

BỘ CÔNG THƯƠNG  
TRƯỜNG ĐẠI HỌC KINH TẾ - KỸ THUẬT CÔNG NGHIỆP

**KHÓA LUẬN TỐT NGHIỆP**

TÊN ĐỀ TÀI KHÓA LUẬN:  
**NGHIÊN CỨU MÔ HÌNH MẠNG MASK R-CNN, ỨNG  
DỤNG ĐỒ TÌM ĐỐI TƯỢNG TRONG ẢNH**

Ngành đào tạo : Công nghệ thông tin

Mã số ngành : 7480201

Họ và tên sinh viên : Nguyễn Đình Hoàng  
Châu Hoàng Phong

Người hướng dẫn khóa luận tốt nghiệp  
ThS. Lương Thị Thảo Hiếu

Hà Nội - 2024

**BỘ CÔNG THƯƠNG**  
**TRƯỜNG ĐẠI HỌC KINH TẾ - KỸ THUẬT CÔNG NGHIỆP**

**KHÓA LUẬN TỐT NGHIỆP**

**TÊN ĐỀ TÀI KHÓA LUẬN:**  
**NGHIÊN CỨU MÔ HÌNH MẠNG MASK R-CNN, ỨNG DỤNG ĐỒ TÌM ĐỐI TƯỢNG TRONG ẢNH**

**Ngành đào tạo : Công nghệ thông tin**  
**Mã số ngành : 7480201**

**Họ và tên sinh viên : Nguyễn Đình Hoàng**  
**Châu Hoàng Phong**

**Người hướng dẫn khóa luận tốt nghiệp**  
**ThS. Lương Thị Thảo Hiếu**

**Hà Nội - 2024**

## LỜI CAM ĐOAN

Chúng tôi xin cam đoan đây là kết quả nghiên cứu của nhóm, không sao chép ở bất kì một công trình hoặc một khóa luận, luận văn, luận án của tác giả khác. Các số liệu, kết quả nêu trong khóa luận là hoàn toàn trung thực và chưa được công bố trong bất kỳ một công trình nghiên cứu, khóa luận nào khác. Các trích dẫn, các số liệu và kết quả tham khảo đều có nguồn trích dẫn rõ ràng.

Nhóm thực hiện  
Nguyễn Đình Hoàng  
Châu Hoàng Phong

## LỜI CẢM ƠN

Để hoàn thành khóa luận tốt nghiệp, chúng tôi xin bày tỏ lòng biết ơn sâu sắc đến cô Ths. Lương Thị Thảo Hiếu – người đã đồng hành và hướng dẫn chúng tôi trong quá trình nghiên cứu và hoàn thiện khóa luận tốt nghiệp này. Sự hỗ trợ và hướng dẫn của cô là một phần không thể thiếu trong thành công khóa luận.

Chúng tôi xin được gửi lời cảm ơn và bày tỏ lòng biết ơn chân thành đến tất cả các thầy, cô Trường Đại học Kinh tế - Kỹ thuật Công nghiệp đã giúp đỡ, truyền dạy các kiến thức cho chúng tôi trong suốt thời gian học tập tại trường.

Và chúng tôi không quên gửi lời cảm ơn để bố mẹ và tất cả người thân trong gia đình đã luôn ở bên động viên, chia sẻ những khó khăn cũng như niềm vui, đặc biệt là luôn tạo điều kiện tốt nhất để chúng con hoàn thành khóa học này.

Mặc dù trong quá thực hiện khóa luận chúng tôi đã có sự vận dụng những kiến thức được học nhưng việc nghiên cứu và viết bài vẫn gặp phải những khó khăn và thiếu sót khi làm bài. Chúng tôi rất mong nhận được ý kiến đóng góp và sự chỉ dẫn của thầy cô để khóa luận được hoàn thiện hơn.

Chúng tôi xin chân thành cảm ơn!

## MỤC LỤC

LỜI CAM ĐOAN .....	ii
LỜI CẢM ƠN .....	iii
MỤC LỤC.....	iv
DANH MỤC TỪ VIẾT TẮT .....	vi
DANH MỤC BẢNG BIỂU .....	viii
MỤC LỤC HÌNH ẢNH .....	ix
MỞ ĐẦU .....	1
1. Lý do chọn đề tài .....	1
2. Mục tiêu đề tài.....	1
3. Đối tượng và phạm vi nghiên cứu.....	2
4. Phương pháp nghiên cứu.....	2
4.1. Cách thức nghiên cứu.....	2
4.2. Xây dựng môi trường .....	2
CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ AI.....	3
1.1. TỔNG QUAN VỀ AI.....	3
1.2. MACHINE LEARNING.....	4
1.2.1. Cách thức hoạt động.....	4
1.2.2. Phân loại Machine Learning .....	7
1.3. DEEP LEARNING .....	9
1.3.1. Khái Niệm .....	9
1.3.2. Cách thức hoạt động.....	10
1.4. MÔI TRƯỜNG FRAMEWORK.....	11
1.5. KẾT LUẬN CHƯƠNG 1.....	13
CHƯƠNG 2: MẠNG MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK (MASK R-CNN).....	15
2.1. CÁC MÔ HÌNH MẠNG HUẤN LUYỆN PHỔ BIẾN .....	15
2.1.1. Mạng huấn luyện LeNet.....	15
2.1.2. Mạng huấn luyện AlexNet .....	16
2.1.3. Mạng huấn luyện VGGNet .....	17

2.1.4. Mạng huấn luyện Inception.....	18
2.1.5. Mạng huấn luyện ResNet .....	20
2.1.6. Mạng huấn luyện DenseNet.....	21
2.1.7. Mạng huấn luyện MobileNet .....	23
2.1.8. Mạng huấn luyện EfficientNet .....	24
2.2. GIỚI THIỆU VỀ MẠNG MASK R-CNN.....	26
2.2.1. Giới thiệu chung về Mask R-CNN.....	26
2.2.2. Đặc trưng của mạng Mask R-CNN.....	27
2.3. KIẾN TRÚC MẠNG MASK R-CNN .....	28
2.3.1. Backbone .....	29
2.3.2. RPN .....	30
2.3.3. RoIAlign.....	31
2.3.4. Nhánh phát hiện đối tượng (Object detection branch).....	32
2.3.5. Nhánh tạo mặt nạ (Mask generation branch):.....	33
2.4. KẾT LUẬN CHƯƠNG 2.....	34
CHƯƠNG 3: ỨNG DỤNG MÔ HÌNH MASK R-CNN DÒ TÌM ĐỐI TƯỢNG TRONG ẢNH .....	36
3.1. MÔI TRƯỜNG LÀM VIỆC .....	36
3.1.1. Giới thiệu chung về Anaconda .....	36
3.1.2. Giới thiệu chung về Spyder.....	37
3.2. CHUẨN BỊ DỮ LIỆU (DATASET) .....	38
3.3. MODEL TRAINNING.....	41
3.4. KẾT LUẬN CHƯƠNG 3.....	49
KẾT LUẬN VÀ ĐÁNH GIÁ KHÓA LUẬN.....	50
TÀI LIỆU THAM KHẢO.....	51

## DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Tên đầy đủ	Ý nghĩa
1	Mask R-CNN	Mask Region-based Convolutional Neural Network	Mạng Nơ-ron chuyển đổi dựa trên vùng mặt nạ
2	AI	Artificial Intelligence	Trí tuệ nhân tạo
3	ML	Machine Learning	Máy học, là một nhánh của trí tuệ nhân tạo
4	SVM	Support Vector Machine	Máy vector hỗ trợ
5	PCA	Principal Component Analysis	Phân tích thành phần chính
6	DNN	Deep Neural Network	Mạng nơ-ron sâu
7	CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
8	RNN	Recurrent Neural Network	Mạng nơ-ron hồi quy
9	GPU	Graphics Processing Unit	Bộ vi xử lý đồ họa
10	CPU	Central Processing Unit	Bộ xử lý trung tâm
11	CUDA	Compute Unified Device Architecture	Kiến trúc thiết bị tính toán hợp nhất
12	ResNet	Residual Network	Mạng dư
13	R-CNN	Region-based Convolutional Neural Network	Mạng nơ-ron tích chập theo vùng
14	FCN	Fully Connected Neural Network	Mạng tích chập đầy đủ

15	FPN	Feature Pyramid Network	Mạng kim tự tháp đặc trưng
16	RPN	Region Proposal Network	Mạng đề xuất vùng
17	RoI, RoIs	Region of Interest	Vùng đề xuất
18	RoIAlign	Region of Interest Alignment	Liên kết vùng đề xuất
19	Spyder	Scientific Python Development Environment	Môi trường phát triển Python dành cho ứng dụng khoa học
20	IDE	Integrated Development Environment	Môi trường phát triển tích hợp
21	JSON	JavaScript Object Notation	Định dạng đối tượng JavaScript
22	GUI	Graphical User Interface	Giao diện đồ họa người dùng



**DANH MỤC BẢNG BIỂU**

<i>Bảng 3.1. Kết quả Loss và Val_loss qua 20 epoch.....</i>	<i>45</i>
---	-----------

## MỤC LỤC HÌNH ẢNH

<i>Hình 1.1. Mối quan hệ giữa Artificial intelligence với Machine Learning và Deep Learning.....</i>	<i>4</i>
<i>Hình 1.2. Dữ liệu phân tách thành Training set và Test set.....</i>	<i>5</i>
<i>Hình 1.3. Một số loại học máy.....</i>	<i>7</i>
<i>Hình 1.4. Mô hình học máy có giám sát .....</i>	<i>8</i>
<i>Hình 1.5. Mô hình học máy không giám sát .....</i>	<i>8</i>
<i>Hình 1.6. Mô hình học máy bán giám sát.....</i>	<i>9</i>
<i>Hình 1.7. Mô hình mạng CNN .....</i>	<i>11</i>
<i>Hình 1.8. Mô hình mạng RNN.....</i>	<i>11</i>
<i>Hình 1.9. Một số môi trường Framework phổ biến.....</i>	<i>12</i>
<i>Hình 2.1. Kiến trúc mạng LeNet .....</i>	<i>15</i>
<i>Hình 2.2. Kiến trúc mạng AlexNet.....</i>	<i>16</i>
<i>Hình 2.3. Kiến trúc mạng VGGNet.....</i>	<i>17</i>
<i>Hình 2.4. Mô hình mạng Inception .....</i>	<i>19</i>
<i>Hình 2.5. Mối tương quan giữa độ sâu và hiệu suất mạng.....</i>	<i>20</i>
<i>Hình 2.6. ResNet sử dụng kết nối tắt xuyên qua một hay nhiều lớp .....</i>	<i>21</i>
<i>Hình 2.7. Ví dụ một DenseBlock gồm 5 layer.....</i>	<i>22</i>
<i>Hình 2.8. Cấu trúc của một Depthwise Separable Convolution.....</i>	<i>24</i>
<i>Hình 2.9. Mô hình mạng huấn luyện EfficientNet .....</i>	<i>25</i>
<i>Hình 2.10. Mô hình phương pháp Mask R-CNN .....</i>	<i>27</i>
<i>Hình 2.11. Kiến trúc cơ bản của một mạng Mask R-CNN.....</i>	<i>28</i>
<i>Hình 2.12. Cấu trúc Backbone.....</i>	<i>29</i>
<i>Hình 2.13. Cấu trúc RPN.....</i>	<i>30</i>
<i>Hình 2.14. Cấu trúc RoIAlign .....</i>	<i>31</i>
<i>Hình 2.15. Cấu trúc của nhánh phát hiện đối tượng.....</i>	<i>32</i>
<i>Hình 2.16. Cấu trúc của nhánh tạo mặt nạ.....</i>	<i>33</i>
<i>Hình 3.1. Logo Anaconda .....</i>	<i>36</i>
<i>Hình 3.2. Giao diện Anaconda.....</i>	<i>37</i>

<i>Hình 3.3. Giao diện phần mềm Spyder .....</i>	<i>38</i>
<i>Hình 3.4. Dữ liệu hình ảnh và gắn nhãn.....</i>	<i>39</i>
<i>Hình 3.5. Kết quả thu được sau khi gắn nhãn .....</i>	<i>39</i>
<i>Hình 3.6. Dataset .....</i>	<i>40</i>
<i>Hình 3.7. Thư mục chứa file JSON gắn nhãn .....</i>	<i>41</i>
<i>Hình 3.8. File list.txt chứa tên data .....</i>	<i>41</i>
<i>Hình 3.9. Thiết lập các thư viện.....</i>	<i>41</i>
<i>Hình 3.10. Tải tập dữ liệu từ thư mục chỉ định.....</i>	<i>42</i>
<i>Hình 3.11. Tải các mặt nạ (masks) cho hình ảnh.....</i>	<i>42</i>
<i>Hình 3.12. Trả về đường dẫn của một hình ảnh cụ thể trong tập dữ liệu.....</i>	<i>43</i>
<i>Hình 3.13. Các bước chuẩn bị tập dữ liệu.....</i>	<i>43</i>
<i>Hình 3.14. Lấy mẫu ngẫu nhiên từ một hình ảnh từ tập huấn luyện .....</i>	<i>43</i>
<i>Hình 3.15. Cấu hình Config .....</i>	<i>44</i>
<i>Hình 3.16. Thông số của model .....</i>	<i>44</i>
<i>Hình 3.17. Xác định thư mục lưu trữ, mô hình và tiến hành training .....</i>	<i>45</i>
<i>Hình 3.18. Huấn luyện mô hình .....</i>	<i>45</i>
<i>Hình 3.19. Kết quả thu về sau khi train .....</i>	<i>46</i>
<i>Hình 3.20. Một số kết quả dò tìm đối tượng trong ảnh.....</i>	<i>48</i>

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Trong thời đại hiện nay, sự tiến bộ của công nghệ thông tin và trí tuệ nhân tạo đã mở ra những cánh cửa mới cho ứng dụng của mô hình xử lý hình ảnh. Với sự phát triển nhanh chóng của các công nghệ như xử lý ảnh số, học sâu, và tính toán đám mây, khả năng xử lý và phân tích hình ảnh đã trở nên mạnh mẽ và linh hoạt hơn bao giờ hết.

Mô hình xử lý hình ảnh không chỉ đơn thuần là một công cụ để nhận diện đối tượng trong hình ảnh, mà còn là một công cụ quan trọng để hiểu và tương tác với thế giới xung quanh. Trong lĩnh vực y tế, theo một nghiên cứu của Nature Medicine, mô hình xử lý hình ảnh có thể giúp chẩn đoán bệnh đột biến trước khi các triệu chứng xuất hiện với độ chính xác đến 97%. Trong lĩnh vực an ninh và giám sát, một báo cáo của Global Market Insights dự báo thị trường giám sát thông minh sẽ đạt 23,8 tỷ đô la Mỹ vào năm 2027, với sự phát triển của các hệ thống nhận diện khuôn mặt và phân loại hành vi. Đồng thời, trong các lĩnh vực như tự động hóa và robot, mô hình phân loại hình ảnh giúp các hệ thống tự động nhận biết và tương tác với môi trường xung quanh một cách thông minh, như được minh họa trong việc phát triển xe tự lái và robot dịch vụ.

Với sự bùng nổ của dữ liệu hình ảnh từ các nguồn như camera giám sát, thiết bị di động, và Internet of Things (IoT), việc có các mô hình xử lý hình ảnh mạnh mẽ và hiệu quả là cực kỳ cần thiết để tận dụng và phân tích dữ liệu này một cách hiệu quả.

Từ các nhận định trên và với những chỉ dẫn của giảng viên hướng dẫn, nhóm đã quyết định chọn nội dung “Nghiên cứu mô hình mạng Mask R-CNN, ứng dụng dò tìm đối tượng trong ảnh” làm đề tài nghiên cứu thực hiện khóa luận tốt nghiệp của nhóm.

## 2. Mục tiêu đề tài

Khóa luận tập trung vào nghiên cứu, tìm hiểu về mạng nơ ron tích chập. Sau đó tìm hiểu về cách hoạt động và xây dựng mô hình Mask R-CNN. Ở phần ứng dụng, nhóm sử dụng mô hình Mask R-CNN ứng dụng dò tìm đối tượng trong ảnh.

### **3. Đối tượng và phạm vi nghiên cứu**

Đối tượng nghiên cứu của khóa luận tập trung vào lĩnh vực Machine Learning và Deep Learning, đặc biệt là tập trung sâu vào mạng Mask Region-based Convolutional Neural Network (Mask R-CNN) cùng một số mô hình liên quan.

Phạm vi nghiên cứu của khóa luận được xây dựng trên Anaconda là một nền tảng tích hợp cho việc quản lý môi trường Python và các gói khoa học dữ liệu.

### **4. Phương pháp nghiên cứu**

#### **4.1. Cách thức nghiên cứu**

Nghiên cứu chi tiết, tổng hợp kiến thức về Machine Learning, Deep Learning, Supervised Learning và Mask Region-based Convolutional Neural Network (Mask R-CNN) để hiểu rõ về các phương pháp và mô hình liên quan.

Tiến hành chuẩn bị dữ liệu một cách kỹ lưỡng, xây dựng mô hình xử lý và giải quyết bài toán Deep Learning dựa trên dữ liệu thu thập. Sử dụng các công cụ và ngôn ngữ lập trình như Python để tối ưu hóa quá trình này.

Ứng dụng kiến thức đã nghiên cứu vào việc xây dựng một mô hình dò tìm đối tượng trong ảnh.

#### **4.2. Xây dựng môi trường**

- Ngôn ngữ được sử dụng để viết code là Python.
- Deep Learning được chọn để sử dụng bao gồm các framework Tensorflow và Keras.
- Hệ điều hành Windows hoặc hệ điều hành mã nguồn mở Linux (khuyến nghị).

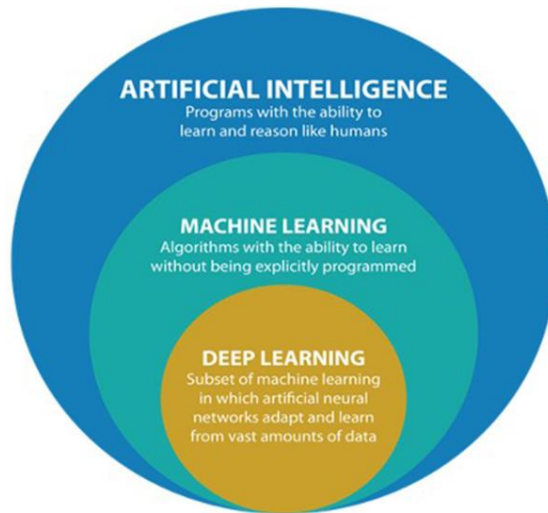
## CHƯƠNG 1:GIỚI THIỆU TỔNG QUAN VỀ AI

### 1.1. TỔNG QUAN VỀ AI

Trong những thập kỷ gần đây, không thể phủ nhận rằng Artificial Intelligence (trí tuệ nhân tạo) đang trở thành tâm điểm của sự chú ý và làm thay đổi đáng kể cả thế giới xã hội và kinh tế, là một bằng chứng đầy rõ nét của cuộc cách mạng công nghiệp lần thứ tư. Đặc biệt, hai khái niệm con quan trọng là Machine Learning (học máy) và Deep Learning (học sâu) đã mọc lên như những "nhân tố thần kỳ" mang lại khả năng tính toán vô cùng mạnh mẽ và khả năng tự học tập [1] [2].

Sự tiến bộ của trí tuệ nhân tạo đang ngày càng thâm hơn, và chúng ta thường không nhận ra điều này trong cuộc sống hàng ngày. Nó đã lan rộng vào mọi lĩnh vực, từ xe tự hành của Google và Tesla trên đường, đến hệ thống tự động nhận diện và gắn thẻ khuôn mặt trên Facebook, cũng như trợ lý ảo như Siri của Apple và thậm chí là camera trí tuệ nhận diện độ tuổi trên smartphone. Trong thực tế, sự hiện diện của trí tuệ nhân tạo đã thâm vào cuộc sống hàng ngày của chúng ta một cách mạnh mẽ và rộng lớn hơn chúng ta có thể tưởng tượng.

Machine Learning, như một tập con của trí tuệ nhân tạo, không chỉ là một lĩnh vực nhỏ trong khoa học máy tính, mà còn đánh dấu bước tiến quan trọng trong khả năng học hỏi tự động của máy tính. Không cần phải được lập trình cụ thể, Machine Learning mở rộng khả năng của máy tính, giúp ta dễ dàng tiếp cận và sử dụng công nghệ một cách linh hoạt. Và từ sự nở rộ của Machine Learning, Deep Learning, hay còn được biết đến với tên gọi học sâu, đã mở ra một thế giới mới với khả năng xử lý thông tin phức tạp và giải quyết những vấn đề mà trước đây được xem là quá khó khăn. Khả năng phân loại cả ngàn vật thể trong ảnh, tạo chú thích tự động cho hình ảnh, thậm chí là bắt chước giọng nói và chữ viết của con người, Deep Learning đang làm thay đổi cách ta nhìn nhận về sức mạnh của máy tính.



Hình 1.1. Mối quan hệ giữa Artificial intelligence với Machine Learning và Deep Learning

Vì vậy, nếu ta muốn đặt ra một quan hệ giữa Artificial Intelligence, Machine Learning, và Deep Learning, đó có thể được mô tả như sau: "Deep Learning là một tập con của Machine Learning, trong khi Machine Learning lại là một phần của Artificial Intelligence." Điều này thực sự là một cuộc hội ngộ của trí thức và sự sáng tạo, mở ra những triển vọng hứa hẹn cho tương lai của công nghệ và con người.

## 1.2. MACHINE LEARNING

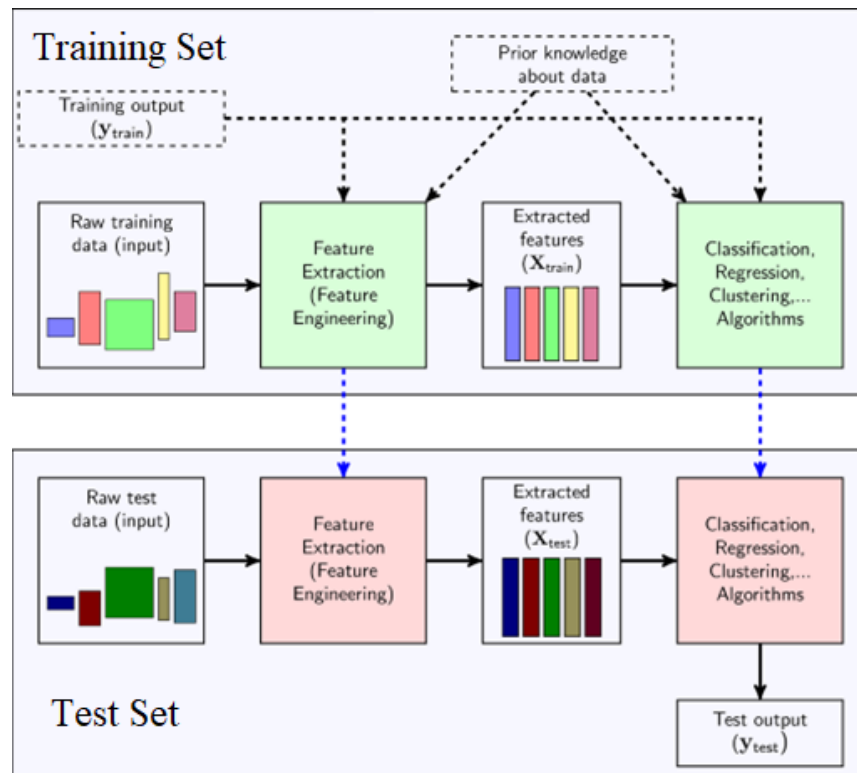
Machine learning (ML) đặt ra như một lĩnh vực quan trọng trong Trí tuệ Nhân tạo (AI), nơi mà các hệ thống máy tính được thiết kế để tự động học từ dữ liệu và tự cải thiện mà không yêu cầu sự can thiệp trực tiếp của người lập trình. Khái niệm cơ bản của machine learning là mang đến khả năng cho máy tính "học" thông qua các kinh nghiệm trước đó, và từ đó, máy tính có thể tự động điều chỉnh hoạt động của mình để phản ánh dữ liệu mới <sup>[1]</sup>. Trong quá trình machine learning, ta sử dụng các thuật toán để phân tích dữ liệu và xây dựng các mô hình, giúp máy tính thực hiện nhiều nhiệm vụ khác nhau mà không cần sự can thiệp trực tiếp của con người. Các ứng dụng của machine learning là vô cùng đa dạng, từ nhận diện hình ảnh và dự đoán dữ liệu cho đến tự động lái xe, dịch ngôn ngữ tự động, và nhiều lĩnh vực khác.

### 1.2.1. Cách thức hoạt động

Machine Learning (ML) đang là một phần quan trọng của Trí tuệ Nhân tạo (AI), mở ra những cơ hội mới trong cách ta tương tác với thế giới xung quanh. ML có thể đơn giản được định nghĩa là quá trình máy tính học từ dữ liệu để đưa ra quyết định hoặc dự đoán mà không cần sự can thiệp lập

trình chi tiết cho từng tình huống. Khác với lập trình truyền thống, nơi có các quy tắc được định nghĩa một cách rõ ràng và cứng nhắc, ML mang lại khả năng học và thích ứng với tình huống mới thông qua phân tích dữ liệu. Dữ liệu đóng một vai trò cực kỳ quan trọng trong ML, đó là nguồn nguyên liệu chính để "nuôi dưỡng" các mô hình. Chất lượng và lượng dữ liệu quyết định độ chính xác và hiệu suất của mô hình. Trong môi trường ML, dữ liệu thường được chia thành hai loại chính: Bộ Dữ liệu Huấn Luyện (Training Set) và Bộ Dữ liệu Kiểm Thử (Testing Set).

- Bộ Dữ liệu Huấn Luyện là nền tảng để xây dựng và "huấn luyện" mô hình. Mô hình ML học từ dữ liệu này, tự phát hiện các mẫu và quy luật trong dữ liệu. Sự đa dạng và lượng lớn dữ liệu trong Bộ Dữ liệu Huấn Luyện càng nhiều, mô hình càng có khả năng học tốt và trở nên thông minh hơn.
- Bộ Dữ liệu Kiểm Thử, ngược lại, được sử dụng để đánh giá mô hình. Sau khi "huấn luyện" trên Bộ Dữ liệu Huấn Luyện, mô hình được kiểm tra trên Bộ Dữ liệu Kiểm Thử để kiểm tra khả năng áp dụng những gì đã học vào dữ liệu mới mà nó chưa từng gặp trước đây. Quá trình này giúp đánh giá khả năng tổng quát hóa của mô hình một cách khách quan.



Hình 1.2. Dữ liệu phân tách thành Training set và Test set



### a. Training Set

Trong mỗi giai đoạn, hai thành phần quan trọng nhất được sử dụng là Trích chọn đặc trưng (Feature Engineering hoặc Feature Extraction) và các thuật toán phân loại hồi quy. Cả hai đều đóng vai trò quan trọng trong việc ảnh hưởng đến kết quả cuối cùng, đòi hỏi sự cẩn thận và chuyên môn cao từ người thiết kế.

Khởi Trích chọn đặc trưng có thể nhận đầu vào từ các yếu tố sau:

**Dữ liệu thô ban đầu (raw training input):** Đây là dữ liệu chưa được xử lý, bao gồm tất cả thông tin liên quan đến dữ liệu. Điều này có thể là các véc tơ, ma trận, hoặc định dạng khác, không có quy chuẩn cụ thể. Dữ liệu thô thường chứa nhiều và thông tin không cần thiết, do đó việc xử lý và rút gọn là quan trọng để tối ưu tính toán sau này.

**Dữ liệu đầu ra (output của training set):** Là thông tin hữu ích được tính toán từ dữ liệu đầu vào, đã loại bỏ nhiễu và vô nghĩa. Việc rút gọn kích thước giúp tối ưu hóa hiệu suất tính toán, làm cho quá trình phân loại hoặc dự đoán hiệu quả hơn.

**Thông tin biết trước về dữ liệu (prior knowledge about data):** Đây là yếu tố tùy chọn, xuất hiện khi có thông tin trước về các đặc trưng quan trọng cho mô hình dự đoán. Thông tin này giúp người thiết kế lựa chọn đặc trưng và phương pháp tính toán phù hợp, làm tăng độ chính xác của mô hình.

### b. Test Set

Khi tiếp cận dữ liệu mới, quá trình sử dụng bộ trích chọn đặc trưng đã được xác định trước là bước quan trọng. Bộ trích chọn đặc trưng này giúp tạo ra vector đặc trưng tương ứng với dữ liệu thô, mà sau đó sẽ được đưa vào thuật toán chính đã được chọn để đưa ra quyết định.

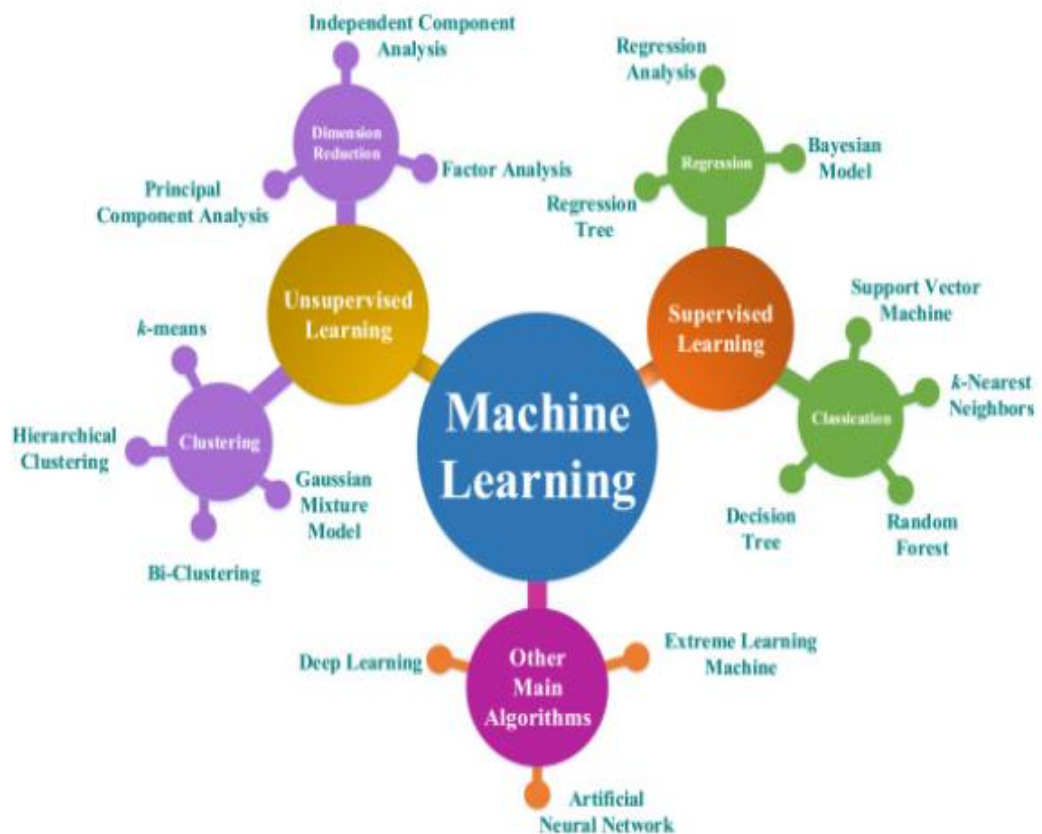
Các thuật toán Machine Learning thường được phân loại theo hai cách phổ biến: phương thức học (Supervised và Unsupervised Learning) và chức năng (hồi quy, phân loại, gom nhóm...). Điều này tạo ra sự đa dạng trong cách mà ta tiếp cận vấn đề.

Một ưu điểm lớn của Machine Learning là độ chính xác của mô hình dự đoán phụ thuộc lớn vào chất lượng của các đặc trưng được chọn. Sự phù hợp giữa đặc trưng và bài toán đặt ra sẽ quyết định đến hiệu suất của mô hình. Điều này không chỉ là điểm mạnh mà còn là điểm yếu vì quá

trình trích chọn đặc trưng đòi hỏi sự hiểu biết chi tiết về bài toán, thuật toán sử dụng, và các thông số của mô hình. Mỗi bài toán đều đòi hỏi các đặc trưng được thiết kế riêng biệt, và chúng không thể được áp dụng một cách chung chung cho mọi tình huống mới mà không cần sửa đổi hoặc thay thế chúng.

### 1.2.2. Phân loại Machine Learning

Học máy bao gồm nhiều kỹ thuật và biến thể đa dạng, nhưng thường được phân loại thành ba loại chính: Học máy có giám sát (Supervised Learning), Học máy không giám sát (Unsupervised Learning), và Học máy bán giám sát (Semi-supervised Learning). Mặc dù chúng khác nhau về cách tiếp cận, nhưng có điểm tương đồng quan trọng: chúng đều có xu hướng mô phỏng cách con người học

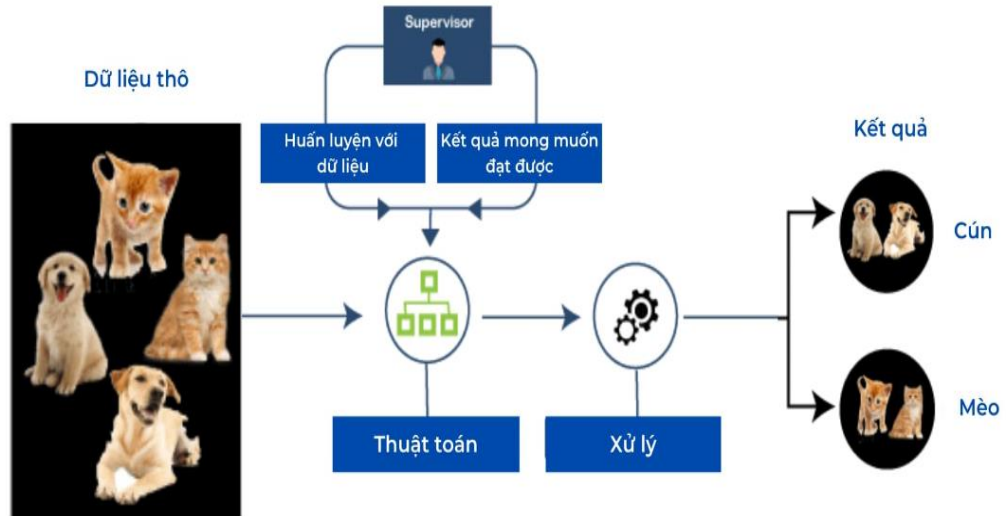


Hình 1.3. Một số loại học máy

#### a. Học máy có giám sát

Học máy có giám sát sử dụng bộ dữ liệu được gán nhãn để huấn luyện mô hình, nhằm phân loại dữ liệu hoặc dự đoán kết quả một cách chính xác. Trong quá trình huấn luyện, mô hình điều chỉnh trọng số của mình để phản ánh đúng nhãn của dữ liệu đầu vào. Các phương pháp bao gồm mạng

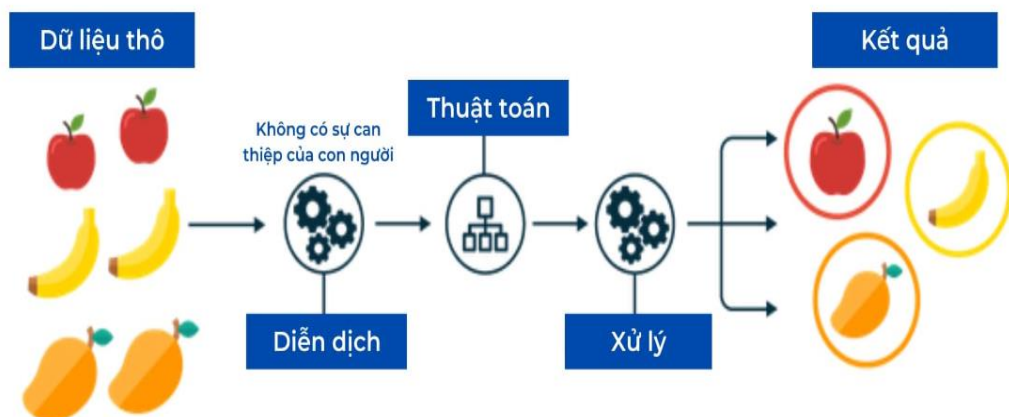
lưới thần kinh, vịnh ngây thơ, hồi quy tuyến tính, hồi quy logistic, rừng ngẫu nhiên và máy vector hỗ trợ (SVM).



Hình 1.4. Mô hình học máy có giám sát

#### b. Học máy không giám sát

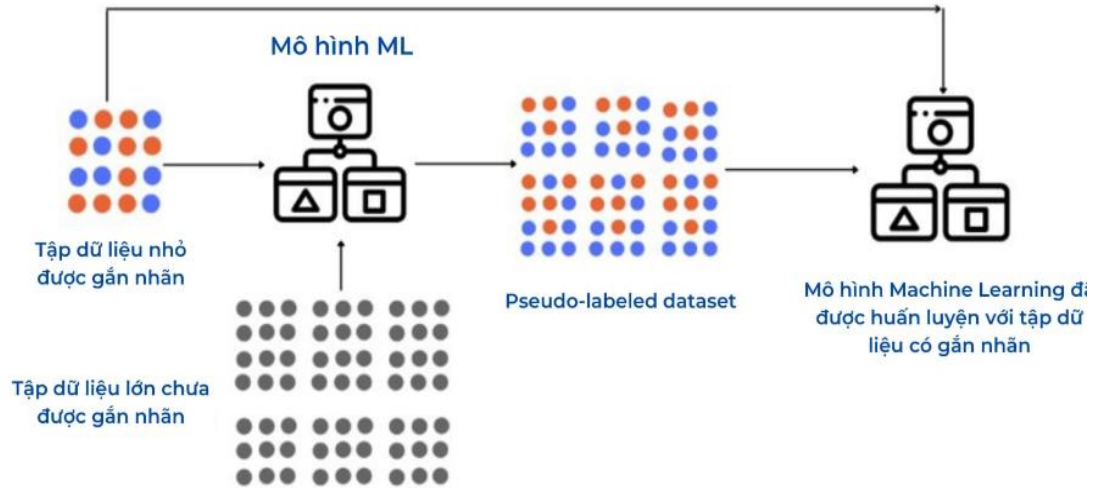
Học máy không giám sát sử dụng một số thuật toán để phân tích và phân cụm nhiều tập dữ liệu không được gắn nhãn. Thuật toán này tự động khám phá các mẫu và nhóm trong dữ liệu mà không cần sự can thiệp của con người. Điều này làm cho nó phù hợp cho việc phân tích dữ liệu thăm dò, chiến lược bán chéo, phân khúc khách hàng và nhận dạng hình ảnh và mẫu. Các phương pháp bao gồm mạng lưới thần kinh, phân cụm k-mean, phương pháp phân cụm xác suất, PCA và SVD.



Hình 1.5. Mô hình học máy không giám sát

### c. Học máy bán giám sát

Học máy bán giám sát là sự kết hợp giữa học có giám sát và không giám sát. Nó sử dụng tập dữ liệu được gắn nhãn nhỏ để hướng dẫn quá trình đào tạo, trong khi cố gắng trích xuất tính năng từ tập dữ liệu lớn hơn, không được gắn nhãn. Điều này giúp giải quyết vấn đề thiếu dữ liệu được gắn nhãn trong học có giám sát và giảm chi phí của việc dán nhãn...



Hình 1.6. Mô hình học máy bán giám sát

## 1.3. DEEP LEARNING

### 1.3.1. Khái Niệm

Học sâu, một phần quan trọng của học máy, sử dụng mạng lưới thần kinh nhiều lớp, thường được gọi là mạng lưới thần kinh sâu, để tái tạo khả năng đưa ra quyết định phức tạp như bộ não con người [3]. Định nghĩa chặt chẽ của mạng nơ-ron sâu (DNN) là một mạng nơ-ron có ít nhất ba lớp, nhưng thực tế, hầu hết chúng có nhiều lớp hơn. DNN được đào tạo trên lượng lớn dữ liệu để nhận diện và phân loại hiện tượng, xây dựng mô hình và mối quan hệ, đánh giá khả năng, cũng như đưa ra dự đoán và quyết định.

Mặc dù mạng nơ-ron một lớp có thể đưa ra dự đoán hữu ích, nhưng sự bổ sung của nhiều lớp trong mạng nơ-ron sâu giúp tinh chỉnh và tối ưu hóa kết quả để đạt được độ chính xác cao hơn. Học sâu đóng góp vào nhiều ứng dụng và dịch vụ, nâng cao tự động hóa và thực hiện nhiều nhiệm vụ phân tích và vật lý mà không yêu cầu sự can thiệp của con người.

Học sâu đã thúc đẩy sự tiện lợi trong cuộc sống hàng ngày của ta thông qua các sản phẩm và dịch vụ như trợ lý kỹ thuật số, điều khiển giọng nói trên tivi, và phát hiện gian lận thẻ tín dụng. Ngoài ra, nó còn đóng vai trò quan trọng trong các công nghệ mới như ô tô tự lái và tổng hợp trí tuệ nhân tạo. Nhờ vào sự tiên tiến của học sâu, các ứng dụng và dịch vụ ngày càng trở nên thông minh và hiệu quả, tạo ra một cuộc cách mạng trong lĩnh vực công nghiệp và cuộc sống hàng ngày của ta.

### **1.3.2. Cách thức hoạt động**

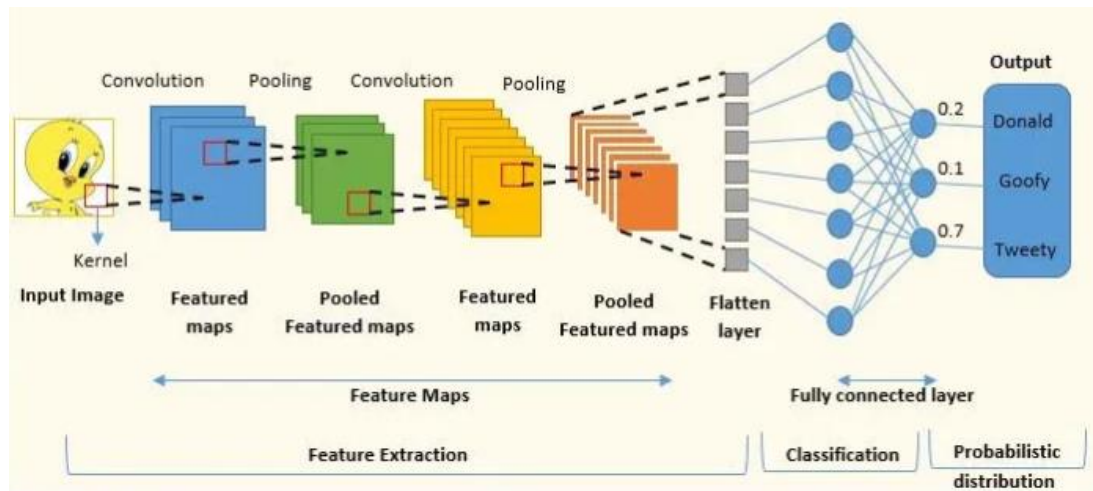
Mạng lưới thần kinh học sâu, hay còn gọi là mạng lưới thần kinh nhân tạo, đặt mục tiêu bắt chước quá trình làm việc của bộ não con người thông qua sự kết hợp thông tin từ dữ liệu đầu vào, trọng số và độ lệch. Các yếu tố này tương tác với nhau để nhận dạng, phân loại và mô tả chính xác các đối tượng trong dữ liệu.

Mạng lưới thần kinh sâu bao gồm nhiều lớp nút kết nối với nhau, mỗi lớp được xây dựng dựa trên lớp trước để điều chỉnh và tối ưu hóa dự đoán hoặc phân loại. Quá trình tính toán này được thực hiện qua mạng theo hình thức lan truyền tiến. Lớp đầu vào và đầu ra của mạng được gọi là các lớp hiển thị. Lớp đầu vào là nơi mô hình nhập dữ liệu để xử lý và lớp đầu ra là nơi đưa ra dự đoán hoặc phân loại cuối cùng.

Lan truyền ngược là một quy trình khác sử dụng thuật toán như giảm độ dốc để tính toán lỗi trong dự đoán. Sau đó, nó điều chỉnh trọng số và độ lệch của hàm bằng cách di chuyển ngược qua các lớp, mục tiêu là huấn luyện mô hình. Lan truyền thuận và lan truyền ngược cùng nhau giúp mạng lưới thần kinh đưa ra dự đoán và sửa chữa bất kỳ lỗi nào tương ứng. Theo thời gian, thuật toán ngày càng trở nên chính xác.

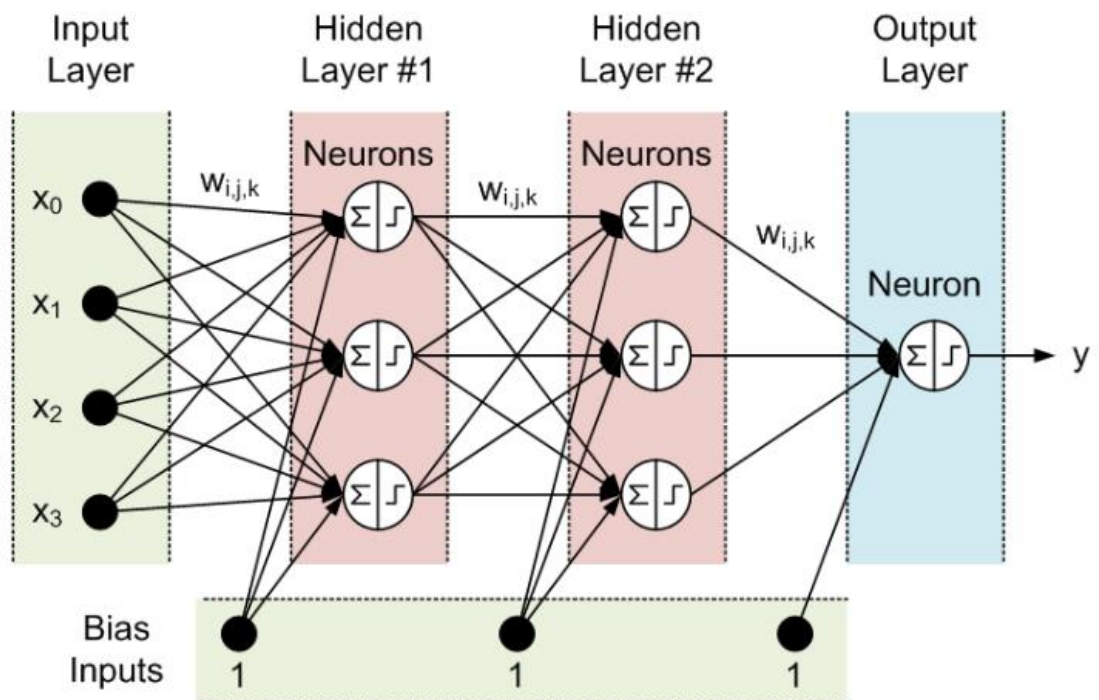
Trên đây là mô tả đơn giản về mạng lưới thần kinh sâu và quy trình hoạt động của nó. Tuy nhiên, các thuật toán học sâu thực sự rất phức tạp và có nhiều loại mạng thần kinh khác nhau để giải quyết các vấn đề hoặc xử lý các bộ dữ liệu cụ thể.

Mạng thần kinh tích chập (CNN), ví dụ, chủ yếu được áp dụng trong phân loại hình ảnh và thị giác máy tính. Nó có khả năng phát hiện các tính năng và mẫu trong hình ảnh, giúp thực hiện các nhiệm vụ như nhận dạng và phân loại đối tượng. Năm 2015, CNN đã lần đầu tiên vượt qua khả năng của con người trong thử thách nhận dạng vật thể.



Hình 1.7. Mô hình mạng CNN

Mạng thần kinh tái phát (RNN) thường được sử dụng trong nhận dạng giọng nói và ngôn ngữ tự nhiên do khả năng tận dụng dữ liệu chuỗi thời gian hoặc tuần tự.



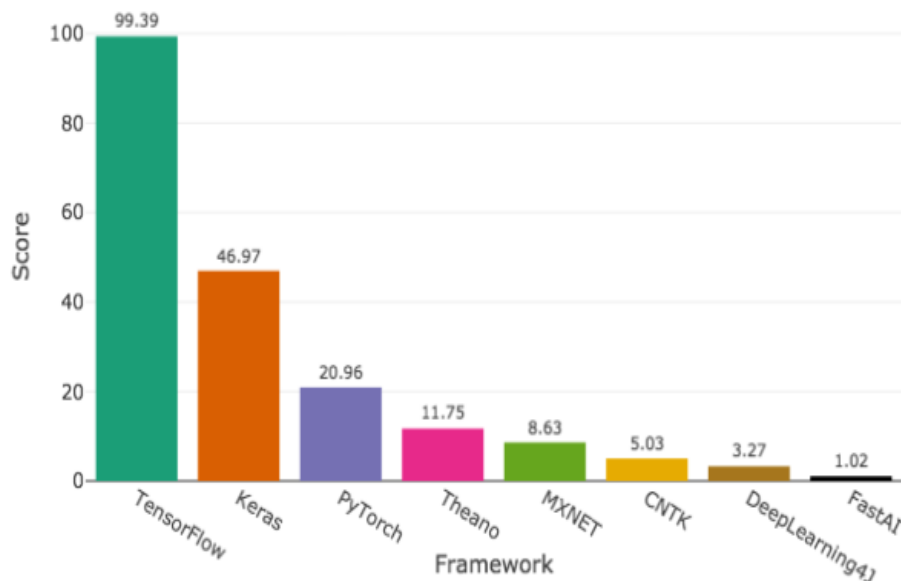
Hình 1.8. Mô hình mạng RNN

#### 1.4. MÔI TRƯỜNG FRAMEWORK

Để xây dựng các mô hình sử dụng Neural Network (NN) và Convolutional Neural Network (CNN) từ đầu bằng Python có thể trở nên phức tạp, đặc biệt là trong quá trình thực hiện backpropagation. Việc triển khai và tối ưu hóa tốc độ tính toán trở nên khó khăn. Do đó, các framework



Machine Learning và Deep Learning được phát triển để giải quyết những thách thức này và mang lại những ưu điểm sau:



Hình 1.9. Một số môi trường Framework phổ biến

**Tự động Backpropagation:** Người dùng chỉ cần định nghĩa mô hình và hàm mất mát (loss function), framework sẽ tự động thực hiện quá trình backpropagation.

**Dễ định nghĩa Layers và Activation Functions:** Các framework cho phép người dùng dễ dàng định nghĩa các lớp (layers), hàm kích hoạt (activation functions) một cách đơn giản. Ví dụ, để thêm một lớp trong mạng Neural Network, người dùng chỉ cần xác định số node và hàm kích hoạt.

**Tối ưu hóa Tốc độ Tính toán GPU và CPU:** Các framework được tối ưu hóa để chạy trên cả CPU và GPU, đặc biệt tận dụng được tốc độ tính toán của GPU thông qua các mở rộng CUDA.

Trong số các framework, TensorFlow là một trong những framework phổ biến nhất, nhưng có thể thấy nó khá phức tạp đối với người mới bắt đầu. Đó là lý do tại sao Keras, một giao diện API cao cấp, trở nên phổ biến. Keras làm cho việc xây dựng mô hình Machine Learning và Deep Learning trở nên dễ dàng hơn, với tính thân thiện với người dùng.

TensorFlow, là một thư viện mã nguồn mở, chạy trên nhiều loại phần cứng khác nhau và có thể thực hiện nhiều loại nhiệm vụ nhận thức và hiểu

ngôn ngữ. Keras, mặc dù là một framework độc lập, thường được tích hợp trực tiếp vào TensorFlow, giúp làm cho việc sử dụng TensorFlow trở nên thuận tiện và dễ dàng. Những tối ưu hóa này làm cho việc xây dựng và huấn luyện mô hình trở nên thuận lợi và hiệu quả, đặc biệt là khi sử dụng GPU của NVIDIA.

Trong thời đại hiện nay, TensorFlow đang trở thành một trong những framework phổ biến được sử dụng rộng rãi. Không chỉ là một thư viện mạnh mẽ cho học máy, TensorFlow còn tích hợp sẵn Keras, giúp người dùng dễ dàng xây dựng mô hình của mình mà không gặp quá nhiều khó khăn. Đặc biệt, tích hợp của TensorFlow với GPU của NVIDIA mang lại sự thuận lợi và tiện ích đáng kể cho người sử dụng, tối ưu hóa hiệu suất tính toán và tăng cường trải nghiệm phát triển. Sự kết hợp giữa TensorFlow và Keras không chỉ mang lại sự linh hoạt trong xây dựng mô hình, mà còn tận dụng được khả năng tích hợp sẵn của Keras để đơn giản hóa quá trình lập trình. Điều này đặc biệt quan trọng đối với người mới bắt đầu trong lĩnh vực học máy và deep learning, nơi mà tính thân thiện và dễ sử dụng của Keras đóng một vai trò quan trọng. Với việc tích hợp dễ dàng trên các GPU của NVIDIA, TensorFlow không chỉ cung cấp một nền tảng mạnh mẽ cho việc phát triển mô hình, mà còn tận dụng được sức mạnh tính toán của các thiết bị này. Điều này mang lại lợi ích lớn cho người sử dụng, giúp họ tiết kiệm thời gian và nỗ lực trong quá trình huấn luyện và triển khai mô hình của mình.

## 1.5. KẾT LUẬN CHƯƠNG 1

Trong chương 1, giới thiệu chung về trí tuệ nhân tạo (AI), học máy (ML) và học sâu (DL). Trong số đó, học sâu đã nổi lên như một trong những lĩnh vực quan trọng nhất, với khả năng học và hiểu được dữ liệu phức tạp.

Các con số thực tế chỉ ra sức hút mạnh mẽ của học sâu. Theo báo cáo từ Gartner, vào năm 2023, 40% các dự án AI sẽ sử dụng DL, so với chỉ 5% vào năm 2019. Điều này minh chứng cho sự gia tăng đáng kể trong việc áp dụng học sâu trong các ứng dụng thực tiễn.

Các framework học sâu như TensorFlow và PyTorch đã chứng tỏ sức mạnh của mình. Theo một nghiên cứu của Gradient Ventures, vào năm 2022, TensorFlow và PyTorch là hai framework phổ biến nhất, chiếm hơn



85% thị phần trong số các nhà nghiên cứu và doanh nghiệp. Sự phát triển của các framework này đã đóng vai trò quan trọng trong việc đẩy mạnh sự tiến bộ trong lĩnh vực học sâu.

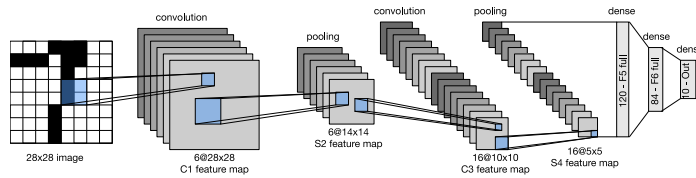
Với sự tiếp tục của sự đầu tư và nghiên cứu, học sâu sẽ tiếp tục đóng vai trò quan trọng trong việc phát triển các ứng dụng AI tiên tiến và tạo ra những tiến bộ đáng kể trong nhiều lĩnh vực khác nhau, từ y tế đến tự động hóa công nghiệp.

## CHƯƠNG 2: MẠNG MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK (MASK R-CNN)

### 2.1. CÁC MÔ HÌNH MẠNG HUẤN LUYỆN PHỔ BIẾN

#### 2.1.1. Mạng huấn luyện LeNet

Mạng LeNet là một trong những mạng nơ-ron sâu đầu tiên được phát triển bởi Yann LeCun, Léon Bottou, Yoshua Bengio và Patrick Haffner và đã được sử dụng rộng rãi trong các ứng dụng nhận diện hình ảnh và nhận dạng ký tự. Mạng này được giới thiệu vào năm 1998 cho việc nhận dạng chữ số viết tay trong bài báo của LeCun và nhóm nghiên cứu của mình.



Hình 2.1. Kiến trúc mạng LeNet

LeNet có cấu trúc sâu đối với thời điểm của nó, bao gồm các lớp tích chập (convolutional layers) xen kẽ với các lớp tầng đặt hàng (pooling layers) và các lớp kích hoạt phi tuyến tính như ReLU. Điều này cho phép nó hiệu quả trong việc học các đặc trưng từ dữ liệu hình ảnh.

Mạng LeNet đã đặt nền móng cho sự phát triển của các mô hình học sâu sau này, và nó thường được coi là một trong những bước tiến quan trọng trong lịch sử của trí tuệ nhân tạo và học sâu.

LeNet giải quyết một loạt các vấn đề trong lĩnh vực thị giác máy tính. Một trong những vấn đề quan trọng nhất mà nó giải quyết là việc nhận dạng ký tự và chữ số trong hình ảnh. Điều này có ứng dụng rộng rãi trong nhiều lĩnh vực, từ việc đọc chữ số trên biển số xe đến xử lý tài liệu và nhận diện chữ viết tay.

Ngoài ra, LeNet cũng được sử dụng để phân loại hình ảnh, đặc biệt là trong việc phân loại chữ số từ bộ dữ liệu MNIST. Khả năng của nó trong việc phân loại hình ảnh giúp giải quyết nhiều vấn đề thực tế, từ nhận dạng đối tượng trong ảnh đến phân loại các loại sản phẩm trong bản đồ siêu thị.

Một điểm mạnh của LeNet là khả năng xử lý ảnh dưới dạng dữ liệu không đánh số. Điều này có nghĩa là nó có thể làm việc với ảnh chưa được gán nhãn hoặc có cấu trúc phức tạp, giúp nó trở thành một công cụ linh hoạt và mạnh mẽ trong nhiều ứng dụng thị giác máy tính.

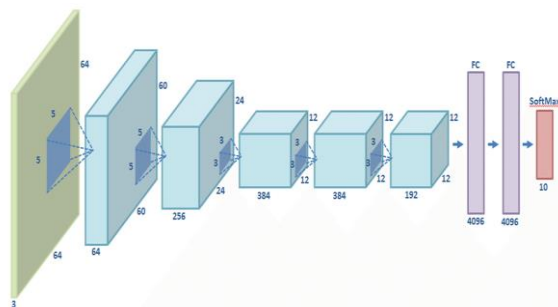
LeNet là một trong những bước tiến quan trọng đối với thị giác máy tính, mở ra cánh cửa cho việc sử dụng mạng nơ-ron tích chập trong nhiều ứng dụng thực tế. Không chỉ giải quyết vấn đề nhận dạng ký tự và chữ số trong hình ảnh, mà còn mang lại những tiềm năng lớn trong việc phân loại và xử lý hình ảnh.

Sự linh hoạt và hiệu suất của LeNet đã tạo ra một tiền đề quan trọng cho việc phát triển các mạng nơ-ron sâu hơn như AlexNet, VGG, và sau này là các mô hình hiện đại như ResNet và EfficientNet. Điều này chứng tỏ vai trò quan trọng của LeNet trong việc định hình hướng phát triển của trí tuệ nhân tạo và thị giác máy tính.

### 2.1.2. Mạng huấn luyện AlexNet

AlexNet là một kiến trúc mạng nơ-ron sâu (deep neural network) được giới thiệu vào năm 2012 bởi Alex Krizhevsky, Ilya Sutskever và Geoffrey Hinton. Được đào tạo trên tập dữ liệu ImageNet, AlexNet đã đạt được thành công lớn trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2012.

Vấn đề chính mà AlexNet giải quyết là vấn đề của nhận diện hình ảnh trong lĩnh vực thị giác máy tính. Trước sự xuất hiện của AlexNet, việc sử dụng mạng nơ-ron sâu để nhận diện hình ảnh gặp phải nhiều thách thức lớn, bao gồm độ phức tạp của dữ liệu hình ảnh, overfitting, và tính tổng quát hóa kém.



Hình 2.2. Kiến trúc mạng AlexNet

AlexNet đã đưa ra một giải pháp hiệu quả bằng cách thiết kế một kiến trúc mạng nơ-ron sâu với nhiều lớp convolutional và fully connected

layers. Bằng cách này, nó có khả năng học các đặc trưng phức tạp từ dữ liệu hình ảnh ở nhiều mức độ khác nhau của không gian và thời gian. Bên cạnh đó, AlexNet sử dụng các kỹ thuật chuẩn hóa dữ liệu, dropout, và hàm kích hoạt ReLU để giảm overfitting và tăng tốc quá trình học. Sử dụng GPU trong quá trình huấn luyện cũng giúp mô hình có thể học từ các tập dữ liệu lớn trong thời gian ngắn.

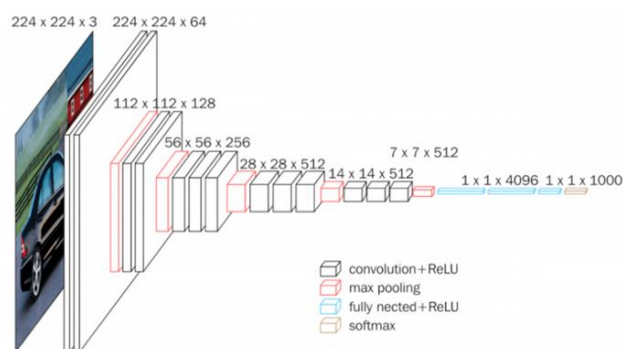
Bên cạnh đó, AlexNet sử dụng các kỹ thuật chuẩn hóa dữ liệu, dropout, và hàm kích hoạt ReLU để giảm overfitting và tăng tốc quá trình học. Sử dụng GPU trong quá trình huấn luyện cũng đóng vai trò quan trọng, giúp mô hình có thể học từ các tập dữ liệu lớn trong thời gian ngắn.

Nhờ những cải tiến này, AlexNet đã mang lại kết quả ấn tượng trong việc nhận diện và phân loại hình ảnh. Điều này đã mở ra một thời kỳ mới trong lĩnh vực thị giác máy tính sử dụng deep learning, đồng thời làm thay đổi cách chúng ta tiếp cận và xử lý dữ liệu hình ảnh trong nhiều ứng dụng thực tế.

### 2.1.3. Mạng huấn luyện VGGNet

Mạng VGGNet (Visual Geometry Group Network) là một kiến trúc mạng nơ-ron sâu (deep neural network) được giới thiệu bởi Karen Simonyan và Andrew Zisserman từ Đại học Oxford vào năm 2014. Tên "VGG" được đặt theo viết tắt của "Visual Geometry Group" tại Đại học Oxford, nơi nghiên cứu này được thực hiện.

VGGNet nổi tiếng với kiến trúc sâu và đơn giản, tập trung vào việc sử dụng các lớp convolutional nhỏ với kích thước  $3 \times 3$  và stride là 1. Mạng này thường sử dụng các lớp convolutional kết hợp với các lớp max pooling để giảm kích thước của đầu ra và tạo ra một biểu diễn đặc trưng cấp cao cho hình ảnh đầu vào.



Hình 2.3. Kiến trúc mạng VGGNet

Vấn đề chính mà VGGNet giải quyết là vấn đề của nhận diện và phân loại hình ảnh trong lĩnh vực thị giác máy tính. Trước khi VGGNet xuất hiện, việc sử dụng mạng nơ-ron sâu để nhận diện hình ảnh gặp phải nhiều thách thức lớn.

Một trong những thách thức chính là khả năng học được các đặc trưng phức tạp từ dữ liệu hình ảnh. Hình ảnh thường chứa nhiều chi tiết, biến thể và đặc điểm khác nhau, từ đó tạo ra một không gian dữ liệu rất lớn và phức tạp. Mạng nơ-ron cần có khả năng học được các đặc trưng này để hiểu và phân loại các đối tượng trong hình ảnh.

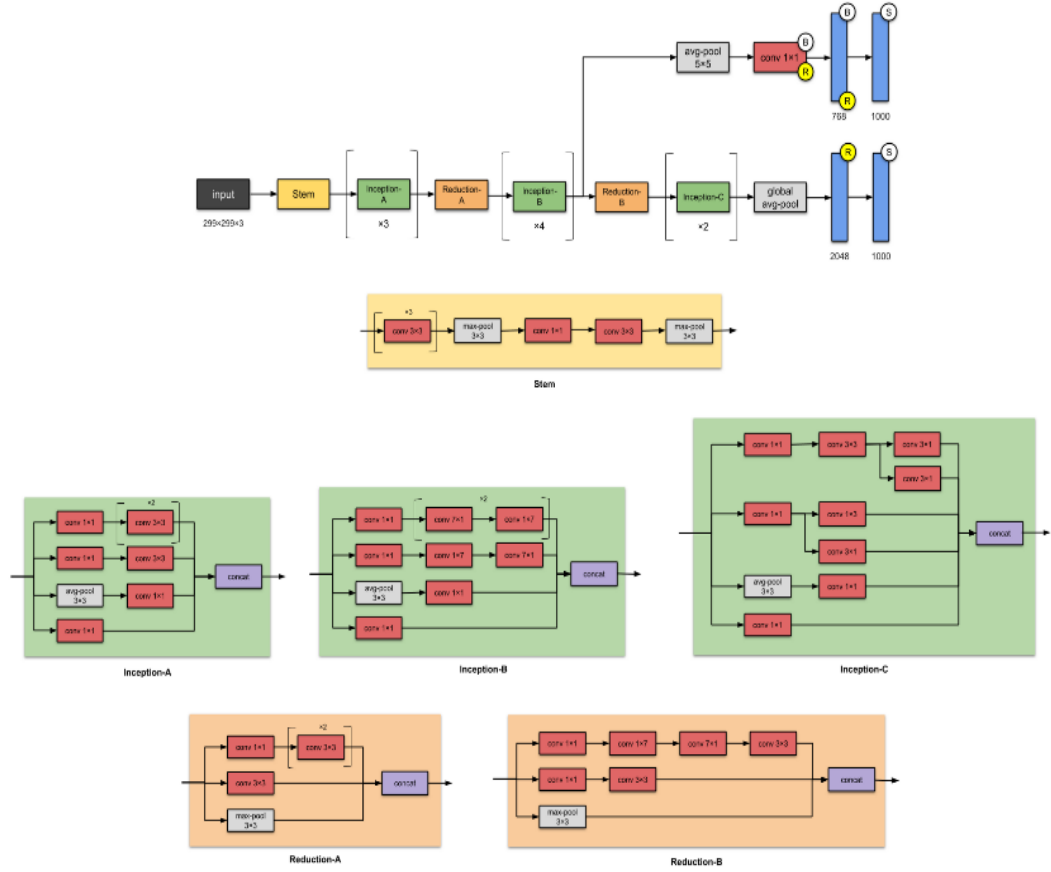
VGGNet đã đưa ra một giải pháp hiệu quả bằng cách thiết kế một kiến trúc mạng nơ-ron sâu với nhiều lớp convolutional và fully connected layers. Thông qua các lớp này, mạng có khả năng học các đặc trưng phức tạp từ dữ liệu hình ảnh ở nhiều mức độ khác nhau của không gian và thời gian.

Bên cạnh đó, VGGNet cũng tập trung vào việc sử dụng các lớp convolutional với kích thước nhỏ và đơn giản hóa kiến trúc, giúp dễ dàng triển khai và hiểu được mô hình. Điều này giúp cải thiện khả năng tổng quát hóa của mô hình và giảm nguy cơ overfitting, từ đó tăng cường hiệu suất của mạng trong việc nhận diện và phân loại hình ảnh.

#### **2.1.4. Mạng huấn luyện Inception**

Mạng Inception, còn được biết đến là GoogLeNet, là một kiến trúc mạng nơ-ron sâu (deep neural network) được phát triển bởi Google Research vào năm 2014. Mạng Inception là một trong những cải tiến đáng chú ý trong lĩnh vực thị giác máy tính và đã đạt được hiệu suất rất cao trong các nhiệm vụ nhận diện và phân loại hình ảnh.

Đặc điểm nổi bật của mạng Inception là việc sử dụng các module gọi là "Inception modules" để xây dựng kiến trúc mạng. Các Inception modules là một loạt các lớp convolutional và pooling được thực hiện song song và kết hợp lại với nhau để trích xuất đặc trưng từ hình ảnh. Thay vì sử dụng các lớp convolutional có kích thước lớn, Inception sử dụng các lớp convolutional với kích thước nhỏ và kết hợp chúng với nhau để tạo ra một biểu diễn đa dạng và phong phú của hình ảnh.



Hình 2.4. Mô hình mạng Inception

Vấn đề chính mà mạng Inception giải quyết là vấn đề về hiệu suất và hiệu quả trong việc nhận diện và phân loại hình ảnh trong lĩnh vực thị giác máy tính. Trước khi Inception xuất hiện, một trong những thách thức lớn đối với các mạng nơ-ron sâu là cân bằng giữa số lượng tham số của mô hình và hiệu suất của nó.

Các mạng nơ-ron sâu truyền thống thường có số lượng tham số lớn, đặc biệt là khi kích thước của các lớp convolutional lớn. Điều này có thể dẫn đến các vấn đề như overfitting (quá mức) và khó khăn trong việc huấn luyện trên các tập dữ liệu lớn.

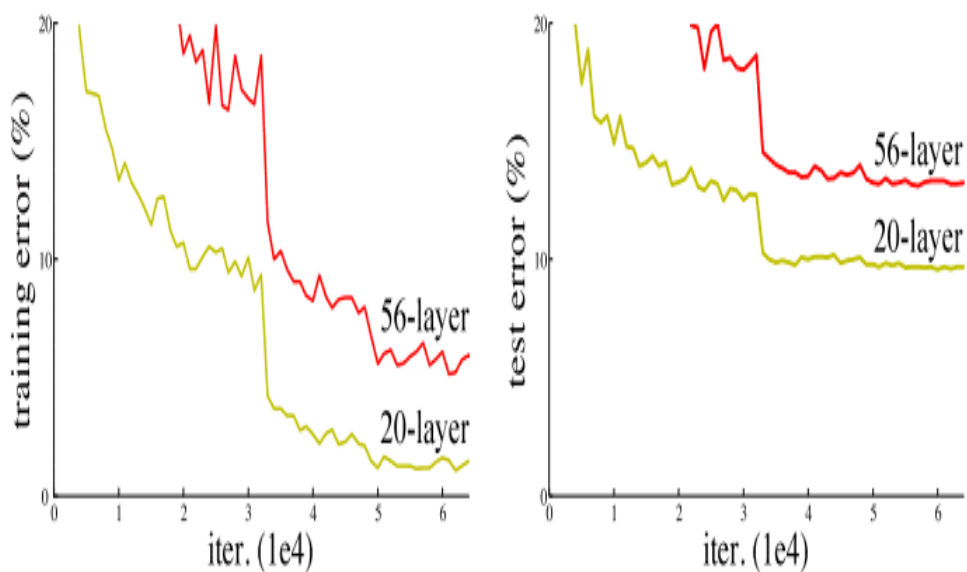
Inception giải quyết vấn đề này bằng cách sử dụng các Inception modules, là một loạt các lớp convolutional và pooling được thực hiện song song và kết hợp lại với nhau. Các Inception modules cho phép mạng học được các đặc trưng ở nhiều mức độ và tỷ lệ khác nhau trong dữ liệu hình ảnh mà không cần tăng số lượng tham số quá nhiều. Thay vào đó, Inception sử dụng các lớp convolutional nhỏ và kết hợp chúng để tạo ra một biểu diễn đa dạng và phong phú của hình ảnh.

Kết quả là, Inception có thể đạt được hiệu suất cao trong việc nhận diện và phân loại hình ảnh mà vẫn giữ được số lượng tham số trong mức độ quản lý được. Điều này làm cho mạng trở nên hiệu quả hơn và giảm nguy cơ overfitting trong quá trình huấn luyện.

### 2.1.5. Mạng huấn luyện ResNet

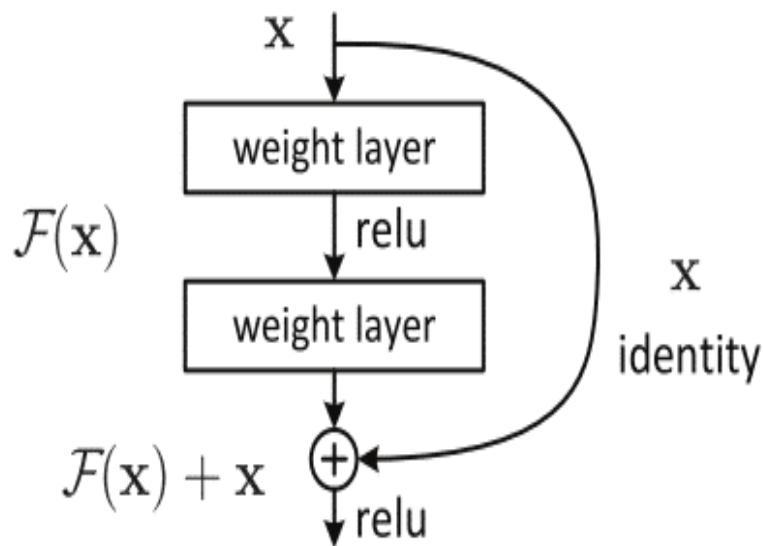
Mạng ResNet (Residual Network) là một kiến trúc mạng nơ-ron sâu đột phá trong lĩnh vực học sâu (deep learning), được giới thiệu bởi Kaiming He, Xiangyu Zhang, Shaoqing Ren và Jian Sun vào năm 2015. ResNet đã giành chiến thắng trong cuộc thi ILSVRC (ImageNet Large Scale Visual Recognition Challenge) năm 2015 và mở ra một cách tiếp cận mới cho việc xây dựng các mạng nơ-ron sâu với độ sâu lớn.

Vấn đề chính mà ResNet giải quyết là hiện tượng suy biến độ sâu (vanishing gradient) khi mạng có nhiều lớp. Suy biến độ sâu là hiện tượng mất mát thông tin và khó khăn trong việc học các thông tin trong các lớp sâu hơn. Điều này dẫn đến việc khó khăn trong việc xây dựng các mạng nơ-ron sâu có độ sâu lớn [7].



Hình 2.5. Mối tương quan giữa độ sâu và hiệu suất mạng

ResNet giải quyết vấn đề này bằng cách sử dụng các đơn vị residual (residual units) để xây dựng mạng. Mỗi đơn vị residual bao gồm một nhánh "đường tắt" (shortcut branch) và một nhánh "chính" (main branch). Nhánh chính thực hiện các phép tính phi tuyến (non-linear transformations), trong khi nhánh đường tắt chỉ truyền thông tin qua mạng mà không thay đổi.



Hình 2.6. ResNet sử dụng kết nối tắt xuyên qua một hay nhiều lớp

Cấu trúc đường tắt cho phép thông tin được truyền qua mạng một cách dễ dàng, cho dù mạng có nhiều lớp. Điều này giúp giải quyết vấn đề suy biến độ sâu và cho phép xây dựng các mạng nơ-ron sâu với độ sâu lớn hơn mà không gặp vấn đề hiệu suất.

Mạng ResNet có nhiều biến thể và độ sâu khác nhau, bao gồm ResNet-18, ResNet-34, ResNet-50, ResNet-101 và ResNet-152. Số trong tên mạng thể hiện số lượng lớp và đơn vị residual trong mạng.

ResNet đã được áp dụng rộng rãi và mang lại hiệu suất tốt trong nhiều nhiệm vụ như nhận dạng hình ảnh, phân đoạn hình ảnh, nhận dạng vật thể và phân loại hình ảnh. Các biến thể của ResNet cũng đã được sử dụng trong việc trích xuất đặc trưng (feature extraction) cho các mạng nơ-ron sâu khác.

Trong Mask R-CNN, ResNet thường được sử dụng làm mạng backbone. Điều này có nghĩa là ResNet được dùng để trích xuất các đặc trưng từ hình ảnh đầu vào, sau đó các đặc trưng này được sử dụng bởi các phần khác của Mask R-CNN để phát hiện đối tượng và phân đoạn thực thể. Sự kết hợp giữa ResNet và Mask R-CNN tạo nên một hệ thống mạnh mẽ cho việc phát hiện và phân đoạn đối tượng với độ chính xác cao.

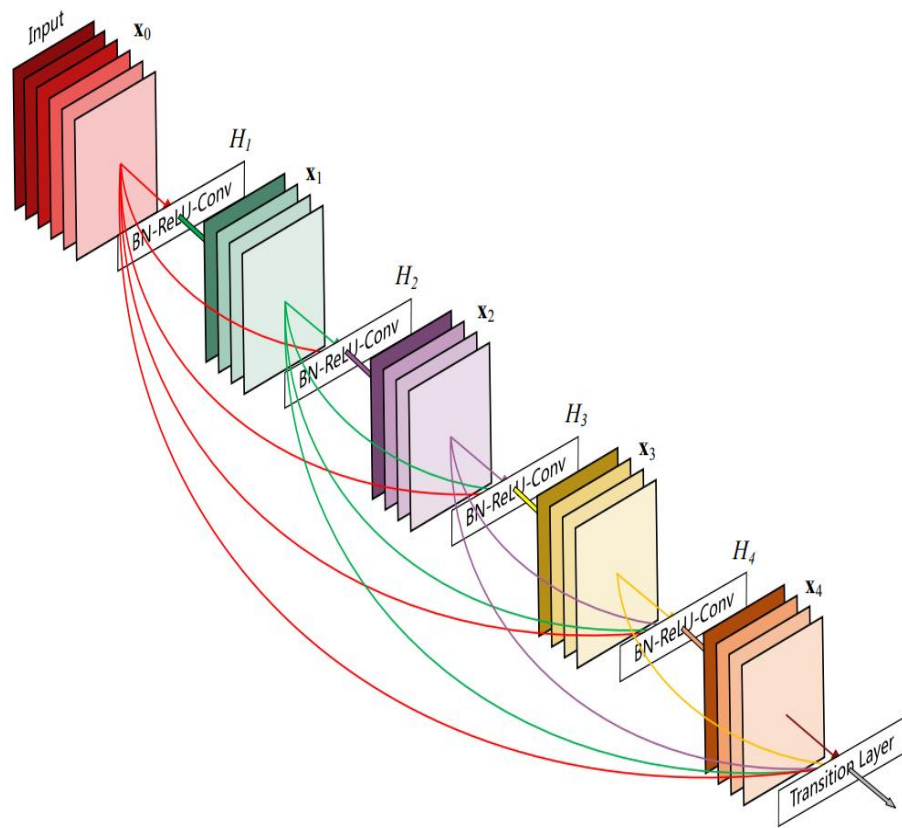
#### 2.1.6. Mạng huấn luyện DenseNet

Mạng DenseNet (Densely Connected Convolutional Networks) là một kiến trúc mạng nơ-ron sâu được giới thiệu bởi Gao Huang, Zhuang Liu, và Kilian Q. Weinberger từ Đại học Cornell vào năm 2017 [9]. DenseNet



là một trong những cải tiến đáng chú ý trong lĩnh vực thị giác máy tính và đã đạt được hiệu suất tốt trong nhiều nhiệm vụ nhận diện và phân loại hình ảnh.

Đặc điểm nổi bật của DenseNet là sự kết nối mật độ cao giữa các lớp trong mạng. Thay vì các lớp chỉ kết nối với các lớp kế tiếp, trong DenseNet, mỗi lớp nhận đầu vào từ tất cả các lớp trước đó trong mạng. Điều này tạo ra một kiến trúc mạng rất dày đặc, với sự kết nối chặt chẽ giữa các lớp, từ đó tăng cường khả năng trích xuất và tái sử dụng đặc trưng.



Hình 2.7. Ví dụ một DenseBlock gồm 5 layer

Cụ thể, trong DenseNet, các đầu vào từ các lớp trước được kết hợp lại thông qua việc nối chúng theo chiều sâu, thay vì cộng hoặc kết hợp theo cách thông thường. Điều này giúp mỗi lớp có thể học được các đặc trưng từ tất cả các lớp trước đó, từ đó cải thiện hiệu suất của mạng.

Vấn đề chính mà DenseNet giải quyết là vấn đề về hiệu suất và hiệu quả trong việc huấn luyện mạng nơ-ron sâu. Trong lĩnh vực thị giác máy tính, các mạng nơ-ron sâu thường gặp phải một số thách thức lớn như

overfitting, biến mất đạo hàm, và khó khăn trong việc học các đặc trưng cần thiết từ dữ liệu.

DenseNet giải quyết vấn đề này bằng cách sử dụng sự kết nối mật độ cao giữa các lớp trong mạng. Thay vì các lớp chỉ kết nối với các lớp kế tiếp, trong DenseNet, mỗi lớp nhận đầu vào từ tất cả các lớp trước đó trong mạng. Điều này tạo ra một kiến trúc mạng rất dày đặc, với sự kết nối chặt chẽ giữa các lớp, từ đó tăng cường khả năng trích xuất và tái sử dụng đặc trưng.

Sự kết nối mật độ cao trong DenseNet giúp mỗi lớp học được các đặc trưng từ tất cả các lớp trước đó, thay vì chỉ từ lớp trước đó như trong các mạng truyền thống. Điều này làm cho mạng trở nên hiệu quả hơn trong việc học các đặc trưng phức tạp từ dữ liệu, từ đó cải thiện độ chính xác và khả năng tổng quát hóa của mô hình.

Nhờ vào cấu trúc này, DenseNet giảm đáng kể số lượng tham số cần thiết trong mạng, giảm nguy cơ overfitting và tăng tốc độ huấn luyện. Đồng thời, nó cũng cải thiện đáng kể hiệu suất của mô hình trong các nhiệm vụ nhận diện và phân loại hình ảnh.

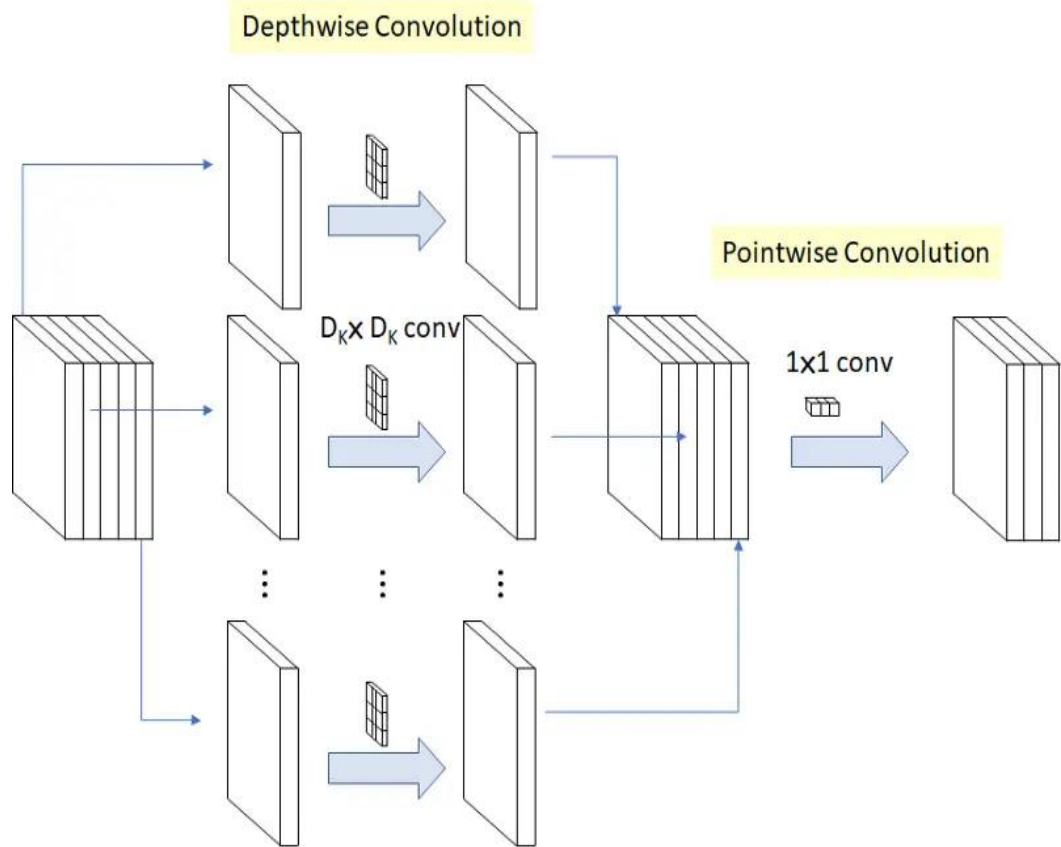
### **2.1.7. Mạng huấn luyện MobileNet**

MobileNet là một kiến trúc mạng nơ-ron sâu được thiết kế để đạt được hiệu suất cao trong các ứng dụng di động và thiết bị có tài nguyên tính toán hạn chế [5]. Nó đã được phát triển bởi nhóm nghiên cứu của Google vào năm 2017. Mục tiêu chính của MobileNet là tối ưu hóa kích thước mô hình và lượng tính toán cần thiết, nhưng vẫn giữ được độ chính xác cao.

Trong thế giới ngày nay, khi các thiết bị di động và IoT trở nên phổ biến và quan trọng hơn, việc triển khai các mô hình AI trên các thiết bị nhỏ gọn và có tài nguyên hạn chế đặt ra một thách thức lớn. MobileNet ra đời nhằm đáp ứng nhu cầu này, cung cấp một giải pháp hiệu quả cho việc triển khai AI trên các thiết bị di động và IoT.

Để đạt được mục tiêu của mình, MobileNet sử dụng một số kỹ thuật thiết kế đặc biệt. Trong đó, kỹ thuật "depthwise separable convolution" là một phần quan trọng. Thay vì sử dụng các lớp tích chập truyền thống, MobileNet chia mỗi lớp tích chập thành hai giai đoạn: depthwise convolution và pointwise convolution. Quá trình này giúp giảm đáng kể

số lượng tham số và lượng tính toán cần thiết trong mạng, tạo ra một mô hình nhỏ gọn và dễ triển khai trên các thiết bị có tài nguyên hạn chế.



Hình 2.8. Cấu trúc của một Depthwise Separable Convolution

Ngoài ra, MobileNet vẫn đảm bảo hiệu suất cao trong các nhiệm vụ như nhận diện đối tượng và phân loại hình ảnh. Mặc dù có kích thước nhỏ, MobileNet vẫn giữ được khả năng nhận diện và phân loại hình ảnh tốt, đảm bảo rằng mô hình vẫn có thể hoạt động hiệu quả trên các thiết bị di động và IoT.

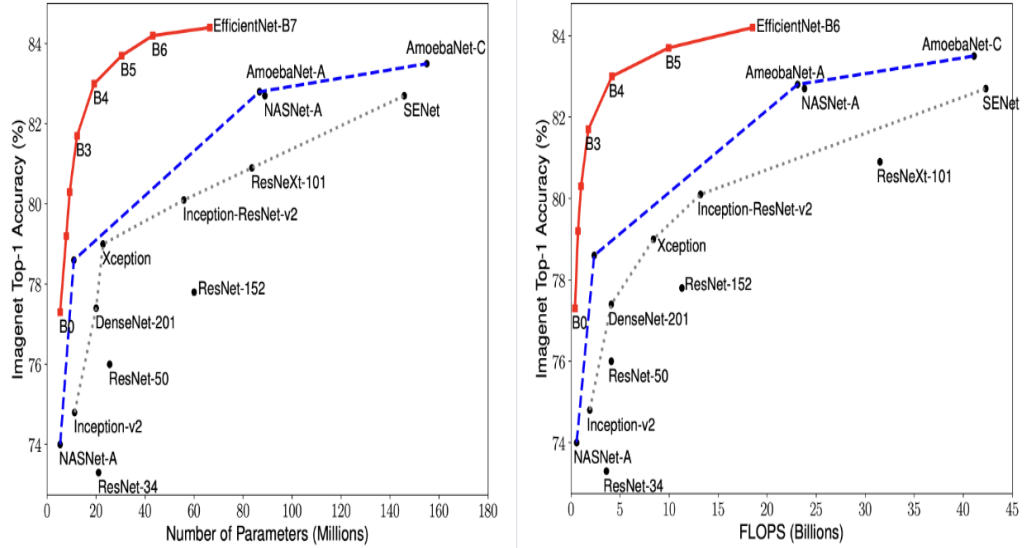
Nhờ vào sự kết hợp giữa kích thước nhỏ gọn và hiệu suất cao, MobileNet đã trở thành một trong những lựa chọn phổ biến cho việc triển khai các ứng dụng trí tuệ nhân tạo trên các thiết bị di động và IoT. Điều này mở ra cơ hội cho việc phát triển các ứng dụng mới và sáng tạo trong lĩnh vực AI, đồng thời giúp tăng cường trải nghiệm người dùng trên các thiết bị di động.

### 2.1.8. Mạng huấn luyện EfficientNet

EfficientNet là một kiến trúc mạng nơ-ron sâu (deep neural network) được thiết kế để đạt được hiệu suất cao và tối ưu trong việc huấn luyện và

triển khai trên các tác vụ liên quan đến thị giác máy tính [8]. Nó đã được giới thiệu bởi các nhà nghiên cứu từ Google Brain vào năm 2019.

Mục tiêu chính của EfficientNet là tối ưu hóa cả hai yếu tố quan trọng trong mạng nơ-ron sâu: hiệu suất và tài nguyên tính toán. Mạng này tận dụng một kỹ thuật gọi là "biểu đồ phân phối chiều sâu" (compound scaling), kết hợp việc mở rộng chiều sâu, chiều rộng và độ phân giải của mạng để đạt được hiệu suất cao hơn.



Hình 2.9. Mô hình mạng huấn luyện EfficientNet

Một trong những đặc điểm nổi bật của EfficientNet là việc sử dụng hệ số mở rộng (compound coefficient) để điều chỉnh kích thước của mạng. Hệ số mở rộng này phản ánh mức độ tăng kích thước của mạng theo các chiều sâu, rộng, và độ phân giải. Bằng cách thay đổi hệ số mở rộng, EfficientNet có thể điều chỉnh kích thước của mạng để phù hợp với yêu cầu tài nguyên tính toán và độ chính xác mong muốn.

Mạng EfficientNet đã đạt được hiệu suất tốt trên nhiều tác vụ liên quan đến thị giác máy tính, bao gồm phân loại hình ảnh, nhận dạng đối tượng và nhận dạng khuôn mặt. Nó đã trở thành một công cụ quan trọng trong lĩnh vực học sâu và có thể được sử dụng trong các ứng dụng thực tế như xử lý ảnh y khoa, xe tự hành, nhận dạng vật thể trong ảnh và nhiều lĩnh vực khác.

Mask R-CNN là một mô hình mạng đa nhiệm mạnh mẽ trong việc giải quyết bài toán nhận diện và phân đoạn đối tượng trên ảnh. Nó kết hợp cả khả năng phát hiện vùng quan tâm của đối tượng (object detection) và

phân đoạn các vùng đối tượng đó (instance segmentation). Điều này cho phép ta có được thông tin chi tiết về đối tượng trong ảnh, bao gồm cả vị trí, lớp và đường viền của chúng. Mạng Mask R-CNN sử dụng các kiến trúc mạng như ResNet và EfficientNet để trích xuất đặc trưng từ ảnh đầu vào. ResNet là một mạng nơ-ron tích chập sâu với khả năng học được các đặc trưng phức tạp từ ảnh, trong khi EfficientNet là một kiến trúc mạng hiệu quả với hiệu suất cao và độ phức tạp thấp. Sự kết hợp của các kiến trúc mạng này giúp Mask R-CNN có khả năng mạnh mẽ trong việc nhận diện và phân đoạn đối tượng.

Với những ưu điểm và khả năng mà Mask R-CNN mang lại, mô hình này được sử dụng trong khóa luận để giải quyết bài toán dò tìm đối tượng trong ảnh.

## 2.2. GIỚI THIỆU VỀ MẠNG MASK R-CNN

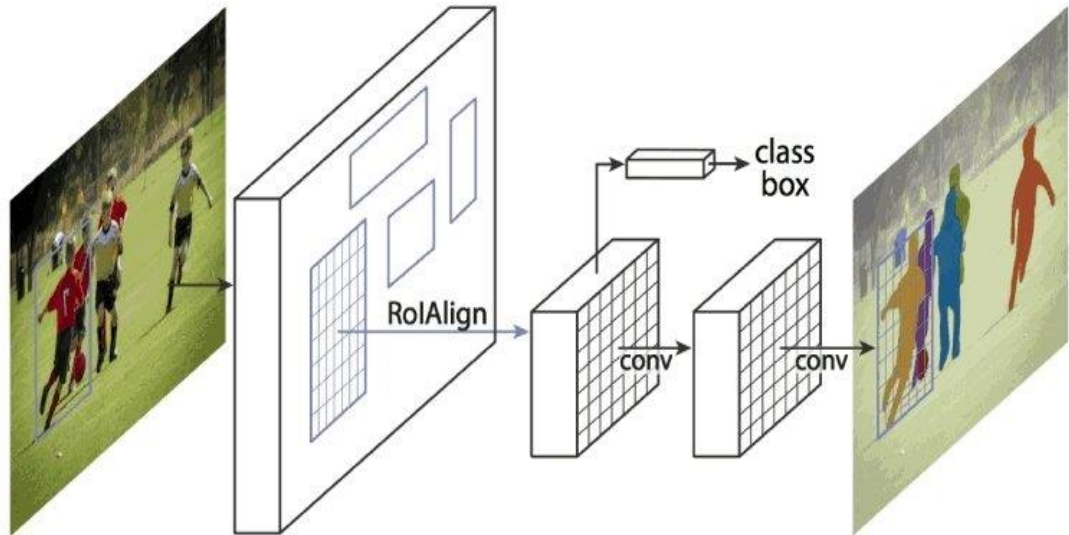
### 2.2.1. Giới thiệu chung về Mask R-CNN

Mask R-CNN là một mô hình học sâu hàng đầu trong lĩnh vực thị giác máy tính, đặc biệt là trong bài toán nhận dạng đối tượng và phân đoạn hình ảnh. Nó là một tiến bộ đáng chú ý trong lĩnh vực phát hiện đối tượng và phân đoạn hình ảnh, xây dựng trên nền tảng của mạng nơ-ron tích chập (CNN). Mô hình này không chỉ có khả năng phát hiện và phân loại đối tượng trong hình ảnh mà còn có khả năng tạo ra các mặt nạ (masks) chi tiết cho từng đối tượng đã được phát hiện.

Sự phát triển của Mask R-CNN phần lớn được truyền cảm hứng từ thách thức về việc xử lý hình ảnh và phân đoạn hình ảnh một cách chính xác trong các ứng dụng thực tế. Trong quá trình này, mạng nơ-ron tích chập đã chứng minh khả năng của mình trong việc trích xuất đặc trưng từ hình ảnh. Tuy nhiên, việc xác định ranh giới chi tiết của các đối tượng trong hình ảnh vẫn là một thách thức đối với các phương pháp truyền thống.

Mask R-CNN vượt qua thách thức này bằng cách kết hợp cả việc phát hiện vị trí của đối tượng và việc tạo ra mặt nạ cho từng đối tượng đó. Điều này cho phép mô hình không chỉ nhận diện và phân loại đối tượng mà còn cung cấp thông tin chi tiết về hình dạng của từng đối tượng. Nhờ đó, Mask R-CNN không chỉ cải thiện chính xác của việc phát hiện đối tượng mà còn mở ra nhiều ứng dụng mới trong lĩnh vực thị giác máy tính và xử lý hình

ảnh, bao gồm nhận dạng đối tượng trong ảnh y tế, giám sát an ninh, và xử lý hình ảnh tự động trong các ứng dụng công nghệ.



Hình 2.10. Mô hình phương pháp Mask R-CNN

### 2.2.2. Đặc trưng của mạng Mask R-CNN

- Kiến trúc nền tảng: Mask R-CNN kế thừa và mở rộng từ kiến trúc R-CNN và Faster R-CNN. Kiến trúc của nó bao gồm ba phần chính: một mô-đun phát hiện vùng (region proposal network - RPN) để đề xuất vùng chứa đối tượng, một mạng nơ-ron tích chập (CNN) để phân loại và điều chỉnh vị trí của các đối tượng, và một mạng nơ-ron thêm vào để tạo ra mặt nạ (mask) cho từng đối tượng.

- Phân đoạn đối tượng và phát hiện đối tượng: Mask R-CNN không chỉ có khả năng phát hiện và phân loại đối tượng mà còn có khả năng tạo ra mặt nạ chi tiết cho từng đối tượng đã được phát hiện. Điều này giúp cải thiện chính xác và độ chi tiết của việc phân đoạn hình ảnh.

- Sử dụng mạng nơ-ron tích chập: Mask R-CNN sử dụng mạng nơ-ron tích chập (CNN) để trích xuất đặc trưng từ hình ảnh. Điều này giúp mô hình tự động học và nhận diện các đặc trưng quan trọng trong hình ảnh, cũng như tạo ra các mặt nạ chi tiết cho các đối tượng.

- Cải tiến hiệu suất: Mask R-CNN đã cải thiện hiệu suất so với các phương pháp trước đó trong việc phát hiện và phân đoạn đối tượng trong hình ảnh. Nhờ vào việc kết hợp các mạng nơ-ron và kỹ thuật tiên tiến, mô hình này đạt được kết quả đáng kể trong các bài toán thị giác máy tính.

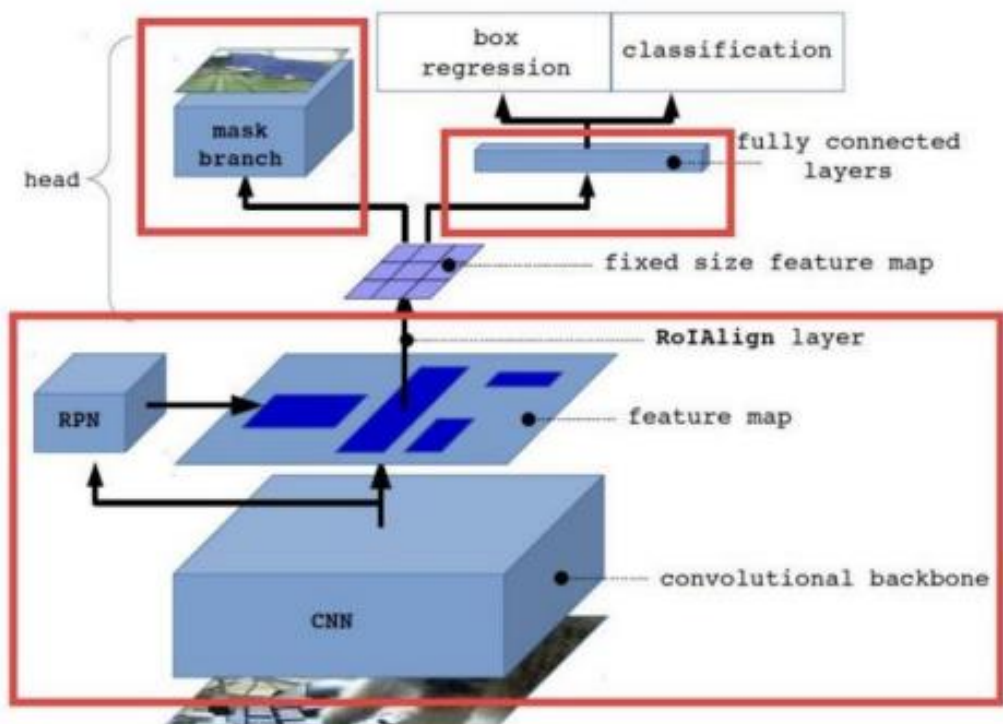
- Ứng dụng đa dạng: Mask R-CNN có thể được áp dụng trong nhiều lĩnh vực, bao gồm nhận dạng đối tượng trong hình ảnh y tế, giám sát an ninh, và xử lý hình ảnh tự động trong các ứng dụng công nghệ khác. Điều này làm cho nó trở thành một công cụ quan trọng trong các ứng dụng thực tế đòi hỏi tính linh hoạt và chính xác cao trong việc xử lý hình ảnh.

### 2.3. KIẾN TRÚC MẠNG MASK R-CNN

Mask R-CNN là một khung mô hình học sâu phổ biến trong lĩnh vực thị giác máy tính để thực hiện tác vụ phân đoạn theo đối tượng. Nó được phát triển dựa trên Faster R-CNN bằng cách thêm một nhánh để dự đoán mặt nạ đối tượng song song với nhánh hiện có để nhận dạng hộp giới hạn [4]. Trong khi Faster R-CNN, Fast R-CNN và R-CNN được sử dụng để phát hiện đối tượng bằng cách tạo ra các khung giới hạn.

Mask R-CNN bao gồm các thành phần sau: backbone (số xương) là một mạng cơ bản, mạng RPN (Region Proposal Network) để đề xuất các vùng quan tâm, lớp RoIAlign (Region of Interest alignment) để điều chỉnh các vùng quan tâm, một đầu phát hiện đối tượng bằng khung giới hạn và một đầu tạo mặt nạ. Các thành phần đầu tiên tạo nên mô hình Faster R-CNN.

Đây là cấu trúc tổng thể của Mask R-CNN:

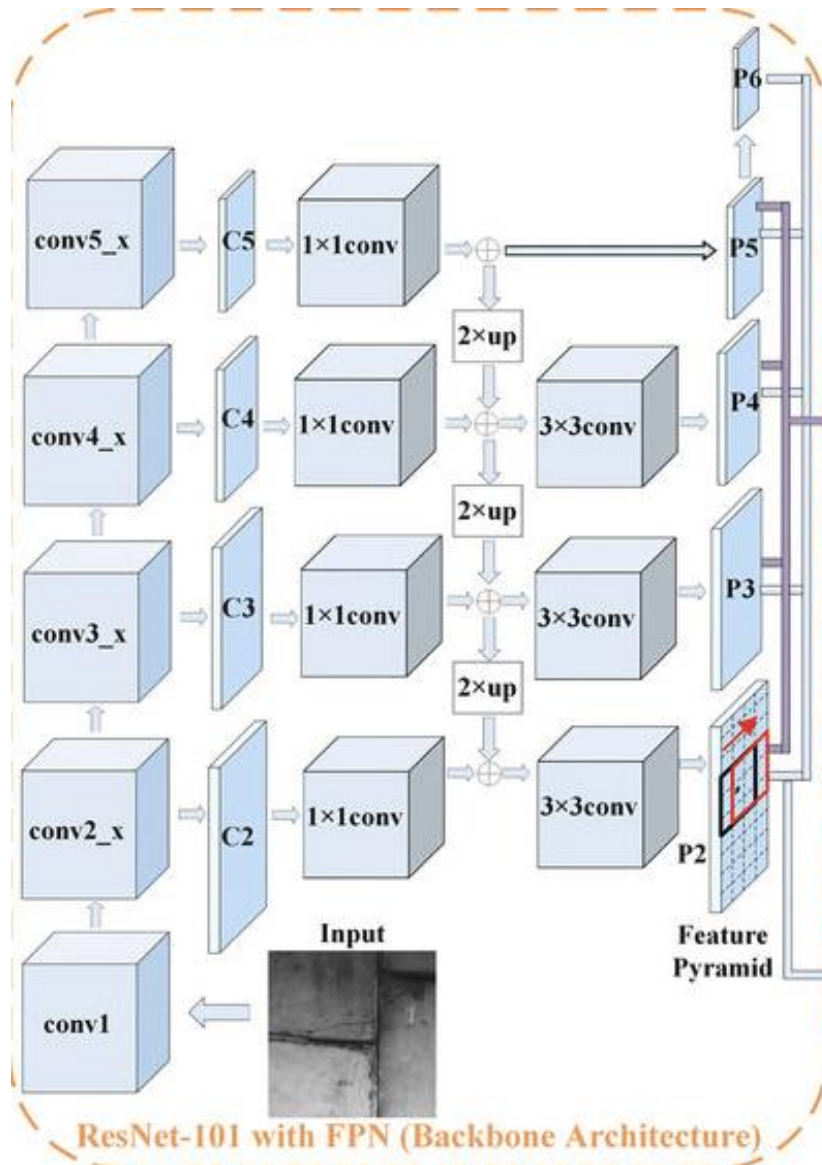


Hình 2.11. Kiến trúc cơ bản của một mạng Mask R-CNN



### 2.3.1. Backbone

Backbone là bộ trích xuất đặc trưng chính của Mask R-CNN. Thông thường, ta sử dụng mạng Residual (ResNet) với hoặc không có FPN (Feature Pyramid Network) cho phần này. Trong trường hợp đơn giản, ta sẽ sử dụng ResNet mà không có FPN làm backbone. Khi đưa một hình ảnh gốc vào backbone ResNet, dữ liệu sẽ đi qua nhiều khối bottleneck hồi quy và biến thành một bản đồ đặc trưng.



Hình 2.12. Cấu trúc Backbone

Như hình ảnh trên, ta có nhiều khối bottleneck hồi quy xếp chồng lên nhau, mỗi khối có cấu hình kênh d/d' khác nhau. Trong mỗi khối bottleneck, đầu vào sẽ đi qua hai đường dẫn. Một đường dẫn là một chuỗi các lớp tích chập và đường dẫn khác là một kết nối ngắn giống nhau. Sau đó, đầu ra từ cả hai đường dẫn sẽ được cộng lại phần tử-wise. Bằng cách

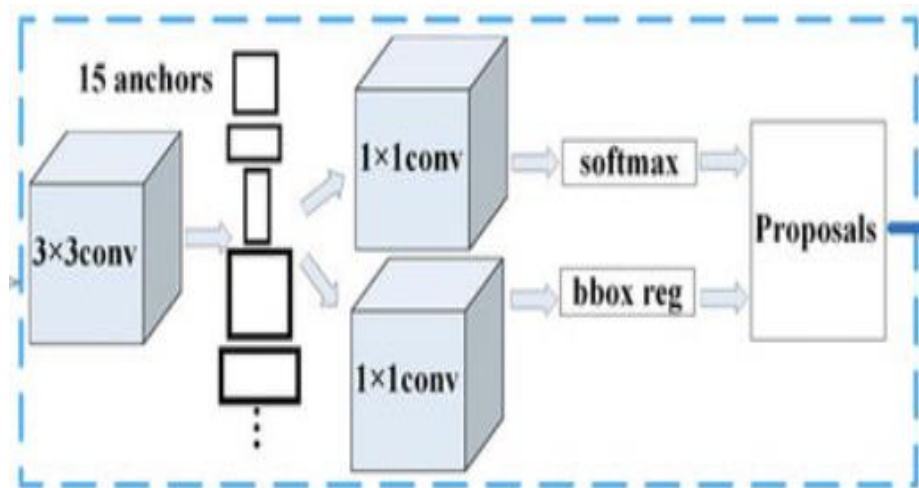


này, gradient có thể dễ dàng truyền qua các khối và mỗi khối có thể học một hàm định danh.

Bản đồ đặc trưng từ lớp tích chập cuối cùng của backbone chứa thông tin trừu tượng về hình ảnh, bao gồm các đối tượng khác nhau, lớp của chúng và các thuộc tính không gian của chúng. Sau đó, bản đồ đặc trưng này sẽ được đưa vào RPN (Region Proposal Network).

### 2.3.2. RPN

RPN là viết tắt của Region Proposal Network, hay mạng đề xuất vùng. Chức năng của RPN là quét qua bản đồ đặc trưng và đề xuất các vùng có thể chứa đối tượng (gọi là Region of Interest hay RoI) [6].



Hình 2.13. Cấu trúc RPN

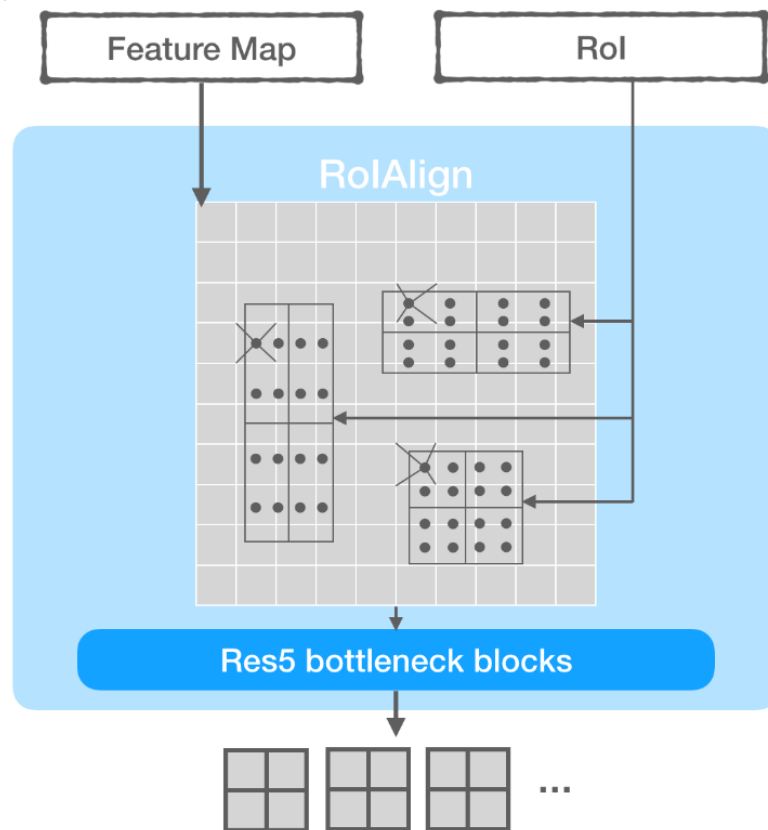
Cụ thể, RPN sử dụng một lớp tích chập để xử lý bản đồ đặc trưng và tạo ra một tensor có kênh  $c$ . Mỗi vector không gian của tensor này (có  $c$  kênh) được liên kết với một trung tâm anchor. Từ mỗi trung tâm anchor, ta tạo ra một tập hợp các hộp anchor có tỷ lệ và tỷ lệ khía cạnh khác nhau. Những hộp anchor này được phân bố đều trên toàn bộ hình ảnh và bao phủ nó hoàn toàn. Tiếp theo, ta sử dụng hai lớp tích chập  $1 \times 1$  song song để xử lý tensor có kênh  $c$ . Lớp đầu tiên là bộ phân loại nhị phân, dự đoán xem mỗi hộp anchor có chứa đối tượng hay không. Nó ánh xạ mỗi vector không gian của tensor (có  $c$  kênh) thành một vector có kênh  $k$ , biểu thị cho  $k$  hộp anchor có tỷ lệ và tỷ lệ khía cạnh khác nhau và chia sẻ cùng một trung tâm anchor. Lớp thứ hai là bộ dự đoán hộp giới hạn, dự đoán các sai số giữa hộp giới hạn thực sự của đối tượng và hộp anchor. Nó ánh xạ mỗi vector không gian của tensor (có  $c$  kênh) thành một vector có  $4k$  kênh. Đối với những hộp giới hạn chồng chéo mà có thể gợi ý cùng một đối tượng, ta

chọn những hộp có điểm tin cậy đối tượng cao nhất và loại bỏ những hộp khác đi. Quá trình này được gọi là quá trình không gian cực đại (Non-max suppression).

Sau quá trình RPN, ta thu được một tập hợp các vùng đề xuất (RoIs). Bước tiếp theo là tìm chính xác vị trí của mỗi vùng đề xuất trong bản đồ đặc trưng. Quá trình này được gọi là RoIAlign.

### 2.3.3. RoIAlign

RoIAlign (Region of Interest Alignment) là một quá trình quan trọng trong mô hình Mask R-CNN. Quá trình này có nhiệm vụ trích xuất các vector đặc trưng từ bản đồ đặc trưng dựa trên vùng đề xuất RoI mà RPN đã đề xuất, và chuyển chúng thành một tensor có kích thước cố định để tiếp tục xử lý.



Hình 2.14. Cấu trúc RoIAlign

Cách thức hoạt động của RoIAlign có thể được minh họa bằng hình ảnh trên. Quá trình này bắt đầu bằng việc căn chỉnh các vùng RoI với các vùng tương ứng trên bản đồ đặc trưng bằng cách tỷ lệ. Vì các vùng RoI có thể nằm ở các vị trí, tỷ lệ và tỷ lệ khía cạnh khác nhau, ta cần lấy mẫu các vùng căn chỉnh liên quan trên bản đồ đặc trưng để có được các vector đặc trưng có hình dạng đồng nhất.

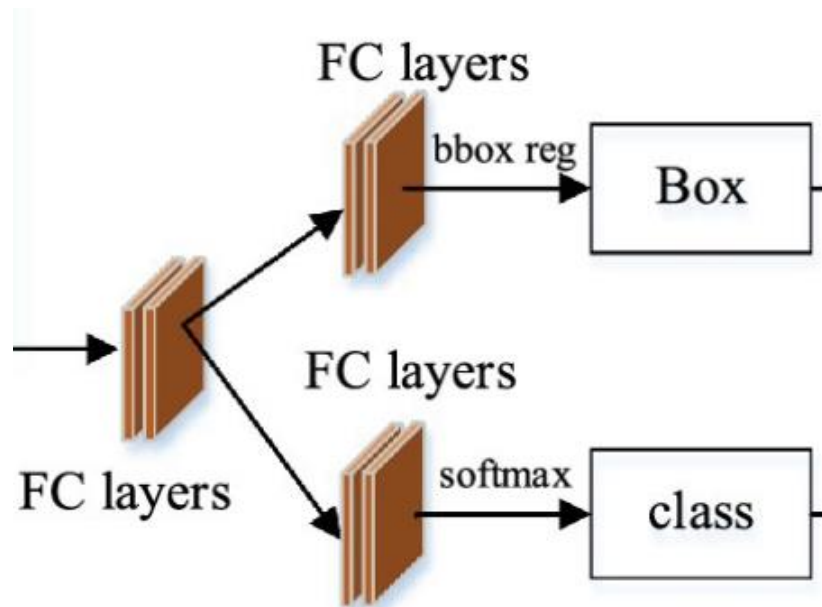
Để làm điều này, ta chia mỗi vùng RoI thành một lưới con có kích thước cố định. Trên mỗi ô lưới con, ta chọn một số điểm mẫu. Ta lấy mẫu các vector đặc trưng từ bản đồ đặc trưng xung quanh mỗi điểm và tính toán giá trị tương tự tuyến tính của chúng. Các vector tương tự này được gọi là vector điểm. Tiếp theo, ta tập hợp các vector điểm trong mỗi ô lưới con để tạo ra một bản đồ đặc trưng có kích thước cố định cho mỗi vùng RoI.

Sau khi có được các bản đồ đặc trưng cố định cho từng vùng RoI, ta áp dụng các khối bottleneck dư để tiếp tục trích xuất đặc trưng từ các vùng này. Kết quả là một bản đồ đặc trưng chi tiết hơn cho mỗi vùng RoI.

Bản đồ đặc trưng chi tiết này được sử dụng trong hai nhánh song song tiếp theo của mô hình Mask R-CNN: nhánh phát hiện đối tượng và nhánh tạo mặt nạ. Nhánh phát hiện đối tượng sẽ dự đoán lớp và hộp giới hạn cho từng vùng RoI, trong khi nhánh tạo mặt nạ sẽ tạo ra một mặt nạ chi tiết cho mỗi vùng RoI để phục vụ cho việc tạo mặt nạ đối tượng.

#### 2.3.4. Nhánh phát hiện đối tượng (Object detection branch)

Sau khi có được bản đồ đặc trưng riêng cho từng vùng quan tâm (RoI), ta có thể tiến hành dự đoán danh mục đối tượng và tạo ra một khung giới hạn chi tiết hơn cho từng vùng đó. Nhánh phát hiện đối tượng trong mô hình sử dụng một lớp kết nối đầy đủ (fully-connected layer) để ánh xạ các vector đặc trưng từ vùng RoI sang các lớp cuối cùng, bao gồm  $n$  lớp đối tượng khác nhau và  $4n$  tọa độ khung giới hạn cho mỗi đối tượng.



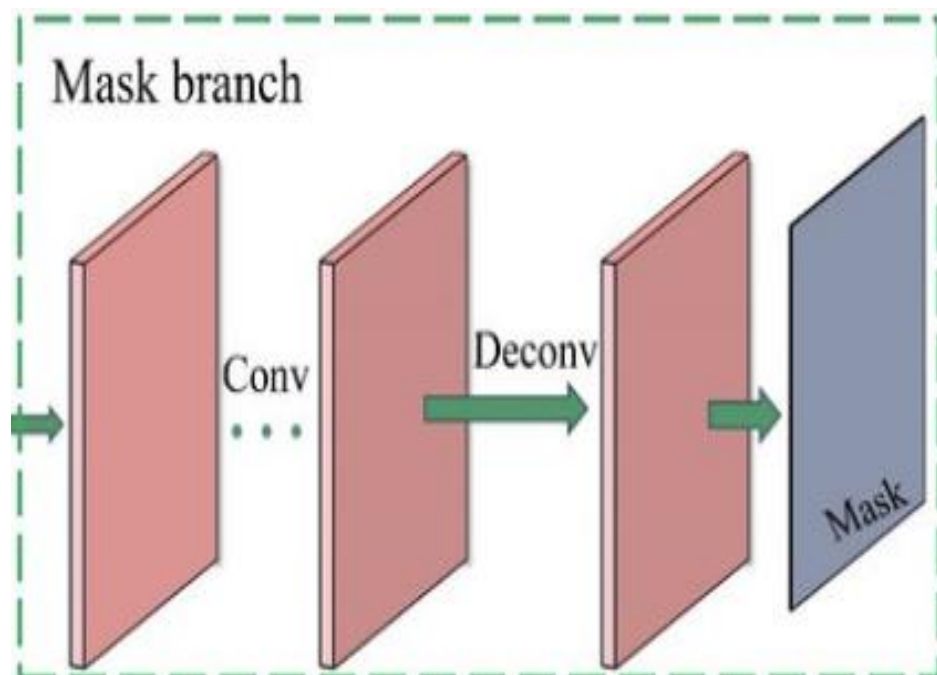
Hình 2.15. Cấu trúc của nhánh phát hiện đối tượng

Trong quá trình này, nhánh phát hiện đối tượng sẽ tính toán xác suất cho từng lớp đối tượng trên bản đồ đặc trưng và dự đoán lớp cuối cùng cho mỗi vùng RoI. Điều này giúp xác định danh mục đối tượng của từng vùng RoI trong ảnh. Đồng thời, nó cũng tính toán tọa độ khung giới hạn chi tiết cho mỗi vùng RoI, bao gồm tọa độ  $(x, y)$  của góc trên bên trái và  $(x, y)$  của góc dưới bên phải. Nhờ đó, ta có thể xác định vị trí chính xác của đối tượng trong ảnh.

Nhánh phát hiện đối tượng trong mô hình Mask R-CNN đóng vai trò quan trọng trong quá trình nhận diện và phân loại các đối tượng trong ảnh. Bằng cách kết hợp thông tin từ bản đồ đặc trưng và sử dụng lớp kết nối đầy đủ, mô hình có khả năng dự đoán danh mục đối tượng và tạo ra khung giới hạn chi tiết cho từng đối tượng trong ảnh một cách chính xác và tỉ mỉ. Điều này giúp cải thiện hiệu suất của quá trình phát hiện đối tượng và thúc đẩy sự hiểu biết về nội dung của ảnh.

### 2.3.5. Nhánh tạo mặt nạ (Mask generation branch):

Trong nhánh tạo mặt nạ (mask generation branch), ta sử dụng bản đồ đặc trưng RoI (Region of Interest) đã thu được và đưa chúng qua một chuỗi các lớp tích chập Transposed convolutional và Convolutional. Mục tiêu của nhánh này là tạo ra các mặt nạ nhị phân chi tiết cho từng lớp đối tượng.



Hình 2.16. Cấu trúc của nhánh tạo mặt nạ

Nhánh tạo mặt nạ được xây dựng dưới dạng một mạng tích chập đầy đủ, cho phép ta ánh xạ các bản đồ đặc trưng từ RoI sang các mặt nạ nhị phân tương ứng. Mỗi lớp đối tượng sẽ có một mặt nạ riêng, đại diện cho vùng của đối tượng đó trên ảnh.

Quá trình tạo mặt nạ bao gồm việc sử dụng lớp tích chập transposed convolutional để thay đổi kích thước bản đồ đặc trưng và tạo ra một bản đồ có kích thước tương tự như RoI ban đầu. Sau đó, ta sử dụng lớp tích chập convolutional để biến đổi bản đồ đặc trưng này thành một mặt nạ nhị phân, trong đó mỗi điểm ảnh chỉ có giá trị 0 (nền) hoặc 1 (đối tượng).

Sau khi có được các mặt nạ nhị phân cho từng lớp đối tượng, ta chọn mặt nạ tương ứng dựa trên dự đoán lớp trong nhánh phát hiện đối tượng. Điều này đảm bảo rằng mỗi điểm ảnh trong mặt nạ sẽ được gán cho một lớp đối tượng duy nhất và tránh sự cạnh tranh giữa các lớp khác nhau.

Qua quá trình này, việc dự đoán mặt nạ cho từng điểm ảnh được thực hiện một cách độc lập và chi tiết, đảm bảo tính chính xác trong việc xác định vị trí và phân loại đối tượng trong ảnh.

## 2.4. KẾT LUẬN CHƯƠNG 2

Trong chương 2, đã tìm hiểu về mạng Mask R-CNN cùng với một số mô hình mạng huấn luyện phổ biến khác. Từ sự tiên tiến của Mask R-CNN, cho thấy cách mà mô hình này có thể chính xác phát hiện và phân loại đối tượng trong ảnh, đồng thời còn có khả năng tạo ra các mặt nạ (masks) chính xác cho các vùng của đối tượng. Bên cạnh đó, việc nắm bắt các mô hình mạng huấn luyện phổ biến khác cũng giúp hiểu rõ hơn về sự phát triển của các phương pháp nhận diện đối tượng trong thời gian gần đây.

Từ việc nghiên cứu và so sánh các mô hình này, có thể nhận thấy rằng việc lựa chọn mô hình phù hợp phụ thuộc vào nhiều yếu tố như độ chính xác, tốc độ xử lý và yêu cầu về tài nguyên tính toán. Bằng cách hiểu rõ các ưu nhược điểm của từng mô hình, chúng ta có thể áp dụng chúng một cách hiệu quả vào các ứng dụng thực tiễn.

Chương 2 đã cung cấp cái nhìn tổng quan về Mask R-CNN và các mô hình mạng huấn luyện phổ biến khác, giúp ta hiểu rõ hơn về cách thức hoạt động và ứng dụng của chúng trong lĩnh vực nhận diện đối tượng.

Điều này sẽ là cơ sở quan trọng để tiếp tục nghiên cứu và phát triển các phương pháp nhận diện đối tượng hiệu quả trong tương lai.

## CHƯƠNG 3: ỨNG DỤNG MÔ HÌNH MASK R-CNN ĐÒ TÌM ĐỐI TƯỢNG TRONG ẢNH

### 3.1. MÔI TRƯỜNG LÀM VIỆC

#### 3.1.1. Giới thiệu chung về Anaconda

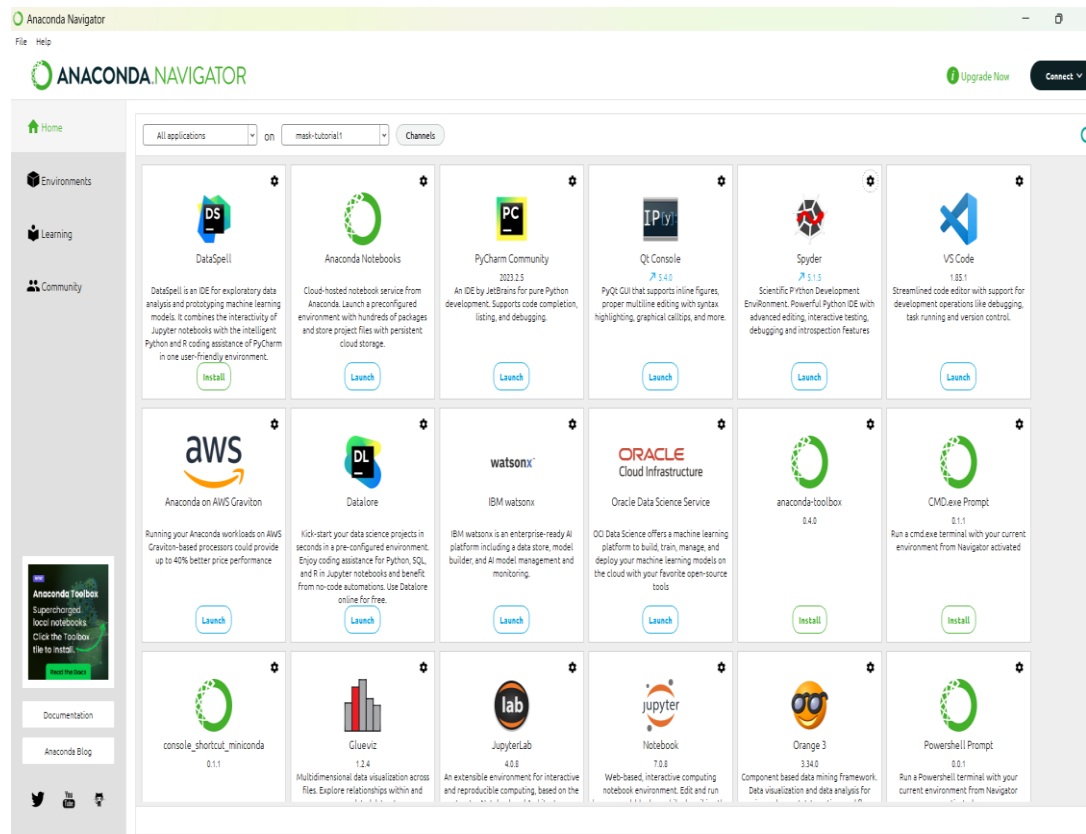
Anaconda là một bản phân phối các ngôn ngữ lập trình Python và R cho tính toán khoa học, nhằm mục đích đơn giản hóa việc quản lý và triển khai gói. Được phát triển bởi công ty Anaconda, Inc., nền tảng này cung cấp một bộ công cụ mạnh mẽ cho các nhà phân tích dữ liệu, nhà khoa học dữ liệu và nhà phát triển phần mềm để làm việc với dữ liệu lớn và phân tích số liệu [10].



Hình 3.1. Logo Anaconda

Anaconda đi kèm với một bộ công cụ rộng lớn của Python và các gói khoa học dữ liệu phổ biến như NumPy, Pandas, Matplotlib, Scikit-learn, và một loạt các gói khác, đã được tối ưu hóa để hoạt động tốt trên nền tảng này. Nó cũng cung cấp một cách tiện lợi để quản lý môi trường Python, giúp người dùng tạo và quản lý các môi trường ảo để phát triển và triển khai các dự án Python một cách hiệu quả.

Một trong những tính năng nổi bật của Anaconda là Anaconda Navigator, một ứng dụng desktop cung cấp giao diện đồ họa để quản lý các môi trường và gói Python, cũng như các công cụ và ứng dụng hỗ trợ như Jupyter Notebook, Spyder, và JupyterLab.



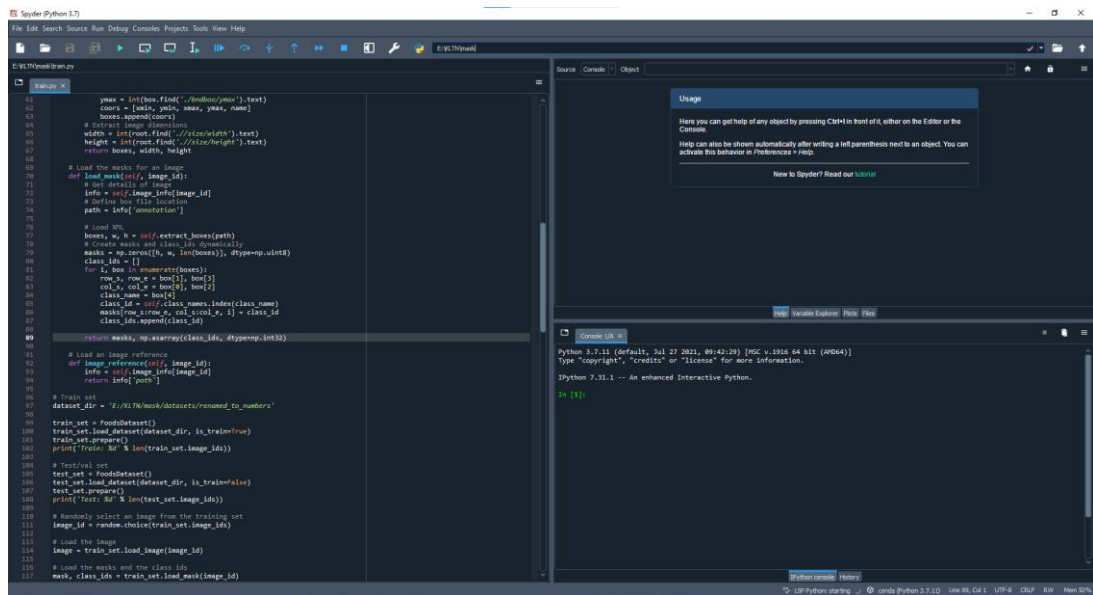
Hình 3.2. Giao diện Anaconda

Với tính năng linh hoạt, đa nền tảng và khả năng tích hợp mạnh mẽ, Anaconda đã trở thành một công cụ quan trọng trong lĩnh vực khoa học máy tính (khoa học dữ liệu, ứng dụng học máy, xử lý dữ liệu lớn, phân tích dự đoán số liệu, ...).

### 3.1.2. Giới thiệu chung về Spyder

Spyder (Scientific Python Development Environment) là một môi trường phát triển tích hợp (Integrated Development Environment - IDE) dành riêng cho việc phát triển ứng dụng sử dụng Python với mục tiêu chính trong lĩnh vực khoa học dữ liệu, tính toán khoa học và phân tích dữ liệu. Nó cung cấp một giao diện người dùng dễ sử dụng và tích hợp các công cụ và tính năng quan trọng để hỗ trợ quá trình phân tích dữ liệu, phân tích số liệu và phát triển các ứng dụng Python [11].





Hình 3.3. Giao diện phần mềm Spyder

Một trong những đặc điểm nổi bật của Spyder là giao diện người dùng thân thiện và dễ sử dụng. Giao diện của nó được tổ chức một cách logic và cung cấp các công cụ tiện ích như trình soạn thảo mã, trình trợ giúp, trình quản lý biến và mô-đun, cũng như trình duyệt biến số và dữ liệu, giúp người dùng dễ dàng tương tác và thực hiện các tác vụ phức tạp một cách hiệu quả.

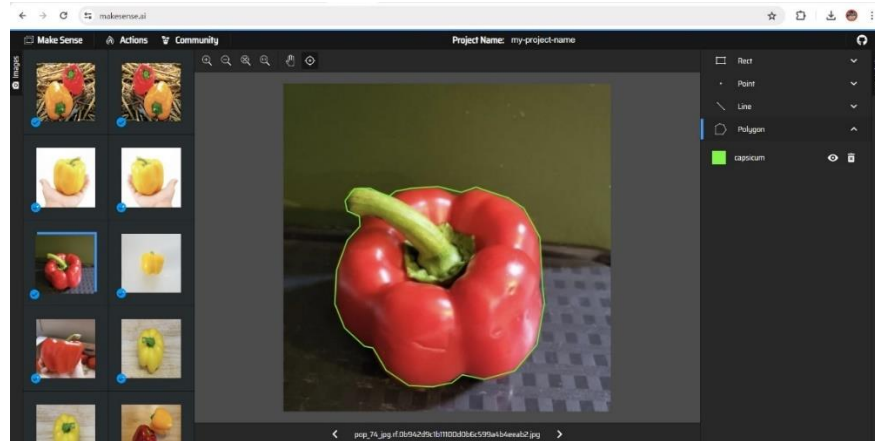
Spyder tích hợp sâu với một số thư viện Python phổ biến trong lĩnh vực khoa học dữ liệu như NumPy, pandas và Matplotlib. Điều này cho phép người dùng thực hiện phân tích dữ liệu, trực quan hóa dữ liệu và phát triển các mô hình máy học một cách thuận tiện và linh hoạt.

Ngoài ra, Spyder cũng tích hợp với IPython, một phiên bản nâng cao của Python interpreter, cung cấp các tính năng như tự hoàn thiện mã, đánh dấu cú pháp, và khả năng hiển thị kết quả trực tiếp trong cửa sổ console.

Với những tính năng và công cụ mạnh mẽ của mình, Spyder đã trở thành một lựa chọn phổ biến trong cộng đồng khoa học dữ liệu và phân tích dữ liệu. Nó được sử dụng rộng rãi trong các lĩnh vực như nghiên cứu khoa học, phân tích dữ liệu, và phát triển các ứng dụng Python phức tạp.

### 3.2. CHUẨN BỊ DỮ LIỆU (DATASET)

Để huấn luyện mô hình nhận diện đối tượng, cần một tập dữ liệu chứa các hình ảnh kèm theo các tệp JSON định dạng, trong đó mỗi đối tượng trên hình được định danh bằng một nhãn cụ thể. Ta sẽ sử dụng <https://www.makesense.ai/> để tạo tập dữ liệu gắn nhãn.

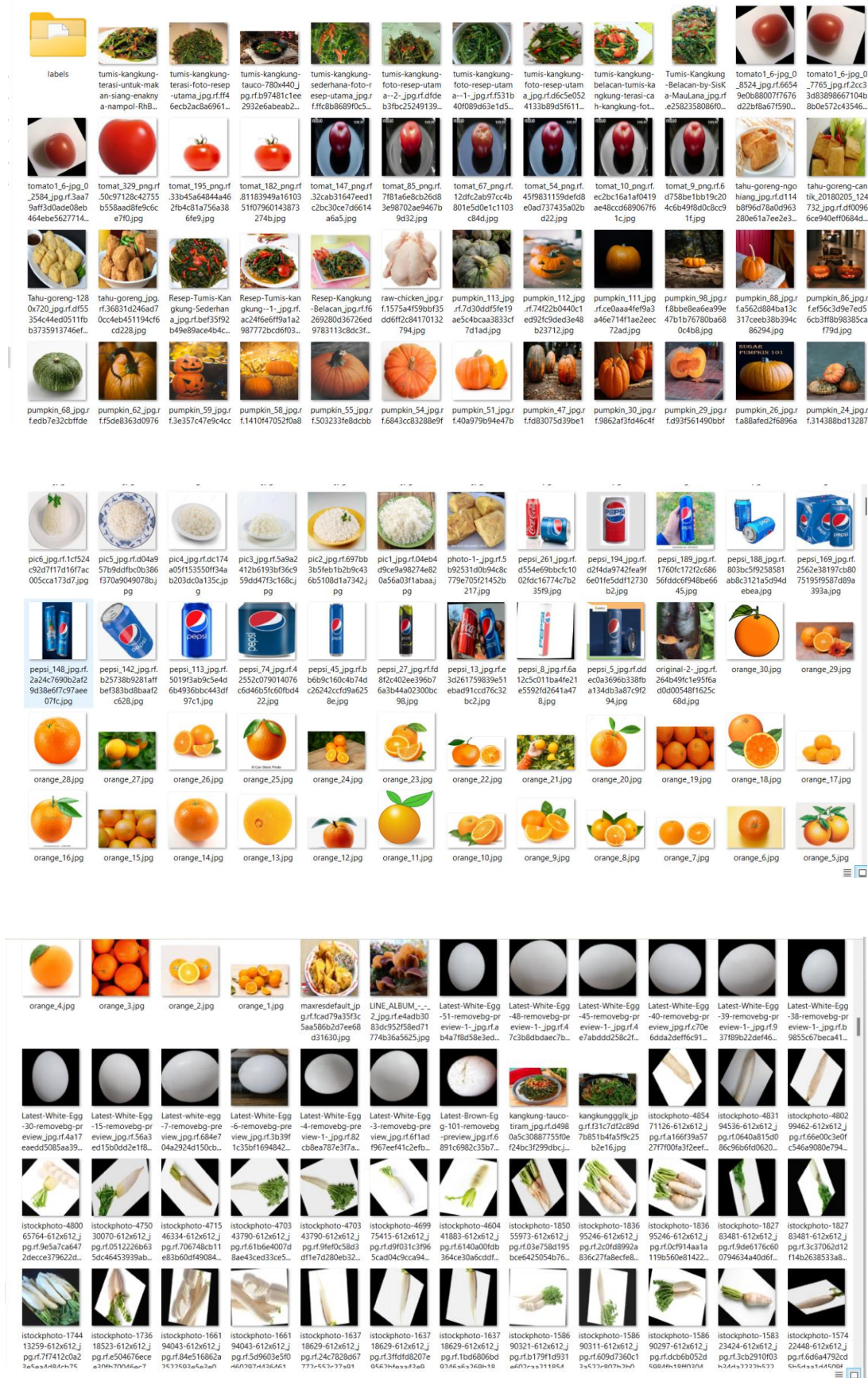


Hình 3.4. Dữ liệu hình ảnh và gán nhãn




Hình 3.5. Kết quả thu được sau khi gán nhãn

Sau khi gán nhãn xong, ta sẽ lưu ảnh và nhãn chú thích ở thư mục có tên là “train”. Trong thư mục “train” ta sẽ tạo thư mục “labels” để lưu tệp JSON riêng để phục vụ cho huấn luyện mô hình trong tương lai và một file “list.txt” chứa tên data chuẩn bị huấn luyện. Ở đây nhóm tạo ra dataset về các loại thực phẩm, món ăn phổ biến ở Việt Nam.

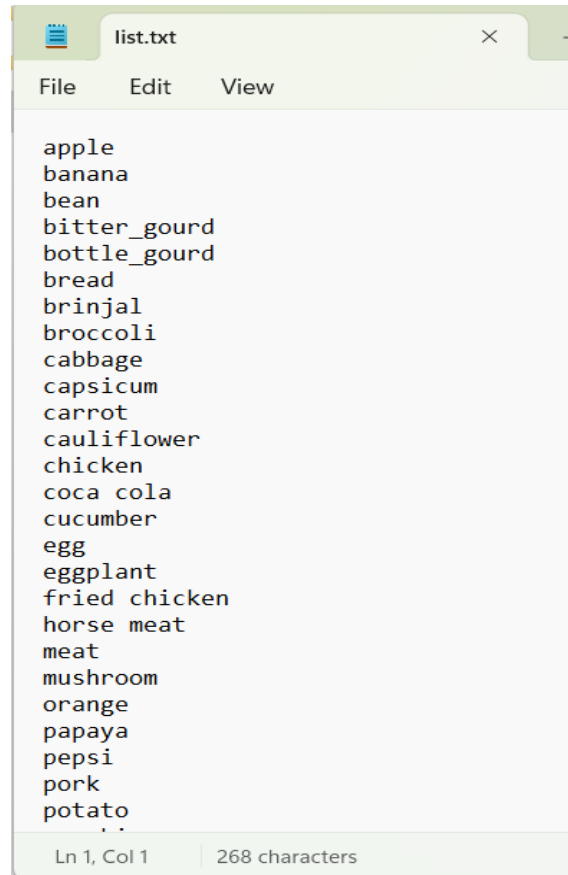


Hình 3.6. Dataset



Name	Date modified	Type	Size
 labels_food.json	2/15/2024 2:26 PM	Adobe After Effect...	2,699 KB

Hình 3.7. Thư mục chứa file JSON gắn nhãn



Hình 3.8. File list.txt chứa tên data

### 3.3. MODEL TRAINNING

#### Bước 1: Import thư viện

Trước tiên ta cần import các thư viện cần thiết.

```
import os
import sys
import json
import random
import numpy as np
import skimage.draw
from os import listdir
from mrcnn.visualize import display_instances
from mrcnn.utils import extract_bboxes
from mrcnn.utils import Dataset
from mrcnn.config import Config
from mrcnn.model import MaskRCNN
from keras.optimizers import Adam
from xml.etree import ElementTree
import matplotlib.pyplot as plt
from keras.callbacks import ModelCheckpoint
from mrcnn import utils
```

Hình 3.9. Thiết lập các thư viện

## Bước 2: Xác định lớp tập dữ liệu

Trong bước này ta sẽ xác định và định nghĩa một lớp tập dữ liệu tùy chỉnh để có thể tải và xử lý dữ liệu một cách hiệu quả.

```
class CustomDataset(utils.Dataset):
    def load_custom(self, dataset_dir, subset):
        """Load a subset of the custom dataset.
        dataset_dir: Root directory of the dataset.
        subset: Subset to load: train or val
        """
        # Add classes according to the number of classes required to detect
        class_labels = self.load_class_labels(os.path.join(dataset_dir, "list.txt"))
        for idx, label in enumerate(class_labels, start=1):
            self.add_class("custom", idx, label)

        # Train or validation dataset?
        assert subset in ["train", "val"]
        dataset_dir = os.path.join(dataset_dir, subset)

        # Load annotations
        annotations = json.load(open(os.path.join(dataset_dir, "labels/labels_food.json")))
        annotations = list(annotations.values()) # don't need the dict keys

        # Add images
        for a in annotations:
            polygons = [r['shape_attributes'] for r in a['regions'].values()]

            custom = [s['region_attributes'] for s in a['regions'].values()]
            num_ids = [class_labels.index(n['label']) + 1 for n in custom if 'label' in n]

            # Load image
            image_path = os.path.join(dataset_dir, a['filename'])
            image = skimage.io.imread(image_path)
            height, width = image.shape[:2]

            self.add_image(
                "custom",
                image_id=a['filename'],
                path=image_path,
                width=width, height=height,
                polygons=polygons,
                num_ids=num_ids
            )

    def load_class_labels(self, path):
        """Load class labels from a file."""
        with open(path, 'r') as file:
            class_labels = [line.strip() for line in file if line.strip()]
        return class_labels
```

Hình 3.10. Tải tập dữ liệu từ thư mục chỉ định

```
def load_mask(self, image_id):
    """Generate instance masks for an image."""
    image_info = self.image_info[image_id]
    if image_info["source"] != "custom":
        return super().load_mask(image_id)
    num_ids = image_info['num_ids']

    info = self.image_info[image_id]
    mask = np.zeros([info["height"], info["width"], len(info["polygons"])], dtype=np.uint8)
    for i, p in enumerate(info["polygons"]):
        rr, cc = skimage.draw.polygon(p['all_points_y'], p['all_points_x'])
        mask[rr, cc, i] = 1

    num_ids = np.array(num_ids, dtype=np.int32)
    return mask, num_ids
```

Hình 3.11. Tải các mặt nạ (masks) cho hình ảnh

```
def image_reference(self, image_id):
    """Return the path of the image."""
    info = self.image_info[image_id]
    if info["source"] == "custom":
        return info["path"]
    else:
        super().image_reference(image_id)
```

Hình 3.12. Trả về đường dẫn của một hình ảnh cụ thể trong tập dữ liệu

### Bước 3: Chuẩn bị tập dữ liệu và tiến hành Training

```
# Initialize dataset
dataset_train = CustomDataset()
dataset_train.load_custom("datasets/", "train")
dataset_train.prepare()
print('Train: %d' % len(dataset_train.image_ids))
```

Hình 3.13. Các bước chuẩn bị tập dữ liệu

(\*) Ta thực hiện một quá trình lấy mẫu ngẫu nhiên một hình ảnh từ tập huấn luyện và hiển thị nó cùng với các đối tượng được chú thích trên đó:

```
image_id = random.choice(dataset_train.image_ids)
image = dataset_train.load_image(image_id)
mask, class_ids = dataset_train.load_mask(image_id)

# Display image with masks and bounding boxes
bbox = extract_bboxes(mask)
display_instances(image, bbox, mask, class_ids, dataset_train.class_names)
print("Number of bounding boxes:", len(bbox))
print("Shape of masks array:", mask.shape)
print("Number of class IDs:", len(class_ids))
```



Hình 3.14. Lấy mẫu ngẫu nhiên từ một hình ảnh từ tập huấn luyện

(\*) Cấu hình Config cho mô hình:

```
class FoodsConfig(Config):
    NAME = "foods_cfg"
    NUM_CLASSES = 1 + 32
    STEPS_PER_EPOCH = 507
    LEARNING_RATE = 0.001
    ADAM_DECAY = 0.0
    ADAM_EPSILON = 1e-7

    def __init__(self):
        super().__init__()
        self.OPTIMIZER = Adam(lr=self.LEARNING_RATE, decay=self.ADM_DECAY, epsilon=self.ADM_EPSILON)

# Prepare config
config = FoodsConfig()
config.display()
```

Hình 3.15. Cấu hình Config

```
Configurations:
ADAM_DECAY                0.0
ADAM_EPSILON              1e-07
BACKBONE                  resnet101
BACKBONE_STRIDES          [4, 8, 16, 32, 64]
BATCH_SIZE                2
BBOX_STD_DEV              [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE    None
DETECTION_MAX_INSTANCES   100
DETECTION_MIN_CONFIDENCE  0.7
DETECTION_NMS_THRESHOLD   0.3
FPN_CLASSIF_FC_LAYERS_SIZE 1024
GPU_COUNT                 1
GRADIENT_CLIP_NORM        5.0
IMAGES_PER_GPU            2
IMAGE_CHANNEL_COUNT        3
IMAGE_MAX_DIM             1024
IMAGE_META_SIZE           45
IMAGE_MIN_DIM             800
IMAGE_MIN_SCALE           0
IMAGE_RESIZE_MODE         square
IMAGE_SHAPE               [1024 1024   3]
LEARNING_MOMENTUM          0.9
LEARNING_RATE             0.001
LOSS_WEIGHTS              {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0,
                           'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE            14
MASK_SHAPE                [28, 28]
MAX_GT_INSTANCES          100
MEAN_PIXEL                 [123.7 116.8 103.9]
MINI_MASK_SHAPE            (56, 56)
NAME                      foods_cfg
NUM_CLASSES                33
OPTIMIZER                  <keras.optimizers.Adam object at 0x000002A880015208>
POOL_SIZE                 7
POST_NMS_ROIS_INFERENCE   1000
POST_NMS_ROIS_TRAINING    2000
PRE_NMS_LIMIT              6000
ROI_POSITIVE_RATIO        0.33
RPN_ANCHOR_RATIOS         [0.5, 1, 2]
RPN_ANCHOR_SCALES         (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE         1
RPN_BBOX_STD_DEV          [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD         0.7
RPN_TRAIN_ANCHORS_PER_IMAGE 256
STEPS_PER_EPOCH           507
TOP_DOWN_PYRAMID_SIZE     256
```

Hình 3.16. Thông số của model

(\*) Xác định thư mục lưu trữ logs và mô hình, định nghĩa một mô hình Mask R-CNN trong chế độ huấn luyện và tải trọng số từ mô hình đã được

huấn luyện trước trên tập dữ liệu MSCOCO, sau đó huấn luyện mô hình trên tập dữ liệu đã chuẩn bị:

```
ROOT_DIR = os.path.abspath("../")
DEFAULT_LOGS_DIR = os.path.join(ROOT_DIR, "logs_model")
COCO_WEIGHTS_PATH = os.path.join(ROOT_DIR, "weights/mask_rcnn_coco.h5")

model = MaskRCNN(mode='training', model_dir=DEFAULT_LOGS_DIR, config=config)
model.load_weights(COCO_WEIGHTS_PATH, by_name=True, exclude=["mrcnn_class_logits", "mrcnn_bbox_fc", "mrcnn_bbox", "mrcnn_mask"])

# define checkpoint callback
checkpoint_path = os.path.join(ROOT_DIR, "best.hdf5")
checkpoint = ModelCheckpoint(checkpoint_path, monitor='val_loss', save_best_only=True, mode='auto')
callback_list = [checkpoint]
print(checkpoint_path)

# Train weights (output layers or 'heads')
model.train(dataset_train, dataset_train, learning_rate=config.LEARNING_RATE, epochs=20, layers='heads')

history = model.keras_model.history.history
print(history)
```

Hình 3.17. Xác định thư mục lưu trữ, mô hình và tiến hành training

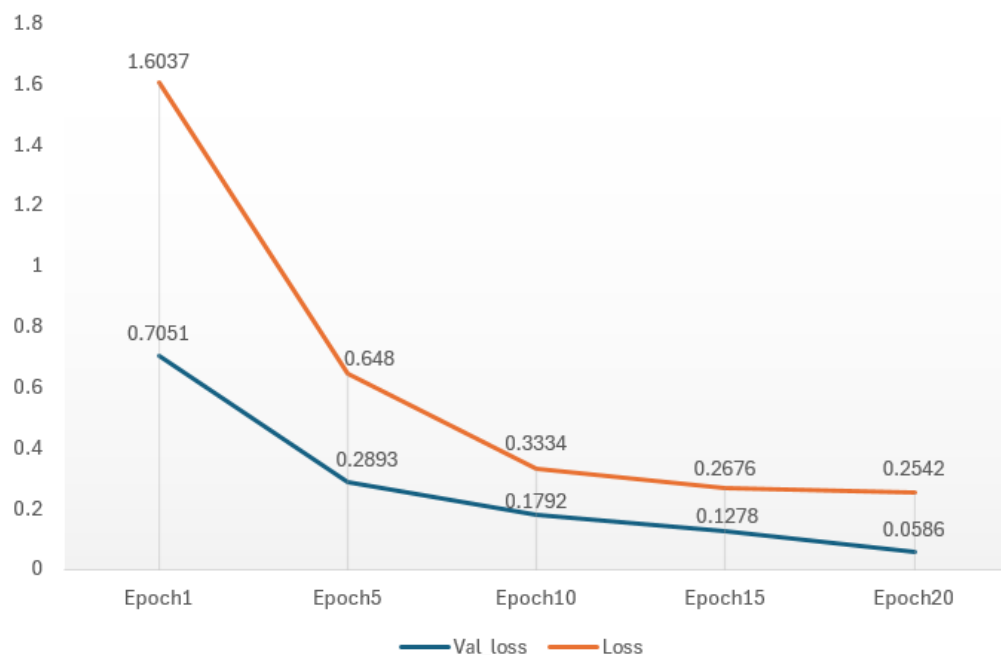
Cuối cùng, ta tiến hành huấn luyện mô hình.

```
507/507 [=====] - 19604s 39s/step - loss: 1.6037 - val_loss: 0.7051
Epoch 2/20
507/507 [=====] - 21251s 42s/step - loss: 0.9648 - val_loss: 0.5731
Epoch 3/20
507/507 [=====] - 19432s 38s/step - loss: 0.8070 - val_loss: 0.6604
Epoch 4/20
507/507 [=====] - 19187s 38s/step - loss: 0.7035 - val_loss: 0.3155
Epoch 5/20
507/507 [=====] - 26203s 52s/step - loss: 0.6480 - val_loss: 0.2893
Epoch 6/20
507/507 [=====] - 19852s 39s/step - loss: 0.5891 - val_loss: 0.9712
Epoch 7/20
144/507 [=====>.....] - ETA: 4:06:08 - loss: 0.5328
```

Hình 3.18. Huấn luyện mô hình

Kết quả sau khi train.

Bảng 3.1. Kết quả Loss và Val\_loss qua 20 epoch

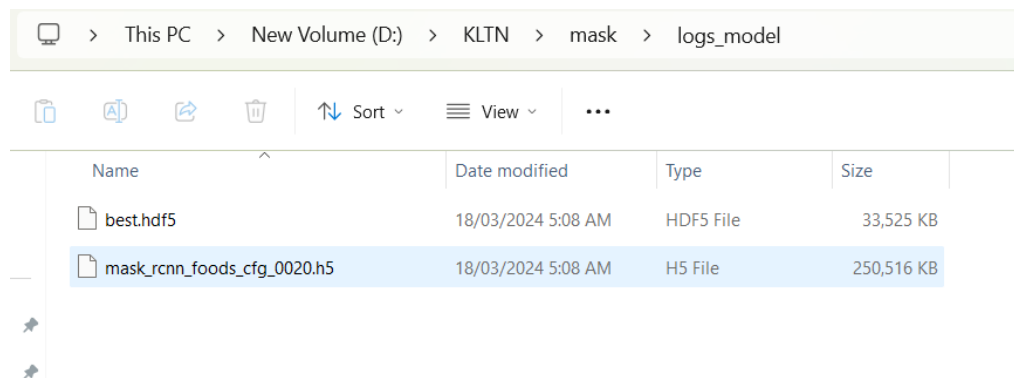


Bảng 3.1 cho thấy độ sai lệch trên tập dữ liệu train (loss) và trên tập dữ liệu validation qua từng epoch.



Trong quá trình huấn luyện, ta chia dữ liệu thành hai phần chính: tập dữ liệu huấn luyện và tập dữ liệu validation. Tập dữ liệu huấn luyện được sử dụng để cập nhật trọng số của mô hình trong quá trình huấn luyện, trong khi tập dữ liệu validation được sử dụng để đánh giá hiệu suất của mô hình và tránh việc overfitting.

Độ sai lệch (loss) là một thước đo để đo lường sự chênh lệch giữa giá trị thực tế và giá trị dự đoán của mô hình. Thông qua việc theo dõi độ sai lệch trên cả hai tập dữ liệu này, ta có thể đánh giá được hiệu suất của mô hình trên dữ liệu huấn luyện (train) cũng như khả năng tổng quát hóa của mô hình trên dữ liệu mà nó chưa từng gặp (validation). Nếu độ sai lệch trên tập dữ liệu huấn luyện giảm dần trong quá trình huấn luyện nhưng trên tập dữ liệu validation lại tăng lên, điều này có thể chỉ ra mô hình đang overfitting - tức là mô hình đã học "quá nhớ" các đặc điểm của dữ liệu huấn luyện mà không thể tổng quát hóa được cho dữ liệu mới. Ngược lại, nếu cả hai độ sai lệch đều giảm dần hoặc ổn định, điều này cho thấy mô hình đang học được các đặc trưng cần thiết và có khả năng tổng quát hóa tốt hơn.



Name	Date modified	Type	Size
best.hdf5	18/03/2024 5:08 AM	HDF5 File	33,525 KB
mask_rcnn_foods_cfg_0020.h5	18/03/2024 5:08 AM	H5 File	250,516 KB

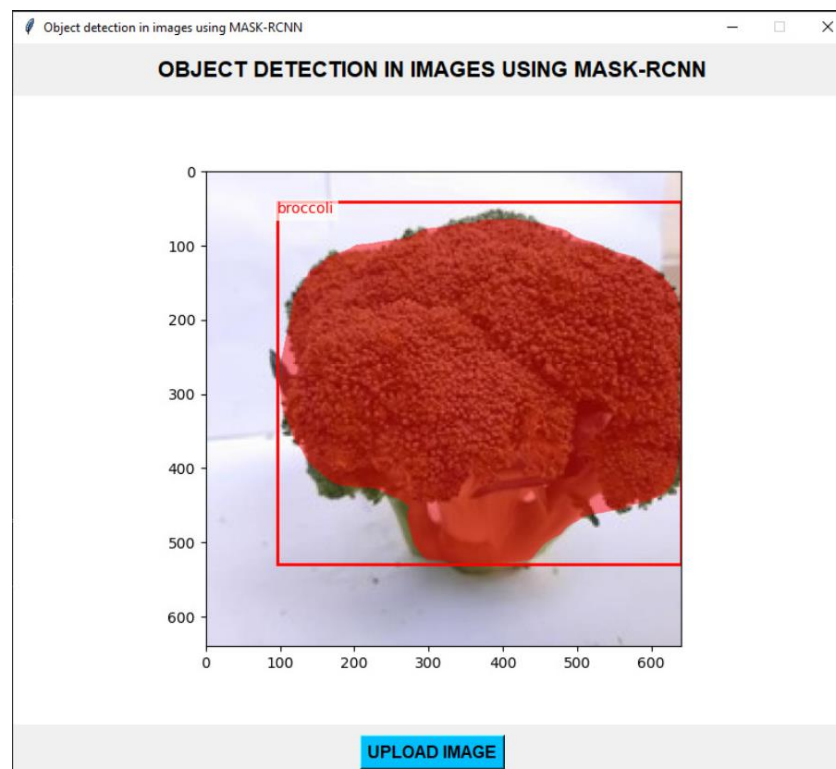
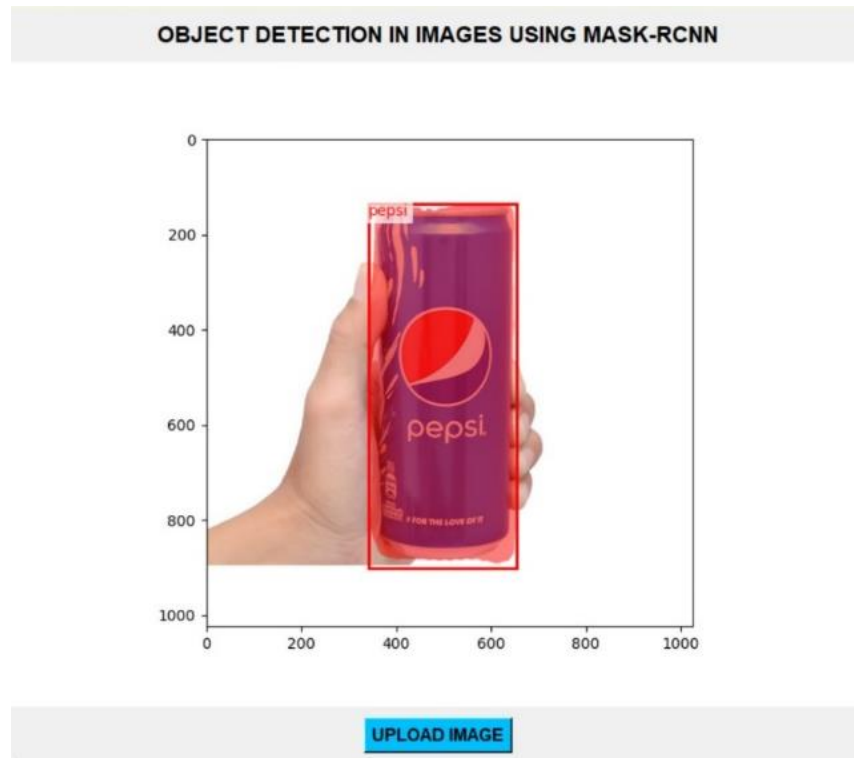
Hình 3.19. Kết quả thu về sau khi train

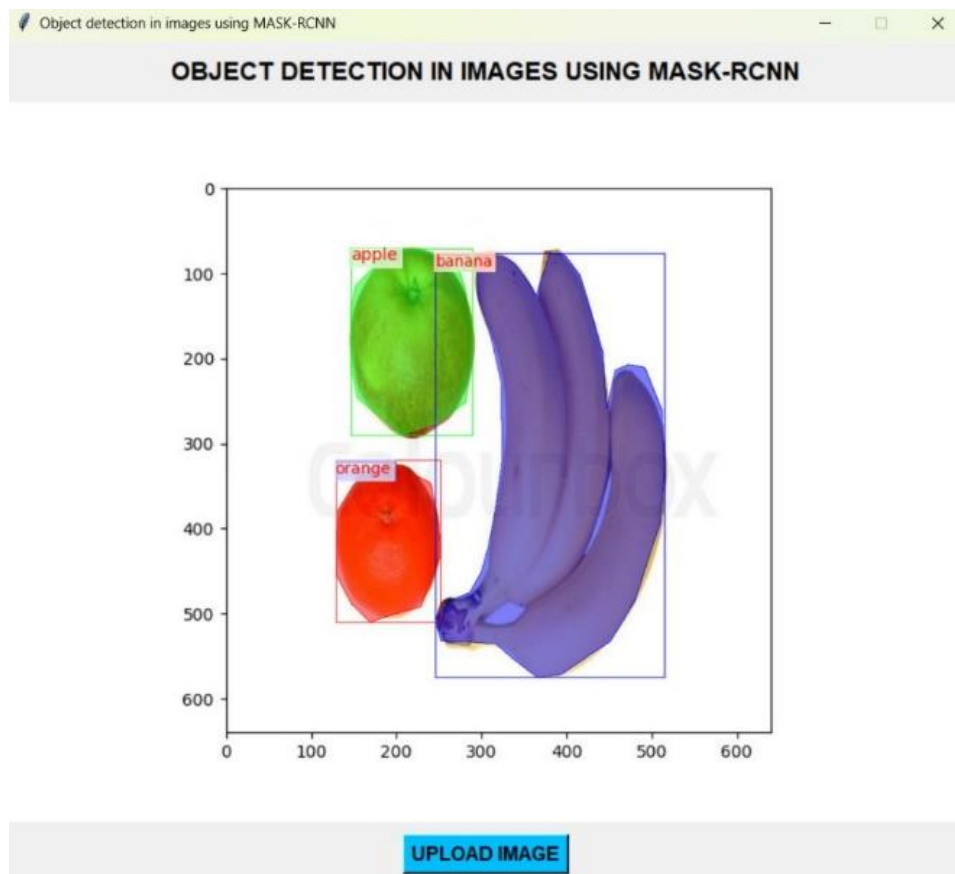
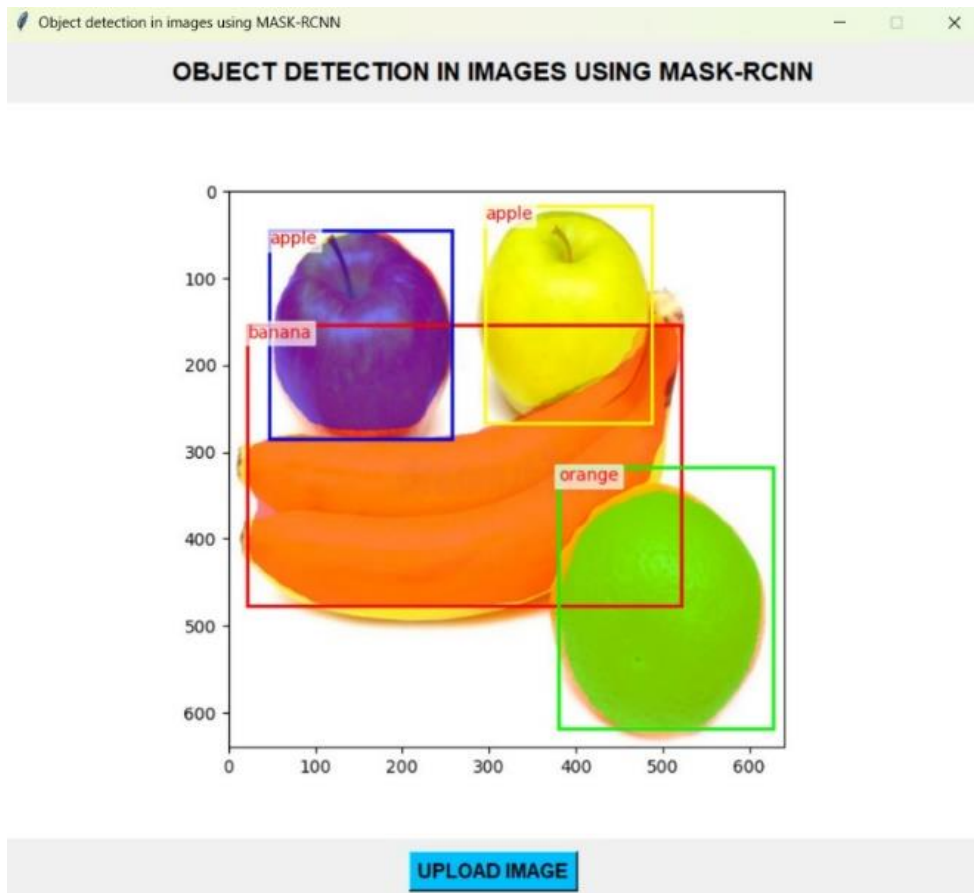
Trong mask\_rcnn\_foods\_cfg\_0020.h5 lưu lại kết quả dự đoán dựa trên ảnh được đưa vào từ 32 loại thực phẩm khác nhau.

#### Bước 4: Thiết kế giao diện người dùng (GUI)

Xây dựng giao diện người dùng bằng Tkinter và thư viện Matplotlib để thực hiện phát hiện đối tượng trên hình ảnh sử dụng mô hình MaskRCNN. Tkinter là một bộ dụng cụ GUI trong thư viện python tiêu chuẩn. Giao diện người dùng cho phép người dùng tải lên hình ảnh hoặc kéo và thả từ hệ thống tệp. Khi hình ảnh được chọn, mô hình MaskRCNN

được sử dụng để phát hiện các đối tượng trong hình ảnh. Các đối tượng được phát hiện được đánh dấu bằng các hộp giới hạn và nhãn tương ứng. Kết quả được hiển thị trên một cửa sổ mới sử dụng Matplotlib. Người dùng cũng có thể tải lên hình ảnh bằng cách nhấp vào nút "Upload Image".





Hình 3.20. Một số kết quả dò tìm đối tượng trong ảnh

### 3.4. KẾT LUẬN CHƯƠNG 3

Trong chương 3, đã khám phá ứng dụng mô hình MASK R-CNN trong việc dò tìm đối tượng trong ảnh qua đó cho thấy MASK R-CNN không chỉ có khả năng phát hiện và phân loại đối tượng một cách chính xác, mà còn có khả năng tạo ra các mặt nạ (masks) chi tiết cho từng vùng của đối tượng, mang lại kết quả đáng kinh ngạc.

Qua việc áp dụng MASK R-CNN vào các ứng dụng thực tiễn như phát hiện đối tượng trong hình ảnh cho thấy mô hình này có thể giúp tăng cường hiệu suất và chính xác trong quá trình phân tích ảnh.

Ngoài ra, việc kết hợp MASK R-CNN với các công nghệ khác như học sâu và xử lý ngôn ngữ tự nhiên cũng mở ra những triển vọng mới trong việc phát triển các hệ thống thông minh và tự động.

Chương 3 đã minh họa rõ ràng về tiềm năng và ứng dụng rộng rãi của mô hình MASK R-CNN trong việc dò tìm đối tượng trong ảnh. Điều này mở ra nhiều cơ hội cho việc sử dụng và phát triển các ứng dụng AI tiên tiến trong tương lai.

## KẾT LUẬN VÀ ĐÁNH GIÁ KHÓA LUẬN

Sau khi nghiên cứu, tìm hiểu Mask R-CNN và ứng dụng vào dò tìm đối tượng trong ảnh nhóm đã nắm bắt và hiểu biết hơn về trí tuệ nhân tạo (AI), Machine learning, Deep learning, Mask R-CNN và ứng dụng trong thực tế. Cũng cố thêm kiến thức và hành trang sau khi tốt nghiệp, tạo một bước đệm cho sự phát triển của bản thân và xã hội. Các kết quả chính mà khóa luận đã đạt được tương ứng với các mục tiêu đề ra ban đầu:

- Hoàn thiện xây dựng cơ sở lý thuyết tìm hiểu chung về trí tuệ nhân tạo (AI), Machine Learning, Deep Learning, môi trường framework, ...
- Hoàn thiện xây dựng cơ sở lý thuyết Mask Region-based Convolutional Neural Network (Mask R-CNN).
- Ứng dụng Mask R-CNN trong dò tìm đối tượng trong ảnh.

Thực nghiệm với mô hình mạng Mask R-CNN trong dò tìm đối tượng trong ảnh đã cho ra kết quả khá chính xác. Với việc phát hiện đối tượng trong ảnh có thể ứng dụng vào nhiều lĩnh vực như y tế, xã hội và đặc biệt là sự phát triển của các thiết bị thông minh.

Một lần cuối chúng em xin gửi lời cảm ơn sâu sắc tới tất cả các thầy, cô Trường Đại học Kinh tế - Kỹ thuật Công nghiệp và sâu sắc nhất tới ThS. Lương Thị Thảo Hiếu – Giảng viên hướng dẫn của nhóm đã giúp đỡ hoàn thành khóa luận tốt nghiệp này.

Mặc dù đã có nhiều cố gắng trong quá trình nghiên cứu, song do khả năng và kinh nghiệm có hạn, nên khóa luận không tránh khỏi những tồn tại, hạn chế và thiếu sót. Kính mong sự chỉ dẫn và đóng góp của các thầy cô giáo để khóa luận của chúng tôi được hoàn thiện hơn nữa.

## TÀI LIỆU THAM KHẢO

- [1]. Đinh Mạnh Tường (Trí Tuệ Nhân Tạo, NXB Khoa Học Và Kỹ Thuật, 2002)
- [2]. PGS.TS. Từ Minh Phương (Giáo Trình Nhập Môn Trí Tuệ Nhân Tạo, Học Viện Bưu Chính Viễn Thông, NXB Thông Tin Và Truyền Thông, 2014)
- [3]. Nguyễn Thanh Tuấn (Tác Giả Sách Deep Learning Cơ Bản, 2019)
- [4]. Kaiming He, Georgia Gkioxari, Piotr Dollár, và Ross Girshick, “Mask R-CNN” ICCV, 2017
- [5]. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications” 2017
- [6]. Wenhui Li, Zhaoyang Liu, Tao Cheng, Liang Lin, và Xiaowei Hu, “Object Detection via Region-Based Fully Convolutional Networks” ECCV, 2016
- [7]. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “ResNet: Deep Residual Learning for Image Recognition” 2016
- [8]. Mingxing Tan, Quoc V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks” ICML 2019
- [9]. Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, “DenseNet: Densely Connected Convolutional Networks” CVPR 2017
- [10]. <https://www.anaconda.com/>
- [11]. <https://www.spyder-ide.org/>