến thức chuyên môn về ảnh y tế để thiết kế các đặc trưng phù hợp. Tuy nhiên nhược điểm sẽ là quá trình trích xuất đặc trưng tốn thời gian và công sức, hiệu suất phụ thuộc nhiều vào đặc trưng được thiết kế
- Phương pháp dựa trên học sâu (Deep Learning):
 - Mạng nơ-ron tích chập (CNNs): Sử dụng các kiến trúc CNN (ví dụ: Alexnet, VGG, ResNet, Inception, EfficientNet) để tự động học các đặc trưng từ ảnh MRI. Mô hình CNN sẽ trực tiếp nhận ảnh MRI làm đầu vào và đưa ra dự đoán về loại khối u. Phương pháp này có ưu điểm là khả năng tự động học các đặc trưng phức tạp, thường đạt hiệu

suất cao hơn các phương pháp truyền thống khi có đủ dữ liệu. Tuy nhiên cần đòi hỏi lượng lớn dữ liệu huấn luyện, chi phí tính toán cao

- Transfer Learning: Tận dụng các mô hình CNN đã được huấn luyện trước dựa trên các bộ dữ liệu ảnh lớn (ví dụ: imagenet) và tinh chỉnh (fine-tune) chúng dựa trên bộ dữ liệu MRI, giúp giảm nhu cầu về dữ liệu huấn luyện lớn, tăng tốc quá trình huấn luyện và có thể cải thiện hiệu suất
- Phương pháp kết hợp (Hybrid Approaches):
 - Kết hợp đặc trưng thủ công và đặc trưng học sâu: Trích xuất cả đặc trưng thủ công và đặc trưng tự động từ các lớp ẩn của mạng CNN, sau đó kết hợp chúng để huấn luyện bộ phân loại.
 - Phương pháp dựa trên attention: Sử dụng các cơ chế attention trong mạng nơ-ron để giúp mô hình tập trung vào các vùng quan trọng nhất của ảnh MRI khi đưa ra quyết định phân loại, tăng tính giải thích của mô hình và có thể cải thiện hiệu suất.
 - Phương pháp Ensemble: Huấn luyện nhiều mô hình phân loại khác nhau (có thể là các kiến trúc CNN khác nhau hoặc kết hợp CNN với các bộ phân loại truyền thống) và kết hợp dự đoán của chúng để đưa ra kết quả cuối cùng, tăng tính ổn định và độ chính xác của hệ thống.

2.3. Máy vector hỗ trợ (Support Vector Machine - SVM)

Support Vector Machine là một trong số những thuật toán phổ biến và được sử dụng nhiều nhất trong học máy trước khi mạng nơ ron nhân tạo trở lại với các mô hình deep learning. Nó được biết đến rộng rãi ngay từ khi mới được phát triển vào những năm 1990. Mục tiêu của SVM là tìm ra một siêu phẳng trong không gian N chiều (ứng với N đặc trưng) chia dữ liệu thành hai phần tương ứng với lớp của chúng. Nói theo ngôn ngữ của đại số tuyến tính, siêu phẳng này phải có lẽ cực đại và phân chia hai bao lồi và cách đều chúng.



Hình 2.1 Mô tả thuật toán SVM

Sau đây, tôi xin giới thiệu sơ lược về mô hình SVM. Xét bài toán phân lớp nhị phân. Cho trước một tập dữ liệu huấn luyện gồm n mẫu:

$$X = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \in \mathbb{R}^{n \times (d+1)}$$

trong đó, x_i là một véc tơ trong không gian \mathbb{R}^d và $y_i \in \{-1, 1\}$ là tập các nhãn lớp. Một siêu phẳng phân tách tập X thành hai miền có dạng:

$$\langle w, x \rangle + b = 0$$

với $w \in \mathbb{R}^d$ và $b \in \mathbb{R}$. Mục tiêu của bài toán huấn luyện SVM là tìm ra một siêu phẳng phân tách “tốt nhất” tập X theo nghĩa là lề của siêu phẳng (margin) đạt cực đại. Để tìm được bộ (w, b) như vậy, ta giải bài toán tối ưu sau:

$$\min_{w, b, \xi} \left\{ \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^n \xi_i \right\}$$

(Cortes, C., & Vapnik, V. (1995). *Support-Vector Networks*. *Machine Learning*, 20(3), 273-297)

sao cho thoả mãn:

$$\Omega: \begin{cases} (w, b, \xi) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}_+^n \\ y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \quad \forall 1 \leq i \leq n \end{cases}$$

(Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks." *Machine Learning*, 20(3), 273-297. DOI: 10.1007/BF00994018)

trong đó, $\langle w, x_i \rangle$ là một tích vô hướng trong không gian R^n , ξ_i là các biến *slack* được thêm vào để nới lỏng điều kiện phân lớp và C là tham số điều chỉnh. Thay vì giải bài toán trên, ta thường xem xét bài toán đối ngẫu của nó như sau:

$$\min_{\alpha} \frac{1}{2} \alpha^T H \alpha - \vec{1}^T \alpha$$

trong đó thỏa mãn $\Delta: \begin{cases} y^T \alpha = 0 \\ 0 \leq \alpha_i \leq C, i = 1, \dots, n \end{cases}$. Trong đó $y = (y_1, y_2, \dots, y_n)$, $\vec{1}$ là vector với toàn bộ thành phần đều bằng 1 và H là một ma trận đối xứng được xác định bởi:

$$H_{i,j} = y_i y_j \langle \phi(x_i), \phi(x_j) \rangle = y_i y_j K(x_i, x_j)$$

ở đây, $\phi()$ là một ánh xạ từ không gian ban đầu (input space) sang không gian đặc trưng (feature space) có số chiều cao hơn nhằm xử lý trường hợp dữ liệu không phân tách tuyến tính. Hàm $K(.)$ được gọi là hàm nhân (kernel function) được định nghĩa:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

Ưu và nhược điểm của SVM:

*** Ưu điểm:**

- Xử lý trên không gian số chiều cao: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn
- Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết định
- Hạn chế overfitting: Với sự lựa chọn hợp lý của tham số điều chỉnh (C) và kernel, SVM có khả năng kiểm soát overfitting và đưa ra mô hình phân loại

chính xác.

- Tính linh hoạt - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.
- Có thể xử lý các lớp không cân bằng: Bằng cách điều chỉnh trọng số của các lớp, SVM có thể cải thiện hiệu suất phân loại trên các tập dữ liệu không cân bằng.

*** Nhược điểm:**

- Bài toán số chiều cao: Trong trường hợp số lượng thuộc tính của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu thì SVM cho kết quả khá tồi
- SVM có thể gặp khó khăn khi xử lý dữ liệu rất lớn về mặt số lượng mẫu và số lượng đặc trưng, dẫn đến thời gian huấn luyện lâu.
- Không hiệu quả với dữ liệu rất lớn và dữ liệu có độ nhiễu cao: Với dữ liệu quá lớn hoặc dữ liệu có nhiễu nhiều, SVM có thể không đạt hiệu suất tốt và yêu cầu nhiều tài nguyên tính toán.
- Chưa thể hiện rõ tính xác suất: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào. Tuy nhiên hiệu quả của việc phân lớp có thể được xác định dựa vào khái niệm margin từ điểm dữ liệu mới đến siêu phẳng phân lớp mà chúng ta đã bàn luận ở trên.
- Khả năng phân loại đa lớp phức tạp: SVM nguyên bản được thiết kế cho phân loại nhị phân, và việc mở rộng cho phân loại đa lớp có thể phức tạp hơn so với các phương pháp khác như cây quyết định hay mạng nơ ron.

2.4. Mạng nơ ron tích chập (Convolution Neural Networks - CNNs)

2.4.1. Tổng quan

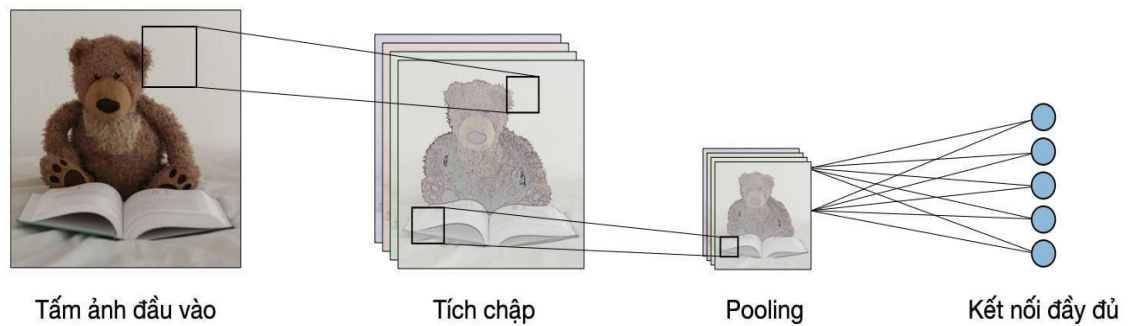
Mạng nơ ron tích chập (CNN) là một thiết kế mạng học sâu mà học trực

tiếp từ đầu vào, loại bỏ nhu cầu về việc trích xuất đặc trưng từ con người. CNN rất hiệu quả trong việc phát hiện các mẫu trong hình ảnh để phân biệt các đối tượng, khuôn mặt và cảnh quan. Chúng cũng hữu ích trong việc phân loại dữ liệu không phải hình ảnh như âm thanh, dãy số thời gian và dữ liệu tín hiệu. CNN là mạng nơ ron đa tầng được điều chuẩn. Mạng nơ ron đa tầng thường là mạng kết nối hoàn toàn, có nghĩa là mỗi nơ ron ở một tầng liên kết với tất cả các nơ ron ở tầng tiếp theo. Do sự "kết nối hoàn toàn" này, các mạng này dễ bị quá mức với dữ liệu. Điều chuẩn, hoặc ngăn chặn quá mức, thường được thực hiện bằng cách trừng phạt các tham số trong quá trình huấn luyện (như giảm trọng lượng) hoặc giảm kết nối (kết nối bỏ qua, dropout, v.v.). CNN giải quyết vấn đề điều chuẩn một cách khác: chúng sử dụng mẫu phân cấp trong dữ liệu để xây dựng các mẫu có độ phức tạp tăng dần bằng cách sử dụng các mẫu nhỏ và đơn giản hơn in vào các bộ lọc của chúng. Do đó, CNN ở dải thấp nhất của phổ kết nối và độ phức tạp.

CNN (Convolutional Neural Network) dần chiếm lĩnh vai trò quan trọng trong lĩnh vực Deep Learning. Nó được sử dụng rộng rãi trong thị giác máy tính và các nhiệm vụ học nơ ron khác. Để giới hạn số lượng tham số, việc sử dụng mô hình CNN là một chiến lược khôn ngoan. Khi đào tạo một CNN, cùng một tham số được sử dụng nhiều lần, dẫn đến một mạng không kết nối đầy đủ. Do đó, đây là một cách hợp lý để giảm thời gian đào tạo. Kể từ khi ra đời, CNNs nhanh chóng phát triển từ xử lý hình ảnh sang văn bản, nhận dạng giọng nói và trở thành tiêu chuẩn vàng trong những lĩnh vực này và nhiều lĩnh vực trí tuệ nhân tạo khác.

2.4.2. Cấu trúc

Kiến trúc truyền thống của một mạng CNN được cấu thành bởi các tầng sau:



Hình 2.2 Minh họa kiến trúc mạng CNN

Tầng tích chập và tầng pooling có thể được hiệu chỉnh theo các siêu tham số (hyperparameters).

* Tầng tích chập:

Tầng tích chập là cái tên có phần không chính xác, vì phép toán mà chúng biểu diễn là phép tương quan chéo (cross correlation). Trong một tầng tích chập, một mảng đầu vào và một mảng hạt nhân (kernel) tương quan hay filter (bộ lọc) được kết hợp để tạo ra mảng đầu ra bằng phép toán tương quan chéo. Như trong ví dụ dưới đây, ta có một input là một mảng 2 chiều với kích thước là 5×5 và một ma trận hạt nhân là ma trận 3×3 .

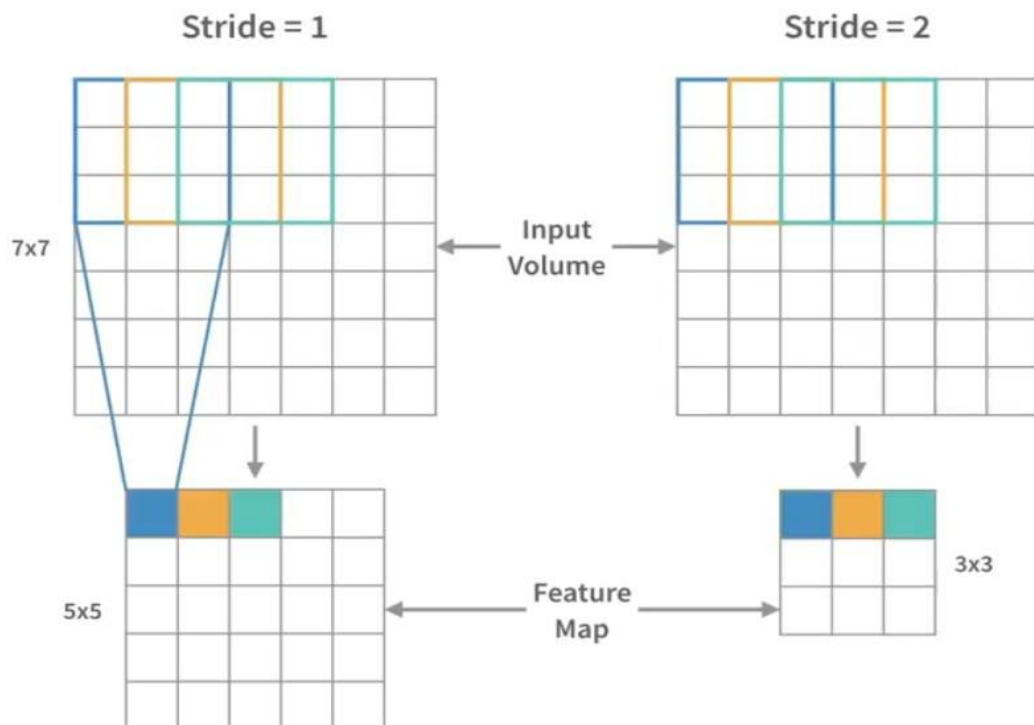
Trong phép tương quan chéo hai chiều, ta bắt đầu với cửa sổ tích chập với kích thước bằng với ma trận hạt nhân đặt tại vị trí góc trên bên trái của mảng đầu vào và di chuyển cửa sổ này từ trái sang phải và từ trên xuống dưới. Khi cửa sổ tích chập được đẩy tới một vị trí nhất định, mảng con đầu vào nằm trong cửa sổ đó và ma trận hạt nhân được nhân theo từng phần tử, rồi sau đó ta lấy tổng các phần tử trong mảng kết quả để có được một giá trị số vô hướng duy nhất. Giá trị này được ghi vào mảng đầu ra tại vị trí tương ứng.

Lưu ý rằng theo mỗi trục, kích thước đầu ra nhỏ hơn một chút so với đầu vào. Bởi vì hạt nhân có chiều dài và chiều rộng lớn hơn một, ta chỉ có thể tính độ tương quan chéo cho những vị trí mà ở đó hạt nhân nằm hoàn toàn bên trong ảnh,

kích thước đầu ra được tính bằng cách lấy đầu vào $H \times W$ trừ kích thước của bộ lọc tích chập $h \times w$ bằng $(H - h + 1) \times (W - w + 1)$. Điều này xảy ra vì ta cần đủ không gian để ‘dịch chuyển’ hạt nhân tích chập qua tấm hình (sau này ta sẽ xem làm thế nào để có thể giữ nguyên kích thước bằng cách đệm các số không vào xung quanh biên của hình ảnh sao cho có đủ không gian để dịch chuyển hạt nhân).

Độ trượt (Stride)

- Stride hay độ trượt là số pixel mà kernel di chuyển qua dữ liệu đầu vào, ở đây là ma trận
- Một độ trượt có nghĩa là kernel trượt từng pixel một, dẫn đến bản đồ tính năng đầu ra được tính toán dày đặc.
- Một bước tiến lớn hơn như hai hoặc nhiều pixel bỏ qua, dẫn đến bản đồ tính năng đầu ra thưa thớt hơn và giảm kích thước không gian.
- Một bước tiến lớn hơn có thể làm tăng hiệu quả tính toán nhưng cũng có thể dẫn đến mất thông tin không gian.

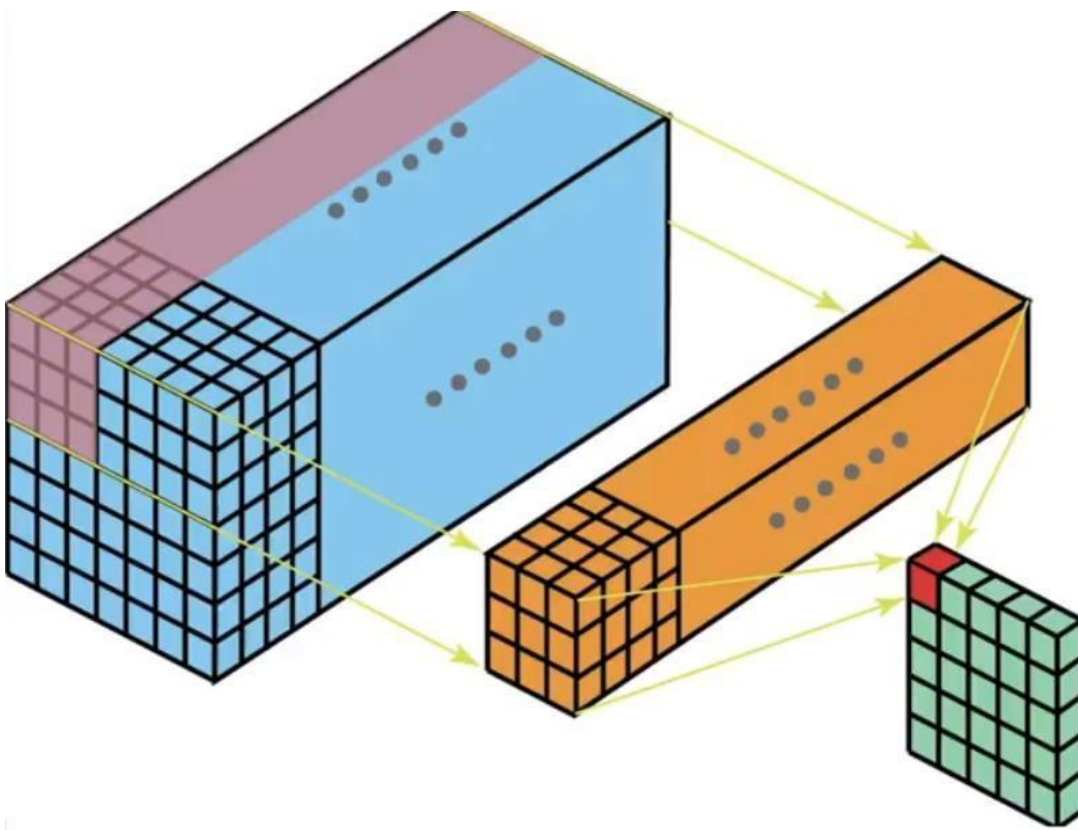


Hình 2.3 Minh họa độ trượt (stride)

Các kiểu tích chập

* Tích chập 2 chiều

- Là kiểu tích chập phổ biến trong CNNs
- Kernel là một ma trận 2 chiều và đầu vào thường cũng là một ảnh có số chiều tương ứng
- Tổng các tích toán tử của cửa sổ tích chập với kernel như ví dụ trên
- Có thể trích xuất các thông tin như cạnh, góc hoặc kết cấu hình dạng



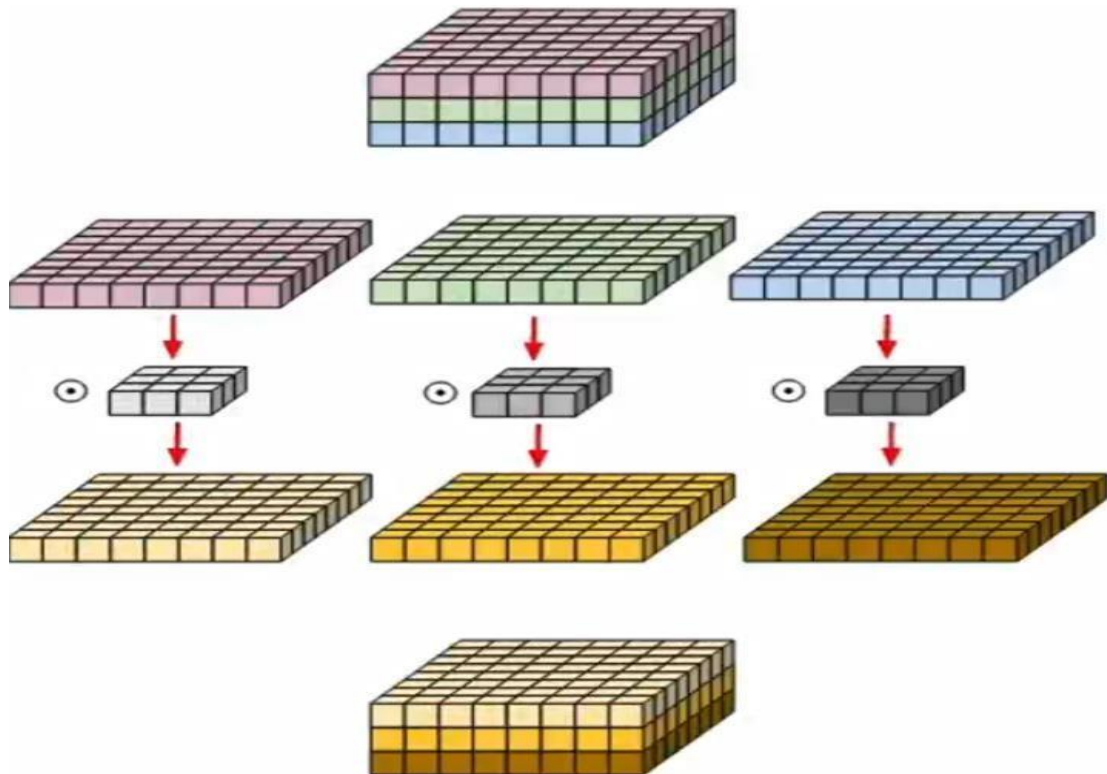
Hình 2.4 Minh họa kiểu tích chập 2 chiều

* Tích chập theo chiều sâu (Depthwise Convolution)

- Tích chập theo chiều sâu là một biến thể tích chập mang lại hiệu quả về mặt tính toán
- Mỗi kênh đầu vào sẽ được thực hiện tích chập với 1 filter (bộ lọc) tương ứng.

Các kênh ở đây có thể hiểu là các màu riêng biệt có trong ảnh được tách thành các ma trận pixel tương ứng

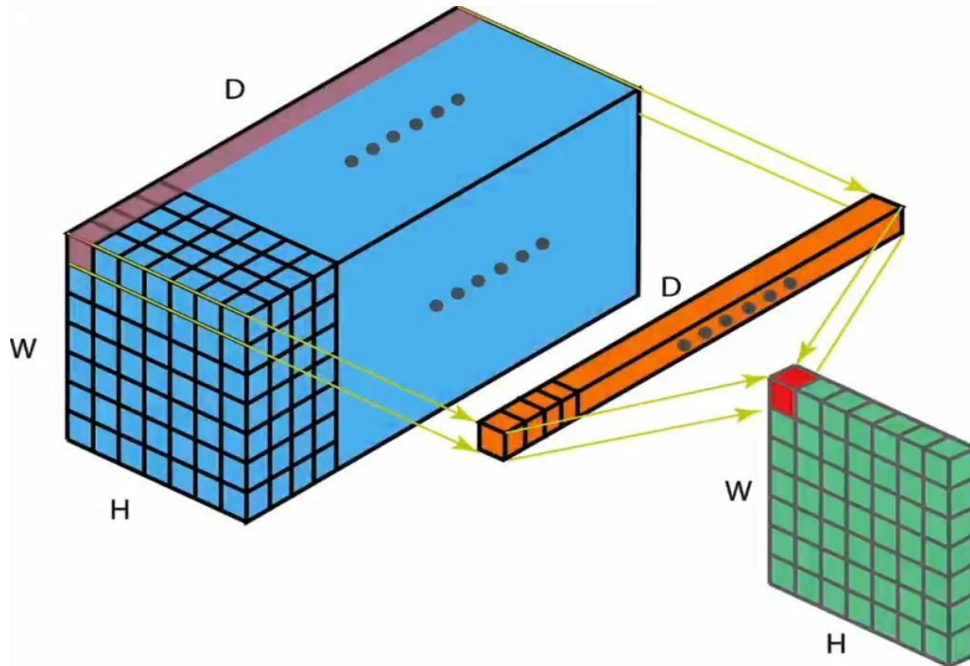
- Các bản đồ đặc trưng sẽ được ghép xếp với nhau để được đầu ra cuối cùng
- Kiểu tích chập này giảm các phép tính toán và tham số, nâng cao tốc độ và cải thiện bộ nhớ



Hình 2.5 Minh họa kiểu tích chập theo chiều sâu

* *Tích chập theo từng điểm (Pointwise Convolution)*

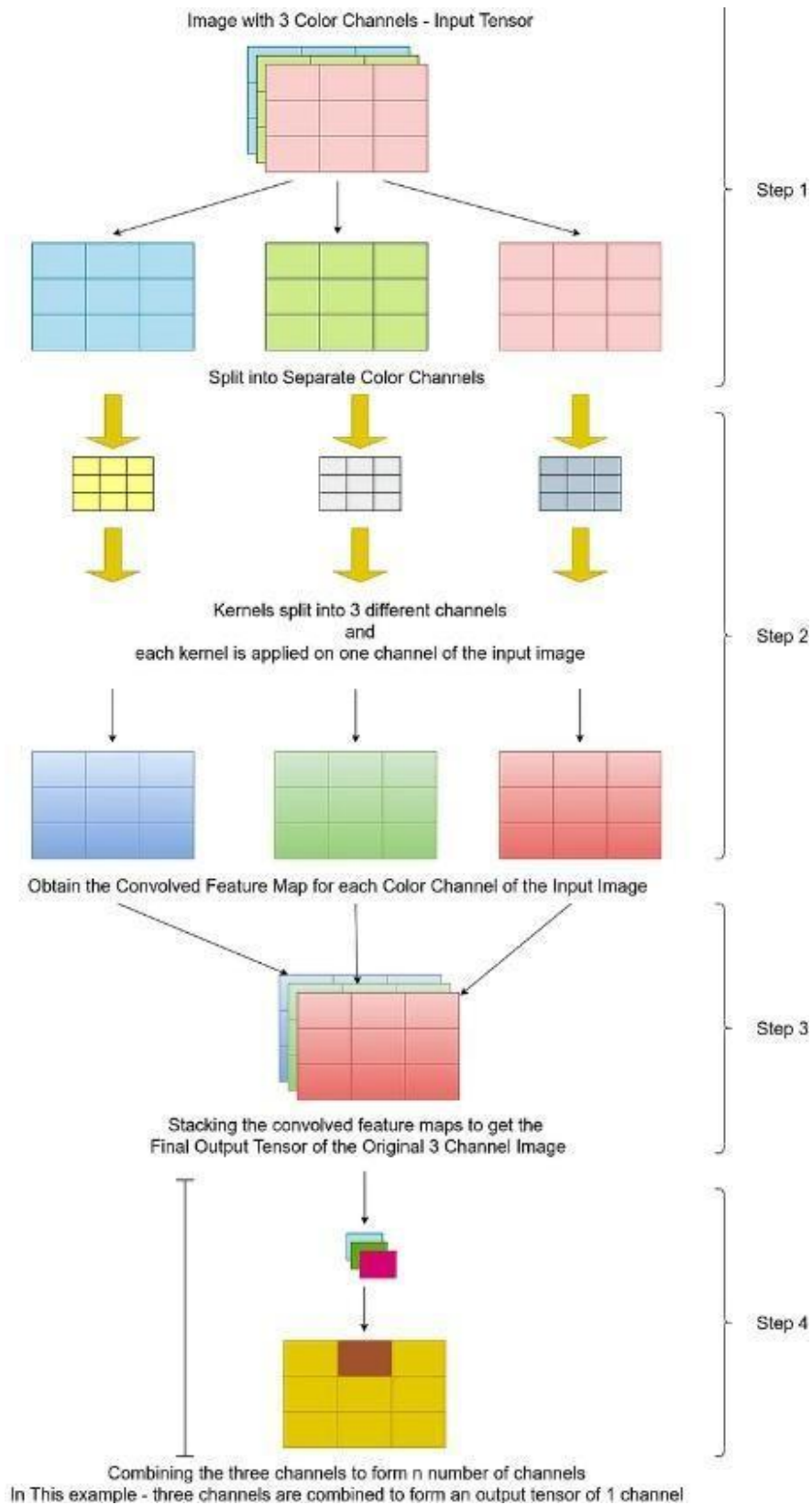
- Các lớp tích chập 1x1 làm thay đổi độ sâu bản đồ đặc trưng mà không làm biến đổi kích thước không gian
- Giảm số kênh và các tổ hợp thông tin
- Tính toán hiệu quả hơn, thường được sử dụng cùng với tích chập theo chiều sâu
- Trong các kiến trúc phức tạp, chúng giảm độ phức tạp tính toán



Hình 2.6 Minh họa kiểu tích chập theo từng điểm

* *Tích chập phân tách theo chiều sâu (Depthwise separable convolution)*

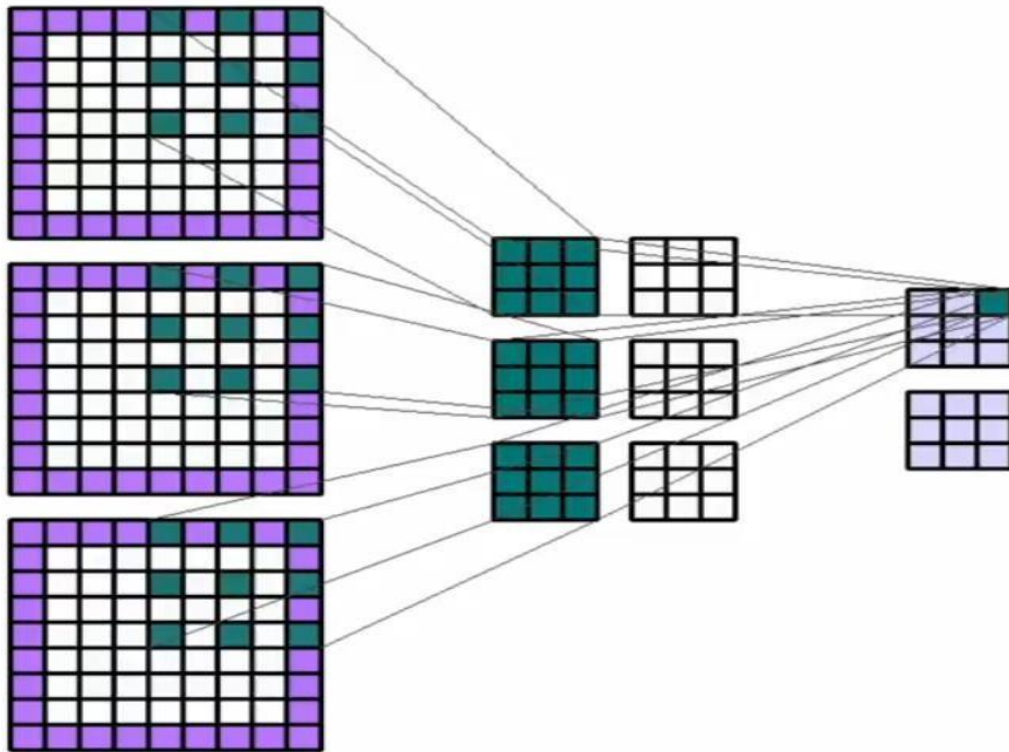
- Kiểu tích chập này là tổ hợp của tích chập theo chiều sâu và tích hợp theo từng điểm
- Tích chập theo chiều sâu sẽ lấy các thông tin về không gian
- Tích chập theo từng điểm đóng vai trò tăng độ sâu và tổng hợp đặc trưng
- Giảm số lượng tính toán phức tạp so với loại tích chập chuẩn (tích chập 2 chiều)
- Phổ biến với các mô hình nhẹ như MobileNet



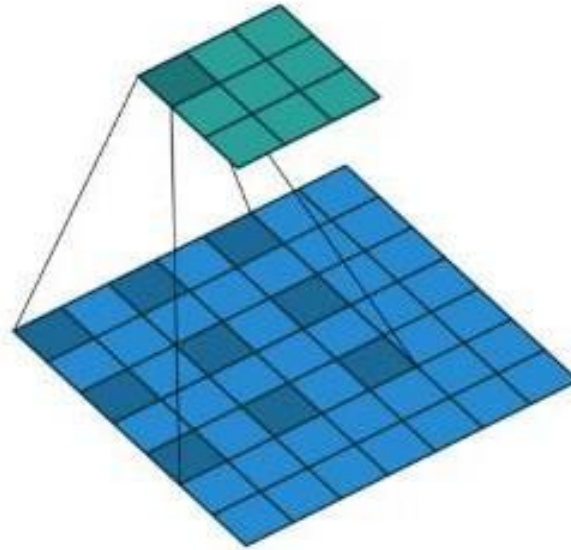
Hình 2.7 Minh họa tích chập phân tách theo chiều sâu

* *Tích chập giãn (Dilated Convolution)*

- Tích chập giãn giới thiệu 1 tác nhân giãn
- Việc giãn mang lại khả năng tăng receptive field mà không tăng kích cỡ hay số tham số
- Receptive field trong deep learning được định nghĩa là kích thước của một vùng (region) trong không gian đầu vào (input space) được nhìn thấy bởi pixel output qua một kernel/filter
- Được sử dụng để lấy thông tin trong các tấm ảnh có độ phân giải lớn và độ phức tạp cao
- Thường được sử dụng cho các bài toán phân loại hình ảnh và nhận diện đối tượng



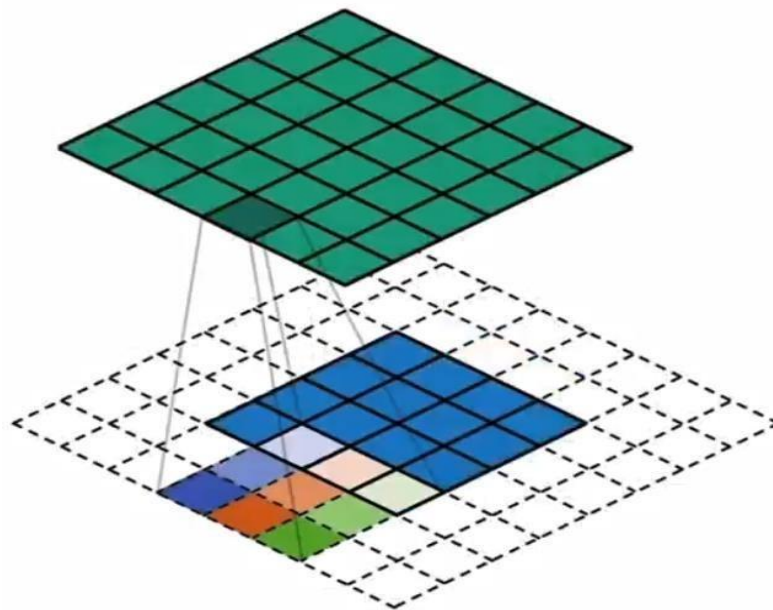
Hình 2.8 Minh họa tích chập giãn



Hình 2.9 Minh họa tích chập giãn từng phần

* *Tích chập chuyển đổi (Transposed Convolution)*

- Tích chập chuyển đổi làm tăng kích cỡ không gian (tăng số mẫu)
- Thường được sử dụng trong phân loại hình ảnh và các mô hình tạo sinh
- Mục tiêu là cải thiện độ phân giải cho đầu ra tốt hơn so với đầu vào
- Các bộ lọc có thể học sẽ làm tăng và không làm giảm kích cỡ không gian



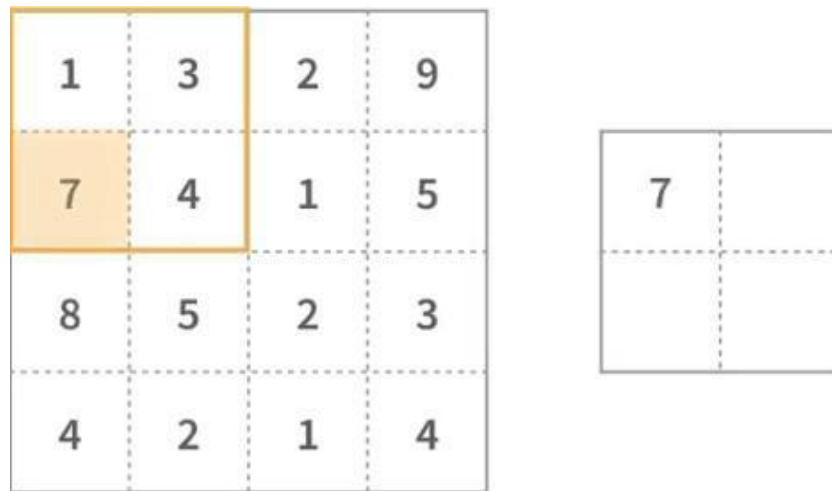
Hình 2.10 Minh họa tích chập chuyển đổi

* **Tầng Pooling**

Tầng pooling là một phép downsampling (giảm kích thước), thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra. Ngoài ra, còn có global average pooling và adaptive pooling cũng là các dạng pooling trong CNNs.

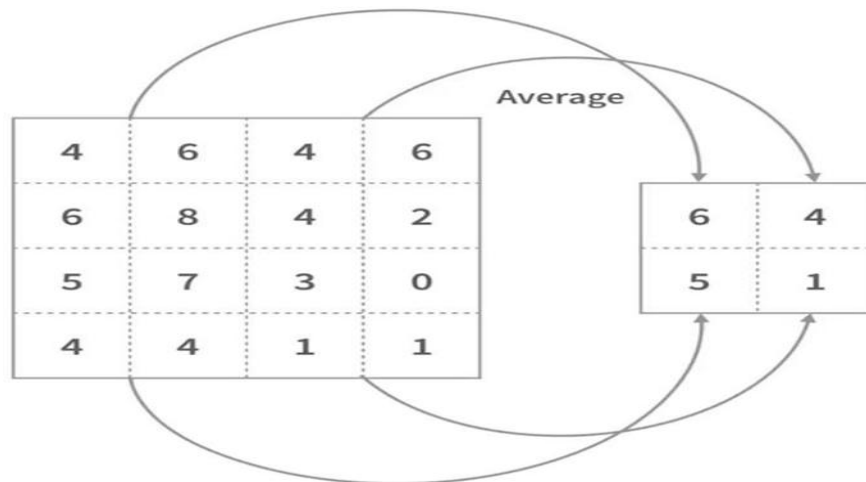
- Pooling và Hierarchical feature learning trong CNNs
 - Pooling layer giảm kích thước bản đồ đặc trưng, cải thiện độ hiệu quả
 - Pooling thêm các không gian bất biến (invariance), nâng cao tính tổng quan
 - Hierarchical feature learning tổ hợp Pooling và Convolution layer
 - Các tầng sâu hơn trong kiến trúc sẽ có Receptive field rộng hơn và có khả năng học được các đặc trưng có cấp độ cao hơn và phức tạp hơn
 - Phân tích được sự phức tạp từ các dữ liệu đầu vào qua đó cải thiện hiệu năng
- Max pooling
 - Là tầng Pooling phổ biến nhất hiện nay được sử dụng để tổng hợp đặc trưng
 - Một cửa sổ sẽ trượt bên trong không gian của ma trận đặc trưng đầu vào
 - Giữ lại các giá trị lớn nhất trong cửa sổ và ghi vào một ma trận nhỏ hơn

- Việc này đảm bảo giữ lại các đặc trưng quan trọng trong khi xóa bỏ các thông tin kém quan trọng hơn



Hình 2.11 Minh họa Max pooling

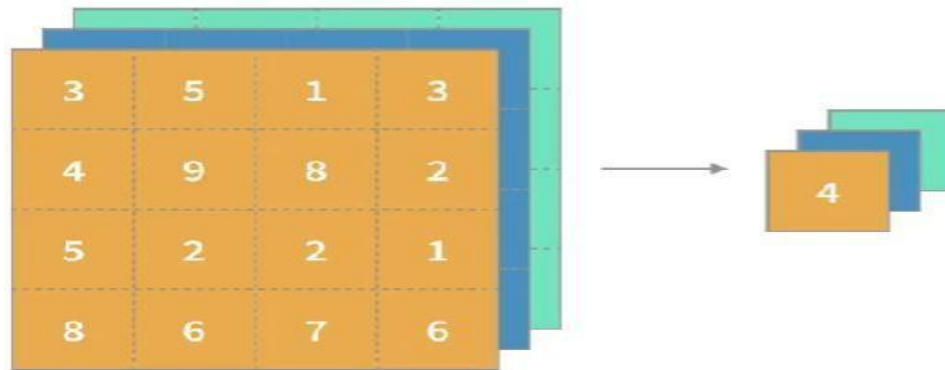
- Average pooling
 - Ảnh hưởng tới tính năng và tính tổng quan của giá trị dựa trên các tác vụ và dữ liệu đầu vào Average pooling tính toán giá trị trung bình trong cửa sổ pooling
 - Cân bằng giữ các tính năng quan trọng và kém quan trọng trong cửa sổ
 - Đảm bảo giữ lại những thông tin có tính tổng quan



Hình 2.12 Minh họa Average pooling

- Global average pooling

- Giảm toàn bộ các bản đồ đặc trưng thành 1 giá trị đơn lẻ
- Lấy được toàn bộ nội dung trên các kênh và nén các thông tin đa chiều
- Phải đảm bảo cân bằng giữa các đặc trưng quan trọng và kém quan trọng
- Thường được sử dụng trước lớp phân lớp cuối cùng, làm giảm số lượng tham số cần thiết



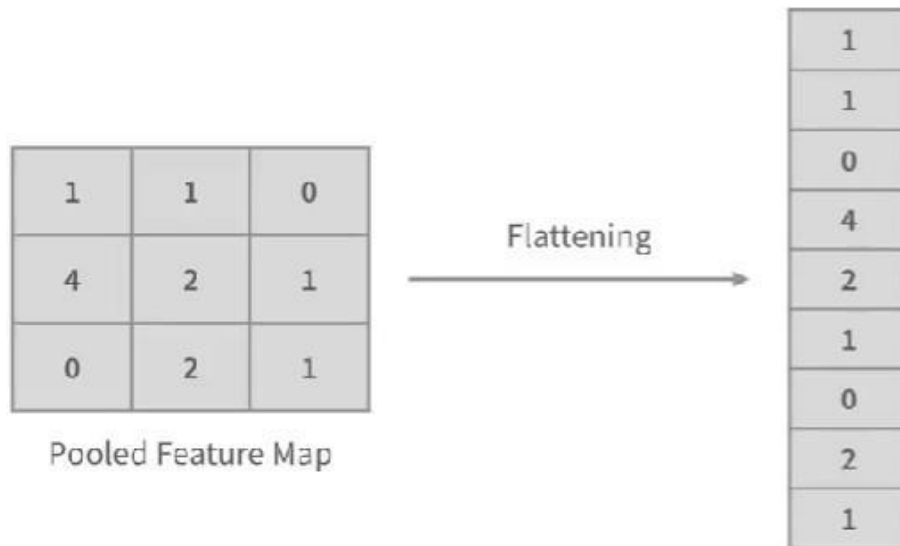
Hình 2.13 Minh họa Global average pooling

- Adaptive pooling
 - Adaptive pooling tự động xác định kích cỡ cửa sổ pooling dựa trên kích cỡ dữ liệu đầu vào
 - Hiệu quả hơn cho các đầu vào với kích cỡ khác nhau so với các kích cỡ đã được cấu hình sẵn
 - Trong khi các dữ liệu với kích cỡ được cấu hình sẵn sẽ không phù hợp cho tất cả các bản đồ đặc trưng đầu vào
 - Adaptive Pooling sẽ tính toán kích cỡ cửa sổ và thanh trượt phù hợp với mục tiêu đầu ra

* Tầng kết nối đầy đủ (Fully Connected Layer)

Tầng kết nối đầy đủ nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả nơ ron. Trong mô hình mạng CNN, các tầng kết nối đầy đủ thường được đặt ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.

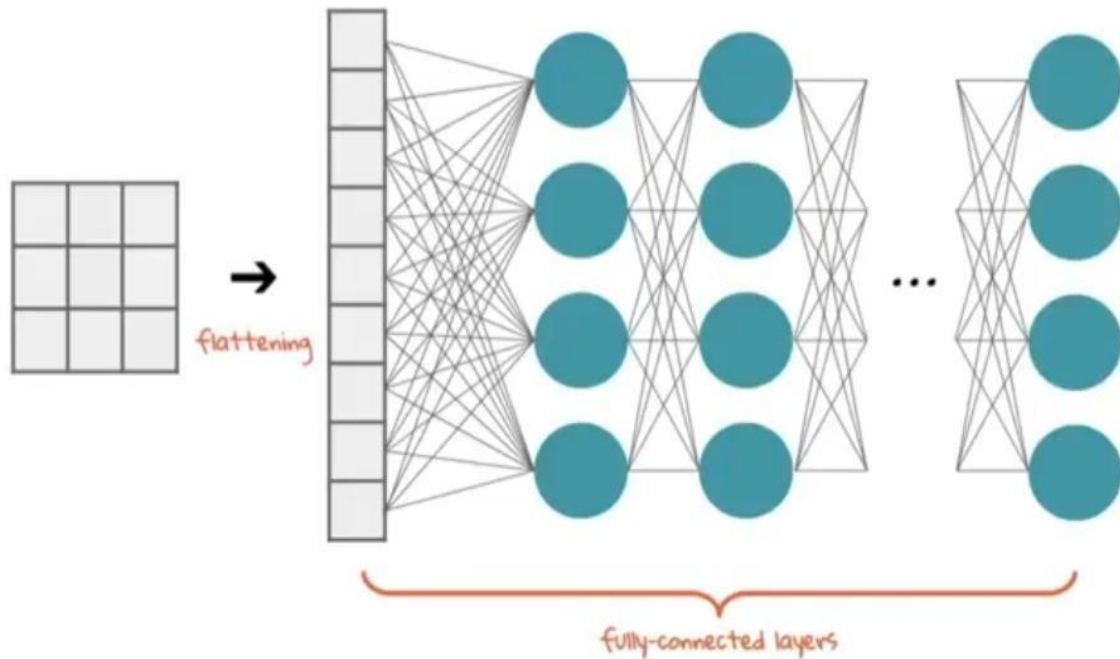
- Các chức năng của tầng kết nối đầy đủ trong CNNs
 - Diễn giải và tổ hợp các đặc trưng từ các tầng tích chập
 - Chuyển đổi các bản đồ đặc trưng đa chiều vào trong một phân phối xác suất
 - Lấy các đặc trưng phức tạp và tóm tắt các đặc trưng phục vụ cho phân loại hình ảnh
 - Học để tổ hợp và diễn giải các đặc trưng phục vụ cho đầu ra mong muốn
 - Đảm bảo các đặc trưng quan trọng và các thông tin phức tạp, hay sự không tuyến tính trong quan hệ giữa các thuộc tính



Hình 2.14 Minh họa phép làm phẳng

- Làm phẳng
 - Các tầng tích chập và pooling tạo ra các bản đồ đặc trưng
 - Các bản đồ đặc trưng là các mảng đa chiều
 - Làm phẳng giúp chuyển các bản đồ đặc trưng sang các vector một chiều
 - Kết hợp các phần tử theo chiều sâu và cho phép đưa các vector vào tầng kết nối đầy đủ
 - Weight (trọng số) của ma trận và bias vector
 - Tầng kết nối đầy đủ bao gồm ma trận trọng số (w) và bias vector (b)

- Ma trận trọng số có kích cỡ $n \times m$ bao gồm n là số nơ ron và m là độ dài của vector đã được làm phẳng. Số chiều của bias vector là số nơ ron
- Các tham số của tầng kết nối đầy đủ có thể học được
- Cho phép chuyển đổi và đảm bảo không tuyến tính giữa các thuộc tính (đặc trưng)
- $Output = W * input + b$



Hình 2.15 Minh họa phép làm phẳng với các tầng nơ ron

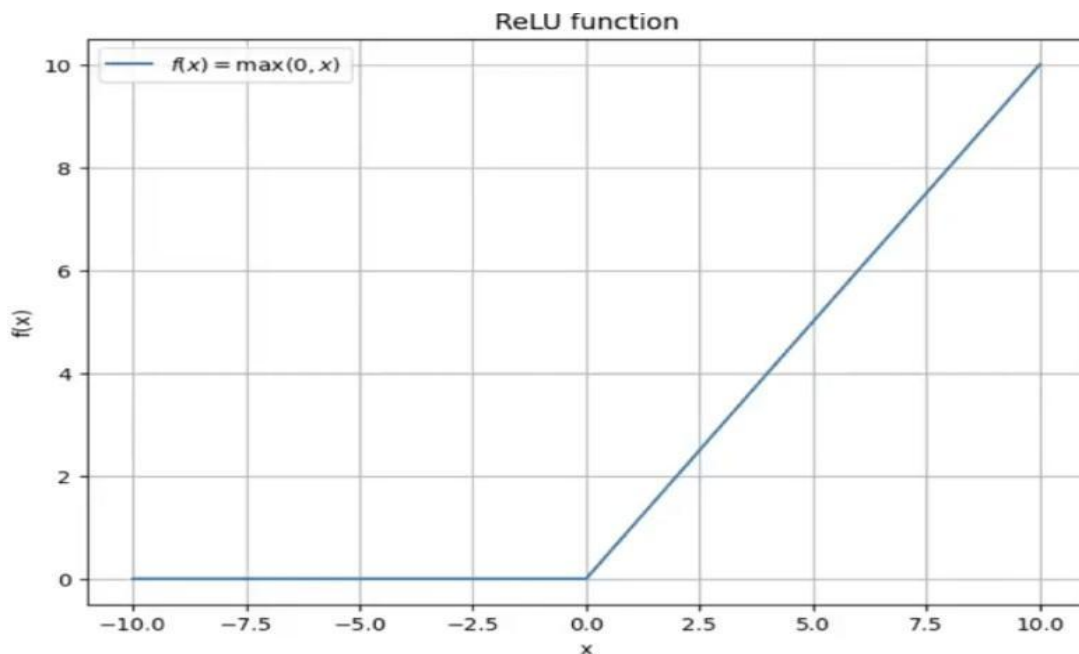
- Output layer
 - Đây là tầng cuối cùng đưa ra dự đoán
 - Các nơ ron trong tầng cuối này sẽ thực hiện các phép toán để đảm bảo đầu ra đảm bảo với các giá trị của các lớp
 - Hàm kích hoạt sẽ chuyển đổi giá trị từ output của tầng kết nối đầy đủ trong tầng này. Hàm softmax sử dụng phổ biến hơn trong các bài toán phân loại

* Các hàm kích hoạt

ReLU

- Hàm kích hoạt này chỉ giữ các giá trị lớn hơn 0

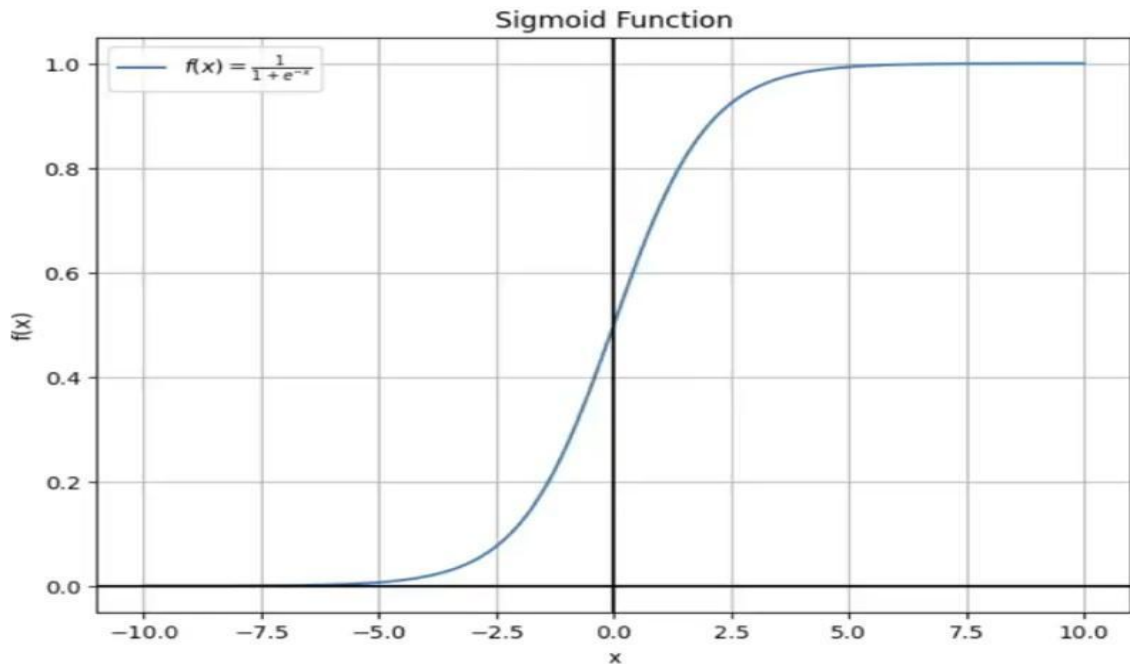
- Giảm thiểu vanishing gradient hay biến mất đạo hàm, cho phép học nhanh hơn
- Vanishing gradient có thể hiểu là việc các lớp ẩn tính toán khiến các giá trị đầu ra ở các lớp này rất nhỏ, gần như bằng 0. Như vậy, các giá trị tiến dần về 0 sẽ làm khó thực hiện được việc cập nhật trọng số trong quá trình điều chỉnh tham số
- Khuyến khích việc biểu diễn giá trị một cách thưa thớt để giảm thiểu tình trạng overfitting
- Có vấn đề xảy ra khi mà giá trị của hàm kích hoạt bằng 0 đó là “dead nơ ron”



Hình 2.16 Minh họa hàm kích hoạt ReLU

Sigmoid

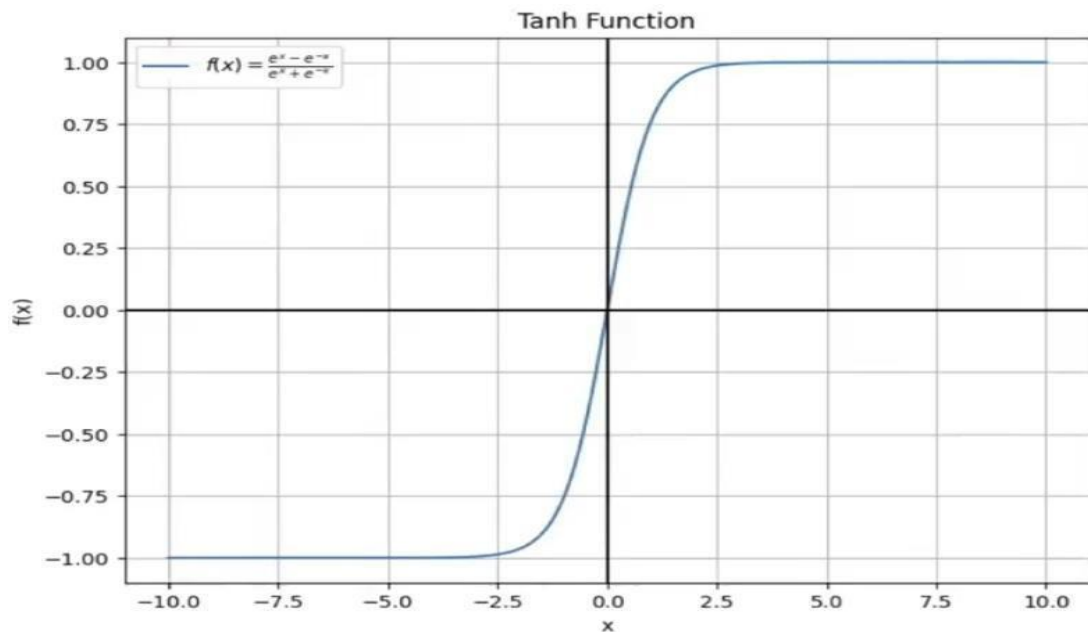
- Các giá trị đầu ra sẽ nằm trong khoảng từ 0 đến 1
- Giúp mượt và phân biệt một các tốt hơn để phù hợp với lan truyền ngược
- Dễ bị biến mất đạo hàm với các giá trị đầu vào lớn
- Việc biến mất đạo hàm thể làm quá trình học diễn ra chậm hơn



Hình 2.17 Minh họa hàm kích hoạt Sigmoid

Tanh

- Các giá trị đầu ra sẽ nằm trong khoảng từ -1 đến 1
- Cung cấp đầu ra cân bằng hơn khi đặt giá trị 0 là giá trị phân tách
- Hàm kích hoạt sẽ mượt và các giá trị đưa ra phân biệt tốt hơn
- Có thể giải quyết được vấn đề biến mất đạo hàm của sigmoid



Hình 2.18 Minh họa hàm kích hoạt Tanh

GELU

GELU là một hàm kích hoạt được giới thiệu vào năm 2016. Nó có dạng phi tuyến mượt mà và được sử dụng rộng rãi trong các mô hình ngôn ngữ lớn như BERT. GELU xấp xỉ phép nhân của đầu vào với hàm phân phối tích lũy của phân phối chuẩn.

Công thức: $GELU(x) = x * \Phi(x)$ Trong đó $\Phi(x)$ là hàm phân phối tích lũy của phân phối chuẩn.

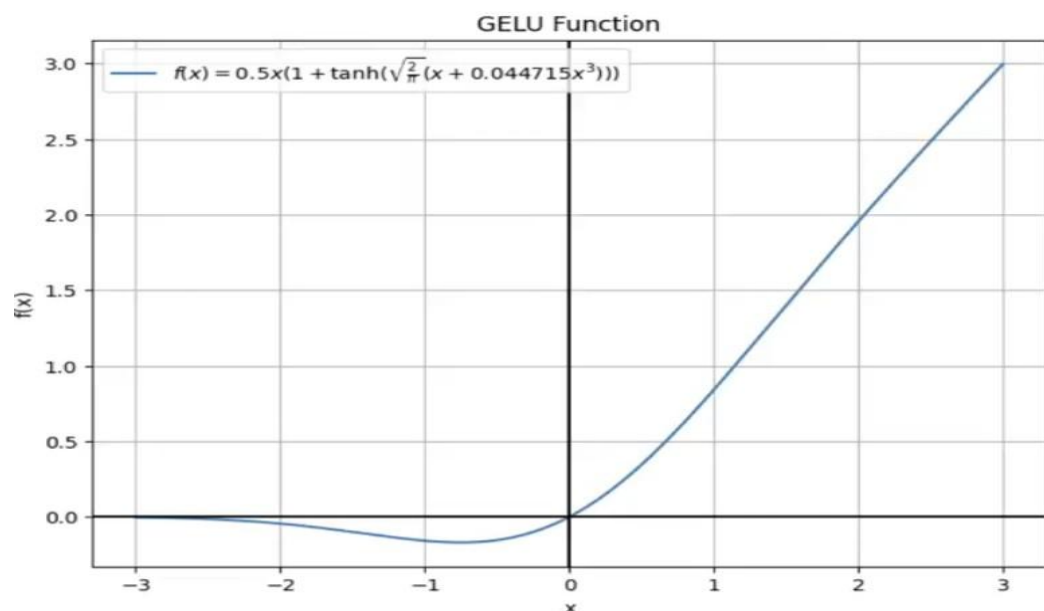
- GELU có các hành vi tương tự ReLU với đầu vào positive ($x \geq 0$)
- Thay vì làm tròn về không với các đầu vào negative như ReLU hàm này đảm bảo với các giá trị đó đầu ra là nonzero
- Hàm kích hoạt này thực hiện tốt với các kiến trúc học sâu, đặc biệt là Transformers

* Ưu điểm:

- Hiệu suất tốt trong nhiều tác vụ học sâu
- Phi tuyến tính mượt mà, giúp gradient flow tốt hơn

* Nhược điểm:

- Tính toán phức tạp hơn so với ReLU



Hình 2.19 Minh họa hàm kích hoạt GELU

SiLU

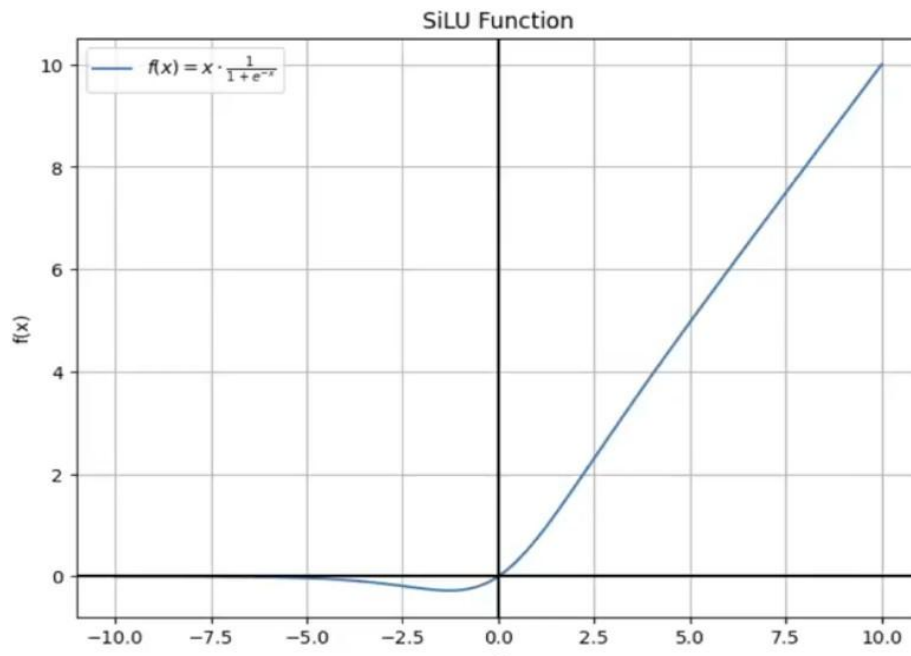
SiLU, còn được gọi là Swish, là một hàm kích hoạt được đề xuất bởi Google Brain vào năm 2017. Nó có dạng phi tuyến mượt mà và được coi là một lựa chọn thay thế cho ReLU trong nhiều ứng dụng.

Công thức: $SiLU(x) = x * sigmoid(x)$

- SiLU có thể coi là tổ hợp của Sigmoid và Linear functions
- Vượt trội hơn ReLU ở công việc phát hiện được những phần phức tạp
- Giới thiệu phương pháp self-gating phục vụ cho adaptive learning
- Cân bằng giữa tính chất hoặc tuyến tính và phi tuyến tính Ưu điểm:
- Hiệu suất tốt trong nhiều tác vụ học sâu
- Phi tuyến tính mượt mà, hỗ trợ gradient flow
- Không bị bão hòa ở các giá trị dương lớn

* Nhược điểm:

- Tính toán phức tạp hơn so với ReLU
- Có thể gây ra vấn đề gradient vanishing ở các giá trị âm lớn



Hình 2.20 Minh họa hàm kích hoạt SiLU

ReLU6

ReLU6 là một biến thể của hàm kích hoạt ReLU (Rectified Linear Unit) phổ biến. Nó giới hạn giá trị đầu ra tối đa là 6, giúp kiểm soát sự kích hoạt của nơ ron.

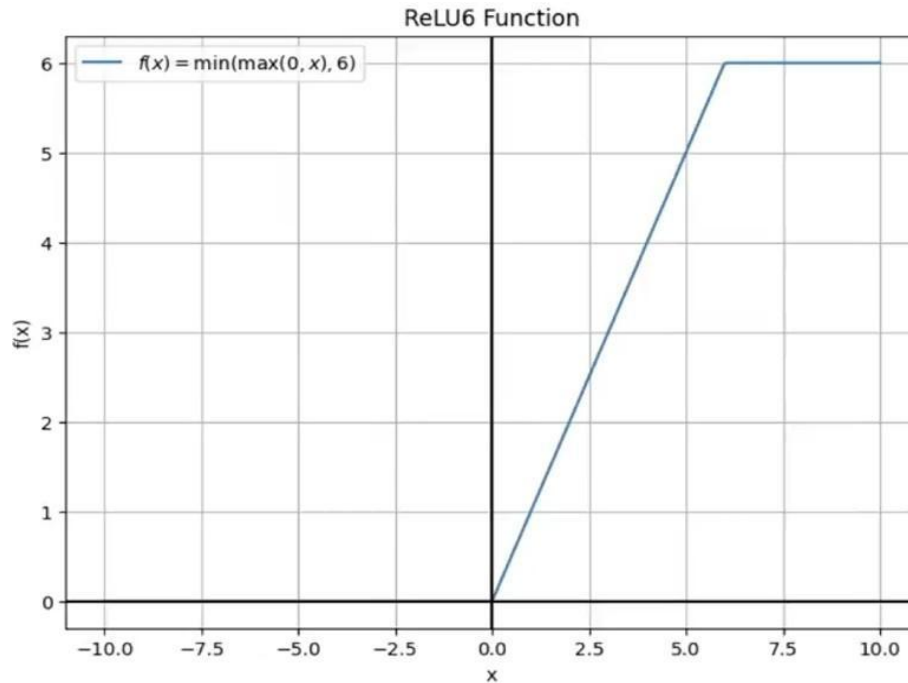
Công thức: $ReLU6(x) = \min(\max(0, x), 6)$

- ReLU6 là một biến thể của ReLU với giá trị đầu ra nằm từ 0 đến 6
- Bằng cách giới hạn đầu ra, ReLU6 đã tăng 1 yếu tố phi tuyến so với ReLU, qua đó cải thiện hiệu suất của mạng trong các tình huống cụ thể
- Được sử dụng trong các mô hình mạng yêu cầu đầu ra bị giới hạn theo yêu cầu
- Đặc biệt hữu ích với các mạng có độ chính xác thấp
- Đơn giản và hiệu quả về mặt tính toán
- Giúp kiểm soát sự kích hoạt, tránh overfitting
- Hữu ích trong các mô hình định lượng hóa (quantization)

* Nhược điểm:

- Có thể gây ra vấn đề "dying ReLU" khi gradient trở thành 0 cho các giá trị âm

- Giới hạn biểu diễn của mạng neural ở các giá trị cao hơn 6



Hình 2.21 Minh họa hàm kích hoạt ReLU6

2.4.3. Cách chọn tham số

- Số các lớp tích chập (convolutional layer): càng nhiều các lớp tích chập thì hiệu suất càng được cải thiện. Sau khoảng 3 hoặc 4 lớp, các tác động được giảm một cách đáng kể nhất.
- Kích thước mặt nạ tích chập (filter size): thường lựa chọn theo kích thước 5×5 hoặc 3×3.
- Kích thước pooling: thường là 2×2 hoặc 4×4 cho ảnh đầu vào lớn.
- Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt

2.5. Mạng nơ ron CNN - VGG16

2.5.1. Giới thiệu

VGG16 là một trong những kiến trúc mạng nơ-ron tích chập (CNN) nổi tiếng được đề xuất bởi nhóm nghiên cứu Visual Geometry Group tại Đại học Oxford. Mạng này được giới thiệu lần đầu trong bài báo “*Very Deep*

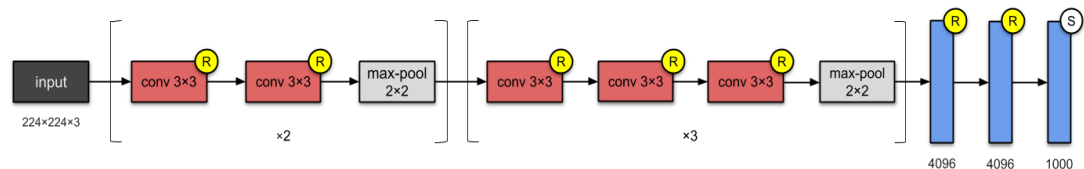
Convolutional Networks for Large-Scale Image Recognition” và đã đạt kết quả ấn tượng trong cuộc thi ImageNet năm 2014.

Điểm đặc biệt của VGG16 là cấu trúc đơn giản nhưng hiệu quả: chỉ sử dụng các lớp tích chập (convolutional) với kích thước kernel 3x3, xen kẽ với các lớp pooling 2x2, và kết thúc bằng ba lớp fully connected. Mạng có tổng cộng 16 lớp có trọng số học được (bao gồm 13 lớp conv và 3 lớp FC).

VGG16 tuy có số lượng tham số lớn (khoảng 138 triệu) nhưng lại rất dễ triển khai và được dùng rộng rãi trong các bài toán nhận dạng ảnh, trích xuất đặc trưng, và transfer learning trong học sâu.

2.5.2. Kiến trúc mạng VGG16

Kiến trúc của VGG16 gồm 5 khối (block) chính, mỗi khối bao gồm nhiều lớp tích chập (convolutional layer) sử dụng bộ lọc kích thước 3x3, stride 1 và padding “same”, giúp giữ nguyên kích thước đầu vào sau mỗi lớp tích chập. Sau mỗi khối là một lớp max pooling 2x2 có stride 2 để giảm kích thước không gian (spatial dimensions) của đặc trưng.



Hình 2.22 Minh họa kiến trúc mạng VGG16

Block 1: 2 lớp conv (64 filters) + max pooling

Block 2: 2 lớp conv (128 filters) + max pooling

Block 3: 3 lớp conv (256 filters) + max pooling

Block 4: 3 lớp conv (512 filters) + max pooling

Block 5: 3 lớp conv (512 filters) + max pooling

Sau các khối này là phần phân loại (classifier) gồm:

- 2 lớp fully connected (4096 neurons mỗi lớp)
- 1 lớp fully connected cuối cùng với số neuron bằng số lớp đầu ra (thường là 1000 đối với ImageNet)

Cuối cùng là hàm softmax để đưa ra xác suất dự đoán.

Tổng cộng, VGG16 có 16 lớp có trọng số học được (13 lớp conv + 3 lớp fully connected), và có khoảng 138 triệu tham số. Mặc dù khá nặng, mạng này lại rất hiệu quả trong việc trích xuất đặc trưng hình ảnh nhờ kiến trúc sâu và đồng nhất.

2.5.3. Ưu và nhược điểm

Ưu điểm:

- Đơn giản và nhất quán: VGG16 sử dụng toàn bộ kernel 3x3 và pooling 2x2 trong suốt mạng, giúp thiết kế mạng đồng đều, dễ triển khai và mở rộng.
- Hiệu quả trích xuất đặc trưng: Nhờ độ sâu lớn và cấu trúc nhiều lớp, VGG16 có khả năng học được các đặc trưng phức tạp của hình ảnh, phù hợp cho các bài toán nhận dạng và phân loại ảnh.
- Được sử dụng rộng rãi: Là một trong những mạng được pretrained phổ biến nhất, VGG16 dễ dàng áp dụng trong transfer learning để huấn luyện nhanh hơn và hiệu quả hơn với tập dữ liệu nhỏ.

Nhược điểm:

- **Rất nặng:** Với khoảng 138 triệu tham số, VGG16 tiêu tốn nhiều bộ nhớ và tài nguyên tính toán, gây khó khăn khi triển khai trên thiết bị có giới hạn như mobile hay edge device.
- **Chậm:** Việc xử lý qua nhiều lớp convolution và fully connected khiến thời gian suy luận (inference) dài hơn so với các mô hình tối ưu hơn như ResNet hay MobileNet.
- **Không tận dụng tốt gradient:** Do mạng sâu nhưng không có kỹ thuật như residual connections (ở ResNet), VGG16 dễ bị vấn đề gradient vanishing khi huấn luyện từ đầu trên tập lớn.

2.6. Mạng nơ-ron CNN – InceptionV3

2.6.1. Giới thiệu

InceptionV1 (GoogLeNet) được giới thiệu vào năm 2014 trong bài báo nổi tiếng "Going Deeper with Convolutions". Nó được xây dựng dựa trên ý tưởng Inception Module – việc kết hợp các bộ lọc khác nhau (1x1, 3x3, 5x5, pooling) song song trong cùng một lớp, giúp mạng nơ-ron có khả năng học các đặc trưng khác nhau ở nhiều kích thước khác nhau. Đã dành chiến thắng ở cuộc thi ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)

Việc cùng kết hợp đồng thời các bộ lọc có kích thước khác vào cùng một block có thể mang lại hiệu quả đó chính là kiến trúc khối Inception

InceptionV2 ra đời nhằm tối ưu hóa hơn về khả năng tính toán, với những cải tiến như batch normalization và factorization giúp giảm số lượng tham số mà không làm giảm độ chính xác.

InceptionV3 bao gồm 42 lớp, 24 triệu tham số là kế thừa từ 2 phiên bản trước, kiến trúc này giải quyết được vấn đề thắt cổ chai (representational bottlenecks) nhờ sử dụng phương pháp phân tích nhân tố (factorisation methods), Convolution-BN-ReLU, thuật toán tối ưu RMSProp, Label Smooth.

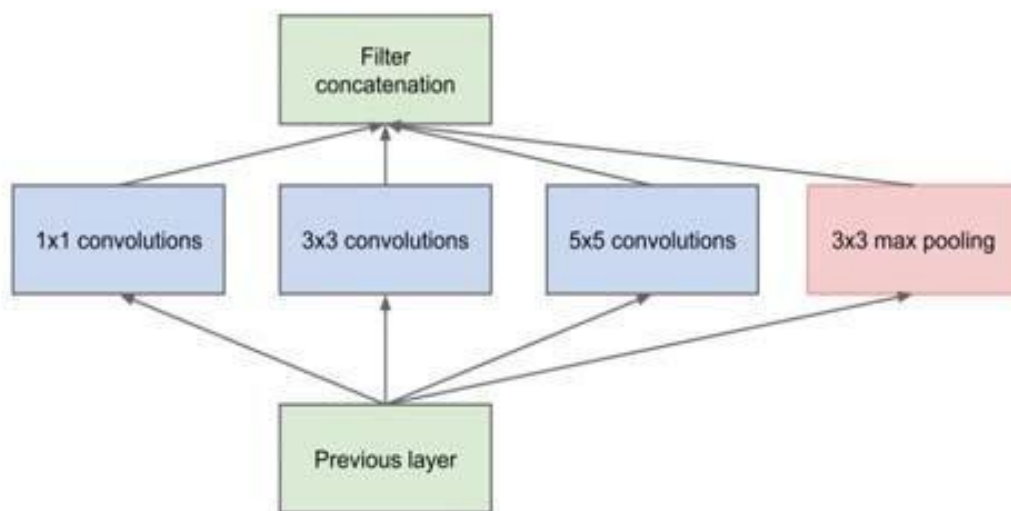
Cũng giống như ImageNet có thể được coi là cơ sở dữ liệu về các đối tượng trực quan được phân loại, Inception giúp phân loại các đối tượng trong thế giới thị giác máy tính. Kiến trúc InceptionV3 đã được sử dụng lại trong nhiều ứng dụng khác nhau, thường được sử dụng "được đào tạo trước" từ ImageNet. Một trong những ứng dụng như vậy là trong khoa học sự sống, nơi nó hỗ trợ nghiên cứu bệnh bạch cầu

Nó có tầm quan trọng về mặt lịch sử như một CNN ban đầu phân tách phần thân (thu thập dữ liệu), phần thân (xử lý dữ liệu) và phần đầu (dự đoán), một thiết kế kiến trúc vẫn tồn tại trong tất cả các CNN hiện đại.

2.6.2. Kiến trúc mạng InceptionV3

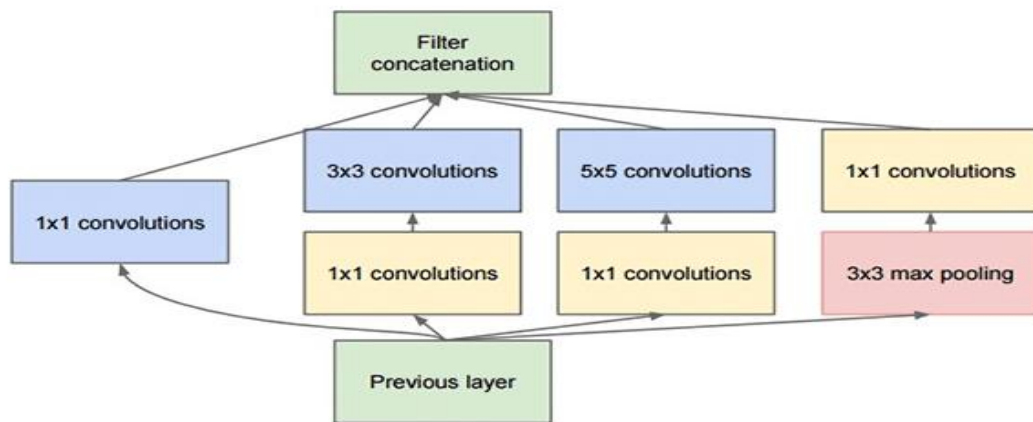
Ý tưởng

Ý tưởng chính được đưa ra của nhóm tác giả GoogleNet (tên chính thức của mô hình trong ILSVRC) là phát triển mô hình không chỉ theo chiều sâu, mà còn là chiều rộng. Thay vì việc chúng ta phải cố định sử dụng một loại kernel size, InceptionNet tạo ra các module sử dụng nhiều Convolution với các loại kernel size khác nhau như 3×3 , 5×5 , 7×7 , sau đó các đặc trưng sẽ được ghép nối với nhau.



Hình 2.23 Inception Module cơ bản

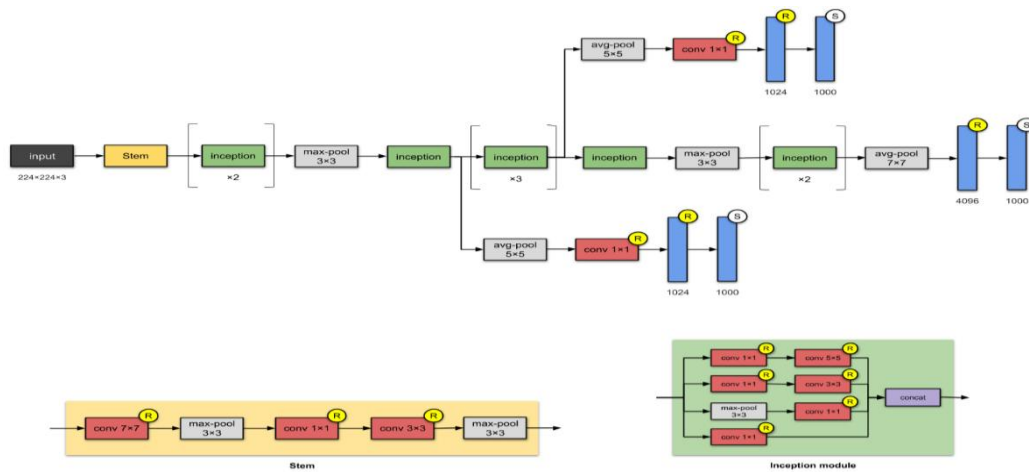
Việc này xuất phát từ ý tưởng từ việc học những phần khác nhau của bức ảnh với các vùng có kích thước khác nhau đòi hỏi ta cần có các kernel size chính xác, việc phối hợp nhiều loại kernel size giúp cho vùng nhận thức của mô hình rộng hơn. Tuy nhiên với việc dùng nhiều loại kernel như vậy sẽ làm tăng chi phí tính toán của mô hình, nhóm tác giả đã khắc phục vấn đề này bằng cách áp dụng convolution 1x1 trước khi đưa vào các loại kernel 3x3, 5x5; với mục tiêu làm giảm kích thước channel của feature map, từ đó giảm chi phí tính toán.



Hình 2.24 Inception Module với Convolution 1x1

Về Convolution 1x1, đây là một phương pháp được lấy ra từ mô hình [Network in Network](#). Vì sao gọi là Network in Network, vì khi tích chập 1 feature map $W \times H \times C$ với Convolution 1x1 với N filter, tại mỗi vị trí trong $W \times H$ vị trí, ta đang tiến hành tạo ra 1 mô hình Perceptron 1 lớp kết hợp C giá trị thành 1 giá trị, và được scale up lại lên N lần tạo ra feature map mới kích thước $W \times H \times N$ “cô đọng” lại từ $W \times H \times C$.

Mô hình Inception V1



Hình 2.25 Mô hình InceptionV1 với các Inception module

Một số tóm tắt về mô hình:

+ Mô hình bao gồm 22 lớp (nếu kể cả các lớp max-pooling là 27), nhưng thực tế số lớp được tính độc lập với nhau là 100 với các lớp Convolution được kết hợp trong các inception module.

+ Module đầu tiên trước khi đưa vào các Inception module là một Stem module với tuần tự 1 convolution 7x7, max pool 3x3, 1 convolution 1x1, 1 convolution 3x3 và max pool 3x3. Mục tiêu của lớp này không được tác giả giải thích, nhưng ta có thể hiểu là những lớp ban đầu với các kernel size lớn và các vùng nhận thức lớn sẽ học tốt cho các đặc trưng cấp thấp như tần số, góc cạnh, ...

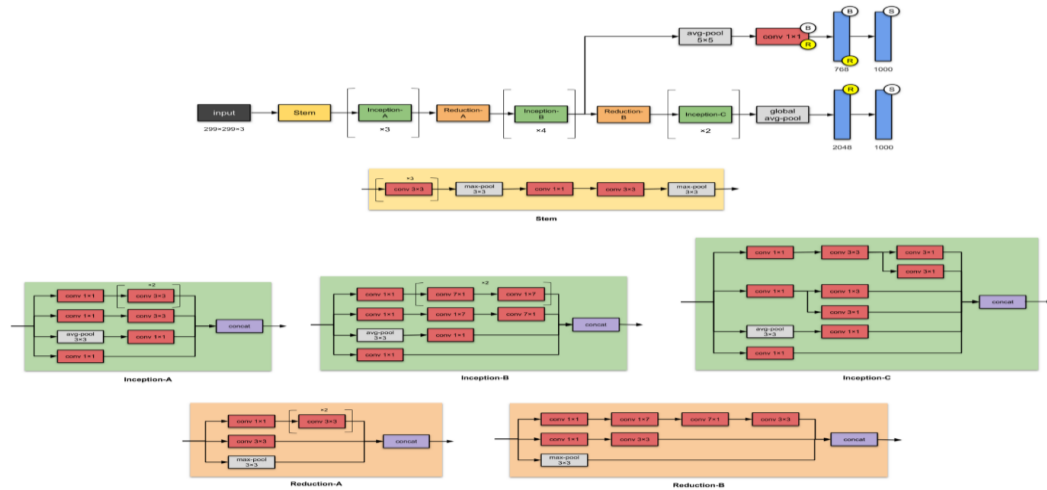
+ Mô hình sử dụng một lớp Global Average Pooling 7x7 trước khi đưa nó vào các lớp FC. Các lớp FC của mô hình Inception nhìn chung khá ít để giảm các tham số và dồn hết các phần học đặc trưng vào các lớp Convolution.

+ Ngoài ra các phiên bản được cập nhật của Inceptionv1 cũng đã thêm vào 2 nhánh phụ Auxiliary Branch (được trích ra từ các lớp khác nhau) đóng góp cho quá trình huấn luyện nhằm giảm thiểu vấn đề vanishing gradient.

Mô hình Inception V2, V3

Mô hình Inception-V2 được đề cập trong bài báo [Rethinking the Inception Architecture for Computer Vision](#) đưa ra các phương pháp cải tiến cho mô hình

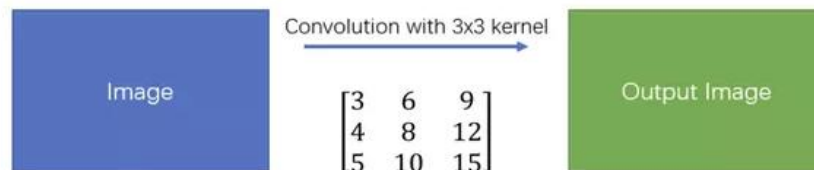
V1. Và mô hình Inception-V3 chính là Inception-V2 với việc thêm vào BatchNorm.



Hình 2.26 Mô hình InceptionV3 với các Inception module

Một số chỉnh sửa được đưa ra nhằm cải tiến để tối ưu các inception module:

Simple Convolution



Spatial Separable Convolution



Hình 2.27 Simple và Spatial Separable Convolution

+ Thay thế convolution 7x7, 5x5 tương ứng thành 3 convolution 3x3 và 2 convolution 3x3, với tác dụng như VGG, giảm số tham số nhưng vẫn giữ nguyên vùng nhận thức 7x7 hay 5x5.

+ Biến đổi các convolution thành các Spatial Seperable Convolution để giảm thiểu tham số cho các convolution 3×3 , 5×5 , 7×7 . Các convolution $n \times n$ sẽ được phân tách thành 2 convolution $n \times 1$ và $1 \times n$, tỉ lệ tham số giảm được tương đối lớn từ $n^2 \rightarrow 2n$.

+ Các thiết kế cho mạng học nhằm rộng hơn và cân bằng giữa chiều sâu và chiều rộng hơn so với mô hình v1. Ngoài ra, các mô hình sau còn thêm 1 số lớp Batch Normalization nhằm tăng tốc độ huấn luyện.

2.6.3. Ưu và nhược điểm

Ưu điểm:

- Hiệu quả tính toán cao: InceptionV3 sử dụng phương pháp factorized convolutions (phân rã các phép tích chập) để giảm thiểu số lượng phép tính, giúp giảm yêu cầu về tài nguyên tính toán mà vẫn đảm bảo độ chính xác cao.

- Chất lượng nhận diện tốt: Mạng InceptionV3 cải tiến khả năng phân loại hình ảnh so với các phiên bản trước. Nó đạt được kết quả tốt hơn trên các bộ dữ liệu hình ảnh lớn như ImageNet, với độ chính xác cao nhờ cấu trúc sâu và rộng của các lớp.

- Label Smoothing: Kỹ thuật này giúp cải thiện khả năng tổng quát hóa của mô hình, làm giảm nguy cơ overfitting bằng cách thêm độ nhiễu vào nhãn huấn luyện, giúp mạng không bị lệ thuộc quá mức vào dữ liệu huấn luyện.

- Phân tách các bộ lọc: Sử dụng bộ lọc 7×7 lớn được phân tách thành các bộ lọc nhỏ hơn (ví dụ: 3×3), giúp tăng hiệu quả tính toán và giảm số lượng tham số cần học.

- Cải tiến Regularization: Inception v3 sử dụng các kỹ thuật regularization như Dropout và auxiliary classifiers, giúp mạng ổn định hơn trong quá trình huấn luyện và tăng cường khả năng khái quát.

Nhược điểm:

- Độ phức tạp mô hình cao: Mặc dù Inception v3 đã cải thiện về mặt hiệu quả, nhưng cấu trúc của nó vẫn khá phức tạp với nhiều tầng tích chập, pool, và các tầng phân tách. Điều này khiến việc hiểu rõ và tối ưu hóa mạng trở nên khó khăn hơn so với các mô hình đơn giản.

- Khó triển khai trên các hệ thống nhẹ: Do số lượng lớp và tham số lớn, Inception v3 vẫn đòi hỏi nhiều tài nguyên tính toán và bộ nhớ. Điều này gây khó khăn khi triển khai trên các thiết bị hạn chế tài nguyên như điện thoại di động hoặc các hệ thống nhúng.

- Yêu cầu thời gian huấn luyện dài: Inception v3, với kích thước mô hình lớn, cần thời gian huấn luyện dài hơn, đặc biệt khi làm việc với các bộ dữ liệu lớn. Điều này có thể gây ra khó khăn nếu không có tài nguyên phần cứng đủ mạnh.

- Không phù hợp cho dữ liệu nhỏ: Mặc dù Inception v3 hoạt động tốt trên các bộ dữ liệu lớn như ImageNet, nhưng nó có thể không hiệu quả với những bộ dữ liệu nhỏ hơn, vì số lượng tham số lớn có thể dễ gây overfitting nếu không có đủ dữ liệu để huấn luyện.

CHƯƠNG 3: THỰC NGHIỆM

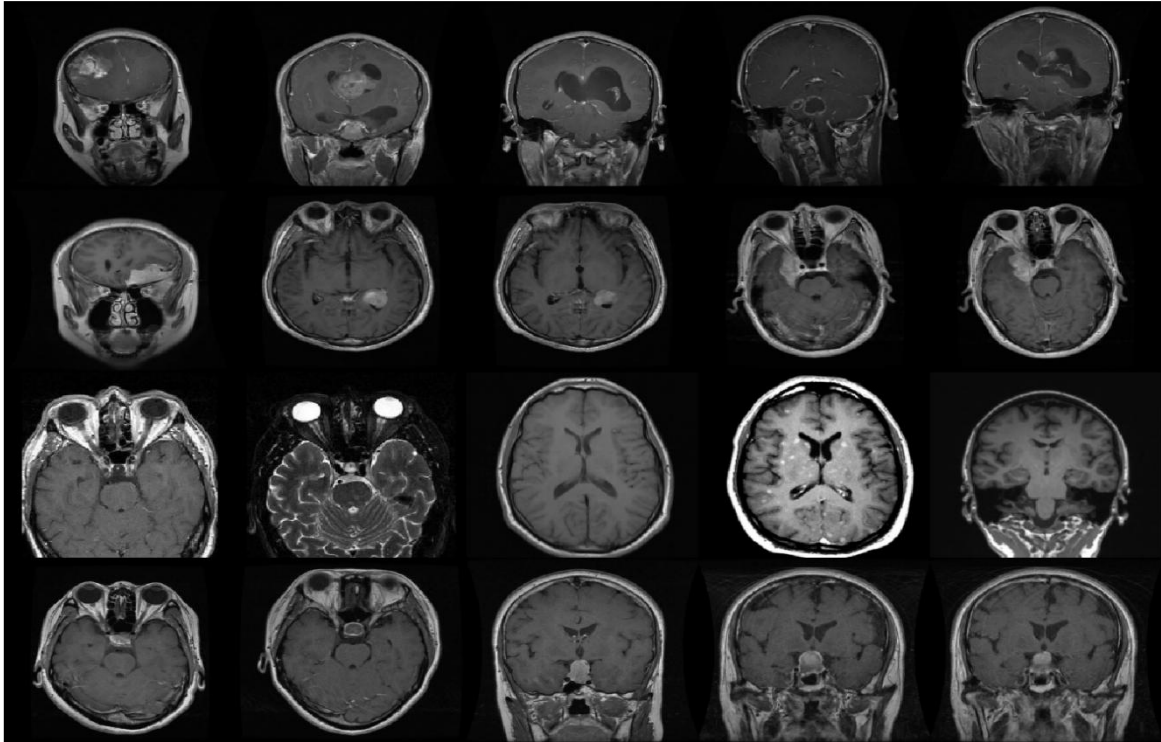
3.1. Dữ liệu thực nghiệm

Trong nghiên cứu này, tôi sử dụng bộ dữ liệu Brain Tumor MRI được thu thập từ Kaggle, bao gồm ảnh cộng hưởng từ (MRI) của não người được phân loại thành 4 loại u não khác nhau: glioma (u thần kinh đệm), meningioma (u màng não), pituitary tumor (u tuyến yên) và no tumor (không có u). Mỗi ảnh là ảnh chụp MRI ở dạng 2D với độ phân giải trung bình và định dạng chuẩn (thường là .jpg hoặc .png).

Bảng 3.1 Mô tả dữ liệu

Thông tin	Chi tiết
Tổng số hình ảnh	7023
Số lượng lớp phân loại	4
Số ảnh trong tập train	5712
Số ảnh trong tập test	1311

Tổng số lượng mẫu trong tập dữ liệu khoảng 7023 ảnh, bao gồm 5712 ảnh cho tập training và 1311 ảnh cho tập testing, được chia đều hoặc gần đều giữa các lớp. Dữ liệu được tổ chức thành các thư mục tương ứng với từng loại u não, thuận tiện cho việc trích xuất và tiền xử lý. Tập dữ liệu được sử dụng phổ biến trong các nghiên cứu về phân loại ảnh y tế, học sâu và nhận dạng bệnh lý não.



Hình 3.1 Một số hình ảnh về u não trong bộ dữ liệu Brain Tumor MRI

Ảnh MRI trong bộ dữ liệu có nhiều khác biệt về độ sáng, tương phản và nhiễu sinh lý. Một số ảnh có độ tương phản thấp hoặc nhiễu nền cao, đòi hỏi bước tiền xử lý như tăng cường tương phản (CLAHE), lọc nhiễu hoặc chuẩn hóa cường độ. Bộ dữ liệu này có xu hướng khá cân bằng giữa các lớp, mặc dù vẫn có sự chênh lệch nhỏ về số lượng ảnh từng loại. Điều này giúp hạn chế tình trạng mô hình thiên lệch nhưng vẫn cần kiểm tra kỹ nếu áp dụng các kỹ thuật học máy nhạy với mất cân bằng.

Trước khi huấn luyện mô hình, ảnh sẽ được tiền xử lý như: chuyển đổi kích thước, chuẩn hóa pixel, và có thể áp dụng augmentation (phép biến đổi ảnh) để tăng độ đa dạng dữ liệu.

3.2. Tiền xử lý dữ liệu

```

train_datagen = ImageDataGenerator(
    rescale = 1 / 255,
    rotation_range = 15,
    width_shift_range = 0.1,
    zoom_range = 0.01,
    shear_range = 0.01,
    brightness_range = [0.3, 1.5],
    horizontal_flip = True,
    vertical_flip = True
)
valid_datagen = ImageDataGenerator(rescale = 1./255, validation_split=0.2)

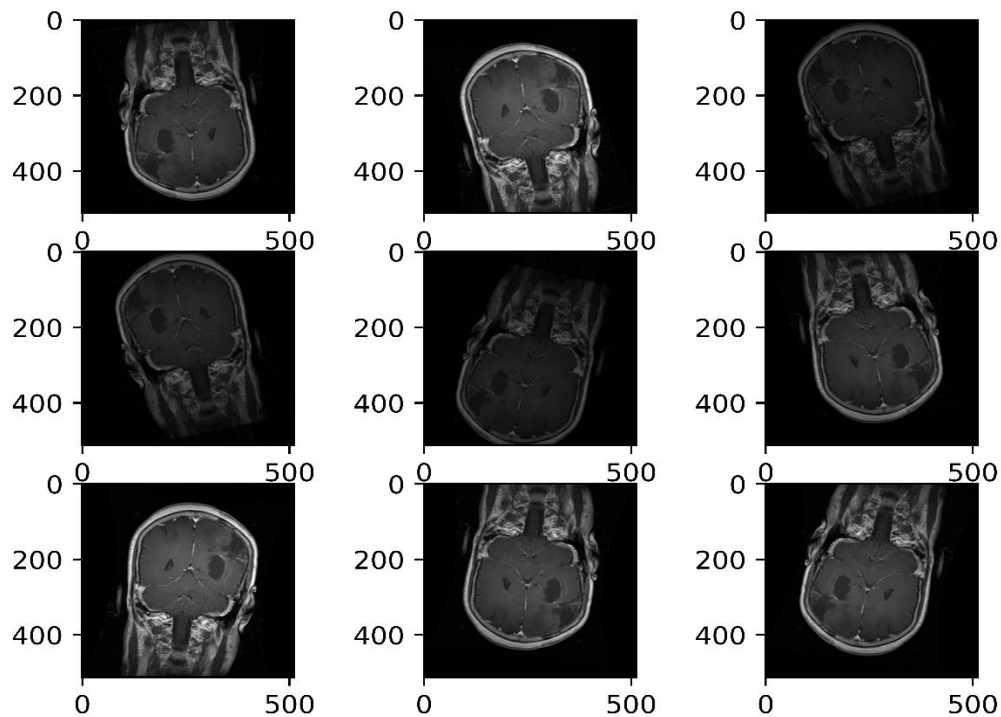
```

Hình 3.2 Ảnh code tiền xử lý dữ liệu

Ở bước này, tôi thực hiện biến đổi hình ảnh chỉ trên tập dữ liệu học (tập training), sử dụng thư viện ImageDataGenerator để thực hiện tăng cường dữ liệu ảnh (data augmentation), giúp cải thiện khả năng tổng quát hóa của mô hình. Cụ thể là chia tập dữ liệu học thành 2 tập nhỏ hơn: 80% cho tập training và 20% cho tập validation. Ở tập validation (tập kiểm định) chỉ thực hiện chuẩn hóa ảnh mà không áp dụng bất kỳ phép tăng cường ảnh nào khác. Còn ở tập training là tập sẽ sinh ra các ảnh để huấn luyện mô hình, sẽ bao gồm các phép biến đổi sau:

- `rescale = 1/255`: Chia toàn bộ pixel từ khoảng $[0, 255]$ về $[0, 1]$. Điều này giúp mô hình học ổn định hơn.
- `rotation_range = 15`: Xoay ảnh ngẫu nhiên trong khoảng ± 15 độ.
- `width_shift_range = 0.1`: Dịch ảnh theo chiều ngang với tỷ lệ tối đa là 10% của chiều rộng ảnh.
- `zoom_range = 0.01`: Phóng to hoặc thu nhỏ ảnh trong khoảng $\pm 1\%$ (tức là rất nhỏ, gần như không đổi).
- `shear_range = 0.01`: Áp dụng biến dạng cắt (shear) nhẹ cho ảnh.
- `brightness_range = [0.3, 1.5]`: Làm tối hoặc sáng ảnh trong khoảng 30% đến 150% độ sáng gốc.
- `horizontal_flip = True`: Lật ảnh theo chiều ngang (trái - phải) một cách ngẫu nhiên.

- `vertical_flip = True`: Lật ảnh theo chiều dọc (trên - dưới) một cách ngẫu nhiên.



Hình 3.3 Ví dụ về ảnh được huấn luyện qua từng epoch

3.3. Chuẩn bị dữ liệu cho mô hình

Sau khi đã cấu hình các biến đổi tiền xử lý dữ liệu, tiến hành chuẩn bị dữ liệu để huấn luyện mô hình. Dữ liệu sẽ được đọc từ thư mục training của tập Brain Tumor MRI và sẽ cố định kích thước ảnh huấn luyện là 224x224 và với kênh màu là RGB (3 kênh màu):

```
: train_generator = train_datagen.flow_from_directory(
    TRAIN_PATH,
    target_size=(img_width, img_height),
    color_mode='rgb',
    batch_size=batch_size,
    class_mode='categorical',
    subset='training',
    shuffle=True,
    seed=1337
)

valid_generator = valid_datagen.flow_from_directory(
    TRAIN_PATH,
    target_size=(img_width, img_height),
    color_mode='rgb',
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    shuffle=True,
    seed=1337
)
```

Hình 3.4 Ảnh code chuẩn bị dữ liệu huấn luyện

Với `train_generator`: Mỗi epoch có thể sử dụng một bộ dữ liệu khác nhau do ảnh được biến đổi ngẫu nhiên qua các phép tăng cường và xáo trộn lại dữ liệu sau mỗi epoch.

Với `valid_generator`: Dữ liệu validation giữ nguyên, nhưng thứ tự ảnh trong batch có thể khác nhau giữa các epoch.

3.4. Huấn luyện các mô hình

Do giới hạn về thời gian và phần cứng, tôi chỉ nghiên cứu tiến hành thực nghiệm trên 2 mô hình mạng nơ-ron tích chập đã nói ở Chương 2 là VGG16 và InceptionV3. Sau đó tiến hành so sánh, đánh giá kết quả huấn luyện của 2 mô hình này dựa trên các chỉ số thống kê. Vì GPU của Google Colab chỉ hỗ trợ trong 12 giờ sử dụng liên tục nhưng quá trình học diễn ra lâu hơn thời gian này, nên thực nghiệm sẽ được tiến hành trên máy tính cá nhân (local) để tránh xảy ra gián đoạn khi mô hình đang được huấn luyện.

3.4.1. Mô hình mạng nơ ron VGG16

Đầu tiên, mạng nơ ron VGG16 mà tôi đề xuất sẽ được huấn luyện và phân lớp trên các bộ dữ liệu. Mạng này sẽ không bao gồm lớp kết nối đầy đủ (fully connected) mà thay vào đó sẽ thêm vào một số lớp khác nhằm để phù hợp với bài toán.

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(img_height, img_width, channels))
for layer in base_model.layers:
    layer.trainable = False # freeze backbone

x = base_model.output
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(4, activation='softmax')(x)
```

Hình 3.5 Cấu trúc mô hình VGG16

Các lớp trong mô hình gốc sẽ được cố định (không huấn luyện lại), nhằm đảm bảo đặc trưng tổng quát. Đầu ra từ mô hình gốc được đưa qua lớp Flatten để làm phẳng các đặc trưng, sau đó đưa vào lớp Dense với 256 đầu ra với hàm

kích hoạt là Relu, kết hợp với Dropout (tỷ lệ 0.5) để giảm thiểu overfitting. Lớp Dense cuối sử dụng 4 đầu ra là 4 lớp cần phân loại với hàm kích hoạt Softmax để dự đoán xác suất cho mỗi lớp

Tiếp đó, thực hiện huấn luyện mô hình:

```
[ ]: model = Model(inputs=base_model.input, outputs=predictions)
    model.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy', metrics=['accuracy'])

[ ]: early_stopping = EarlyStopping(
    monitor='val_loss', # Theo dõi loss trên tập validation
    patience=10, # Dừng nếu val_loss không giảm sau 10 epochs
    restore_best_weights=True, # Lấy lại model tốt nhất
    min_delta=1e-4 # Chỉ dừng nếu giảm không đáng kể
)

callbacks = [early_stopping]

[ ]: history = model.fit(
    train_generator,
    epochs = 100,
    validation_data = valid_generator,
    verbose = 1,
    callbacks = callbacks,
    shuffle = True
)
```

Hình 3.6 Huấn luyện mô hình VGG16

Mô hình được huấn luyện bằng cách sử dụng kiến trúc được xây dựng ở bên trên, sau đó được biên dịch với hàm mất mát `categorical_crossentropy`, tối ưu hóa bằng thuật toán Adam với tốc độ học là 0.0001, và theo dõi độ chính xác (accuracy) trong quá trình huấn luyện. Quá trình huấn luyện được thực hiện với tối đa 100 epoch trên tập dữ liệu huấn luyện (`train_generator`), sử dụng tập validation (`valid_generator`) để đánh giá. Đồng thời, kỹ thuật `EarlyStopping` được áp dụng nhằm dừng sớm quá trình huấn luyện nếu giá trị `val_loss` không cải thiện đáng kể (ít nhất 0.0001) sau 10 epoch liên tiếp. Khi dừng sớm, mô hình sẽ khôi phục lại trọng số tốt nhất trước đó để đảm bảo hiệu suất tối ưu. Dữ liệu đầu vào được shuffle mỗi epoch để tăng tính ngẫu nhiên và giúp mô hình học tổng quát hơn.

```

Epoch 78/100
90/90 ----- 661s 7s/step - accuracy: 0.9463 - loss: 0.1454 - val_accuracy: 0.9798 - val_loss: 0.0740
Epoch 79/100
90/90 ----- 659s 7s/step - accuracy: 0.9432 - loss: 0.1703 - val_accuracy: 0.9667 - val_loss: 0.0915
Epoch 80/100
90/90 ----- 657s 7s/step - accuracy: 0.9388 - loss: 0.1525 - val_accuracy: 0.9720 - val_loss: 0.0725
Epoch 81/100
90/90 ----- 646s 7s/step - accuracy: 0.9403 - loss: 0.1646 - val_accuracy: 0.9649 - val_loss: 0.0942
Epoch 82/100
90/90 ----- 596s 7s/step - accuracy: 0.9466 - loss: 0.1423 - val_accuracy: 0.9781 - val_loss: 0.0648
Epoch 83/100
90/90 ----- 483s 5s/step - accuracy: 0.9441 - loss: 0.1511 - val_accuracy: 0.9772 - val_loss: 0.0741
Epoch 84/100
90/90 ----- 484s 5s/step - accuracy: 0.9489 - loss: 0.1389 - val_accuracy: 0.9711 - val_loss: 0.0911
Epoch 85/100
90/90 ----- 485s 5s/step - accuracy: 0.9447 - loss: 0.1517 - val_accuracy: 0.9755 - val_loss: 0.0748
Epoch 86/100
90/90 ----- 485s 5s/step - accuracy: 0.9473 - loss: 0.1465 - val_accuracy: 0.9763 - val_loss: 0.0658
Epoch 87/100
90/90 ----- 486s 5s/step - accuracy: 0.9488 - loss: 0.1333 - val_accuracy: 0.9720 - val_loss: 0.0766
Epoch 88/100
90/90 ----- 486s 5s/step - accuracy: 0.9453 - loss: 0.1503 - val_accuracy: 0.9772 - val_loss: 0.0649
Epoch 89/100
90/90 ----- 485s 5s/step - accuracy: 0.9437 - loss: 0.1497 - val_accuracy: 0.9746 - val_loss: 0.0727
Epoch 90/100
90/90 ----- 486s 5s/step - accuracy: 0.9396 - loss: 0.1437 - val_accuracy: 0.9807 - val_loss: 0.0696
Epoch 91/100
90/90 ----- 485s 5s/step - accuracy: 0.9451 - loss: 0.1490 - val_accuracy: 0.9702 - val_loss: 0.0872
Epoch 92/100
90/90 ----- 487s 5s/step - accuracy: 0.9516 - loss: 0.1356 - val_accuracy: 0.9711 - val_loss: 0.0842

```

Hình 3.7 Quá trình huấn luyện mô hình VGG16 qua các epoch

Kết quả nhận dạng của mạng này được coi là kết quả của mạng VGG16 và được dùng để so sánh với mô hình InceptionV3.

3.4.2. Mô hình mạng nơ ron InceptionV3

```

: base_model = InceptionV3(input_shape=(img_height, img_width, channels), weights='imagenet', include_top
for layer in base_model.layers[:30]: # Giữ nguyên các Layer đầu, chỉ fine-tune Layer cuối
    layer.trainable = False

inputs = Input(shape=(img_height, img_width, channels))
x = base_model(inputs)
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01))(x)
x = Dropout(0.5)(x)
x = Dense(4, activation='softmax')(x) # 4 Lớp đầu ra
x = Dense(512, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001))(x)
x = BatchNormalization()(x)
x = Dropout(0.3)(x) # Thêm Dropout 30% nữa
x = Dense(4, activation='softmax')(x)

```

Hình 3.8 Cấu trúc mô hình InceptionV3

Xây dựng kiến trúc mô hình dựa trên kiến trúc InceptionV3 đã được huấn luyện sẵn trên tập ImageNet, trong đó các tầng đầu được giữ nguyên và chỉ fine-tune 30 tầng cuối nhằm tận dụng khả năng trích xuất đặc trưng mạnh mẽ của mô hình pretrained đồng thời giảm nguy cơ overfitting. Đầu vào của mô hình có kích thước xác định và được truyền qua base_model, sau đó đi qua tầng

GlobalAveragePooling2D để giảm chiều và giữ lại đặc trưng toàn cục. Phần phân loại gồm nhiều tầng fully-connected được thiết kế xen kẽ với các kỹ thuật giảm overfitting như regularization L2, dropout và batch normalization. Cụ thể, mô hình có một tầng Dense với 1024 nơ-ron (ReLU, L2=0.01) kết hợp với dropout 50%, tiếp theo là một tầng Dense với 512 nơ-ron (ReLU, L2=0.001), batch normalization và dropout 30%. Cuối cùng, tầng đầu ra là một tầng Dense với 4 nơ-ron và hàm kích hoạt softmax để phân loại dữ liệu thành 4 lớp.

Tiếp đó, thực hiện huấn luyện mô hình:

```
early_stopping = EarlyStopping(
    monitor='val_loss', # Theo dõi loss trên tập validation
    patience=10, # Dừng nếu val_loss không giảm sau 10 epochs
    restore_best_weights=True, # Lấy lại model tốt nhất
    min_delta=1e-4 # Chỉ dừng nếu giảm không đáng kể
)
lr_scheduler = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=6, min_delta=0.0001)

callbacks = [early_stopping, lr_scheduler]

model = Model(inputs, x)
model.compile(loss='categorical_crossentropy', optimizer=optimizers.Adam(learning_rate=0.0001), metrics=['accuracy'])
model.summary()

Model: "functional"

history = model.fit(
    train_generator,
    epochs = 100,
    validation_data = valid_generator,
    verbose = 1,
    callbacks = callbacks,
    shuffle = True
)
```

Hình 3.9 Huấn luyện mô hình InceptionV3

Mô hình được huấn luyện với hai callback chính: EarlyStopping và ReduceLROnPlateau. Trong đó, EarlyStopping theo dõi giá trị val_loss trên tập validation và sẽ dừng sớm quá trình huấn luyện nếu val_loss không giảm đáng kể (ít nhất 0.0001) sau 10 epoch liên tiếp, đồng thời tự động khôi phục lại trọng số của mô hình tại thời điểm tốt nhất. Bên cạnh đó, ReduceLROnPlateau giúp điều chỉnh tốc độ học bằng cách giảm learning rate theo hệ số 0.2 nếu loss trên tập huấn luyện không cải thiện sau 6 epoch, giúp mô hình tối ưu tốt hơn ở giai đoạn sau của quá trình huấn luyện. Mô hình được biên dịch với hàm mất mát categorical_crossentropy, tối ưu bằng thuật toán Adam với learning rate là 0.0001, và sử dụng độ chính xác (accuracy) làm thước đo đánh giá hiệu suất.

```

Epoch 87/100
90/90 137s 1s/step - accuracy: 0.9973 - loss: 0.0242 - val_accuracy: 0.9974 - val_loss: 0.0194 - learning_rate: 2.0000e-05
Epoch 88/100
90/90 136s 1s/step - accuracy: 0.9983 - loss: 0.0222 - val_accuracy: 0.9974 - val_loss: 0.0185 - learning_rate: 2.0000e-05
Epoch 89/100
90/90 136s 1s/step - accuracy: 0.9980 - loss: 0.0222 - val_accuracy: 0.9965 - val_loss: 0.0193 - learning_rate: 2.0000e-05
Epoch 90/100
90/90 136s 1s/step - accuracy: 0.9978 - loss: 0.0212 - val_accuracy: 0.9965 - val_loss: 0.0191 - learning_rate: 2.0000e-05
Epoch 91/100
90/90 136s 1s/step - accuracy: 0.9976 - loss: 0.0215 - val_accuracy: 0.9965 - val_loss: 0.0194 - learning_rate: 2.0000e-05
Epoch 92/100
90/90 136s 1s/step - accuracy: 0.9943 - loss: 0.0286 - val_accuracy: 0.9947 - val_loss: 0.0200 - learning_rate: 2.0000e-05
Epoch 93/100
90/90 137s 1s/step - accuracy: 0.9982 - loss: 0.0193 - val_accuracy: 0.9947 - val_loss: 0.0190 - learning_rate: 2.0000e-05
Epoch 94/100
90/90 136s 1s/step - accuracy: 0.9973 - loss: 0.0236 - val_accuracy: 0.9974 - val_loss: 0.0182 - learning_rate: 2.0000e-05
Epoch 95/100
90/90 136s 1s/step - accuracy: 0.9982 - loss: 0.0193 - val_accuracy: 0.9974 - val_loss: 0.0190 - learning_rate: 2.0000e-05
Epoch 96/100
90/90 136s 1s/step - accuracy: 0.9980 - loss: 0.0199 - val_accuracy: 0.9956 - val_loss: 0.0175 - learning_rate: 2.0000e-05
Epoch 97/100
90/90 136s 1s/step - accuracy: 0.9959 - loss: 0.0228 - val_accuracy: 0.9956 - val_loss: 0.0202 - learning_rate: 2.0000e-05
Epoch 98/100
90/90 136s 1s/step - accuracy: 0.9961 - loss: 0.0242 - val_accuracy: 0.9956 - val_loss: 0.0174 - learning_rate: 2.0000e-05
Epoch 99/100
90/90 136s 1s/step - accuracy: 0.9973 - loss: 0.0207 - val_accuracy: 0.9974 - val_loss: 0.0181 - learning_rate: 2.0000e-05
Epoch 100/100
90/90 136s 1s/step - accuracy: 0.9963 - loss: 0.0276 - val_accuracy: 0.9956 - val_loss: 0.0185 - learning_rate: 2.0000e-05

```

Hình 3.10 Quá trình huấn luyện mô hình InceptionV3 qua các epoch

3.5. Các kết quả thực nghiệm

Trước tiên, để kiểm tra tính ổn định của 2 mô hình được huấn luyện ở trên, tôi xây dựng một đoạn mã có nhiệm vụ trực quan hóa quá trình huấn luyện của mô hình qua từng epoch, bao gồm cả độ chính xác (accuracy) và hàm mất mát (loss) cho cả tập huấn luyện và tập kiểm định (validation).

```

plt.subplot()
plt.title('Model Accuracy')
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Training Accuracy', 'Validation Accuracy'])
plt.savefig('baseline_acc_epoch.png', transparent=False, bbox_inches='tight', dpi=DPI)
plt.show()

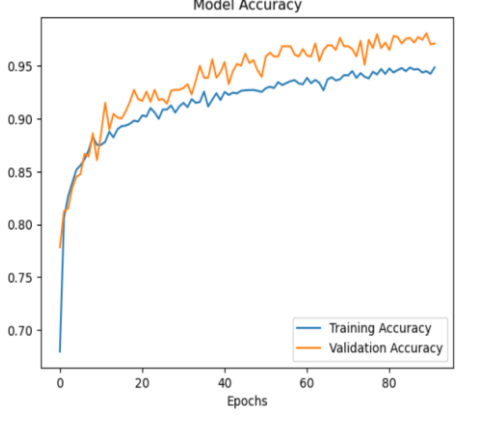
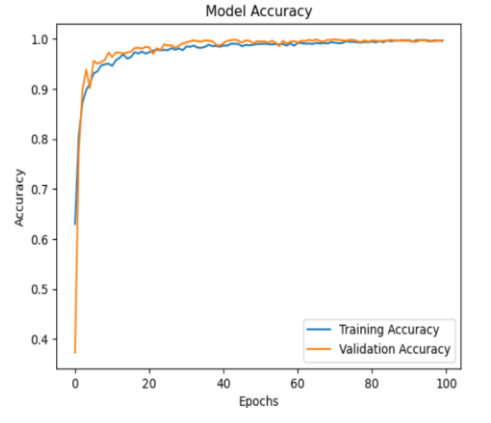
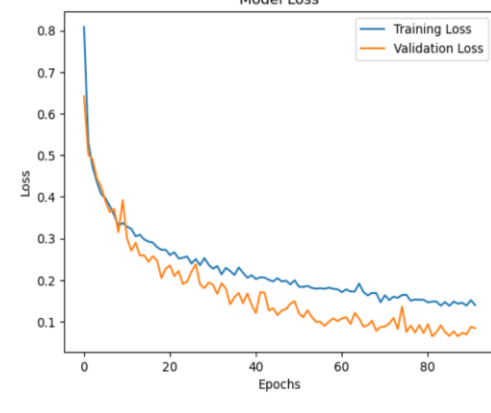
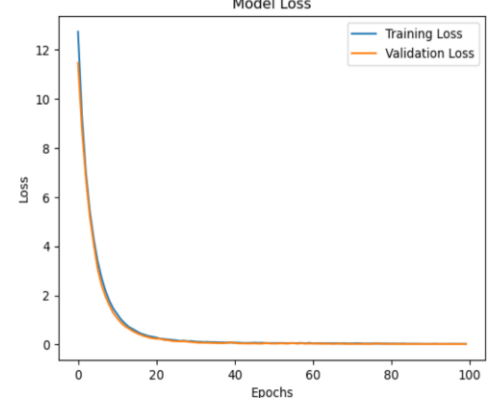
plt.title('Model Loss')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Training Loss', 'Validation Loss'])
plt.savefig('baseline_loss_epoch.png', transparent=False, bbox_inches='tight', dpi=DPI)
plt.show()

```

Hình 3.11 Ảnh code trực quan hóa mô hình

Sau đó tiến hành so sánh các kết quả trực quan của các mô hình để đưa ra đánh giá khách quan:

Bảng 3.1 Mô tả dữ liệu

<div>Mô hình</div> <div>Tiêu chí</div>	VGG16	InceptionV3
Độ chính xác (Accuracy)		
Hàm mất mát (Loss)		

Về độ chính xác:

- VGG16: Độ chính xác huấn luyện và kiểm định tăng đều, đạt khoảng 0.95 sau 80 epochs (dừng sớm do EarlyStopping), cho thấy mô hình học tốt và có sự ổn định giữa tập huấn luyện và kiểm định.
- InceptionV3: Độ chính xác tăng nhanh hơn, đạt gần 1.0 sau 100 epochs, với sự ổn định cao giữa huấn luyện và kiểm định, thể hiện khả năng tổng quát hóa tốt hơn.

Về hàm mất mát:

- VGG16: Loss giảm đều từ 0.8 xuống dưới 0.2 sau 80 epochs, với xu hướng tương đồng giữa huấn luyện và kiểm định, cho thấy không có hiện tượng overfit rõ rệt.
- InceptionV3: Loss giảm mạnh từ 12 xuống gần 0 sau 100 epochs, với sự ổn định tốt, nhưng giá trị ban đầu cao hơn có thể phản ánh dữ liệu hoặc cấu trúc mô hình phức tạp hơn.

Tổng quan, InceptionV3 có vẻ vượt trội hơn VGG16 về độ chính xác tối đa và khả năng học nhanh, trong khi VGG16 cho thấy hiệu suất ổn định với loss thấp hơn ban đầu. Cả hai mô hình đều có xu hướng hội tụ tốt, nhưng InceptionV3 có thể phù hợp hơn với nhiệm vụ yêu cầu độ chính xác cao.

Cuối cùng, tôi thực hiện đánh giá mô hình với bộ dữ liệu test bao gồm 1311 ảnh chưa học thuộc 4 nhóm mà mô hình đã học để kiểm tra hiệu suất. Ảnh của tập dữ liệu này cũng cần qua xử lý để phù hợp với mô hình đã huấn luyện

```
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    TEST_PATH,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

Hình 3.12 Ảnh code xử lý dữ liệu test để phù hợp với mô hình

Tiếp đến là tiến hành dự đoán trên cả hai mô hình và so sánh dựa trên 2 tiêu chí: độ chính xác và giá trị của hàm mất mát, ta được bảng so sánh sau:

Bảng 3.3 So sánh kết quả trên tập test

Mô hình Tiêu chí	VGG16	InceptionV3
Độ chính xác (Accuracy)	0.96	0.99
Giá trị của hàm mất mát	0.11	0.06

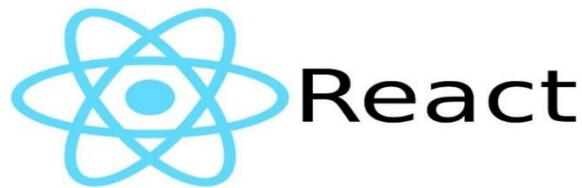
InceptionV3 vượt trội hơn VGG16 với độ chính xác cao hơn và loss thấp hơn trên tập test, cho thấy nó hiệu quả hơn trong việc xử lý dữ liệu mới. Cả hai mô hình đều cho kết quả khả quan, nhưng InceptionV3 có ưu thế rõ rệt. Vì vậy, tôi sử dụng InceptionV3 để ứng dụng vào sản phẩm demo ở chương tiếp theo.

CHƯƠNG 4: XÂY DỰNG SẢN PHẨM DEMO

4.1. Giới thiệu các framework sử dụng

4.1.1. ReactJS

ReactJS (hay còn gọi là React) là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, ra mắt lần đầu vào năm 2013. React được thiết kế nhằm hỗ trợ xây dựng giao diện người dùng (UI) cho các ứng dụng web một cách hiệu quả, linh hoạt và dễ bảo trì.



Hình 4.1 Giới thiệu framework ReactJS

Điểm nổi bật của React là kiến trúc component-based – cho phép chia giao diện thành các thành phần nhỏ, độc lập và có thể tái sử dụng. Điều này giúp việc phát triển, kiểm thử và bảo trì trở nên dễ dàng hơn, đặc biệt với các ứng dụng có giao diện phức tạp. React sử dụng Virtual DOM – một mô hình DOM ảo để tăng hiệu suất. Khi có thay đổi trên giao diện, React sẽ so sánh giữa DOM ảo và DOM thật, sau đó chỉ cập nhật những phần thực sự thay đổi, giúp ứng dụng chạy nhanh và mượt mà hơn. Hiện nay, React đang là một trong những công nghệ phổ biến nhất trong lĩnh vực phát triển web front-end. Với cộng đồng lớn mạnh, tài liệu đầy đủ, cùng hệ sinh thái phong phú, React không chỉ được sử dụng rộng rãi trong các dự án cá nhân mà còn trong các hệ thống quy mô lớn của các công ty hàng đầu.

Với những đặc tính nêu trên, tôi đã chọn ReactJS cho để xây dựng phía giao diện của sản phẩm demo với mục tiêu là thân thiện, tinh tế và dễ sử dụng với người dùng. Hơn thế, tôi hy vọng ứng dụng web này có tiềm năng phát triển thành một ứng dụng thương mại trong tương lai.

4.1.2. FastAPI

FastAPI là một framework backend hiện đại và mạnh mẽ dùng cho việc xây dựng các API web, được phát triển bằng ngôn ngữ Python. Ra mắt lần đầu vào năm 2018, FastAPI nhanh chóng trở nên phổ biến nhờ vào tốc độ xử lý cao, cú pháp rõ ràng và khả năng tự động sinh tài liệu API.



Hình 4.2 Giới thiệu framework FastAPI

Điểm nổi bật của FastAPI là khả năng tận dụng kiểu dữ liệu tĩnh (type hints) của Python để tự động kiểm tra đầu vào/đầu ra, giúp giảm lỗi và tăng độ tin cậy trong quá trình phát triển. FastAPI còn hỗ trợ sẵn chuẩn OpenAPI và JSON Schema, từ đó tự động tạo ra giao diện tài liệu tương tác cho API (Swagger UI và ReDoc), rất thuận tiện cho việc kiểm thử và tích hợp hệ thống.

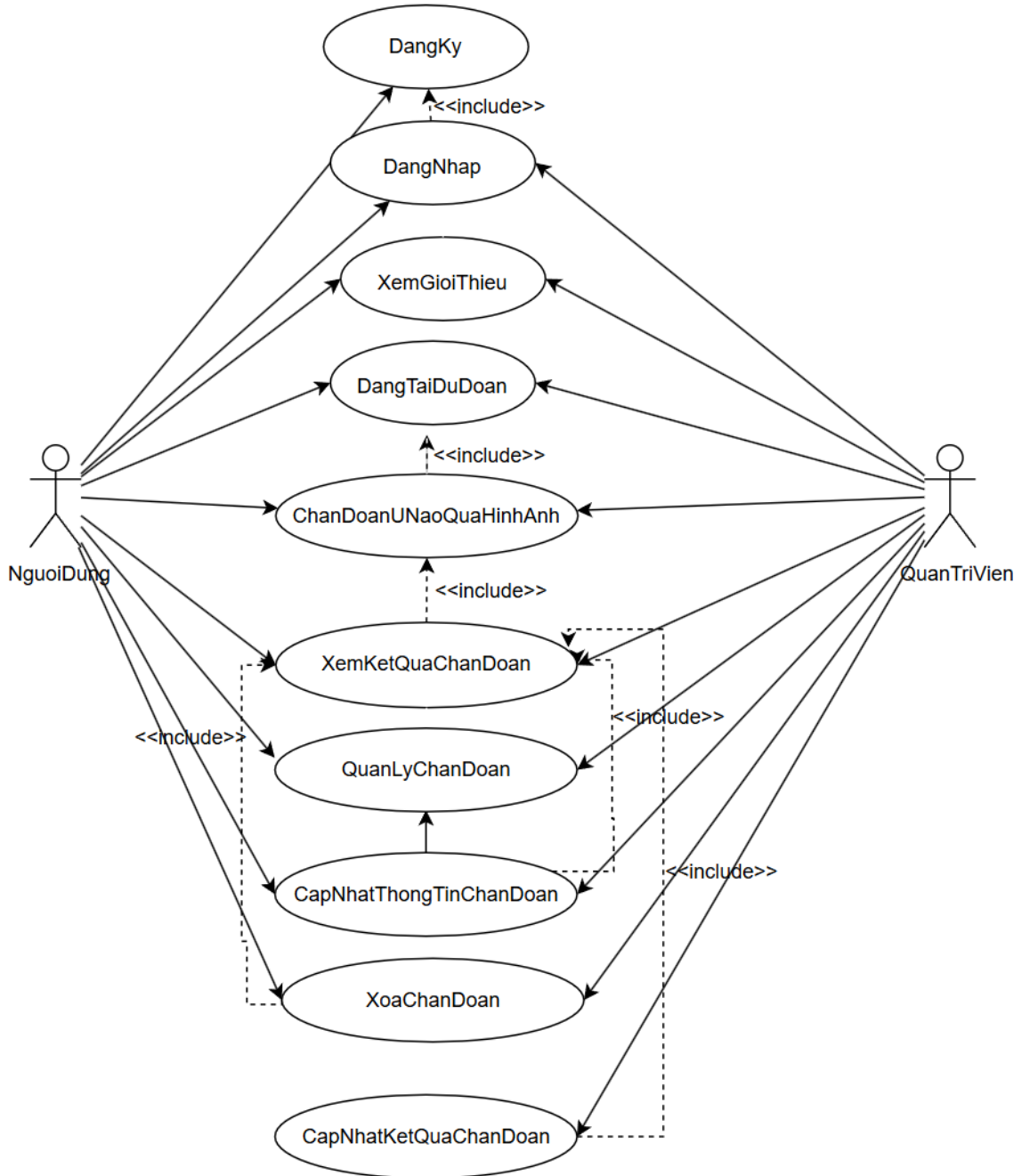
Về hiệu suất, FastAPI được xây dựng trên nền ASGI (Asynchronous Server Gateway Interface) và sử dụng Starlette làm web framework nền tảng, giúp xử lý bất đồng bộ (async/await) một cách hiệu quả. Điều này giúp FastAPI đạt hiệu suất rất cao, tiệm cận với các framework nổi tiếng như NodeJS, Go hoặc Java Spring trong nhiều bài benchmark. Với cú pháp đơn giản, tốc độ cao và tài liệu rõ ràng, FastAPI là lựa chọn lý tưởng để phát triển các hệ thống RESTful API hiện đại, microservices, hoặc backend cho các ứng dụng sử dụng trí tuệ nhân tạo và machine learning.

Tóm lại, sự kết hợp giữa ReactJS ở phía giao diện người dùng (frontend) và FastAPI ở phía máy chủ (backend) mang lại một kiến trúc hiện đại, hiệu quả và dễ mở rộng. ReactJS đảm nhiệm việc hiển thị và tương tác với người dùng một cách linh hoạt, trong khi FastAPI xử lý dữ liệu nhanh chóng và tối ưu ở tầng API.

Cả hai công nghệ đều có cộng đồng phát triển mạnh mẽ, tài liệu rõ ràng và được ứng dụng rộng rãi trong thực tế, giúp tăng tốc quá trình phát triển cũng như đảm bảo chất lượng hệ thống trong dài hạn.

4.2. Phân tích thiết kế hệ thống

4.2.1. Biểu đồ use case tổng quát



Hình 4.3 Biểu đồ use case tổng quát

4.2.2. Mô tả chi tiết use case

4.2.2.1. Mô tả chi tiết use case Đăng tải dự đoán

Mã use case	UC1
Tên use case	Đăng tải dự đoán
Tóm tắt	Người dùng tải lên hình ảnh quét não và nhập thông tin bệnh nhân để hệ thống dự đoán u não
Actor	Người dùng
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống, có quyền truy cập chức năng này và đã điền đầy đủ thông tin cơ bản
Đảm bảo tối thiểu	Hệ thống hiển thị thông báo lỗi nếu tải lên thất bại
Đảm bảo thành công	Hình ảnh và thông tin bệnh nhân được tải lên thành công và hệ thống bắt đầu quá trình dự đoán
Kích hoạt	Người dùng nhấp vào "Chẩn đoán" trên giao diện
Luồng sự kiện: <ol style="list-style-type: none"> 1. Người dùng nhấp vào "Chẩn đoán". 2. Hệ thống hiển thị form với các trường: tên bệnh nhân, tuổi, giới tính, ảnh MRI. 3. Người dùng nhập thông tin (tên, tuổi, giới tính) và chọn file ảnh MRI. 4. Người dùng nhấn nút "Tải lên hình ảnh". 5. Hệ thống kiểm tra: tên không rỗng, tuổi là số dương, giới tính được chọn, file là ảnh. 6. Nếu hợp lệ, hình ảnh và thông tin được gửi đến server để xử lý dự đoán. 	
Ngoại lệ: <ul style="list-style-type: none"> - Tên bệnh nhân rỗng: Hiển thị "Vui lòng nhập tên bệnh nhân". 	

<ul style="list-style-type: none"> - Tuổi không hợp lệ (âm hoặc không phải số): Hiển thị "Tuổi không hợp lệ". - Giới tính không chọn: Hiển thị "Vui lòng chọn giới tính". - Mất kết nối internet: Hiển thị "Kết nối thất bại, vui lòng thử lại". - Server không phản hồi: Hiển thị "Lỗi hệ thống, vui lòng thử lại sau".
Hậu điều kiện: Không có

4.2.2.2. Mô tả use case Quản lý chẩn đoán

Mã use case	UC2
Tên use case	Quản lý chẩn đoán
Tóm tắt	Người dùng hoặc quản trị viên xem và quản lý các chẩn đoán u não, với quyền truy cập khác nhau: quản trị viên xem tất cả chẩn đoán, người dùng chỉ xem chẩn đoán của mình đã tải lên
Actor	Người dùng, quản trị viên
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng hoặc quản trị viên đã đăng nhập vào hệ thống - Có ít nhất một chẩn đoán trong hệ thống (đối với quản trị viên) hoặc chẩn đoán do người dùng tạo (đối với người dùng)
Đảm bảo tối thiểu	Hệ thống hiển thị thông báo lỗi nếu không truy cập được dữ liệu chẩn đoán
Đảm bảo thành công	Người dùng hoặc quản trị viên xem được danh sách chẩn đoán phù hợp với quyền của mình (quản trị viên thấy tất cả, người dùng chỉ thấy của mình)
Kích hoạt	Người dùng hoặc quản trị viên nhấp vào "Quản lý chẩn đoán" trên giao diện
Luồng sự kiện: <ol style="list-style-type: none"> 1. Người dùng hoặc quản trị viên nhấp vào "Quản lý chẩn đoán". 	

<ol style="list-style-type: none"> 2. Hệ thống kiểm tra quyền truy cập của tài khoản (quản trị viên hay người dùng). 3. Nếu là quản trị viên: Hệ thống hiển thị danh sách tất cả các chẩn đoán (bao gồm thông tin: tên bệnh nhân, tuổi, giới tính, ảnh, kết quả dự đoán). 4. Nếu là người dùng: Hệ thống chỉ hiển thị danh sách các chẩn đoán do người dùng đó tạo. 5. Người dùng hoặc quản trị viên có thể thực hiện các hành động: xem chi tiết, cập nhật, hoặc xóa chẩn đoán (nếu được phép). 6. Hệ thống phản hồi dựa trên hành động (ví dụ: hiển thị chi tiết, lưu cập nhật, hoặc xóa dự đoán).
<p>Ngoại lệ:</p> <ul style="list-style-type: none"> - Mất kết nối internet: Hiển thị thông báo "Kết nối thất bại, vui lòng thử lại". - Server không phản hồi: Hiển thị thông báo "Lỗi hệ thống, vui lòng thử lại sau". - Quyền truy cập không hợp lệ (token hết hạn): Hiển thị thông báo "Phiên đăng nhập hết hạn, vui lòng đăng nhập lại".
<p>Hậu điều kiện: Không có</p>

4.2.2.3. Mô tả chi tiết use case Xem kết quả chẩn đoán

Mã use case	UC3
Tên use case	Xem kết quả chẩn đoán
Tóm tắt	Người dùng hoặc quản trị viên xem kết quả chẩn đoán u não, với quyền truy cập khác nhau: quản trị viên xem tất cả chẩn đoán, người dùng chỉ xem dự đoán của mình.
Actor	Người dùng, quản trị viên

Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng hoặc quản trị viên đã đăng nhập vào hệ thống - Có ít nhất một chẩn đoán trong hệ thống (đối với quản trị viên) hoặc chẩn đoán do người dùng tạo (đối với người dùng)
Đảm bảo tối thiểu	Hệ thống hiển thị thông báo lỗi nếu không tải được kết quả
Đảm bảo thành công	Người dùng hoặc quản trị viên xem được thông tin chi tiết của kết quả chẩn đoán phù hợp với quyền của mình
Kích hoạt	Người dùng hoặc quản trị viên nhấp vào "Xem kết quả" trên một dự đoán từ danh sách
Luồng sự kiện: <ol style="list-style-type: none"> 1. Người dùng hoặc quản trị viên nhấp vào "Xem kết quả" trên một chẩn đoán 2. Hệ thống hiển thị popup với thông tin: ID bệnh nhân, tên bệnh nhân, tuổi, giới tính, ngày tải, hình ảnh MRI, và kết quả chẩn đoán (ví dụ: "Không có u"). 3. Người dùng hoặc quản trị viên nhấp "Đóng" để thoát popup. 	
Ngoại lệ: <ul style="list-style-type: none"> - Mất kết nối internet: Hiển thị thông báo "Kết nối thất bại, vui lòng thử lại". - Server không phản hồi: Hiển thị thông báo "Lỗi hệ thống, vui lòng thử lại sau". - Quyền truy cập không hợp lệ (token hết hạn): Hiển thị thông báo "Phiên đăng nhập hết hạn, vui lòng đăng nhập lại". 	
Hậu điều kiện: Không có	

4.2.2.4. Mô tả chi tiết use case Cập nhật kết quả chẩn đoán

Mã use case	UC4
Tên use case	Cập nhật kết quả chẩn đoán

Tóm tắt	Quản trị viên chỉnh sửa thông tin hoặc kết quả của một dự đoán u não đã có trong hệ thống.
Actor	Quản trị viên
Tiền điều kiện	<ul style="list-style-type: none"> - Quản trị viên đã đăng nhập vào hệ thống với quyền quản trị. - Có ít nhất một chẩn đoán tồn tại trong hệ thống. - Chẩn đoán đã được xử lý và có kết quả ban đầu.
Đảm bảo tối thiểu	Hệ thống hiển thị thông báo lỗi nếu cập nhật thất bại
Đảm bảo thành công	Thông tin chẩn đoán được cập nhật thành công trong cơ sở dữ liệu.
Kích hoạt	Quản trị viên nhấp vào "Cập nhật" trên giao diện trên một chẩn đoán từ danh sách
Luồng sự kiện: <ol style="list-style-type: none"> 1. Quản trị viên nhấp vào "Cập nhật" trên một chẩn đoán (ví dụ: ID 6). 2. Hệ thống kiểm tra quyền truy cập (xác nhận tài khoản là quản trị viên). 3. Hệ thống hiển thị form chỉnh sửa với các trường: tên bệnh nhân, tuổi, giới tính, kết quả chẩn đoán (ví dụ: "Không có u", "U màng não",...). 4. Quản trị viên chỉnh sửa thông tin (ví dụ: thay đổi kết quả từ "Không có u" thành "U màng não"). 5. Quản trị viên nhấn nút "Lưu". 6. Hệ thống kiểm tra thông tin (tên không rỗng, tuổi là số dương, giới tính hợp lệ, kết quả hợp lệ). 7. Nếu hợp lệ, hệ thống gửi request cập nhật đến server (PUT /predictions/{id}). 	

<p>8. Hệ thống hiển thị thông báo "Cập nhật thành công".</p> <p>9. Nếu không hợp lệ, hiển thị thông báo lỗi (ví dụ: "Tuổi không hợp lệ").</p>
<p>Ngoại lệ:</p> <ul style="list-style-type: none"> - Quản trị viên không có quyền (token không hợp lệ): Hiển thị thông báo "Bạn không có quyền thực hiện hành động này". - Chẩn đoán không tồn tại: Hiển thị thông báo "Dự đoán không tìm thấy". - Thông tin không hợp lệ (tuổi âm, tên rỗng): Hiển thị thông báo "Thông tin không hợp lệ". - Mất kết nối internet: Hiển thị thông báo "Kết nối thất bại, vui lòng thử lại". - Server không phản hồi: Hiển thị thông báo "Lỗi hệ thống, vui lòng thử lại sau".
<p>Hậu điều kiện: Không có</p>

4.3. Giao diện hệ thống

Giao diện được thiết kế với tông màu sáng, thao tác đơn giản và dễ sử dụng. Hiện tại hệ thống chỉ hỗ trợ ngôn ngữ Tiếng Việt với mục đích chính hướng tới người sử dụng trong nước. Chi tiết giao diện được trình bày qua các hình bên dưới.

The screenshot shows the login interface of the BrainTumorApp. The header is blue with the app name 'BrainTumorApp' and the page title 'Trang chủ' on the left, and buttons for 'Đăng nhập' and 'Đăng ký' on the right. A left sidebar contains the text 'Trang chủ'. The main content area features a white box titled 'Đăng nhập' with two input fields for 'Tên tài khoản' and 'Mật khẩu', a blue 'Đăng nhập' button, and a link 'Chưa có tài khoản? [Đăng ký](#)'.

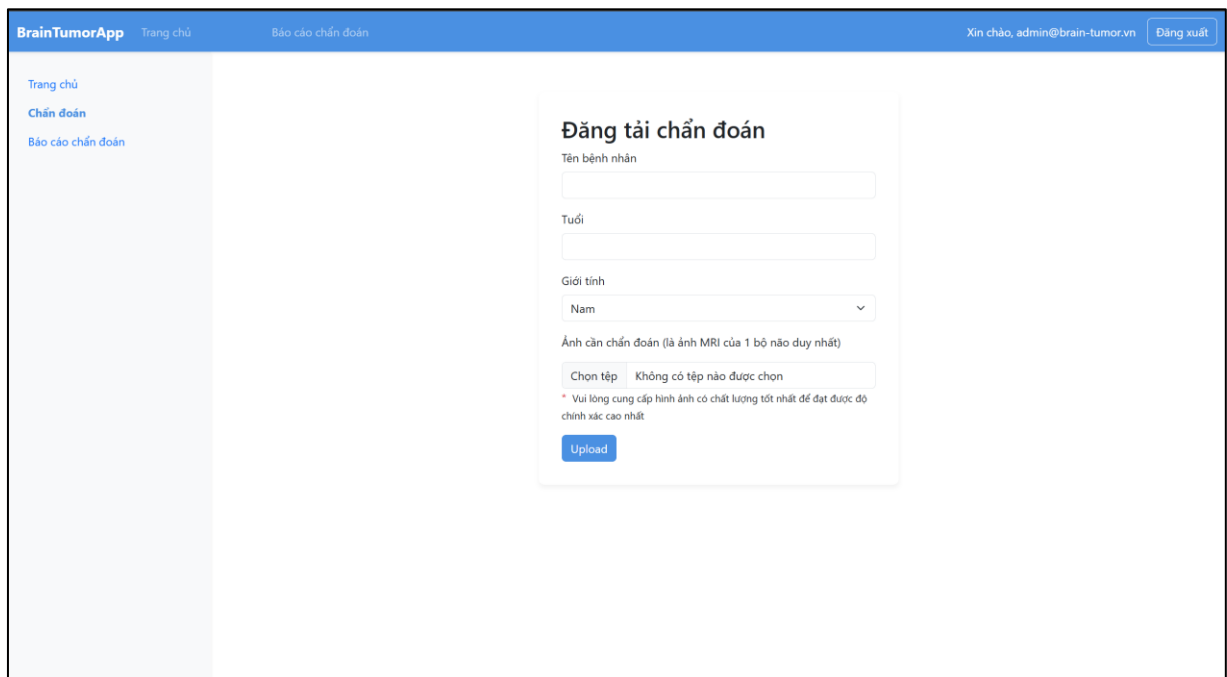
Hình 4.4 Giao diện màn hình đăng nhập

The screenshot shows the registration interface of the BrainTumorApp. The header is blue with the app name 'BrainTumorApp' and the page title 'Trang chủ' on the left, and buttons for 'Đăng nhập' and 'Đăng ký' on the right. A left sidebar contains the text 'Trang chủ'. The main content area features a white box titled 'Đăng ký' with three input fields for 'Tên tài khoản', 'Mật khẩu', and 'Xác nhận mật khẩu', a blue 'Đăng ký tài khoản' button, and a link 'Đã có tài khoản? [Đăng nhập tại đây](#)'.

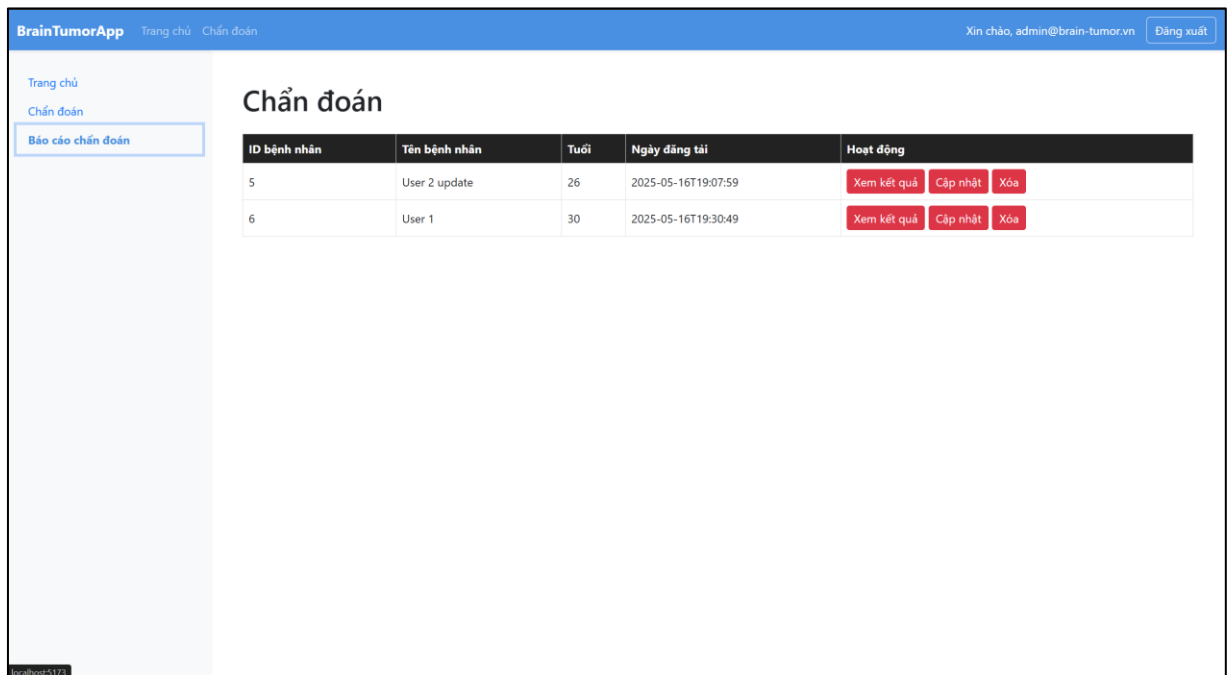
Hình 4.5 Giao diện màn hình đăng ký



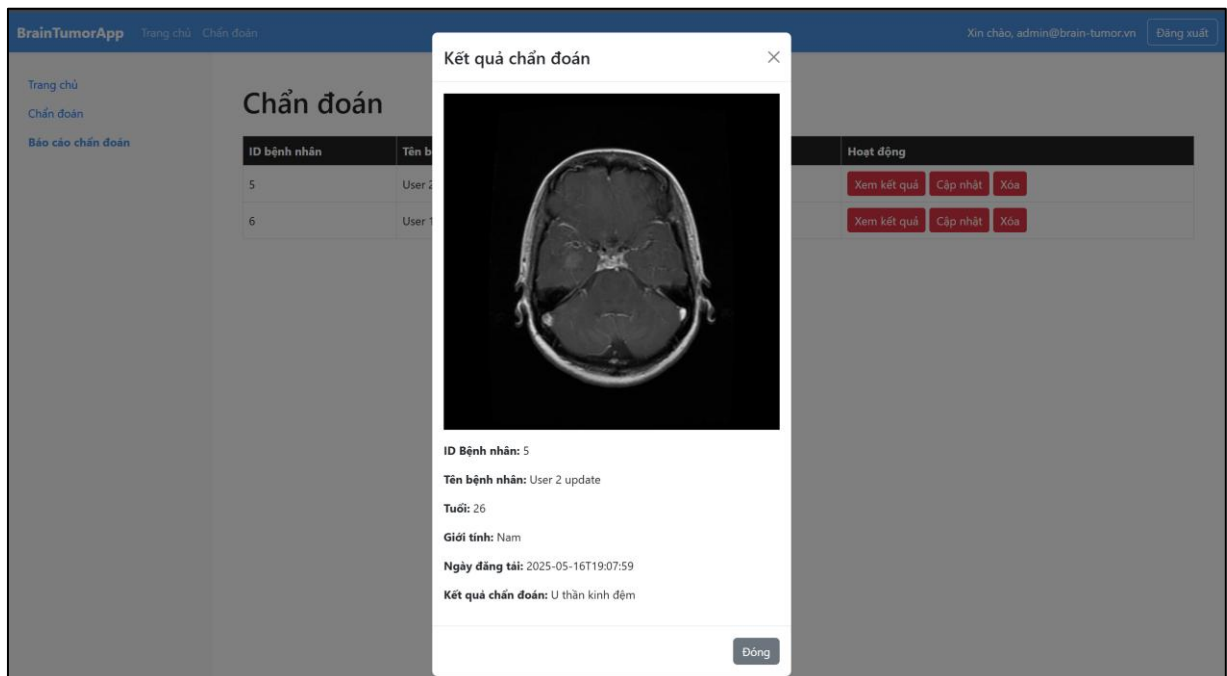
Hình 4.6 Giao diện trang chủ



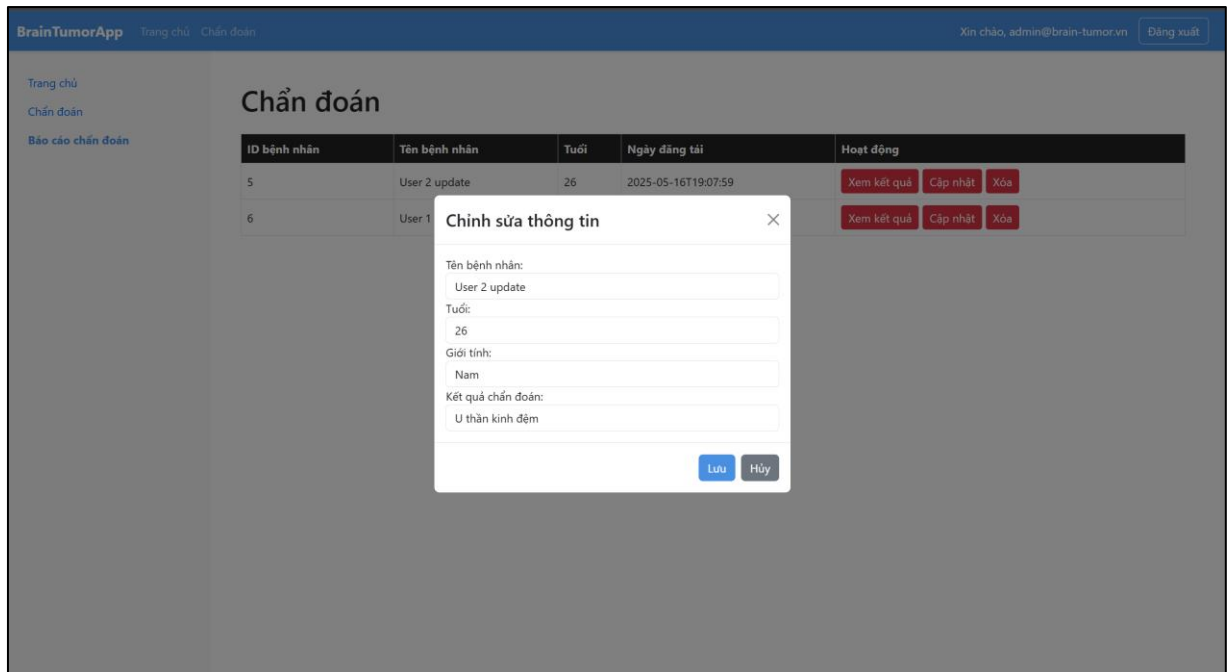
Hình 4.7 Giao diện đăng tải chẩn đoán



Hình 4.8 Giao diện báo cáo chẩn đoán



Hình 4.9 Giao diện xem kết quả chẩn đoán



Hình 4.10 Giao diện cập nhật kết quả chẩn đoán (Chỉ với Quản trị viên)

4.4. Chức năng chính của hệ thống

Hệ thống có chức năng chính là chẩn đoán u não qua dữ liệu hình ảnh MRI nhận được từ người dùng. FastAPI sử dụng thư viện như Pydantic để validate dữ liệu đầu vào và Python-multipart để xử lý file upload. Hình ảnh được kiểm tra định dạng (chỉ chấp nhận .jpg, .png, hoặc MRI) và chất lượng trước khi gửi đến mô hình.

```
# Kiểm tra định dạng file
if not image.filename.lower().endswith((".png", ".jpg", ".jpeg")):
    raise HTTPException(status_code=400, detail="Only PNG, JPG, or JPEG files are allowed")

# Tạo tên file duy nhất để tránh trùng lặp
file_extension = os.path.splitext(image.filename)[1] # Lấy đuôi file (.jpg, .png, v.v.)
unique_filename = f"{uuid.uuid4()}{file_extension}" # Tạo tên file duy nhất bằng UUID
image_path = UPLOAD_DIR / unique_filename # Đường dẫn đầy đủ: uploads/<uuid>.jpg

# Lưu file
try:
    with open(image_path, "wb") as f:
        content = await image.read() # Đọc file
        f.write(content) # Ghi file
except Exception as e:
    raise HTTPException(status_code=500, detail=f"Error saving file: {str(e)}")

# Đặt lại con trỏ file (nếu cần dùng lại image)
# Vì process_and_predict cần file, ta sẽ truyền đường dẫn thay vì đọc lại
image.file.seek(0)
```

Hình 4.11 Ảnh code xử lý ảnh MRI gửi lên từ người dùng

Sau khi nhận dữ liệu, tiến hành tiền xử lý ảnh đầu vào. Tiếp đến, FastAPI gọi mô hình học máy (ví dụ: tích hợp TensorFlow hoặc PyTorch) để phân tích hình ảnh. Kết quả dự đoán (có u hoặc không có u) được tạo ra cùng với xác suất, sau đó được lưu tạm thời trong cơ sở dữ liệu (sử dụng SQLAlchemy với MySQL).

```

try:
    result = process_and_predict(image)
except Exception as e:
    # Xóa file nếu có lỗi
    if image_path.exists():
        image_path.unlink()
    raise HTTPException(status_code=500, detail=f"Error processing image: {str(e)}")

# Save prediction to database
try:
    db_prediction = Prediction(
        patient_name=patient_name,
        age=age,
        gender=gender,
        image_path=str(unique_filename), # Lưu đường dẫn dưới dạng string
        prediction_result=result,
        user_id=current_user.id
    )
    db.add(db_prediction)
    db.commit()
    db.refresh(db_prediction)
except Exception as e:
    # Xóa file nếu có lỗi khi lưu vào database
    if image_path.exists():
        image_path.unlink()
    raise HTTPException(status_code=500, detail=f"Error saving prediction to database: {str(e)}")

return db_prediction

```

Hình 4.12 Ảnh code dự đoán kết quả và lưu vào cơ sở dữ liệu

Cuối cùng là cho phép cập nhật lại thông tin dự đoán (tên, tuổi, giới tính) và chỉ cho phép Quản trị viên cập nhật được kết quả dự đoán nhằm mục đích xây dựng một bộ dữ liệu sạch, có tính ứng dụng để phát triển mô hình hoặc xây dựng các mô hình tối ưu hơn.

KẾT LUẬN

Trong thời gian thực hiện đề tài "Kỹ thuật phân loại khối u não dựa trên hình ảnh bằng InceptionV3", em đã có cơ hội nghiên cứu và áp dụng các kiến thức về học sâu vào giải quyết bài toán thực tế. Trong suốt quá trình triển khai, em đã hiểu sâu hơn về cách thiết kế, huấn luyện, và tối ưu hóa mô hình mạng nơ ron tích chập (CNN), đặc biệt là kiến trúc VGG16, InceptionV3.

Em đã ứng dụng các kỹ thuật tiên tiến như transfer learning và fine-tuning để cải thiện hiệu suất của mô hình. Nhờ đó, mô hình đạt được độ chính xác cao trên tập dữ liệu kiểm thử (gần 99%), chứng minh tiềm năng áp dụng của mạng nơ ron tích chập trong việc phân loại hình ảnh phức tạp. Bên cạnh đó, việc thực hiện các bước tiền xử lý dữ liệu kỹ lưỡng và sử dụng các phương pháp tăng cường dữ liệu cũng giúp cải thiện đáng kể khả năng tổng quát hóa của mô hình.

Qua quá trình làm việc, em không chỉ học hỏi được nhiều kiến thức chuyên môn về trí tuệ nhân tạo mà còn phát triển các kỹ năng làm việc nhóm, lập trình, và phân tích vấn đề. Đồ án này đã mang lại cho em nền tảng vững chắc để nghiên cứu sâu hơn trong lĩnh vực học sâu và ứng dụng của nó vào các bài toán thực tiễn.

Tuy nhiên, do thời gian và tài nguyên còn hạn chế, mô hình vẫn tồn tại một số hạn chế khi phân loại đối với ảnh chất lượng thấp. Trong tương lai, em dự định mở rộng quy mô dữ liệu huấn luyện và cải thiện các tham số mô hình để tăng hiệu quả phân loại.

TÀI LIỆU THAM KHẢO

- [1] Le Thi, H.A., Nguyen, M.C., 2016, Efficient Algorithms for Feature Selection in Multi-class Support Vector Machine, *Annals of Operations Research*.
- [2] Nguyễn Đức Tùng (2019). Nhập môn xử lý ảnh, Nhà xuất bản Bách Khoa.
- [3] Nguyễn Xuân Huy (2018). Trí tuệ nhân tạo và ứng dụng. Nhà xuất bản Khoa học và Kỹ thuật.
- [4] Trần Duy Thanh & Nguyễn Thị Lan (2020). Cơ sở học sâu và ứng dụng trong xử lý ảnh. Nhà xuất bản Đại học Quốc gia TP. Hồ Chí Minh.
- [5] Phạm Văn Tấn & Nguyễn Văn Hiếu (2021). Mạng nơ-ron tích chập và ứng dụng trong nhận diện đối tượng. Nhà xuất bản Giao thông Vận tải.
- [6] EHealth Vietnam Summit. (n.d.). Lợi ích của ứng dụng công nghệ thông tin trong chăm sóc sức khỏe
- [7] Bệnh viện Đa khoa Tâm Anh. (n.d.). U não: Phân loại, triệu chứng, nguyên nhân và cách điều trị
- [8] Sannakki SS, Rajpurohit VS, Nargund VB, Kulkarni P. 2013. Diagnosis and Classification of Grape Leaf Diseases using Neural Networks, In proceeding of 4th International Conference (ICCCNT), IEEE, Tiruchengode, 1-5. DOI: 10.1109/ICCCNT.2013.6726616.
- [9] Ciresan, D. C., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for traffic sign classification. *Neural Networks*.
- [10] Yann LeCun, Léon Bottou, Yoshua Bengio, & Patrick Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.
- [11] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR 2015 Conference Papers*. <https://arxiv.org/abs/1409.1556> (Lần truy cập cuối 05/10/2024)

- [12] Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. Foundations and Trends® in Signal Processing.
- [13] Deep neural networks for traffic sign classification. *Neural* Cirezan, D. C., Meier, U., & Schmidhuber, J. (2012). Multi-column *Networks*.
- [14] Larsson, F., & Felsberg, M. (2011). Using Fourier descriptors and spatial models for traffic sign recognition. *International Conference on Computer Vision Systems*.