BỘ GIÁO DỤC ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

--- 🕮 ---

ĐỒ ÁN TỐT NGHIỆP

NGHIÊN CỨU, ÁP DỤNG VÀ XÂY DỰNG HỆ THỐNG PHÂN LOẠI CHAI THỦY TINH VÀ NHỰA

SINH VIÊN THỰC HIỆN : NGUYỄN ĐÌNH ĐẠT

MÃ SINH VIÊN : 1451020053

KHOA : CÔNG NGHỆ THÔNG TIN

HÀ NỘI - 2023

BỘ GIÁO DỤC ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

NGUYỄN ĐÌNH ĐẠT

NGHIÊN CỨU, ÁP DỤNG VÀ XÂY DỰNG HỆ THỐNG PHÂN LOẠI CHAI THỦY TINH VÀ NHỰA

CHUYÊN NGÀNH : **CÔNG NGHỆ THÔNG TIN**

MÃ SỐ : **74.80.201**

NGƯỜI HƯỚNG DẪN: TS. BÙI HẢI PHONG

HÀ NỘI - 2023

LỜI CAM ĐOAN

Kính thưa quý thầy cô giáo.

Em là: Nguyễn Đình Đạt Mã số sinh viên: 1451020053

Là sinh viên Trường Đại học Đại Nam niên khóa : 2020 – 2024

Em xin cam đoan đề tài "Nghiên cứu, áp dụng và xây dựng hệ thống phân loại chai thuỷ tinh và nhựa" là công trình nghiên cứu độc lập của bản thân, không sao chép của người khác. Mọi số liệu trong đồ án này là hoàn toàn có thật và được lấy từ những nguồn đáng tin cậy. Nếu sai, em xin hoàn toàn chịu trách nhiệm và kỷ luật từ phía nhà trường.

Hà Nội, ngày 25 tháng 05 năm 2024

Sinh viên thực hiện

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới quý thầy cô và toàn thể nhà trường vì đã tạo điều kiện thuận lợi, môi trường học tập tốt cùng sự hỗ trợ nhiệt tình trong suốt quá trình em thực hiện đồ án.

Nhờ có sự quan tâm, chỉ dẫn và các nguồn tài nguyên quý báu từ phía nhà trường, em đã có thể hoàn thành đồ án một cách thành công và đạt được những kết quả tốt đẹp. Môi trường học tập chuyên nghiệp và khuyến khích nghiên cứu của nhà trường đã giúp em phát triển kiến thức, kỹ năng và đam mê trong lĩnh vực của mình.

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy TS. Bùi Hải Phong vì đã giúp đỡ và hỗ trợ em trong suốt quá trình thực hiện đồ án "Nghiên cứu, Áp dụng và Xây dựng hệ thống phân loại phai thủy tinh và nhựa". Nhờ sự hướng dẫn tận tâm, những góp ý quý báu và sự động viên của thầy, em đã có thể hoàn thiện đồ án một cách tốt nhất.

Sự tận tụy và kinh nghiệm của thầy đã truyền cảm hứng cho em không chỉ trong công việc nghiên cứu mà còn trong việc học hỏi và phát triển kỹ năng chuyên môn. Một lần nữa, em xin gửi lời cảm ơn sâu sắc tới thầy. Em mong rằng sẽ tiếp tục nhận được sự hỗ trợ và chỉ dẫn của Thầy trong những dự án và nghiên cứu tiếp theo.

NHẬN XÉT

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

DANH MỤC CÁC CHỮ VIẾT TẮT

Chữ Viết Tắt	Ý Nghĩa
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
FC	Fully Connected
IoT	Internet of Things
RGB	Red Green Blue
RNN	Recurrent Neural Networks
VGG16	Visual Geometry Group 16
VGG19	Visual Geometry Group 19

MỤC LỤC

MỤC LỤC HÌNH ẢNH

LỜI GIỚI THIỆU

Trong thời đại hiện nay, vấn đề ô nhiễm môi trường đang trở thành một trong những thách thức lớn nhất đối với sự phát triển bền vững của xã hội. Sự gia tăng không ngừng của sản xuất và tiêu thụ đã dẫn đến việc tăng cường sự thải ra môi trường, đặc biệt là từ các loại vật liệu như chai thuỷ tinh và nhựa. Nơi nơi mọi ngóc ngách trên đường phố, trên núi, trên bãi biển chai lọ thủy tinh đều có mặt, đây là hai mặt hàng khó phân hủy và tiêu hủy nhất do con người sản xuất khiến cho môi trường ô nhiễm trầm trọng. Hai loại rác thải này đem đến sự đe dọa tới môi trường sinh vật sông tự nhiên cũng như là tác động trực tiếp đến con người chúng ta.

Theo thông tin được công bố từ nhiều nguồn trên mạng cũng như trên thông tin đại chúng thì với chai thuỷ tinh không phân hủy tự nhiên trong điều kiện môi trường thông thường. Chúng có thể tồn tại hàng ngàn năm mà không bị phân hủy. Điều này làm cho chai thuỷ tinh được coi là một vật liệu không gây ô nhiễm môi trường khi bị bỏ đi, nhưng cũng đồng nghĩa rằng chúng có thể đóng vai trò như rác thải vĩnh viễn nếu không được xử lý. Còn đối với chai nhựa thì thời gian phân hủy của nhựa phụ thuộc vào loại nhựa cụ thể. Một số loại nhựa có thể phân hủy trong vài năm dưới điều kiện môi trường thích hợp, trong khi những loại nhựa khác có thể tồn tại hàng trăm năm hoặc thậm chí là hàng ngàn năm mà không phân hủy. Các nhựa sinh học hoặc nhựa tái chế có thể phân hủy nhanh hơn dưới điều kiện môi trường có sự can thiệp của vi sinh vật phân hủy.

Trong bối cảnh hiện nay, con người đã đi tới mọi nơi trên địa cầu bằng nhưng rác thải vứt bừa bãi từ trên đất liền cũng như đáy biển sâu nhất của trái đất, ảnh hưởng tới nhiều sinh vật, động vật. Mặc dù đã có rất nhiều nhóm tình nguyện đi đến nhiều nơi để dọn rác, nhưng vẫn không thể hết được. Vậy việc xử lý và tái chế các loại vật liệu như chai thuỷ tinh và nhựa trở thành một nhu cầu cấp bách. Tuy nhiên, quy trình này đang gặp phải nhiều thách thức, đặc biệt là trong việc phân loại chúng một cách hiệu quả và tiết kiệm chi phí. Đó là lý do em chọn đề tài nghiên cứu và xây dựng hệ thống phân loại chai thuỷ tinh và nhựa. Đây sẽ có thể là một công nghệ sẽ cho mọi người hiểu biết về tầm quan trọng của môi trường và ý thức hơn trong việc xả rác thải.

CHƯƠNG 1. CƠ SỞ KHOA HỌC VÀ TÍNH THỰC TIỄN

1.1. Cơ sở khoa học

Để thực hiện đồ án về xây dựng hệ thống phân loại chai thuỷ tinh và nhựa sử dụng mạng nơ-ron tích chập (CNN),có một số cơ sở khoa học:

Sử dụng các Framework học sâu như Tensorflow, Pytorch hoặc Keras để xây dựng và huấn luyện mô hình cnn sẽ giúp tiết kiệm thời gian và tăng hiệu quả.

Cần có một bộ dữ liệu ảnh đủ lớn và đa dạng về chai thuỷ tinh và nhựa để huấn luyện và đánh giá mô hình. Dữ liệu này cần được gắn nhãn chính xác để có thể huấn luyện mô hình phân loại.

Hiểu biết về các vấn đề môi trường liên quan đến chai thuỷ tinh và nhựa, cũng như về các phương pháp quản lý và xử lý rác thải sẽ giúp hiểu rõ hơn về bối cảnh và ý nghĩa của đề tài.

Kỹ năng phân tích kết quả của mô hình và đánh giá hiệu suất của nó trên tập dữ liệu kiểm tra là quan trọng để đảm bảo tính chính xác và hiệu quả của hệ thống phân loại.

Để thực hiện đồ án này cần có sự kết hợp giữa kiến thức về máy học, xử lý ảnh, môi trường và các kỹ năng lập trình, kết hợp với việc thu thập dữ liệu và phân tích kết quả một cách chặt chẽ và chuyên sâu.

Ngoài ra cần biết thêm về xử lý dữ liệu hình ảnh. Trước khi đưa dữ liệu nào đó vào một mô hình trainning luôn cần một dữ liệu sạch nếu không làm tốt bước này sẽ khiến cho những công đoạn phân tích dữ liệu về sau sẽ không thể thực hiện, và phải bắt đầu lại từ đầu.

1.2. Tính thực tiễn

Việc nghiên cứu và xây dựng hệ thống phân loại chai thủy tinh và nhựa không chỉ là một yếu tố quan trọng mà còn là bước đi thực tiễn trong việc giải quyết các vấn đề môi trường. Trong thời đại ngày càng chú trọng vào bảo vệ môi trường và tái chế, việc

Hình 1.1. Các loại chai lọ (Nguồn https://chaipetsaigon.com/su-dung-chai-nhua-hay-chai-thuy-tinh/)

phát triển các hệ thống phân loại thông minh đóng vai trò quan trọng trong việc giảm thiểu lượng rác thải và tối ưu hóa sử dụng tài nguyên.

Trong quá trình nghiên cứu và phát triển, việc áp dụng công nghệ học sâu (deep learning) là một xu hướng đáng chú ý. Công nghệ này cho phép xây dựng các mô hình máy học có khả năng học và cải thiện từ dữ liệu, từ đó tăng cường khả năng phân loại tự động và chính xác của hệ thống. Thông qua việc huấn luyện trên dữ liệu lớn, các mô hình học sâu có thể nhận biết và phân loại chai thủy tinh và nhựa một cách chính xác và hiệu quả, giúp giảm thiểu lỗi.

Ngoài ra, việc tích hợp các công nghệ mới, như Internet of Things (IoT) và trí tuệ nhân tạo (AI), vào các hệ thống phân loại cũng mở ra nhiều cơ hội mới trong việc tối ưu hóa quy trình và nâng cao hiệu suất. Sự kết hợp giữa các cảm biến thông minh, dữ liệu thời gian thực và thuật toán AI cho phép các hệ thống phân loại tự động điều chỉnh và tối ưu hóa hoạt động của mình, từ đó giảm thiểu lãng phí và tăng cường hiệu quả sản xuất.

1.3. Ứng dụng của kết quả nghiên cứu trong đề tài

Như đã nói ở trong phần đặt vấn đề, ứng dụng của kết quả nghiên cứu và xây dựng hệ thống phân loại chai thủy tinh và trường và sản xuất trong thời đại hiện nay.

Kết quả của nghiên cứu không chỉ giúp tăng cường hiệu quả sản xuất và tái chế, mà còn mở ra nhiều cơ hội mới trong việc ứng dụng công nghệ thông tin và trí tuệ nhân tạo vào các lĩnh vực khác nhau.

Bên cạnh đó, kết quả nghiên cứu cũng có thể được áp dụng vào nhiều lĩnh vực khác nhau, từ ngành tái chế đến công nghiệp sản xuất và đóng gói. Việc tự động hóa

quy trình phân loại và phân tách các loại vật liệu giúp giảm thiểu lãng phí và tối ưu hóa sử dụng tài nguyên, đồng thời tạo ra các sản phẩm có tính thân thiện với môi trường và giảm thiểu tác động đến tài nguyên tự nhiên.

Như vậy, ứng dụng của kết quả nghiên cứu và xây dựng hệ thống phân loại chai thủy tinh và nhựa không chỉ đem lại lợi ích cho doanh nghiệp mà còn đóng góp vào việc bảo vệ môi trường và thúc đẩy phát triển bền vững trong xã hội. Điều này thể hiện tính ứng dụng và giá trị thực tiễn của đề tài nghiên cứu.

1.4. Mục tiêu đề tài

Dự án giải quyết thách thức quản lý chất thải hiệu quả bằng cách xây dựng và cải tiến mô hình học máy có thể phân loại các loại chất thải khác nhau. Cụ thể, nó tập trung vào việc phân loại chất thải thủy tinh và nhựa, vì những vật liệu này có vấn đề phân loại phức tạp hơn.

với hơn 2 tỷ tấn rác thải được tạo ra trên toàn cầu mỗi năm, việc quản lý chất thải sai lầm dẫn đến ô nhiễm môi trường và các vấn đề sức khỏe con người. Những đổi mới như thùng chứa rác thải thông minh và công nghệ phân loại tự học có thể giúp giảm thiểu những vấn đề này. Xây dựng và cải tiến mô hình học máy để phân loại chất thải là một bước hướng tới việc tạo ra các giải pháp này.

Hình 1.2. Rác thải gây ô nhiễm(Nguồn https://www.rfi.fr/vi/quoc-te/20180605-ngay-moi-truong-the-gioi-dai-duong-la-kho-rac-thai-nhua-toan-cau)

Nghiên cứu và phát triển một hệ thống phân loại chai thủy tinh và nhựa thông minh, sử dụng công nghệ học sâu. Hệ thống này không chỉ có khả năng nhận diện và phân loại các loại chai dựa trên các tiêu chí như kích thước, hình dạng và chất liệu, mà còn có khả năng phân loại một cách chính xác và hiệu quả, giúp tăng cường hiệu suất trong quá trình tái chế và chế biến.

Hình 1.3. Tái chế rác thải (Nguồn https://vuachailo.vn/cac-cau-hoi-thuong-gap-ve-tai-che-thuy-tinh)

Việc này giúp tối ưu hóa quy trình sản xuất và tái chế, giảm thiểu lãng phí và tăng cường hiệu suất công việc. Bằng cách áp dụng hệ thống phân loại vào quy trình sản xuất, các nhà sản xuất có thể tăng cường kiểm soát chất lượng và giảm thiểu sự phụ thuộc vào lao động. Ngoài ra, mục tiêu của dự án còn là nghiên cứu và phát triển các phương pháp và công nghệ mới nhằm cải thiện hiệu suất và tích hợp hệ thống phân loại vào hệ thống quản lý rác thải hiện đại. Bằng cách áp dụng công nghệ tiên tiến, ta có thể tối ưu hóa việc thu gom, phân loại và xử lý rác thải, giúp giảm thiểu tác động tiêu cực đến môi trường và tạo ra một môi trường sống lành mạnh hơn cho cộng đồng.

Để đảm bảo rằng dự án nghiên cứu này có hướng tới và giới hạn rõ ràng nhất thì đề tài sẽ tập trung vào việc nghiên cứu và phát triển hệ thống phân loại chai thủy tinh và nhựa. Nghiên cứu sẽ tập trung vào việc phân loại các loại chai theo thu thập dữ liệu, tiền xử lý và đào tạo và triển khai mô hình. Phạm vi của đề tài sẽ giới hạn trong phạm vi nghiên cứu về hệ thống phân loại chai thủy tinh và nhựa, bao gồm cả quy trình phân loại tự động và tích hợp công nghệ học sâu vào hệ thống. Điều này sẽ giúp đảm bảo tính khả thi và hiệu quả của dự án.

Để đảm bảo việc nghiên cứu diễn ra một cách suôn sẻ và hiệu quả, em sẽ cần sử dụng các kỹ thuật xử lý dữ liệu đầu vào giúp đảm bảo rằng chúng có định dạng nhất

quán và sẵn sàng để sử dụng làm đầu vào cho cnn, các kỹ thuật tiền xử lý phổ biến bao gồm thay đổi kích thước, cắt xén và chuẩn hóa., thuật toán Convolutional Neural Network (CNN) để xác định hình ảnh có phải chai thủy tinh hay lọ hay không.

Ngoài ra, mục tiêu là có khả năng tích hợp mô hình vào các hệ thống phức tạp hơn để sử dụng trong các công nghệ phân loại hiện đại. Tập dữ liệu được sử dụng cho dự án này là "bộ dữ liệu hình ảnh để phân loại rác thải sinh hoạt" từ Kaggle. Dự án bắt đầu với mô hình cơ sở hồi quy Logistic, sau đó đào tạo, thử nghiệm và tinh chỉnh các mô hình học sâu khác nhau. Mô hình hoạt động tốt nhất, VGG16 đã được tinh chỉnh, sau đó được trực quan hóa và đề xuất làm giải pháp cuối cùng.

1.5. Giới thiệu về trí tuệ nhân tạo và học máy.

Trí tuệ nhân tạo là một hướng nghiên cứu của lĩnh vực công nghệ thông tin và khoa học máy tính. Phát triển hệ thống thông minh nhằm giải quyết vấn đề các bài toán trong thực tế giống như hoạt động của não bộ con người. Trí tuệ nhân tạo được bắt đầu nghiên cứu từ những năm 50 của thế kỷ 20, trong khoảng 30 năm trở lại đã được cộng đồng khoa học quan tâm mạng mẽ. Rất nhiều các hội thảo lớn về lĩnh vực này được tổ chức hàng năm. Các ứng dụng tiêu biểu của trí tuệ nhân tạo vào đời sống xã hội bao gồm người máy, robot, xử lý ngôn ngữ tự nhiên, nhân dạng, phát hiện đặc tính khác biệt, an ninh quốc phòng, sinh học, khoa học vũ trụ,…

với đồ án tốt nghiệp của em, với việc được trang bị các môn học lý thuyết và thực hành như thuật toán, học máy. Em đã lựa chọn đề tài nghiên cứu, áp dụng và xây dựng hệ thống phân loại chai thuỷ tinh và nhựa. Chủ đề phân lớp dữ liệu là một nhánh quan trọng trong lĩnh vực học máy của trí tuệ nhân tạo, có thể nhận biết các ứng dụng của học máy trong thực tế như người máy, robot, nhận dạng khuôn mặt…

- **Đối tượng và phạm vi nghiên cứu**
    - Đối tượng nghiên cứu
        - Nghiên cứu mạng nơ ron đơn và mạng nơ ron tích chập
        - Nghiên cứu bài toán phân loại chai nhựa và thủy tinh
        - Tìm hiểu xây dựng kiến trúc và thực hiện mô hình nơ ron tích chập để giải quyết bài toán

- Phạm vi nghiên cứu
    - Lý thuyết: nghiên cứu lý thuyết xử lý ảnh, học máy, deep learning, mang nơ ron tích chập
    - Thực nghiệm: Lập trình trên phần mềm python cho chương trình phân loại.

- **Phương pháp nghiên cứu**
    - Thu thập, phân tích dữ liệu thông tin liên quan đến đề tài đưa ra cái nhìn tổng quan, các khó khăn, và các ràng buộc bài toán…
    - Tiến hành phân tích, xây dựng giải pháp nhận dạng gồm : tiền xử lý dữ liệu, trích chọn đặc trưng huấn luyện mô hình, tiền xử lý.
    - Xây dụng kiểm thử đánh giá hiệu quả các phương pháp nơ ron tích chập để chọn ra mô hình thích hợp cho bài toán.

- **Cấu trúc bài báo cáo**

Cấu trúc bài báo cáo nghiên cứu, áp dụng và xây dựng hệ thống phân loại chai thuỷ tinh và nhựa. Và bao gồm nhưng chương sau.

Chương 1: cơ sở khoa học và thực tiễn

Chương 2: cơ sở lý thuyết của deep learning và các mô hình sử dụng trong đồ án

Chương 3: triển khai hệ thống thống phân loại chai thuỷ tinh và nhựa. Chương này sẽ đi sâu về cài đặt hệ thống bằng phương pháp deep learning trong mô hình nơ ron tích chập.

1.6. Tổng quan về xử lý ảnh

1.6.1. Giới thiệu

Con người thu nhận thông tin qua các giác quan, trong đó thị giác đóng vai trò quan trọng nhất. Những năm trở lại đây với sự phát triển của phần cứng máy tính, xử lý ảnh và đồ họa đó phát triển một cách mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Xử lý ảnh và đồ họa đóng một vai trò quan trọng trong tương tác người và máy.

Quá trình xử lý ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh và đưa ra một kết luận.

- **Các bước cơ bản trong hệ thống xử lý ảnh**
    - Khối thu nhận ảnh: có nhiệm vụ tiếp nhận ảnh đầu vào.
    - Khối tiền xử lý: có nhiệm vụ xử lý nâng cao chất lượng ảnh như giảm nhiễu, phân vùng, tìm biên,...
    - Khối trích chọn đặc điểm: có nhiệm vụ trích chọn các đặc trưng quan trọng của các bức ảnh đã được tiền xử lý để sử dụng trong hệ quyết định
    - Khối hậu xử lý: có nhiệm vụ xử lý các đặc điểm đã trích chọn, có thể lược bỏ hoặc biến đổi các đặc điểm này để phù hợp với các kỹ thuật cụ thể sử dụng trong hệ quyết định
    - Khối hệ quyết định và lưu trữ: có nhiệm vụ đưa ra quyết định dựa trên dữ liệu đã học lưu trong khối lưu trữ
    - Khối kết luận: đưa ra kết luận dựa vào quyết định của khối quyết định

1.6.2. Một số khái niệm trong xử lý ảnh

- **Ảnh và điểm ảnh:**
    - Điểm ảnh được xem như là dấu hiệu hay cường độ sáng tại một vị trí nào đó của đối tượng trong không gian và ảnh được xem như là một tập hợp các điểm ảnh.
- **Mức xám, màu:**
    - Là số các giá trị có thể có của các điểm ảnh của ảnh.
- **Nắn chỉnh biến dạng:**
    - Ảnh thu nhận thường bị biến dạng do các thiết bị quang học và điện tử.
- **Khử nhiễu:**

  Có 2 loại nhiễu cơ bản trong quá trình thu nhận ảnh mà ta cần bỏ là nhiễu hệ thống và nhiễu ngẫu nhiên. Nhiễu hệ thống có thể khử bằng các phép biến đổi. Nhiễu ngẫu nhiên là do các vết bẩn không rõ là nguyên nhân có thể khắc phục bằng phép lọc.

- **Chính số mức xám:**

Chỉnh số mức xám là nhằm khắc phục tính không đồng đều của hệ thống xử lý ảnh, thông thường có 2 hướng tiếp cận. Giảm số mức xám thực hiện bằng cách nhóm

các mức xám gần nhau thành một bó. Trường hợp giảm xuống 2 mức xám thì chính là chuyển về ảnh đen trắng. Tăng số mức xám thực hiện nội suy ra các mức xám trung gian bằng kỹ thuật nội suy. Kỹ thuật này nhằm tăng cường độ mịn cho ảnh.

Phân tích ảnh: là khâu quan trọng trong quá trình xử lý ảnh để tiến tới hiểu ảnh. Trong phân tích ảnh việc trích chọn đặc điểm là một bước quan trọng. Các đặc điểm của đối tượng được trích chọn tuỳ theo mục đích nhận dạng trong quá trình xử lý ảnh. Một số đặc điểm của ảnh.

Đặc điểm không gian: phân bỗ mức xám, phân bó xác suắt, biên độ, điểm uốn,...

Đặc điểm biến đổi: các đặc điểm loại này được trích chọn bằng việc thực hiện lọc vùng (Zonal Filtering). Các bộ vùng được gọi là mặt nạ đặc điểm (Feature Mask) thường là các khe hẹp với hình dạng khác nhau (chữ nhật, tam giác, cung tròn)

- **Nền ảnh:**

Nén ảnh là kỹ thuật nhằm giảm thiểu không gian lưu trữ. Có hai hướng tiếp cận chính là nén có bảo toàn và không bảo toàn thông tin. Nén không bảo toàn thì thường, có khả năng nén cao hơn nhưng không phục hồi được ảnh gốc, ngược lại nén có bảo toàn cho phép khôi phục hoàn toàn ảnh gốc. Nén ảnh nói chung có 4 cách tiếp cận.

Nén ảnh thống kê: kỹ thuật nén này dựa vào việc thống kê tần xuất xuất hiện của giá trị các điểm ảnh, trên cơ sở đó mà có chiến lược mã hóa thích hợp.

Nén ảnh sử dụng phép biến đổi: đây là kỹ thuật tiếp cận theo hướng nén không bảo. Toàn và do vậy tỉ lệ nén tương đối cao.

Nén ảnh không gian: kỹ thuật này dựa vào vị trí không gian của các điểm ảnh để tiến hành mã hóa. Kỹ thuật lợi dụng sự giống nhau của các điểm ảnh trong các vùng gần nhau.

Nén ảnh fractal: sử dụng tính chất fractal của các đối tượng ảnh. Tính chất fractal của ảnh thẻ hiện sự lặp lại của các chỉ tiết tại nhiều vị trí khác nhau với kích thước và hướng khác nhau. Kỹ thuật nén sẽ tính toán để chỉ cần lưu trữ phần gốc ảnh và quy. Luật sinh ra ảnh theo nguyên lý fraetal.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Học sâu (Deep learning)

Học máy tập trung vào việc phát triển các thuật toán và mô hình giúp máy tính có thể tự học từ dữ liệu và cải thiện hiệu suất của chúng mà không cần phải lập trình rõ ràng cho mỗi tác vụ cụ thể. Mục tiêu chính của học máy là phát hiện ra các mẫu, xu hướng và mối quan hệ trong dữ liệu để đưa ra dự đoán hoặc quyết định dựa trên dữ liệu đó.

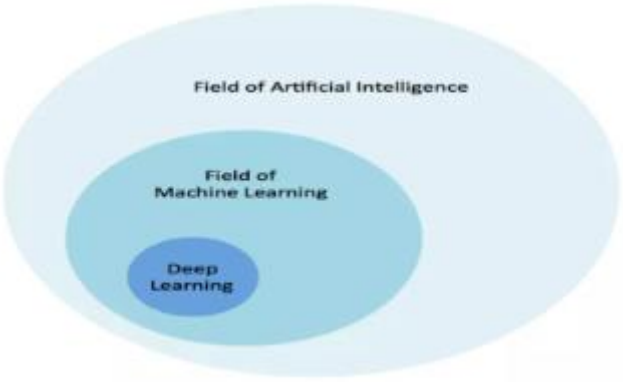
Thông thường học máy có hai loại chính là học có giám sát và học không giám sát. Ở trong đề tài này em sử dụng học máy có giám sát, nhằm đào tạo cho máy biết trước dữ liệu cho máy cần học thuộc loại nào và đưa ra dự đoán cho những ảnh khác.

Học sâu tập trung vào các mạng nơ ron nhân tạo có nhiều lớp. Học sâu nổi bật nhờ khả năng tự động học các đặc trưng từ dữ liệu thô mà không cần đến sự can thiệp của con người để thiết kế đặc trưng.

Đặc điểm của học sâu là sử dụng nhiều lớp nơ-ron để học các biểu diễn phức tạp của dữ liệu. Các lớp đầu vào nhận dữ liệu thô, các lớp ẩn học các đặc trưng từ dữ liệu, và lớp đầu ra đưa ra dự đoán. Học sâu thường yêu cầu lượng dữ liệu lớn và sức mạnh tính toán cao để huấn luyện, nhưng có thể đạt được hiệu suất rất cao trong nhiều bài toán phức tạp như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên.

Hình 2.1. Deep learning (Nguồn https://vietnix.vn/deep-learning-la-gi/)

Trong đề tài phân loại chai thủy tinh và nhựa này em sử dụng Hồi quy logistic, CNN, model VGG16 và VGG19. Nhằm so sánh tỉ lệ chính xác giữa các model cà xác định chọn ra model phù hợp, tốt nhất sử dụng cho đề tài.

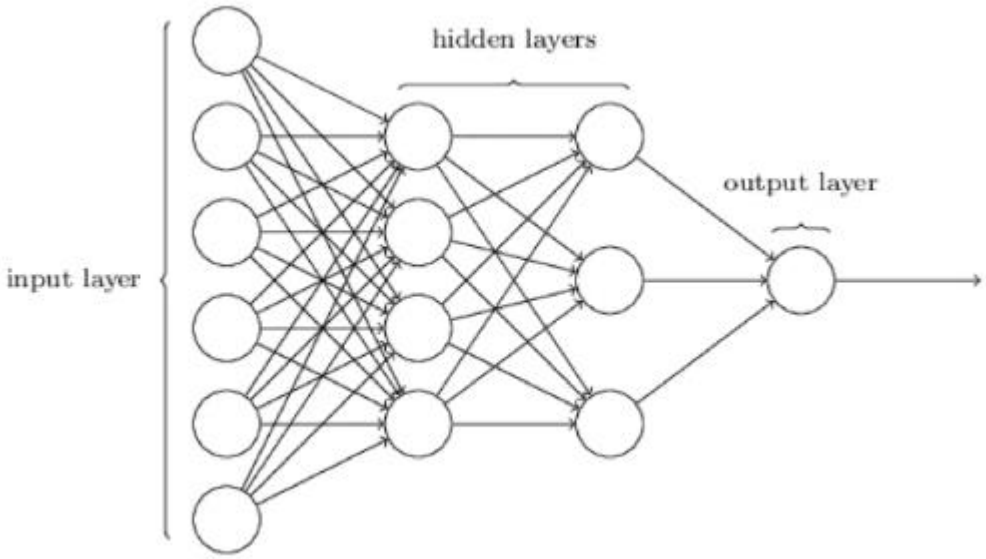


Hình 2.2. Minh họa deep learning

Ở vòng ngoài cùng, bạn có trí thông minh nhân tạo. Bên trong lớp này là machine learning. Với mạng neuron nhân tạo và deep. Iễu hơn của learning tại trung tâm. Nói rộng ra, deep learning là một tên gọi dễ mạng neuron nhân tạo. Từ "dep" trong deep learning để cập đến độ sâu của mạng, lưới. Một mạng neuron nhân tạo cũng có thể rất cạn. Mạng neuron được lấy cảm hứng từ cấu trúc của vỏ não.

Mức cơ bản được gọi là perceptron, biêu diễn toán học của một neuron sinh học. Giống như vỏ não, có thể có nhiều lớp perceptron kết nói với nhau.

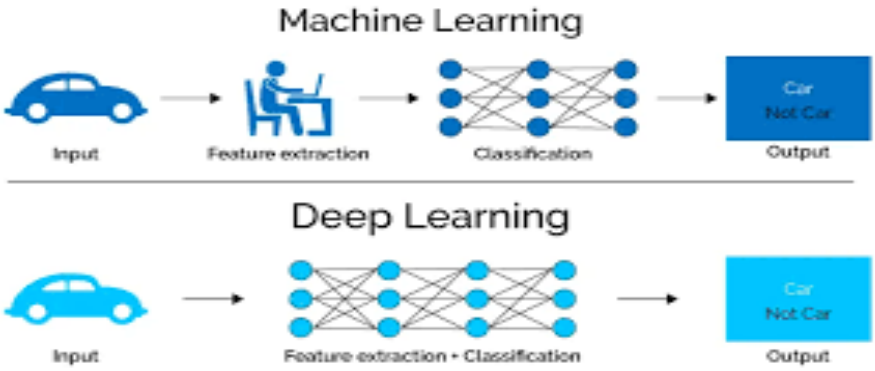
Lớp đầu tiên là lớp input (đầu vào). Mỗi nút trong lớp này lấy một đầu vào, và sau đó chuyển output (đầu ra) của nó làm đầu vào cho mỗi nút trong lớp tiếp theo. Nói chung không có kết nối giữa các nút trong cùng một lớp và lớp cuối cùng tạo nên kết quả đầu ra.



Hình 2.3. Minh họa các lớp trong học sâu

Deep leaming là kỹ thuật học tập trong mạng neuron sử dụng nhiều lớp trừu tượng đề giải quyết các vấn đề nhận dạng khuôn mẫu. Machine learning được coi là một nhánh hoặc phương pháp tiếp cận trí tuệ nhân tạo, trong khi đeep learning là một loại machine learning chuyên biệt.

2.2. Tính đặc biệt của học sâu với học máy



Hình 2.4. Sự khác nhau của deep learning và machine learning (Nguồn https://vietnix.vn/deep-learning-la-gi/)

Deep learning và machine learning đều là các nhánh của trí tuệ nhân tạo (ai), nhưng chúng có những lợi thế và ứng dụng khác nhau. Dưới đây là một số lợi thế của deep learning so với machine learning truyền thống:

- Khả năng xử lý dữ liệu phức tạp:

Deep learning có thể xử lý và phân tích dữ liệu có cấu trúc phức tạp và kích thước lớn như hình ảnh, video, âm thanh và văn bản tự nhiên. Điều này là do các mạng nơ-ron sâu có khả năng học được các đặc trưng phức tạp từ dữ liệu mà không cần phải thiết kế thủ công các đặc trưng này.

- Tự động trích xuất đặc trưng:

Một trong những lợi thế lớn nhất của deep learning là khả năng tự động trích xuất đặc trưng từ dữ liệu thô. Trong machine learning truyền thống, các đặc trưng phải được thiết kế thủ công bởi các chuyên gia, nhưng deep learning có thể tự động học và phát hiện các đặc trưng quan trọng từ dữ liệu mà không cần can thiệp của con người.

- Hiệu suất cao trong các bài toán nhận dạng và phân loại:

Deep learning đã chứng minh được hiệu suất vượt trội trong các bài toán nhận dạng hình ảnh, phân loại hình ảnh, nhận dạng giọng nói và dịch máy. Các mô hình deep learning như CNN (Convolutional Neural Networks) và RNN (Recurrent Neural

Networks) đã đạt được kết quả gần như tương đương hoặc thậm chí vượt qua con người trong một số tác vụ nhất định.

- Khả năng tổng quát hóa tốt:

Các mô hình deep learning thường có khả năng tổng quát hóa tốt hơn khi làm việc với các tập dữ liệu lớn. Chúng có thể học được các mô hình và đặc trưng phức tạp mà các thuật toán machine learning truyền thống có thể bỏ sót.

- Xử lý dữ liệu không cấu trúc:

Deep learning có khả năng xử lý dữ liệu không cấu trúc như văn bản, hình ảnh và âm thanh tốt hơn so với các phương pháp machine learning truyền thống. Điều này làm cho deep learning trở thành lựa chọn lý tưởng cho các ứng dụng như phân tích cảm xúc từ văn bản, nhận diện đối tượng từ hình ảnh và phân tích tín hiệu âm thanh.

- Khả năng cải thiện liên tục:

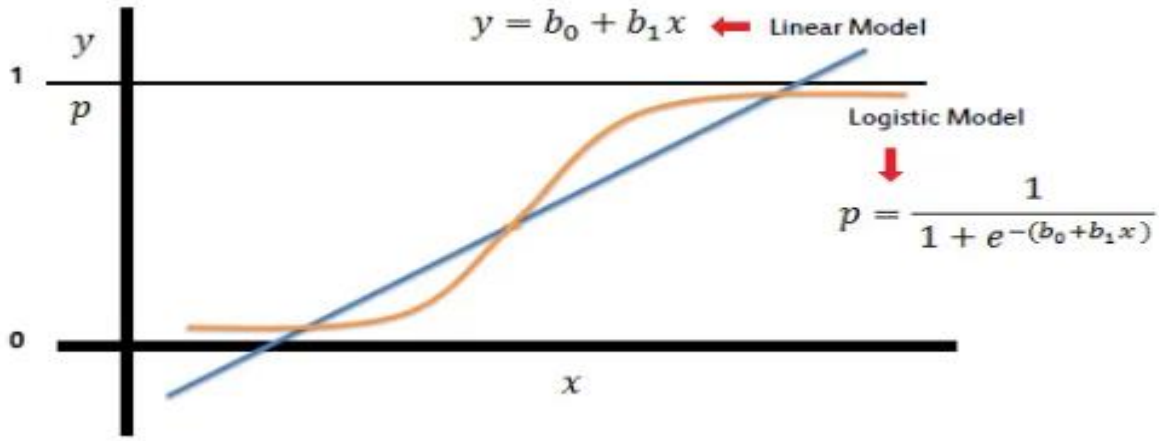
Khi có thêm dữ liệu mới, các mô hình deep learning có thể được huấn luyện lại hoặc tiếp tục huấn luyện (Fine-tuning) để cải thiện độ chính xác và hiệu suất. Điều này giúp các hệ thống deep learning trở nên linh hoạt và dễ dàng thích nghi với sự thay đổi của dữ liệu.

2.3. Lý thuyết về hồi quy logistic

Hồi quy logistic là một kỹ thuật phân tích dữ liệu sử dụng toán học để tìm ra mối quan hệ giữa hai yếu tố dữ liệu. Sau đó, kỹ thuật này sử dụng mối quan hệ đã tìm được để dự đoán giá trị của những yếu tố đó dựa trên yếu tố còn lại. Dự đoán thường cho ra một số kết quả hữu hạn, như có hoặc không.

Hồi quy logic thường là một ký thuật trong lĩnh vực trí tuệ nhân tạo và học máy. Mô hình machine learning là các chương trình phần mềm có thể được đào tạo để thực hiện các tác vụ xử lý dữ liệu phức tạp mà không cần sự can thiệp của con người. Hồi quy logistic là một mô hình thống kê được sử dụng để phân loại nhị phân, tức dự đoán một đối tượng thuộc vào một trong hai nhóm. Hồi quy logistic làm việc dựa trên nguyên tắc của hàm Sigmoid – một hàm phi tuyến tự chuyển đầu vào của nó thành xác suất thuộc về một trong hai lớp nhị phân.

2.3.1. Mô hình hồi quy logistic hoạt động.



Hình 2.5. Logistic Regression sử dụng hàm phi tuyến xác định xác suất lớp 0 và 1

Hồi quy Logistic hoạt động dựa trên hàm Sigmoid, được biểu diễn như sau:

$$S(z)=1/(1+e^{(-z)})$$

Hàm Sigmoid nhận đầu vào là một giá trị z bất kỳ, và trả về đầu ra là một giá trị xác suất nằm trong khoảng [0,1][0,1]. Khi áp dụng vào mô hình Hồi quy Logistic với đầu vào là ma trận dữ liệu X và trọng số w, ta có z = Xw.

Việc huấn luyện của mô hình là tìm ra bộ trọng số w sao cho đầu ra dự đoán của hàm Sigmoid gần với kết quả thực tế nhất.

Để đánh giá được hồi quy Logistic em sử dụng hàm log loss

$$L(w)=-n1\sum i=1n[yilogpi+(1-yi)log(1-pi)]$$

- n: số lượng mẫu dữ liệu trong tập huấn luyện.
- yi: giá trị thực tế của đầu ra thứ i.
- pi: xác suất dự đoán thuộc lớp 1 của mô hình cho đầu vào thứ i.

Hàm log loss đo lường khoảng cách giữa hai phân phối xác suất yi và pi. Khi mô hình dự đoán chính xác, tức là nếu yi = 1 thì pi càng gần 1, và nếu yi = 0 thì pi càng gần 0

2.3.2. Ứng dụng

Mô hình logistic là một công cụ toán học quan trọng, được ứng dụng rộng rãi trong nhiều lĩnh vực để phân tích và dự đoán các hiện tượng bị giới hạn bởi một giá trị nhất định. Chẳng hạn, mô hình hồi quy logistic giúp dự đoán xác suất xảy ra của các sự kiện

nhị phân như khả năng mua hàng của khách hàng hoặc xác suất bệnh nhân mắc bệnh dựa trên các yếu tố rủi ro.

Trong sinh học và y tế, mô hình logistic tăng trưởng được sử dụng để mô phỏng sự gia tăng của quần thể sinh vật hoặc sự lan truyền của dịch bệnh. Trong lĩnh vực quản lý chuỗi cung ứng và bán lẻ, mô hình logistic giúp tối ưu hóa việc quản lý hàng tồn kho, dự đoán nhu cầu và lập kế hoạch phân phối hàng hóa, đảm bảo sự cân đối giữa cung và cầu.

Ngoài ra, trong vận tải, các mô hình logistic hỗ trợ tối ưu hóa lộ trình vận chuyển, quản lý đội xe và lập kế hoạch vận tải, giúp giảm chi phí và nâng cao hiệu quả hoạt động.

2.4. Lý thuyết Convolutional Neural Network

Convolutional Neural Network (CNN) hay còn được gọi là mạng Nơ-ron tích chập là một trong những mô hình của Deep Learning. Tác dụng của thuật toán này chính là tạo ra những hệ thống thông minh, có sự phản ứng với độ chính xác cao. Ví dụ như Facebook, Google, đã đưa vào sản phẩm của mình chức năng nhận diện khuôn mặt,… Ứng dụng cơ bản nhất của thuật toán này là phân lớp, tức là phân biệt hoặc là cái này hoặc là cái kia, tức là khi đưa hình ảnh vào máy tính, nó sẽ là các điểm ảnh hai chiều và điều CNN thực hiện đó là khi các điểm ảnh thanh đổi thì máy tính vẫn biết được đó là hình ảnh gì.

CNN được chia thành 3 chiều: rộng, cao, sâu. Các Nơ-ron trong mạng không liên kết hoàn toàn với toàn bộ Nơ-ron kế đến mà chỉ liên kết tới một vùng nhỏ. Cuối cùng, một tầng đầu ra được tối giản thành vec-tơ của giá trị xác suất.

CNN gồm 2 thành phần: Phần tầng ẩn hay phần rút trích.

Mạng sẽ tiến hành tính toán hàng loạt phép tích chập (Convolutional layer) và phép hợp nhất (pooling) để phát hiện các đặc trưng. Một số lớp các liên kết đầy đủ (Fully Connected) sẽ đóng vai trò như một bộ phân lớp các đặc trưng đã rút trích trước đó. Tầng này sẽ đưa ra xác suất của một đối tượng.

Khả năng đặc trưng của CNN là tự động học các đặc trưng từ dữ liệu ảnh mà không cần sự can thiệp trực tiếp từ con người. Nhờ vào cấu trúc của mình, CNN có thể trích xuất các đặc trưng ở mức độ cao và trừu tượng từ ảnh đầu vào, giúp cải thiện độ chính xác của quá trình nhận diện.

CNN không chỉ được sử dụng trong việc nhận diện đối tượng, mà còn có thể áp dụng cho nhiều bài toán khác như phân loại, nhận dạng vùng quan tâm, hay phát hiện và phân loại các đối tượng di động. Điều này làm cho CNN trở thành một công cụ đa năng trong lĩnh vực xử lý hình ảnh.

2.4.1. Mô hình CNN hoạt động

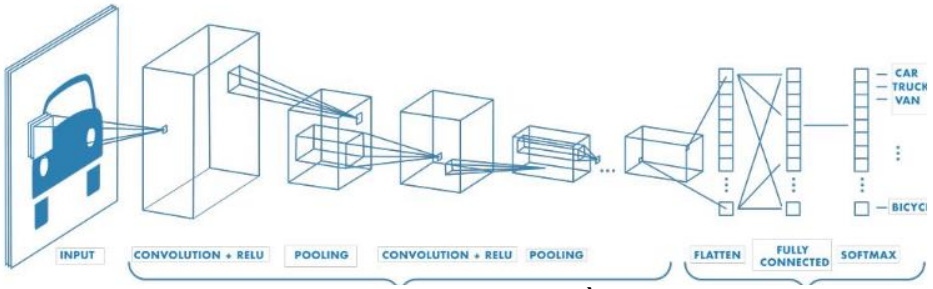
Trong mô hình CNN, các layer liên kết được với nhâu thông qua cơ chế convolution. Layer tiếp theo là kết quả của convolution từ layer trước đó, nhờ vậy có được kết quả kết nối cục bộ. Nghĩa là mỗi nơ ron ở layer tiếp theo sẽ sinh ra từ filter được đặt lên mỗi vùng ảnh của bộ nơ ron layer trước.

Mỗi layer như vậy sẽ được đặt các filter khác nhau, thông thương có vài trăm đến vài nghìn filter. Mỗi layer khác như pooling dùng để lọc lại các thông tin (loại bỏ phần nhiễu).

Trong quá trình huấn luyện, CNN sẽ tự động học được các thông số các filter. Trong phân lớp ảnh, CNN sẽ cố găng tìm ra thông số tối ưu cho các filter tương ứng. layer cuối cùng sẽ được dùng để phân lớp ảnh.

CNN có tính bất biến và tính kết hợp cục bộ. với cùng một đối tượng, nếu đối tượng được chiếu theo các góc độ khác nhau thì dộ chính xác của thuật toán sẽ bị ảnh hưởng.

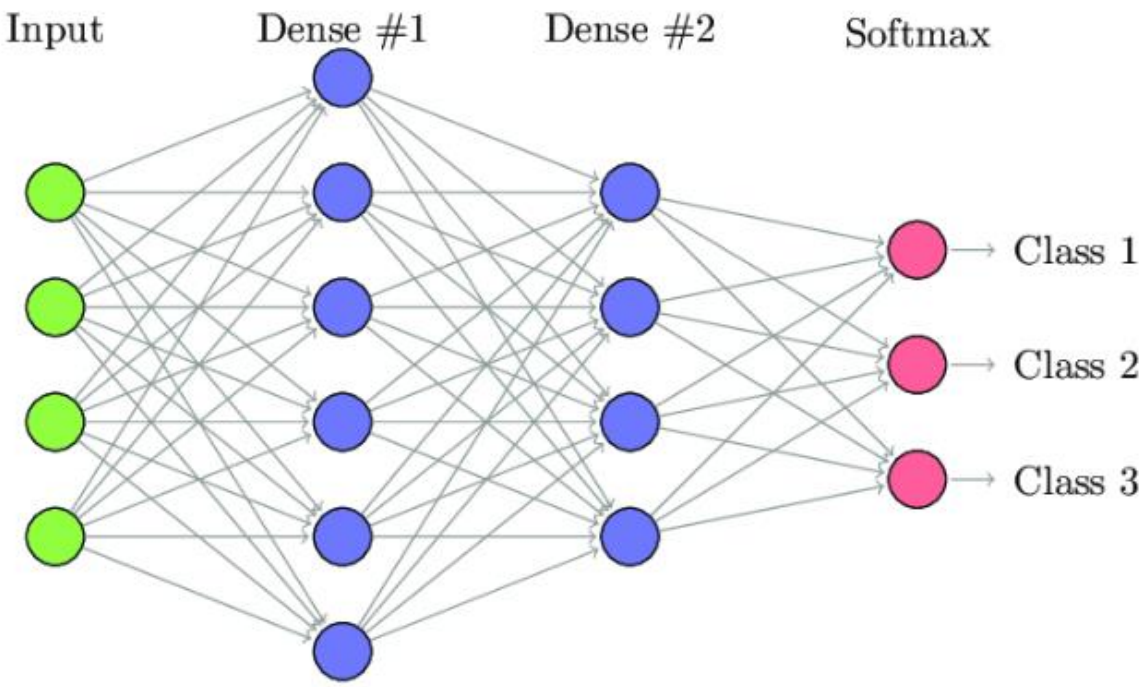
Tính kết cho thấy được các cấp độ biểu diễn thông tin từ mức độ thấp đến muwxc độ casco và trừu tượng hơn thông qua convolution từ các lớp filter. Đó là lý do tại sao CNN cho ra mô hình cới độ chĩnh xác cao. Cũng giống như con người nhận biết được một vật thể trong tự nhiên. Phân biệt được một con chó hay con con mèo nhờ vào các đặc trưng riêng từ mức độ thấp như có 4 chân, có đuôi đến mức độ cao hơn như dáng đi, hình thể, màu lông.



Hình 2.6. Mô hình mạng nơ ron tích chập(Nguồn https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2)

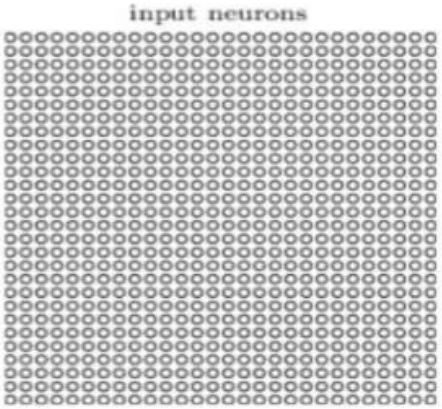
2.4.2. Xây dựng mạng nơ ron tích chập

- **Đầu vào dữ liệu**



Hình 2.7. Mô hình mạng perceptron đa tầng

Đầu vào dữ liệu sẽ là một hình ảnh đã được xử lý thành các điểm ảnh. Đối với mỗi điểm ảnh sẽ được mã hóa cường độ điểm ảnh là giá trị của nơ ron tương ứng trong tầng đầu vào. Đối với ảnh kích thước 28x28 điểm ảnh đang sử dụng, có mạng 784 nơ ron đầu vào. Sau đó huấn luyện trong số và độ lệch để đầu ra của mạng xác đinh chính xác ảnh các chữ số 0, 1,2,3,4,…



Hình 2.8. Các nơ ron đầu vào

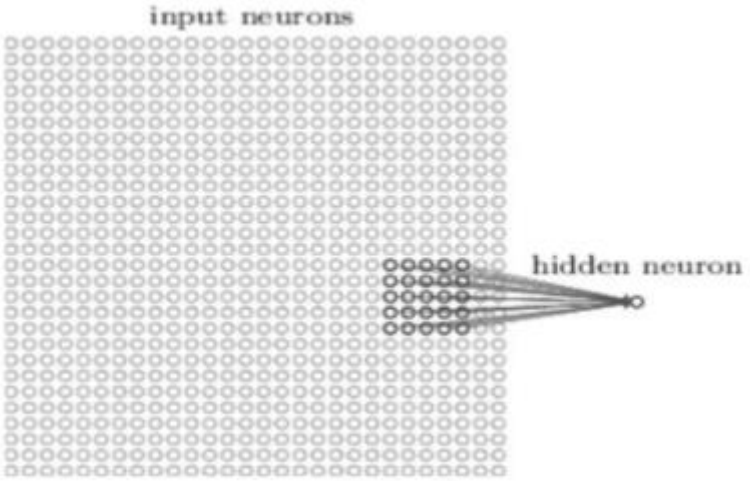
Trong các tầng kết nối đầy đủ, đầu vào được mô tả là một đường thẳng chứa các nơ ron. Trong mạng tích chập, sẽ thay thế đầu vào là 28x28 nơ ron, giá trị tương ứng với 28x28 điểm ảnh.

Bình thường mô hình sẽ kết các điểm ảnh đầu vào cho các nơ ron ở tầng ẩn, nhưng sẽ không kết nối mỗi điểm ảnh đầu vào mỗi no ron. Thay vào đó, sẽ chỉ kết nối trong phạm vị nhỏ, các vùng cục bộ của bước ảnh.

- **Lớp Convolutionl**

Convolutional Neural Network là một trong những phương pháp chính khi sử dụng dữ liệu về ảnh. Kiến trúc mạng này xuất hiện do các phương pháp xử lý dữ liệu ảnh thường sử dụng giá trị của từng pixel.

Mỗi nơ ron trong lớp ẩn đầu tiên sẽ được nối với một vùng nhỏ của các nơ ron đầu vào.



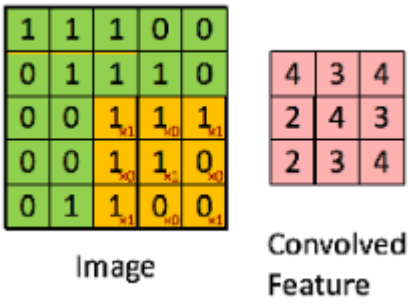
Hình 2.9. Mô hình nơ ron cục bộ

Như hình trên mỗi một vùng 5x5 tương ứng với 25 điểm ảnh đầu vào, hay còn gọi vùng 5x5 là bộ filter. Vùng trong ảnh vùng tiếp nhận cục bộ cho nơ ron ẩn.

Một phép toán trong toán học được gọi là tích chập được thực hiện bằng cách sử dụng Kernel trượt qua ma trận mặt thời gian thực đầu vào. Trên mỗi vị trí, nhân ma trận khuôn mặt được thực hiện và thêm tập hợp kết quả vào bản đồ tính năng cuối cùng. Sử dụng xem xét bộ lọc hạt nhân 2 Chiều là K và đầu vào hình ảnh 2 Chiều là I. Trong trường hợp này, hình ảnh phức hợp được tính như thể hiện bằng công thức.

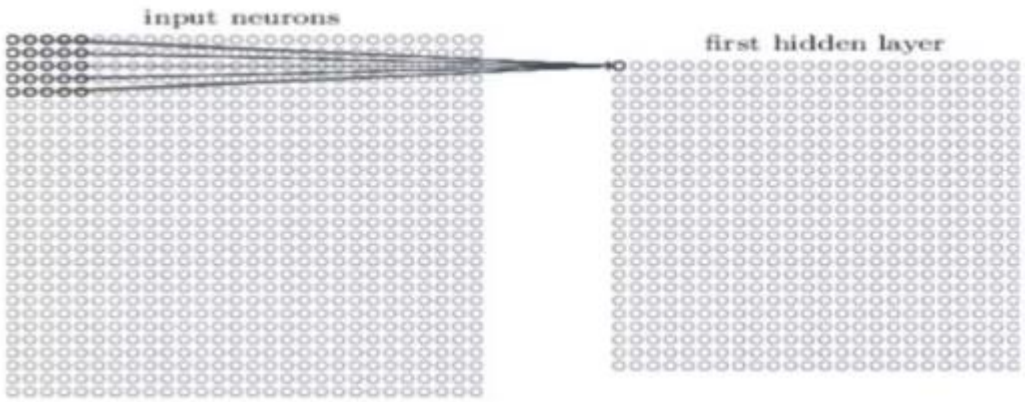
$$(i,j)=\sum m\sum n(m,n)(i-m,j-n)$$

Một hình ảnh được đưa về dạng ma trận và sau đó được nhân với một ma trận lọc (filter matrix) sau khi nhân ma trận với nhau sẽ được một ma trận hình ảnh nhỏ hơn ảnh đầu vào.



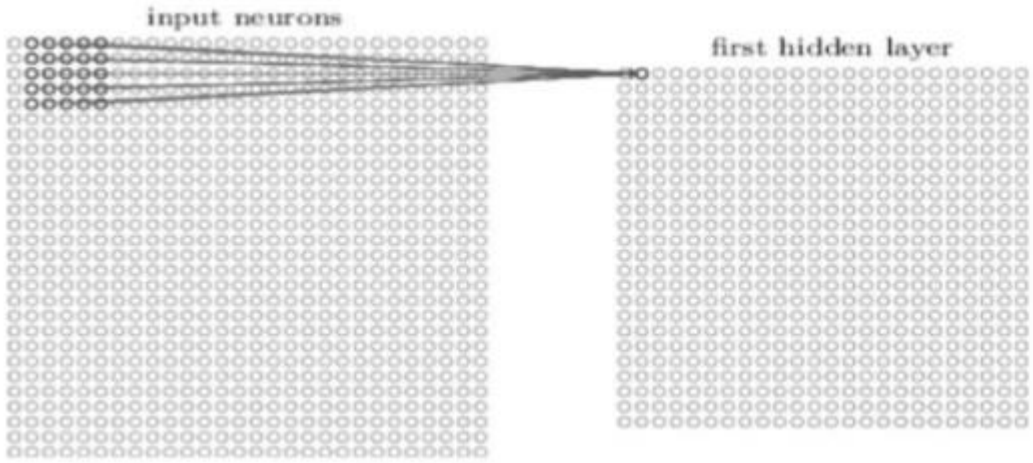
Hình 2.10. Mô hình tích chập (Nguồn https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2)

Từ mô hình và công thức trên ta có thể vẽ mô hình sau.



Hình 2.11. Mô hình nơ ron cục bộ

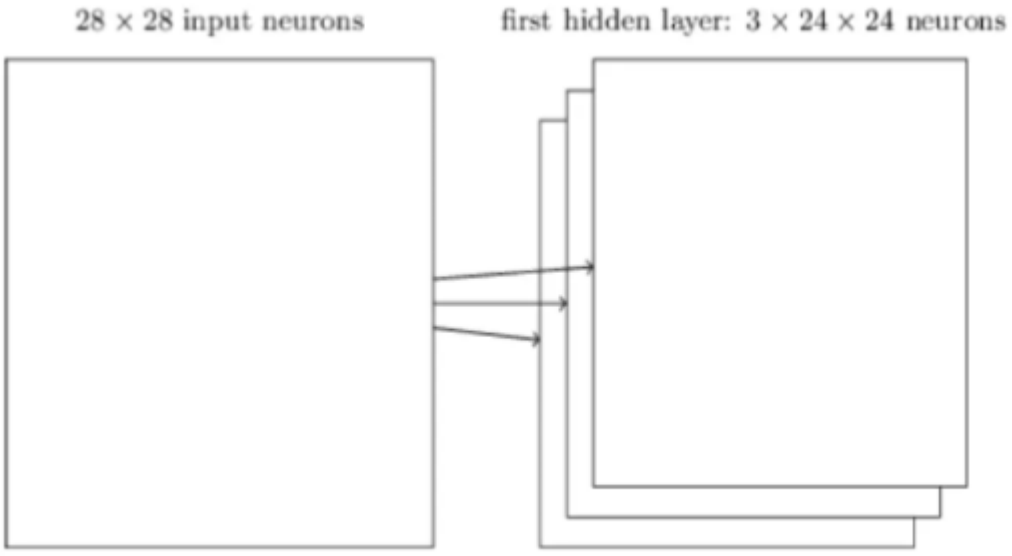
Ta biết được rằng mỗi một vùng 5x5 sẽ tương ứng với một nơ ron. Vùng 5x5 này sẽ trượt trên toàn bộ bức ảnh. Với mỗi một vùng cục bộ sẽ là một nơ ron khác nhau.



Hình 2.12. Mô hình nơ ron cục bộ

Như vậy sau khi vùng cục bộ duyệt qua toàn bộ ảnh thì ta sẽ có được một một ảnh gồm 24x24 nơ ron từ lớp ẩn. Ta có được điều này do chỉ có thể di chuyển qua cục ngang qua 23 nơ ron trước khi chạm vào phía bên phái hoặc dưới của ảnh đầu.

Để nhận phát hiện và nhận dạng hình ảnh thì ta cần nhiều hơn một bản đồ đặc trưng và mỗi lớp tích chập sẽ hoàn chỉnh vài bản đồ đặc trưng.



Hình 2.13. Minh hoạ đặc trưng cấu trúc nơ ron

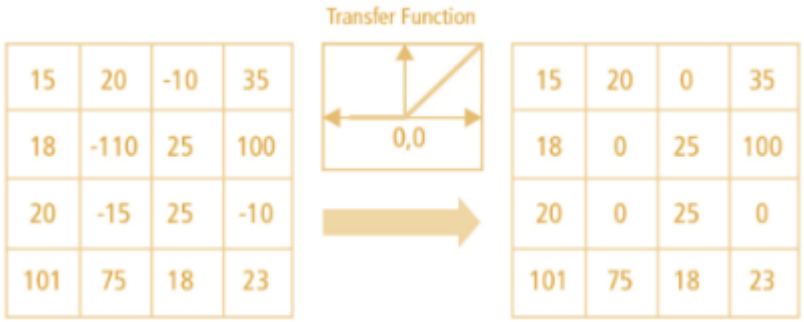
Trên thực tế hình ảnh là một hình 3 chiều nó sẽ vì nó có thêm màu sắc. Với một ảnh có giá trị kích thước 28 x 28 sử dụng kênh RGB ta có tổng cộng ta có 28 * 28 * 3 bằng 2352 nút ở lớp đầu vào, khiến cho số lượng lớn trọng lượng và độ lệch dẫn đến mạng nơ ron trở nên quá nặng cho việc tính toán. Ở đây, màu sắc là một đặc trưng.

Thông tin của các nơ ron thường chỉ chịu tác động bởi các nơ ron ngay gần nó, vậy nên việc bỏ qua một số nút ở tầng đầu vào trong mỗi lần huấn luyện sẽ không làm giảm độ chính xác của mô hình.

Khi thực hiện việc nhân ma trận sẽ rất có nhiều dữ liệu và có nhiều dữ liệu nhỏ 0 mà dữ liệu âm sẽ khiến cho việc tính toán sai mà đôi khi lại gây ra nhưng dữ liệu rất lớn khi nhân vì vậy hàm tuyến tính ReLU sẽ giúp chuyển những dữ liệu âm đó về 0.

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

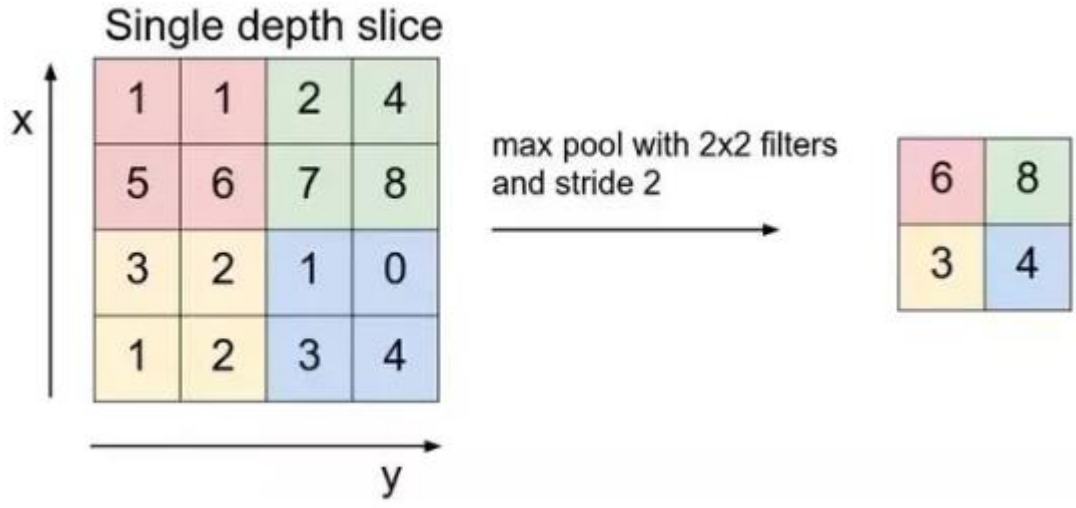
Việc chuyển dữ liệu về 0 sẽ giúp tính toán chính xác mà cũng giúp giảm bớt dữ liệu sai lệch về sau.



Hình 2.14. Mô tả những hàm tuyến tính ReLU (Nguồn https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2)

- **Lớp Pooling**

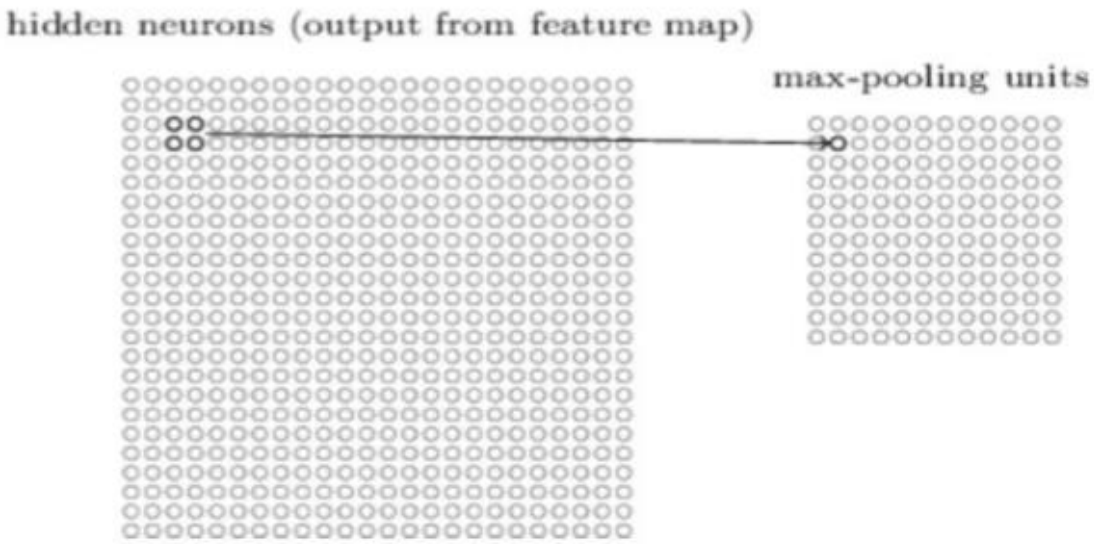
Lớp gộp (Lớp pooling) sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng.



Hình 2.15. Mô tả lớp pooling (https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2)

Có hai lớp pooling chính gồm:

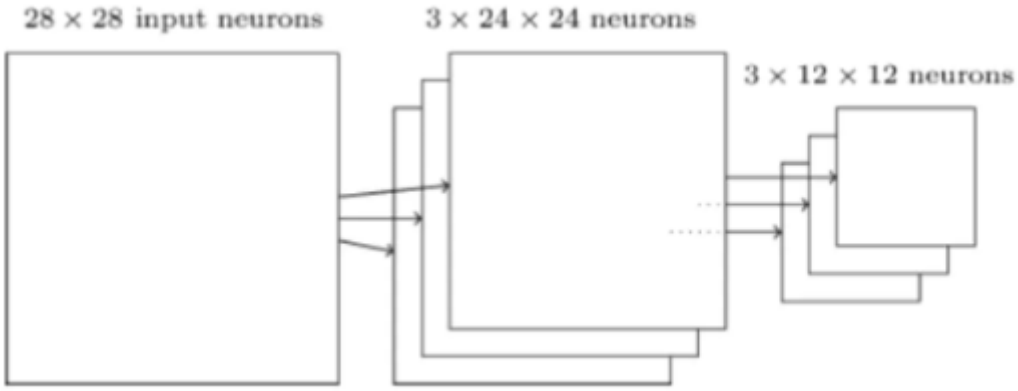
- Max-pooling hay còn gọi là Lớp lấy mẫu cực đại. Max-pooling hoạt động bằng cách áp dụng một cửa sổ di chuyển trên các đặc trưng đầu vào và chọn giá trị lớn nhất từ mỗi cửa sổ. Các cửa sổ thường có kích thước nhỏ, ví dụ như 2x2 hoặc 3x3, và được di chuyển trên toàn bộ đặc trưng đầu vào với một bước nhảy (stride) cố định. Nó có ưu điểm Giữ lại thông tin quan trọng nhất: Bằng cách lựa chọn giá trị lớn nhất, Max Pooling giữ lại các đặc trưng quan trọng nhất từ mỗi cửa sổ và max pooling giúp loại bỏ một phần nhiễu trong dữ liệu bằng cách giữ lại các giá trị cao nhất.
- Average pooling mỗi cửa sổ di chuyển trên các đặc trưng đầu vào và tính trung bình của các giá trị trong cửa sổ. Kết quả là giá trị trung bình của các giá trị trong cửa sổ, tạo ra một biểu diễn trung bình của các đặc trưng trong cửa sổ đó. Giảm nhiễu: Average Pooling giúp giảm đi nhiễu trong dữ liệu bằng cách tạo ra một biểu diễn mềm mại của các đặc trưng trung bình các giá trị có thể giúp tạo ra một biểu diễn đồng đều của dữ liệu.



Hình 2.16. Sơ đồ phân lớp

Trên hình với mỗi đơn vị lớp pooling có thể thu một vùng 2x2 nơ ron trong lớp trước. Pooling phổ biến nhất là max-pooling, một đơn vị pooling chỉ đơn giản là kết quả của đầu ra giá trị lớn nhất trong vùng 2x2.

Vì vó 24x24 nơ ron đầu ra từ lớp tích chập, vậy sau khi pooling sẽ có được 12x12 nơ ron. Như đã phân tích, thì lớp tích chập cho ra nhiều hơn một bản đặc trưng, vì vậy nếu áp dụng max pooling mỗi đặc trưng riêng. Nếu có 3 bản đồ đặc trưng thì sẽ cho ra 3 bản đồ max pooling.

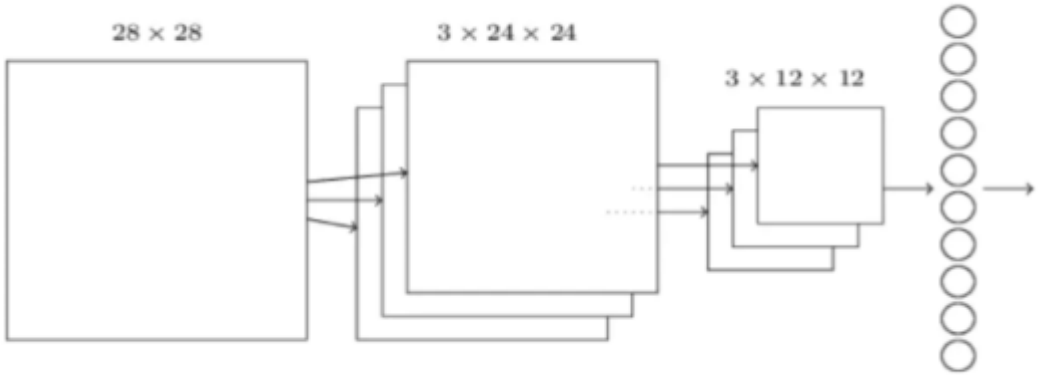


Hình 2.17. Sở đồ phân lớp

Max pooling được hiểu như là một cách để cho biết xem đặc trưng đặc biệt nhất được tìm thấy ở bất cứ đâu trong một khu vực của ảnh. Sau đó nó bỏ đi nhưng thông tin trừ đặc trưng đặc biệt.

- **Lớp Fully Connected**

Lớp được kết nối đầy đủ (FC): Đầu ra cuối cùng của lớp tổng hợp cuối cùng đóng vai trò là đầu vào của Lớp được kết nối đầy đủ trong CNN. Một hoặc nhiều lớp có thể có trong này. Kết nối đầy đủ cho thấy, mọi nút trong lớp ban đầu được kết nối với mọi nút của lớp tiếp theo.



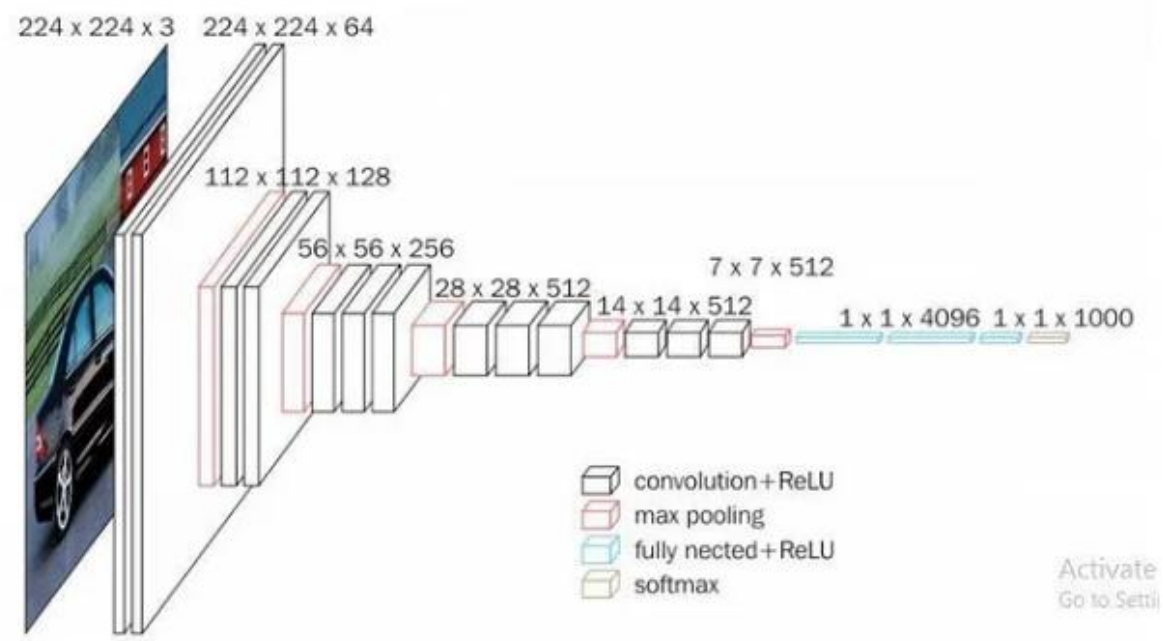
Hình 2.18. Sơ đồ phân lớp

2.5. Lý thuyết về Model VGG16

2.5.1. Model VGG16

Mô hình VGG16 là kiến trúc mạng thần kinh tích chập (CNN). Nó được đặc trưng bởi độ sâu của nó, bao gồm 16 lớp, ở VGG16 trong đó có 13 lớp chập và 3 lớp được kết nối đầy đủ. Kiến trúc của mô hình có một chồng các lớp tích chập, theo sau là các lớp tổng hợp tối đa, với độ sâu tăng dần. Thiết kế này cho phép mô hình tìm hiểu các cách biểu diễn phân cấp phức tạp của các tính năng trực quan, dẫn đến những dự đoán mạnh mẽ và chính xác. Mặc dù đơn giản so với các kiến trúc gần đây hơn, VGG-16 vẫn là lựa chọn phổ biến cho nhiều ứng dụng deep learning do tính linh hoạt và hiệu suất tuyệt vời của nó.

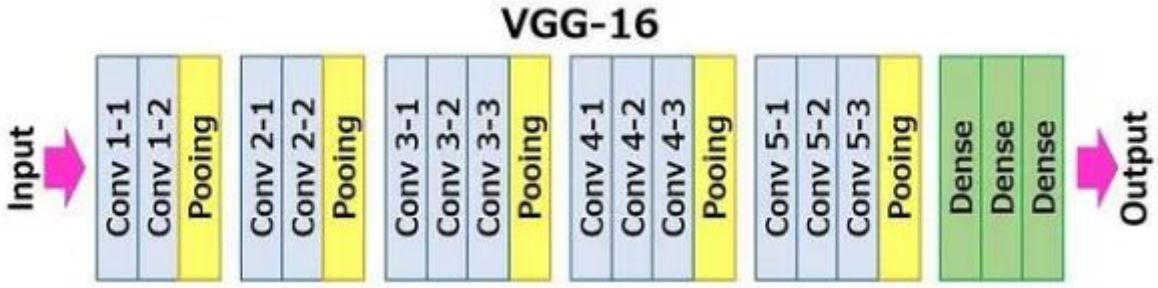
VGG16 là thuật toán phát hiện và phân loại đối tượng có khả năng phân loại 1000 hình ảnh thuộc 1000 danh mục khác nhau với độ chính xác 92,7%. Đây là một trong những thuật toán phổ biến để phân loại hình ảnh và dễ sử dụng với phương pháp học chuyển giao.



Hình 2.19. Mô hình mạng ron VGG16 (Nguồn https://neurohive.io/en/popular-networks/vgg16/)

Model VGG16 dựa trên kiến trúc mạng nơ ron tích chập nhưng nó khác ở số lớp được cố chạy cố định.

2.5.2. Kiến trúc của VGG16



Hình 2.20. Kiến trúc phân lớp VGG16 (https://neurohive.io/en/popular-networks/vgg16/)

- Số 16 trong VGG16 đề cập đến 16 lớp có trọng số. Trong VGG16 cómười ba lớp chập, năm lớp Max Pooling và ba lớp Mật độ cao, tổng cộng có tới 21 lớp nhưng nó chỉ có mười sáu lớp trọng lượng, tức là lớp tham số có thể học được.
- VGG16 lấy kích thước tensor đầu vào là 224, 244 với 3 kênh RGB
- Điều độc đáo nhất về VGG16 là thay vì có một số lượng lớn siêu tham số, họ tập trung vào việc có các lớp tích chập của bộ lọc 3x3 với bước 1 và luôn sử dụng cùng một lớp đệm và lớp maxpool của bộ lọc 2x2 của bước 2.
- Các lớp tích chập và nhóm tối đa được sắp xếp nhất quán trong toàn bộ kiến trúc
- Lớp Conv-1 có 64 bộ lọc, Conv-2 có 128 bộ lọc, Conv-3 có 256 bộ lọc, Conv 4 và Conv 5 có 512 bộ lọc.
- Ba lớp được kết nối đầy đủ (FC) tuân theo một chồng các lớp tích chập: hai lớp đầu tiên có 4096 kênh, mỗi lớp thứ ba thực hiện phân loại ILSVRC 1000 chiều và do đó chứa 1000 kênh (một kênh cho mỗi lớp). Lớp cuối cùng là lớp soft-max.

2.6. Lý thuyết về Model VGG19

2.6.1. Model VGG19

Model VGG19 là một mô hình nâng cao của mô hình VGG16, Mô hình này được thiết kế để nhận dạng hình ảnh và phân loại các đối tượng trong ảnh. VGG19 được biết đến với kiến trúc đơn giản nhưng sâu.

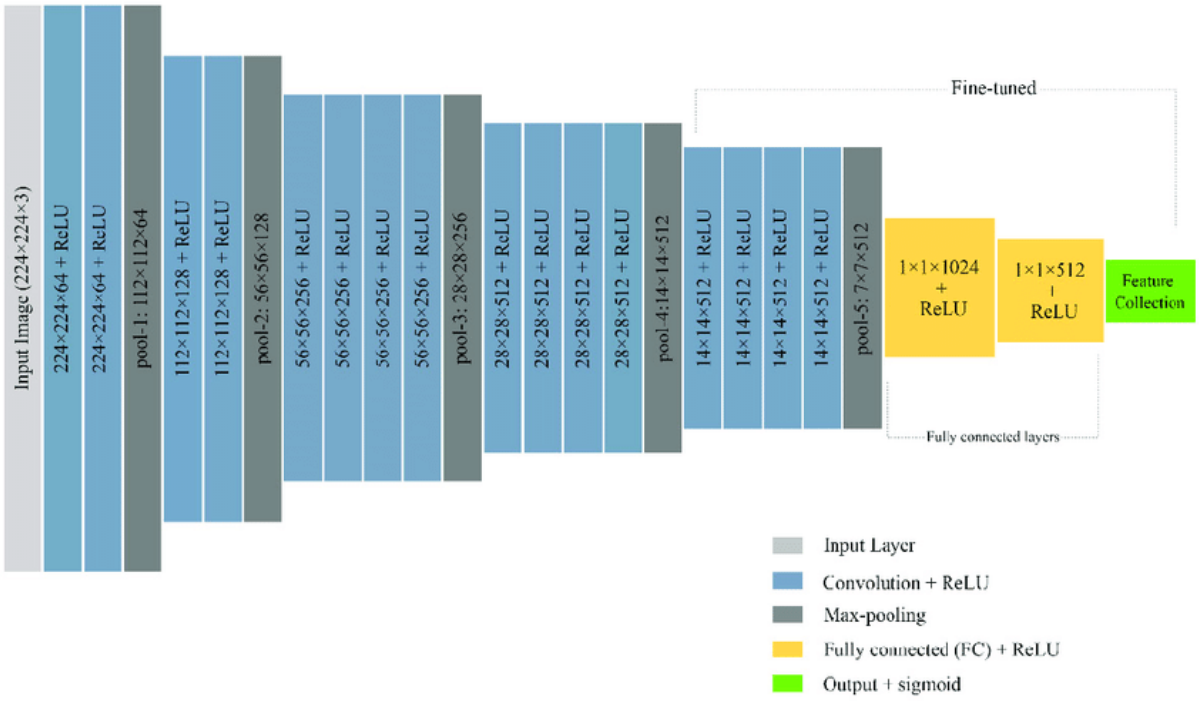
VGG19 bao gồm 19 lớp có trọng số có thể học được, trong đó có 16 lớp tích chập và 3 lớp kết nối đầy đủ. Sử dụng các bộ lọc có kích thước cố định là 3x3 cho tất cả các lớp tích chập. Sử dụng các lớp pooling có kích thước 2x2 với bước nhảy bằng 2 để giảm

kích thước không gian sau mỗi khối tích chập.Mô hình VGG19 yêu cầu kích thước đầu vào cố định là 224x224x3.

2.6.2. Kiến trúc VGG19

Kiến trúc của VGG19 sẽ khác với VGG16 một chút

Kiến trúc của VGG19 được chia thành 5 khối chính (block), mỗi khối bao gồm các lớp tích chập và một lớp pooling:



Hình 2.21. Kiến trúc phân lớp VGG19 (Nguồn https://www.researchgate.net/figure/Illustration-of-fine-tuned-VGG19-pre-trained-CNN-model_fig1_342815128)

CHƯƠNG 3. TRIỂN KHAI HỆ THỐNG PHÂN LOẠI CHAI NHỰA VÀ CHAI THỦY TINH

3.1. Công nghệ sử dụng

Python là một công cụ mạnh mẽ và linh hoạt cho phát triển deep learning, với sự hỗ trợ mạnh mẽ từ các thư viện và framework phong phú, cũng như cộng đồng lớn và nhiệt tình. Sự kết hợp giữa tính linh hoạt, dễ sử dụng và hiệu suất của Python là lý do em chọn nó cho các dự án deep learning. Python có một cộng đồng lớn và phát triển các thư viện và framework mạnh mẽ như TensorFlow, PyTorch, Keras, và scikit-learn, giúp tăng tốc độ quá trình phát triển mô hình deep learning và giảm bớt công sức cần đầu tư.

Hình 3.1. Công nghệ Python (Nguồn http://dien.saodo.edu.vn/nghien-cuu-khoa-hoc/co-ban-ve-ngon-ngu-lap-trinh-python-688.html)

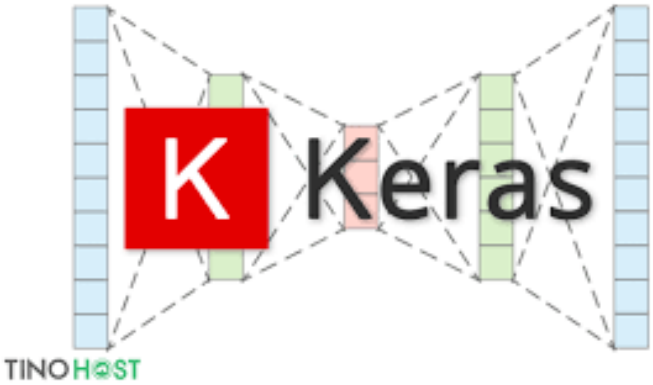
Numpy là thư viện mạnh mẽ cho tính toán khoa học với Python. Nó cung cấp các cấu trúc dữ liệu hiệu quả và các hàm toán học để thao tác với các mảng nhiều chiều (Arrays). Thư viện này thường được sử dụng để chuẩn bị và thao tác với dữ liệu trước khi vẽ biểu đồ bằng Matplotlib.

Pandas cung cấp các cấu trúc dữ liệu và công cụ phân tích dữ liệu mạnh mẽ, đặc biệt là DataFrame, rất hữu ích cho việc thao tác và phân tích dữ liệu dạng bảng. Kết hợp với Matplotlib, Pandas giúp dễ dàng vẽ biểu đồ từ dữ liệu dạng bảng.

Thư viện Học Sâu: TensorFlow và Keras

Hình 3.2. Công nghệ TensorFlow (Nguồn https://blog.cloud-ace.vn/huong-dan-cai-dat-tensorflow-voi-python-tren-window-os/)

- Tensorflow là một thư viện mã nguồn mở phát triển bởi Google Brain Team, được ra mắt lần đầu vào năm 2015. Nó cung cấp một cơ sở hạ tầng mạnh mẽ cho việc xây dựng, huấn luyện và triển khai mô hình deep learning trên một loạt các nền tảng như máy tính để bàn, điện thoại di động và đám mây. TensorFlow cũng cấp các API linh hoạt và mạnh mẽ cho việc xây dựng mạng nơ-ron sâu (deep neural networks) và hỗ trợ tích hợp với nhiều công cụ và framework khác nhau.



Hình 3.3. Công nghệ Keras(Nguồn https://pyimagesearch.com/2020/02/17/autoencoders-with-keras-tensorflow-and-deep-learning/)

- Keras là một thư viện mã nguồn mở đơn giản và dễ sử dụng, được tạo ra bởi François Chollet. Nó được thiết kế để giúp người dùng xây dựng, huấn luyện và triển khai các mô hình deep learning một cách nhanh chóng và dễ dàng. Keras cung cấp một giao diện lập trình ứng dụng (API) cao cấp, cho phép người dùng xây dựng các mô hình deep learning bằng cách sử dụng các tầng (layers), hàm

kích hoạt (Activation functions) và các tối ưu hóa (Optimizers) một cách trực quan và linh hoạt.

Keras và Tensorflow đều là hai công cụ phổ biến và được sử dụng rộng rãi trong cộng đồng deep learning. Sử dụng cả hai giúp nắm bắt được những kỹ thuật và tiêu chuẩn công nghiệp, tạo ra các mô hình có hiệu suất cao và dễ bảo trì. Sử dụng cả hai cùng nhau cung cấp cho sự độc lập trong lựa chọn công cụ. sử dụng Keras cho các tác vụ đơn giản hoặc khi muốn tập trung vào tính năng nhanh chóng, trong khi sử dụng Tensorflow cho các tác vụ phức tạp và yêu cầu tính linh hoạt và tinh chỉnh cao hơn.

Thư viện hình ảnh: Matplotlib

Matplotlib là một thư viện vẽ đồ thị 2D mạnh mẽ và linh hoạt của Python, thường được sử dụng trong việc trực quan hóa dữ liệu khoa học. Matplotlib cung cấp nhiều công cụ và phương pháp để hiển thị và phân tích hình ảnh, từ việc hiển thị ảnh đơn giản đến việc tạo các biểu đồ phức tạp nhằm minh họa kết quả của các mô hình học máy. Em sử dụng để hiển thị các biểu đồ trong trình phân loại chai nhựa và chai thủy tinh.

3.2. Các bước thử nghiệm phân loại

3.2.1. Thu thập dữ liệu.

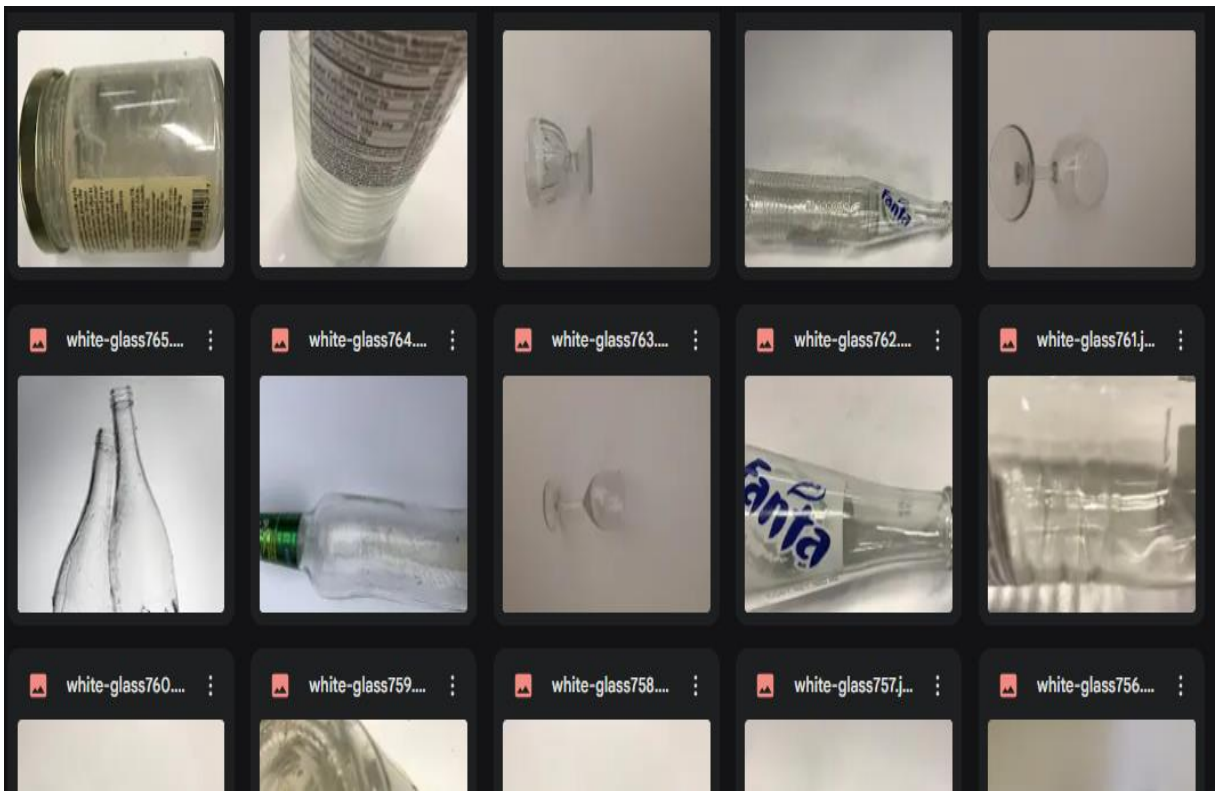
Để lấy hình ảnh chai nhựa và chai thủy tinh nhằm phục vụ cho đề tài phân loại hình ảnh. Em đã sử dụng cho dự án này là một phần của "Images dataset for classifying household garbage" từ Kaggle, do Mostafa Mohamed tạo ra.

Bộ dữ liệu gốc này bao gồm 12 tệp tin về rác thải khác nhau. Trong đề tài này em lấy 4 tệp dữ liệu chính cho đề tài phân loại này.

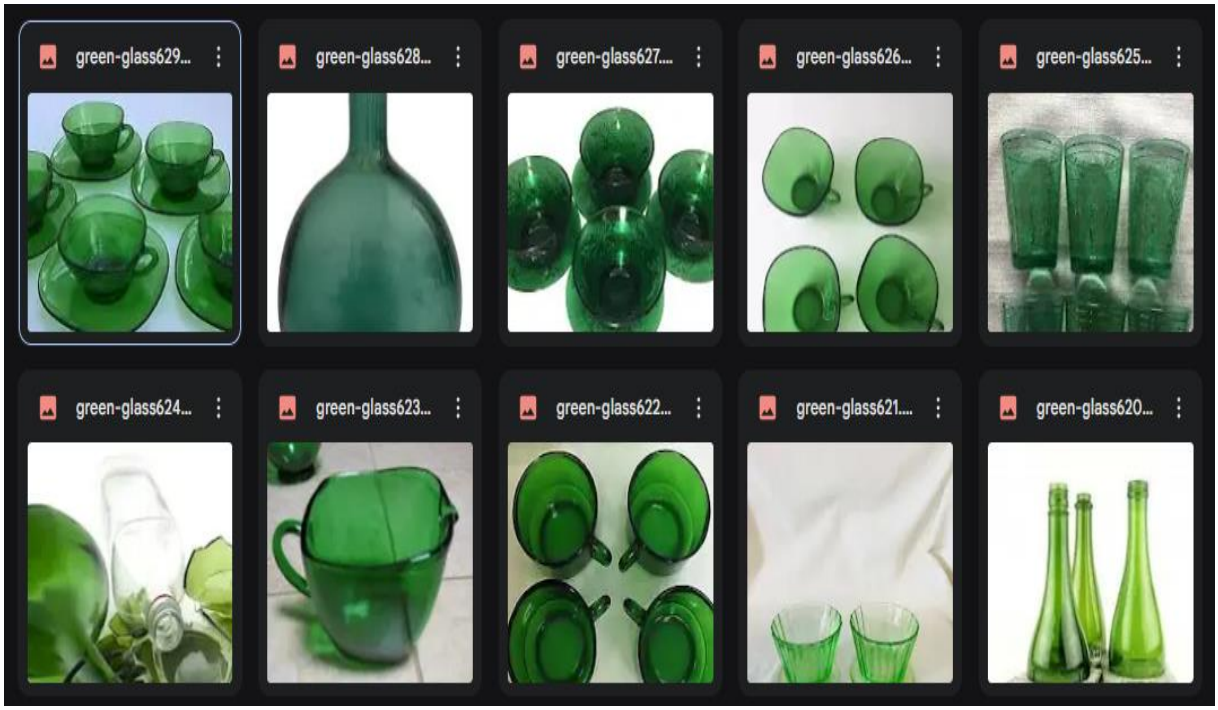
Bộ dữ liệu chứa 2268 hình ảnh RGB, thuộc 4 loại:

- Nhựa (865 hình ảnh)
- Thủy tinh xanh (629 hình ảnh)
- Thủy tinh trắng (775 hình ảnh)
- Thủy tinh nâu (607 hình ảnh)

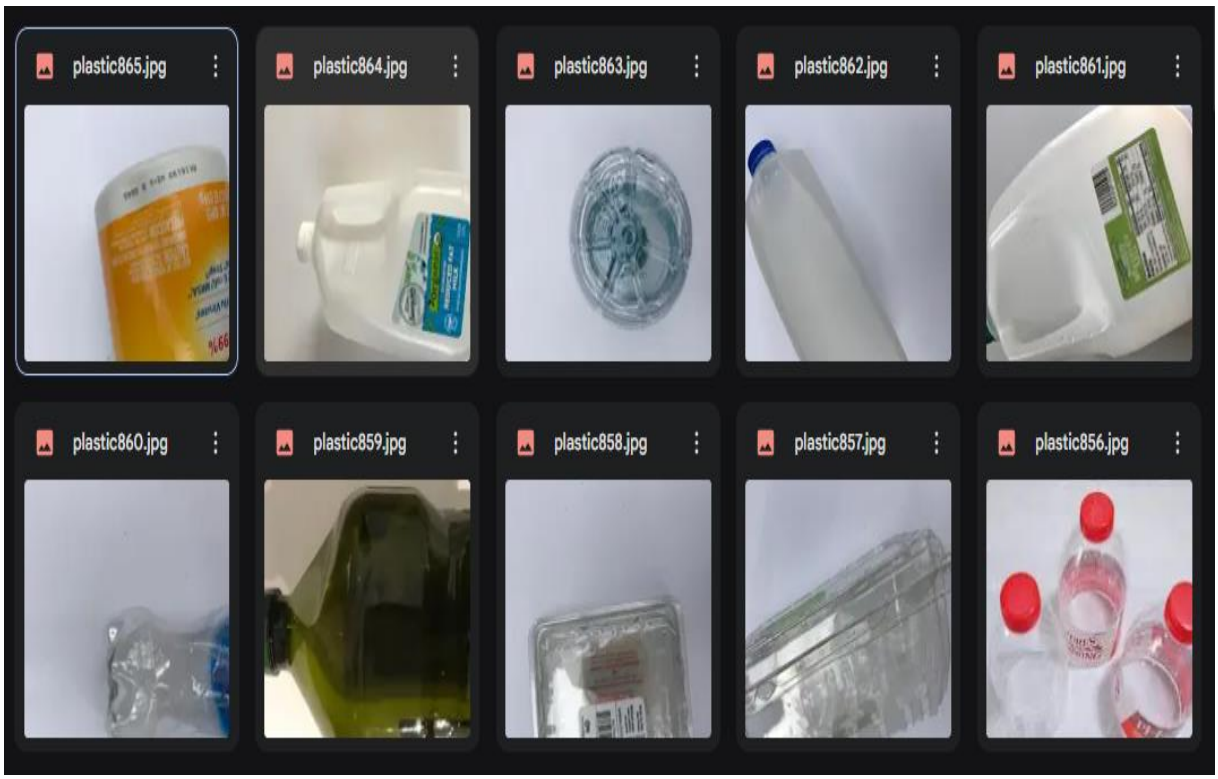
Em phân chia 80% dữ liệu để đào tạo và 20% để thử nghiệm.



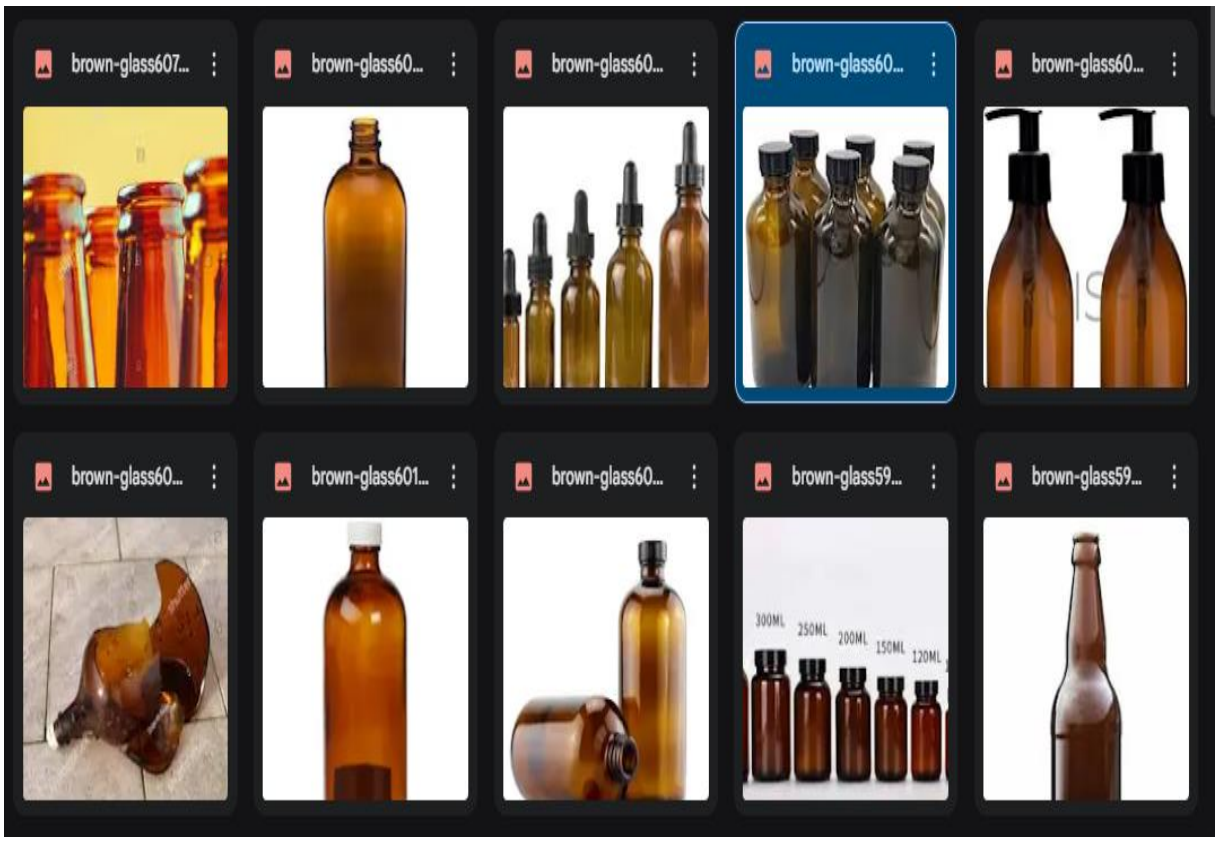
Hình 3.4. Dữ liệu chai thủy tinh trắng



Hình 3.5. Bộ dữ liệu chai thủy tinh xanh



Hình 3.7. Dữ liệu chai nhựa



Hình 3.6. Dữ liệu thủy tinh nâu

3.2.2. Tiền xử lý dữ liệu

- Đầu tiên, xử em xử lý hình ảnh loại bỏ những dữ liệu bị nhiễu như là hình ảnh tờ giấy hoặc những hình ảnh không liên quan đến chai nhựa và thủy tinh.
- Thứ hai là việc xử lý trước hình ảnh có thể giúp đảm bảo rằng chúng có định dạng nhất quán và sẵn sàng để sử dụng làm đầu vào cho CNN. Các kỹ thuật tiền xử lý phổ biến bao gồm thay đổi kích thước, cắt xén và chuẩn hóa.
- Tăng cường dữ liệu liên quan đến việc tạo thêm dữ liệu đào tạo bằng cách áp dụng các phép biến đổi cho dữ liệu hiện có. Điều này có thể giúp ngăn chặn việc trang overfitting và cải thiện hiệu suất của mô hình trên dữ liệu mới, chưa được nhìn thấy
- Do mô hình CNN sẽ mong đợi hình ảnh đầu vào có định dạng cụ thể, chẳng hạn như kích thước và số lượng kênh nhất định. Nên việc Chuyển đổi hình ảnh sang định dạng được yêu cầu là một bước quan trọng trong việc chuẩn bị dữ liệu để sử dụng làm đầu vào cho mô hình.
- Sau khi các dữ liệu đã hoàn thành thì em bắt đầu chia tập huấn luện và tập thử nghiệm để sẵn sàng cho vào model chạy. Em phân chia 80% dữ liệu để đào tạo và 20% để thử nghiệm.

3.2.3. Đào tạo dữ liệu.

- Triển khai CNN bằng TensorFlow và Keras. TensorFlow là một thư viện máy học phổ biến có thể được sử dụng để triển khai và huấn luyện mạng lưới thần kinh. Keras là một API cấp cao để xây dựng và đào tạo các mạng thần kinh chạy trên TensorFlow.
- Sử dụng trọng số lớp để giải quyết các lớp mất cân bằng trong tập dữ liệu. Nếu tập dữ liệu mất cân bằng, với một số lớp có nhiều ví dụ hơn đáng kể so với các lớp khác, bạn có thể sử dụng trọng số lớp để tăng thêm trọng số cho các lớp được trình bày dưới mức. Điều này có thể giúp mô hình học hỏi và phân loại tốt hơn các lớp chưa được trình bày đầy đủ.
- Giám sát hiệu suất của mô hình trên tập xác thực trong quá trình đào tạo, sử dụng các số liệu như độ chính xác và độ mất mát. Nếu mô hình không hoạt động tốt trên tập xác thực, bạn có thể cần điều chỉnh kiến trúc mô hình, các tham số của trình tối ưu hóa hoặc các kỹ thuật tăng cường và tiền xử lý dữ liệu. Khi mô hình

hoạt động tốt trên tập xác thực, bạn có thể huấn luyện nó trên tập dữ liệu đầy đủ và lưu mô hình đã huấn luyện để sử dụng trong tương lai.

3.3. Thực hiện đạt được

3.3.1. Xử lý hình ảnh đầu vào

1. Em muốn xử lý dữ liệu cho về cùng một một kích thước sử dụng hàm sau

```
#chức năng thay đổi kích thước và đệm hình ảnh để sử
dụng sau này trong mô hình

TARGET_HEIGHT = 255
TARGET_WIDTH = 255
def preprocess_img_to_arr(img,
target_height=TARGET_HEIGHT,
target_width=TARGET_WIDTH):
    # Chuyển đổi hình ảnh thành mảng numpy
    img_arr = np.asarray(img)

    # Thay đổi kích thước và đệm hình ảnh
    resized_img =
tensorflow.image.resize_with_pad(img_arr,
target_height, target_width)
    #-------------------------------------------------
-------------------------
        # Lấy kích thước ban đầu của hình ảnh
        #original_height, original_width, _ =
resized_img.shape

        # Tính toán điểm bắt đầu để cắt xén hình ảnh
        #start_x = (original_width - target_width) // 2
        #start_y = (original_height - target_height) //
2

        # Cắt xén hình ảnh
```

```
        #cropped_img = img_arr[start_y:start_y +
target_height, start_x:start_x + target_width, :]
    #-------------------------------------------------
------------------------

    # Chuyển đổi thành tensor TensorFlow
    img_tensor =
tensorflow.convert_to_tensor(resized_img)

    return img_tensor
```

- TARGET_HEIGHT = 255: Đặt chiều cao mục tiêu của hình ảnh sau khi thay đổi kích thước là 255 pixel.
- TARGET_WIDTH = 255: Đặt chiều rộng mục tiêu của hình ảnh sau khi thay đổi kích thước là 255 pixel.
- Defpreprocess_img_to_arr(img,target_height=TARGET_HEIGHT,target_width=TARGET_WIDTH) : Định nghĩa hàm preprocess_img_to_arr với tham số img là hình ảnh đầu vào, và các tham số target_height và target_width là chiều cao và chiều rộng mục tiêu để thay đổi kích thước hình ảnh. Các tham số mặc định được đặt là TARGET_HEIGHT và TARGET_WIDTH.
- img_arr = np.asarray(img): Chuyển đổi hình ảnh img thành một mảng NumPy. Điều này cho phép ta thao tác dễ dàng với các giá trị pixel của hình ảnh.
- resized_img = tf.image.resize_with_pad(img_arr, target_height, target_width):: Thay đổi kích thước hình ảnh img_arr và đệm (padding) để phù hợp với kích thước mục tiêu target_height và target_width. Hàm resize_with_pad sẽ giữ nguyên tỷ lệ của hình ảnh và thêm đệm xung quanh nếu cần thiết để đạt được kích thước mục tiêu.
- img_tensor = tf.convert_to_tensor(resized_img): Chuyển đổi hình ảnh đã thay đổi kích thước resized_img thành một tensor TensorFlow. Tensor là cấu trúc dữ liệu chính trong TensorFlow, được sử dụng trong các bước tiếp theo của quá trình huấn luyện mô hình.

```
[6] array_to_img(preprocess_img_to_arr(img))
```

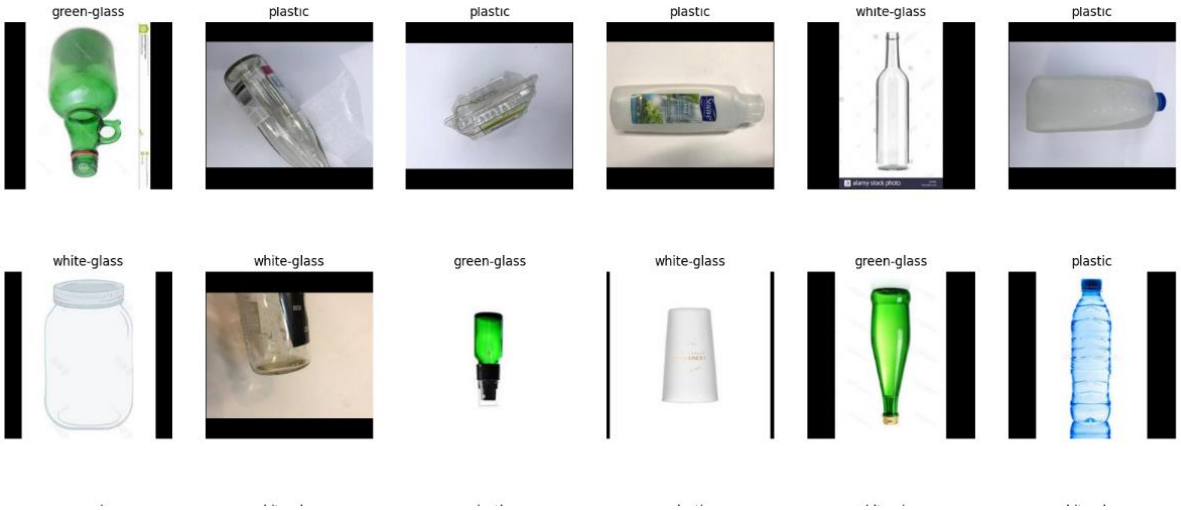
Hình 3.8. Hình ảnh đã được đưa về cùng một kích hước

2. Sau đó lưu trữ dữ liệu đã thay đổi kích thước vào một tệp mới.

```
# lưu trữ hình ảnh được xử lý trước trong thư mục mới
NEW_DIR_PATH = os.path.abspath('new_images')

for folder in folders:
    folder_path = os.path.join(DIR_PATH, folder)
    new_folder_path = os.path.join(NEW_DIR_PATH,
folder)

    if not os.path.exists(new_folder_path):
        os.makedirs(new_folder_path)

    imgs = os.listdir(folder_path)

    for img in imgs:
```

```
        img_path = os.path.join(folder_path, img)
        new_img_path = os.path.join(new_folder_path,
img)

        img = load_img(img_path)
        img_arr = preprocess_img_to_arr(img)
        save_img(new_img_path, img_arr)
```

- if not os.path.exists(NEW_DIR_PATH): : Kiểm tra xem thư mục new_images có tồn tại hay không.
- os.makedirs(NEW_DIR_PATH): Nếu thư mục không tồn tại, tạo thư mục này.
- folder_path = os.path.join(DIR_PATH, folder): : Tạo đường dẫn đầy đủ đến thư mục con hiện tại trong thư mục gốc DIR_PATH.
- new_folder_path = os.path.join(NEW_DIR_PATH, folder): : Tạo đường dẫn đầy đủ đến thư mục con tương ứng trong thư mục new_images.
- imgs = os.listdir(folder_path):: Lấy danh sách tất cả các tệp hình ảnh trong thư mục con hiện tại.
- img_path = os.path.join(folder_path, img_name):: Tạo đường dẫn đầy đủ đến tệp hình ảnh hiện tại.
- new_img_path = os.path.join(new_folder_path, img_name):: Tạo đường dẫn đầy đủ đến tệp hình ảnh trong thư mục new_images.
- img = load_img(img_path):: Tải hình ảnh từ img_path bằng cách sử dụng hàm load_img
- img_arr = preprocess_img_to_arr(img):: Xử lý hình ảnh bằng cách sử dụng hàm preprocess_img_to_arr, chuyển đổi và chuẩn hóa hình ảnh thành tensor TensorFlow.
- save_img(new_img_path, img_arr):: Lưu hình ảnh đã được xử lý vào đường dẫn new_img_path bằng cách sử dụng hàm save_img.



Hình 3.9. Tệp tin lưu các hình ảnh theo kích thước đã sửa đổi.

3. Sau khi đã có một tệp tin lưu trữ ảnh thì ảnh sẽ được mã hóa dưới dạng mã 0 và 1 để bắt đầu sử lý ảnh.

```
#tải dữ liệu và tiền xử lý cho mô hình
def create_image_df(img_folder):
    img_array=[]
    class_name=[]
    img_name=[]

    for img_class_subfolder in os.listdir(img_folder):
        if img_class_subfolder == '.DS_Store':
            continue
        for file in os.listdir(os.path.join(img_folder,
img_class_subfolder)):

            image_path = os.path.join(img_folder,
img_class_subfolder, file)

            img =
tensorflow.keras.utils.load_img(image_path,
target_size=(128, 128), color_mode="rgb")
```

```
            x =
tensorflow.keras.utils.img_to_array(img)
            x = x / 255.0  # Chuẩn hóa giá trị pixel
            img_array.append(x.flatten())
            class_name.append(img_class_subfolder)
            img_name.append(file)

    # Tạo DataFrame từ list
    img_df = pd.DataFrame(img_array)
    img_df['class_name'] = class_name
    img_df['img_id'] = img_name

    return img_df
```

- def create_image_df(img_folder):: Định nghĩa một hàm tên create_image_df với tham số img_folder là đường dẫn đến thư mục chứa các hình ảnh.
- img_array = []: Khởi tạo một danh sách trống để lưu trữ các mảng (array) hình ảnh.
- class_name = []: Khởi tạo một danh sách trống để lưu trữ tên lớp của các hình ảnh.
- img_name = []: Khởi tạo một danh sách trống để lưu trữ tên tệp hình ảnh.
- for img_class_subfolder in os.listdir(img_folder): Lặp qua tất cả các thư mục con trong thư mục img_folder. Mỗi thư mục con tương ứng với một lớp hình ảnh.
- img_class_folder = os.path.join(img_folder, img_class_subfolder): Tạo đường dẫn đầy đủ đến thư mục con hiện tại.
- for file in os.listdir(img_class_folder): Lặp qua tất cả các tệp trong thư mục con hiện tại.
- image_path = os.path.join(img_class_folder, file):Tạo đường dẫn đầy đủ đến tệp hình ảnh hiện tại.
- img = load_img(image_path, target_size=(128, 128), color_mode="rgb"): Tải hình ảnh từ image_path và thay đổi kích thước của nó về (128, 128) pixels với chế độ màu RGB.
- x = img_to_array(img): Chuyển đổi hình ảnh thành một mảng số (numpy array).

- x = x / 255.0: Chuẩn hóa giá trị pixel của hình ảnh về khoảng [0, 1] bằng cách chia các giá trị này cho 255.
- img_array.append(x.flatten()): Chuyển đổi mảng 3D của hình ảnh thành một mảng 1D (flatten array) và thêm vào danh sách img_array.
- class_name.append(img_class_subfolder): Thêm tên lớp của hình ảnh (tên thư mục con) vào danh sách class_name.
- img_name.append(file): Thêm tên tệp hình ảnh vào danh sách img_name.
- img_df = pd.DataFrame(img_array): Tạo một DataFrame từ danh sách img_array.
- img_df['class_name'] = class_name: Thêm cột 'class_name' vào DataFrame với dữ liệu từ danh sách class_name.
- img_df['img_id'] = img_name: Thêm cột 'img_id' vào DataFrame với dữ liệu từ danh sách img_name.
- return img_df: Trả về DataFrame đã được tạo

	0	1	2	3	4	5	6	7	8	9	...	49144	49145	49146	49147
0	1.000000	1.000000	1.000000	1.0	1.0	1.0	1.000000	1.000000	1.000000	1.0	...	1.0	1.0	1.0	1.0
1	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.0
2	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.0
3	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.0
4	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2263	0.007843	0.007843	0.007843	0.0	0.0	0.0	0.015686	0.015686	0.015686	1.0	...	0.0	0.0	0.0	0.0
2264	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.0
2265	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.0
2266	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.0
2267	1.000000	1.000000	1.000000	1.0	1.0	1.0	1.000000	1.000000	1.000000	1.0	...	1.0	1.0	1.0	1.0

2268 rows × 49154 columns

Hình 3.10. Convert dữ liệu hình ảnh về dạng nhị phân

Chia mô hình thành 2 tập

```
features = image_df.iloc[:,2:]
target = image_df.iloc[:,1:2]

# 80%/20%
X_train, X_val, y_train, y_val =
train_test_split(features, target, test_size=.20,
random_state=42, stratify=target)
```

```
#xác định nhãn
y_train_label = y_train.values.astype(object)
y_val_label = y_val.values.astype(object)

#Nhãn mã hóa
encoder = preprocessing.LabelEncoder()
y_train = encoder.fit_transform(y_train.values.ravel())
y_val = encoder.transform(y_val.values.ravel())

#tăng cường dữ liệu
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

#PCA
pca = PCA(n_components=2)
X_train_2PC = pca.fit_transform(X_train_scaled)
X_val_2PC = pca.transform(X_val_scaled)
```

- image_df.iloc[:,2:]: Chọn tất cả các hàng và các cột từ cột thứ 3 đến cột cuối cùng trong image_df. Phần này được gán cho features (đặc trưng).
- image_df.iloc[:,1:2]: Chọn tất cả các hàng và cột thứ 2 trong image_df. Phần này được gán cho target
- train_test_split(features, target, test_size=.20, random_state=42, stratify=target): Chia features và target thành các tập huấn luyện và tập kiểm tra với tỉ lệ 80/20.
- test_size=.20: 20% dữ liệu sẽ được sử dụng cho kiểm tra.
- random_state=42: Đảm bảo chia dữ liệu có thể tái hiện.
- stratify=target: Đảm bảo tỷ lệ các lớp trong target được duy trì trong cả tập huấn luyện và tập kiểm tra.
- y_train.values.astype(object): Chuyển đổi giá trị của y_train sang kiểu object và gán cho y_train_label.
- y_val.values.astype(object): Chuyển đổi giá trị của y_val sang kiểu object và gán cho y_val_label.
- encoder = preprocessing.LabelEncoder(): Khởi tạo một đối tượng LabelEncoder.
- y_train = encoder.fit_transform(y_train.values.ravel()): Mã hóa nhãn trong y_train và chuyển đổi chúng thành định dạng số.

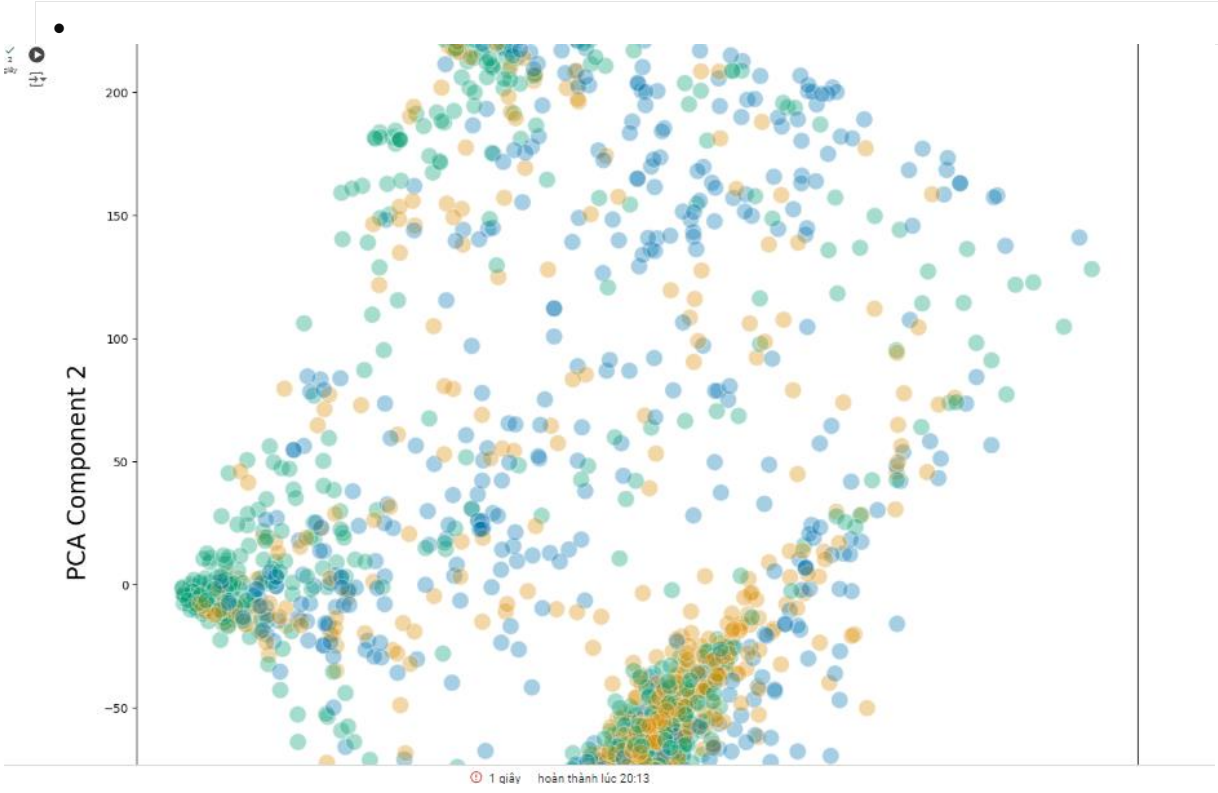
- y_val = encoder.transform(y_val.values.ravel()): Mã hóa nhãn trong y_val sử dụng cùng một bộ mã hóa để đảm bảo tính nhất quán.
- scaler = StandardScaler(): Khởi tạo một đối tượng StandardScaler.
- X_train_scaled = scaler.fit_transform(X_train): Áp dụng chuẩn hóa lên X_train để có trung bình bằng 0 và độ lệch chuẩn bằng 1.
- X_val_scaled = scaler.transform(X_val): Áp dụng chuẩn hóa lên X_val sử dụng cùng một bộ chuẩn hóa để đảm bảo tính nhất quán.
- pca = PCA(n_components=2): Khởi tạo một đối tượng PCA để giảm chiều dữ liệu xuống 2 thành phần chính.
- X_train_2PC = pca.fit_transform(X_train_scaled): Áp dụng PCA lên dữ liệu huấn luyện đã chuẩn hóa và chuyển đổi chúng thành 2 thành phần chính.
- X_val_2PC = pca.transform(X_val_scaled): Chuyển đổi dữ liệu kiểm tra đã chuẩn hóa thành 2 thành phần chính sử dụng cùng một PCA.

3.3.2. Huấn luyện mô hình hồi quy Logistic

```
# Khởi tạo mô hình
model = LogisticRegression(
# Huấn luyện mô hình trên tập huấn luyện
model.fit(X_train_2PC, y_train)
# Dự đoán nhãn trên tập kiểm tra
y_pred = model.predict(X_val_2PC)

# Tính độ chính xác của mô hình trên tập kiểm tra
accuracy = accuracy_score(y_val, y_pred)
print("Độ chính xác của mô hình trên tập kiểm tra:",
accuracy)
```

- model = LogisticRegression(): Khởi tạo một đối tượng LogisticRegression để sử dụng mô hình hồi quy logistic.
- model.fit(X_train_2PC, y_train): Huấn luyện mô hình hồi quy logistic với dữ liệu huấn luyện X_train_2PC (đặc trưng đã giảm chiều) và y_train.
- y_pred = model.predict(X_val_2PC): Sử dụng mô hình đã được huấn luyện để dự đoán nhãn cho dữ liệu kiểm tra X_val_2PC.

- accuracy = accuracy_score(y_val, y_pred): Tính độ chính xác của mô hình bằng cách so sánh nhãn dự đoán y_pred với nhãn thực tế y_val.
- print("Độ chính xác của mô hình trên tập kiểm tra:", accuracy): In ra màn hình độ chính xác của mô hình trên tập kiểm tra.



Hình 3.11. Mô hình chực quan model logistic

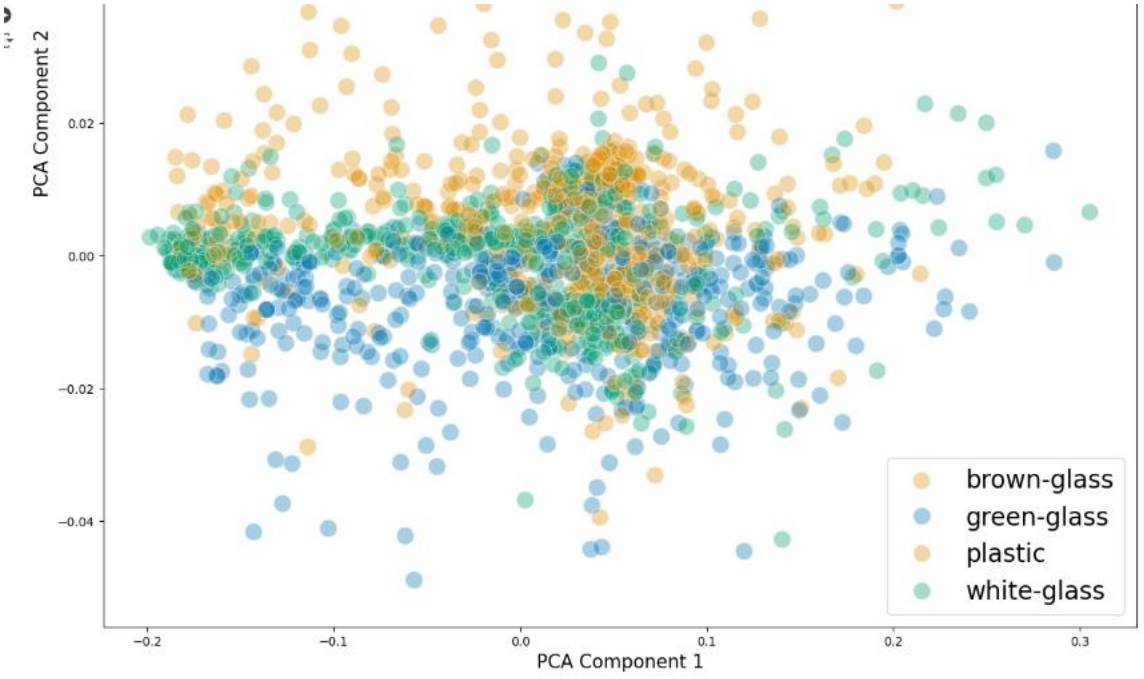
3.3.3. Huấn luyện mô hình CNN

```
CNN = Sequential()

CNN.add(InputLayer(input_shape=X_train_r.shape[1:]))

CNN.add(Conv2D(filters=32, kernel_size=3,
activation='relu', padding='same'))
#CNN.add(Conv2D(filters=16, kernel_size=3,
activation='relu', padding='same'))
CNN.add(MaxPooling2D())

CNN.add(Conv2D(filters=64, kernel_size=3,
activation='relu', padding='same'))
```

```
# NN.add(Conv2D(filters=16, kernel_size=3,
activation='relu', padding='same'))
CNN.add(MaxPooling2D())

CNN.add(Conv2D(filters=128, kernel_size=3,
activation='relu', padding='same'))
#CNN.add(Conv2D(filters=256, kernel_size=3,
activation='relu', padding='same'))

CNN.add(GlobalAveragePooling2D())

CNN.add(Dense(256, activation='relu'))
CNN.add(Dense(128, activation='relu',name = '2D_layer'))
CNN.add(Dense(4, activation='softmax'))

CNN.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=[f1_m],
)
```

- CNN = Sequential(): Tạo một mô hình tuần tự Sequential, nơi em xếp chồng các lớp một cách tuần tự.
- CNN.add(InputLayer(input_shape=X_train_r.shape[1:])): Thêm một lớp InputLayer để xác định hình dạng của dữ liệu đầu vào. X_train_r.shape[1:] là hình dạng của dữ liệu đầu vào, bỏ qua chiều batch.
- CNN.add(Conv2D(filters=32, kernel_size=3, activation='relu', padding='same')): Thêm một lớp Conv2D với 32 bộ lọc, kích thước kernel là 3x3, hàm kích hoạt là ReLU và chế độ padding là "same" để giữ nguyên kích thước đầu ra.
- CNN.add(MaxPooling2D()): Thêm một lớp MaxPooling2D để giảm kích thước của đầu ra bằng cách chọn giá trị lớn nhất từ các vùng.

- CNN.add(Conv2D(filters=64, kernel_size=3, activation='relu', padding='same')): Thêm một lớp Conv2D khác với 64 bộ lọc.
- CNN.add(MaxPooling2D()): Thêm một lớp MaxPooling2D khác.
- CNN.add(Conv2D(filters = 128, kernel_size = 3,activation = 'relu' , padding = 'same')): Thêm một lớp Conv2D với 128 bộ lọc.
- CNN.add(GlobalAveragePooling2D()): Thêm một lớp GlobalAveragePooling2D để chuyển đổi đầu ra từ dạng ma trận 2D thành dạng vector 1D bằng cách tính trung bình của tất cả các giá trị.
- CNN.add(Dense(256, activation='relu')): Thêm một lớp Dense với 256 đơn vị và hàm kích hoạt ReLU.
- CNN.add(Dense(128, activation='relu',name = '2D_layer')): Thêm một lớp Dense khác với 128 đơn vị và hàm kích hoạt ReLU, có tên là '2D_layer'.
- CNN.add(Dense(4, activation='softmax')): Thêm một lớp Dense cuối cùng với 4 đơn vị và hàm kích hoạt softmax để dự đoán xác suất của 4 lớp đầu ra.
- CNN.compile(): Biên dịch mô hình với hàm mất mát là 'categorical_crossentropy', tối ưu hóa là 'adam', và sử dụng metric là f1_m.



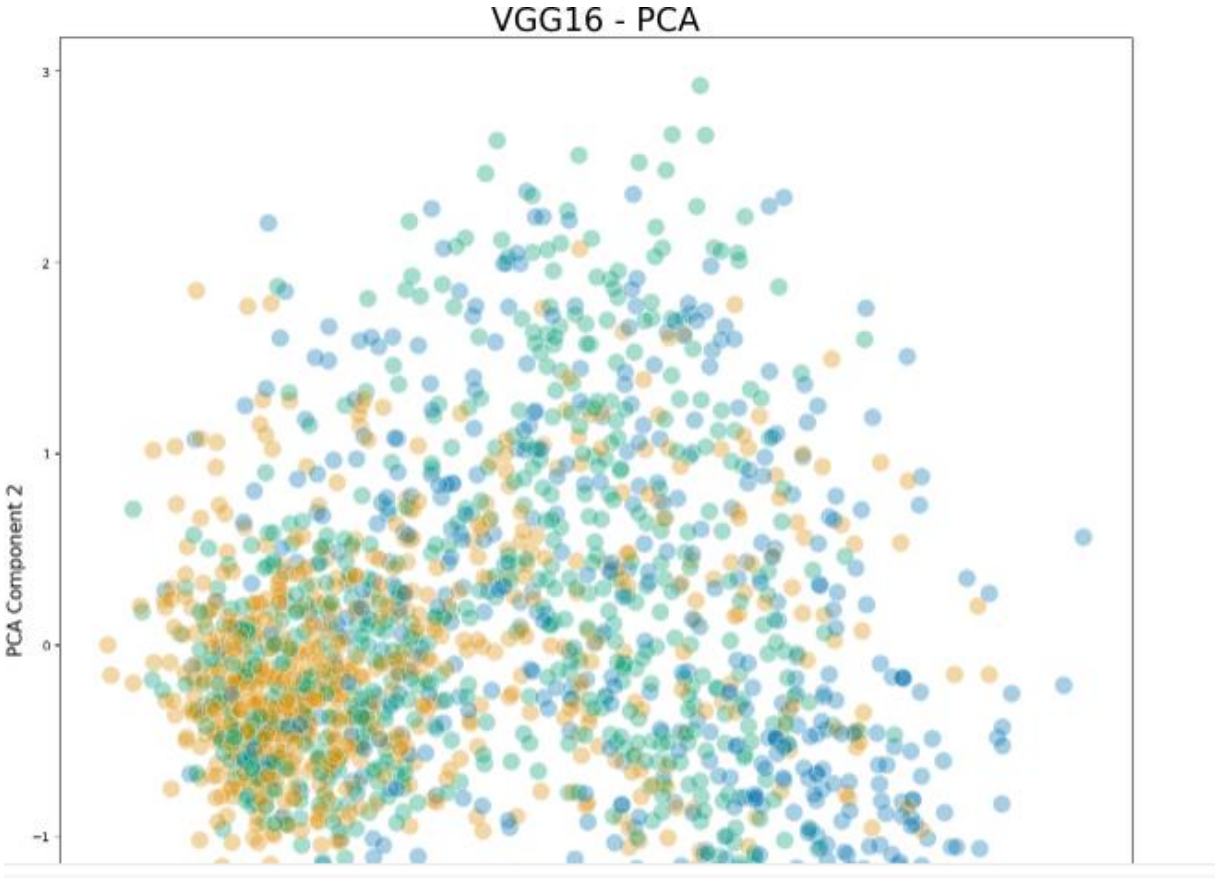
Hình 3.12. Chực quan dữ liệu cho mô hình CNN

3.3.4. Huấn luyện mô hình VGG16

```
VGG16_base_model =
tensorflow.keras.applications.VGG16(weights='imagenet',
include_top=False, input_shape=X_train_r.shape[1:])

for layer in VGG16_base_model.layers:
    layer.trainable = False

x = VGG16_base_model.output

x = Flatten()(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='elu',name = '2D_layer')(x)

predictions = Dense(4, activation='softmax')(x)

VGG16 = Model(inputs=VGG16_base_model.input,
outputs=predictions)
VGG16.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=[f1_m])
```

- VGG16_base_model = tensorflow.keras.applications.VGG16(weights='imagenet', include_top=False, input_shape=X_train_r.shape[1:]): Tạo một mô hình VGG16 sử dụng trọng số được đào tạo trước từ ImageNet, loại bỏ lớp Fully Connected (FC) đầu ra (include_top=False), và xác định hình dạng của dữ liệu đầu vào bằng X_train_r.shape[1:].
- for layer in VGG16_base_model.layers: và layer.trainable = False: Đóng băng các lớp convolutional của mô hình VGG16 để giữ cho trọng số của chúng không được cập nhật trong quá trình huấn luyện.
- x = VGG16_base_model.output: Lấy đầu ra của mô hình VGG16_base_model.
- x = Flatten()(x): Thêm một lớp Flatten để chuyển đổi tensor đầu ra từ dạng ma trận thành dạng vector phẳng để sử dụng trong các lớp kết nối đầy đủ.

- x = Dense(512, activation='relu')(x): Thêm một lớp kết nối đầy đủ với 512 đơn vị và hàm kích hoạt ReLU.
- x = Dense(256, activation='elu',name = '2D_layer')(x): Thêm một lớp kết nối đầy đủ khác với 256 đơn vị và hàm kích hoạt ELU, có tên là '2D_layer'.
- predictions = Dense(4, activation='softmax')(x): Thêm một lớp kết nối đầy đủ cuối cùng với 4 đơn vị và hàm kích hoạt softmax để dự đoán xác suất của 4 lớp đầu ra.
- VGG16 = Model(inputs=VGG16_base_model.input, outputs=predictions): Tạo một đối tượng mô hình mới, sử dụng đầu vào từ VGG16_base_model và đầu ra từ lớp dự đoán.
- VGG16.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[f1_m]): Biên dịch mô hình với tối ưu hóa Adam, hàm mất mát là 'categorical_crossentropy', và sử dụng metric là f1_m.

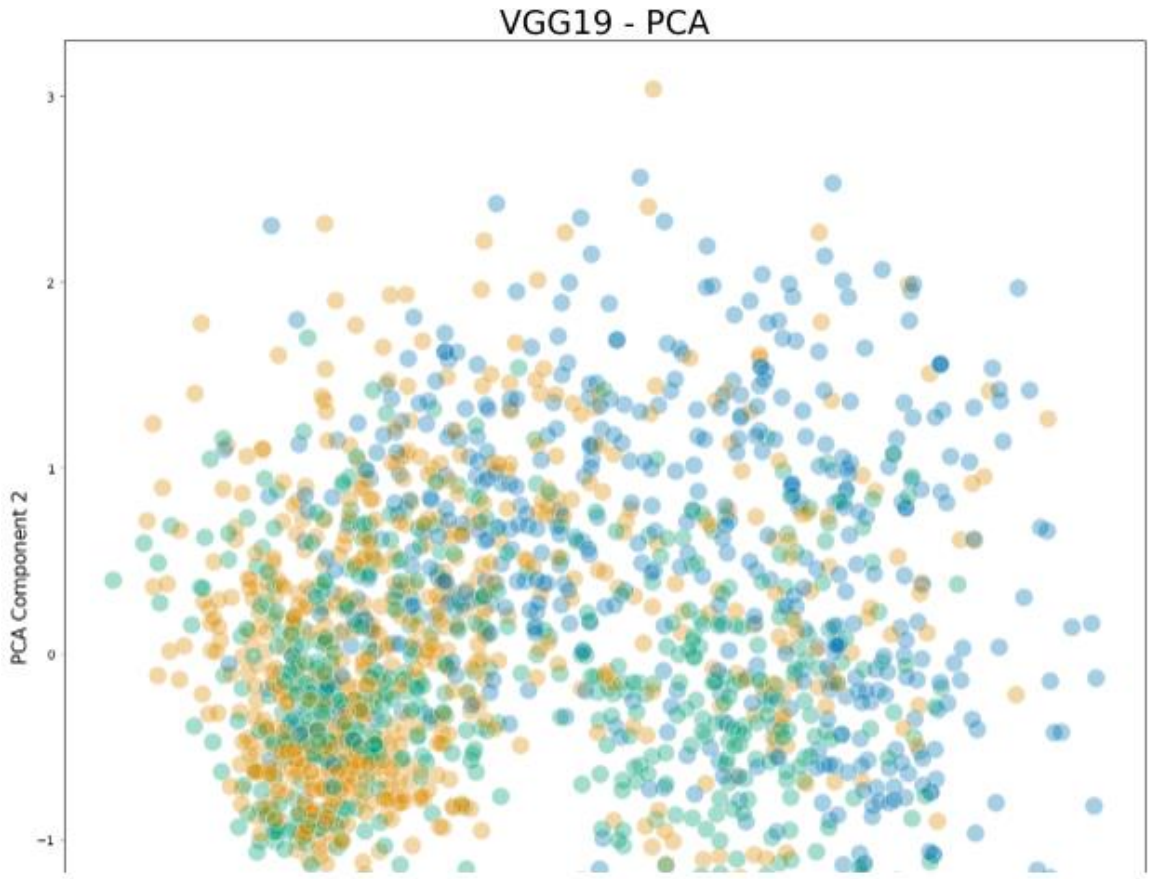


Hình 3.13. Chực quan dữ liệu cho model VGG16

3.3.5. *Huấn luyện mô hình VGG19*

```
VGG19_base_model =
tensorflow.keras.applications.VGG19(weights='imagenet',
include_top=False, input_shape=X_train_r.shape[1:])

#base_model =
tensorflow.keras.applications.Xception(weights='imagenet',
include_top=False, input_shape=X_train.shape[1:])

# Freeze convolutional layers
for layer in VGG19_base_model.layers:
    layer.trainable = False

# Establish new fully connected block
x = VGG19_base_model.output
#x = GlobalAveragePooling2D()(x) #experiment
x = Flatten()(x) # flatten from convolution tensor output
#x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x) # number of layers
and units are hyperparameters, as usual
x = Dense(256, activation='elu',name = '2D_layer')(x)

predictions = Dense(4, activation='softmax')(x) # should
match # of classes predicted

# define formal model object to train and compile it as
usual
VGG19 = Model(inputs=VGG19_base_model.input,
outputs=predictions)
VGG19.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=[f1_m])
```

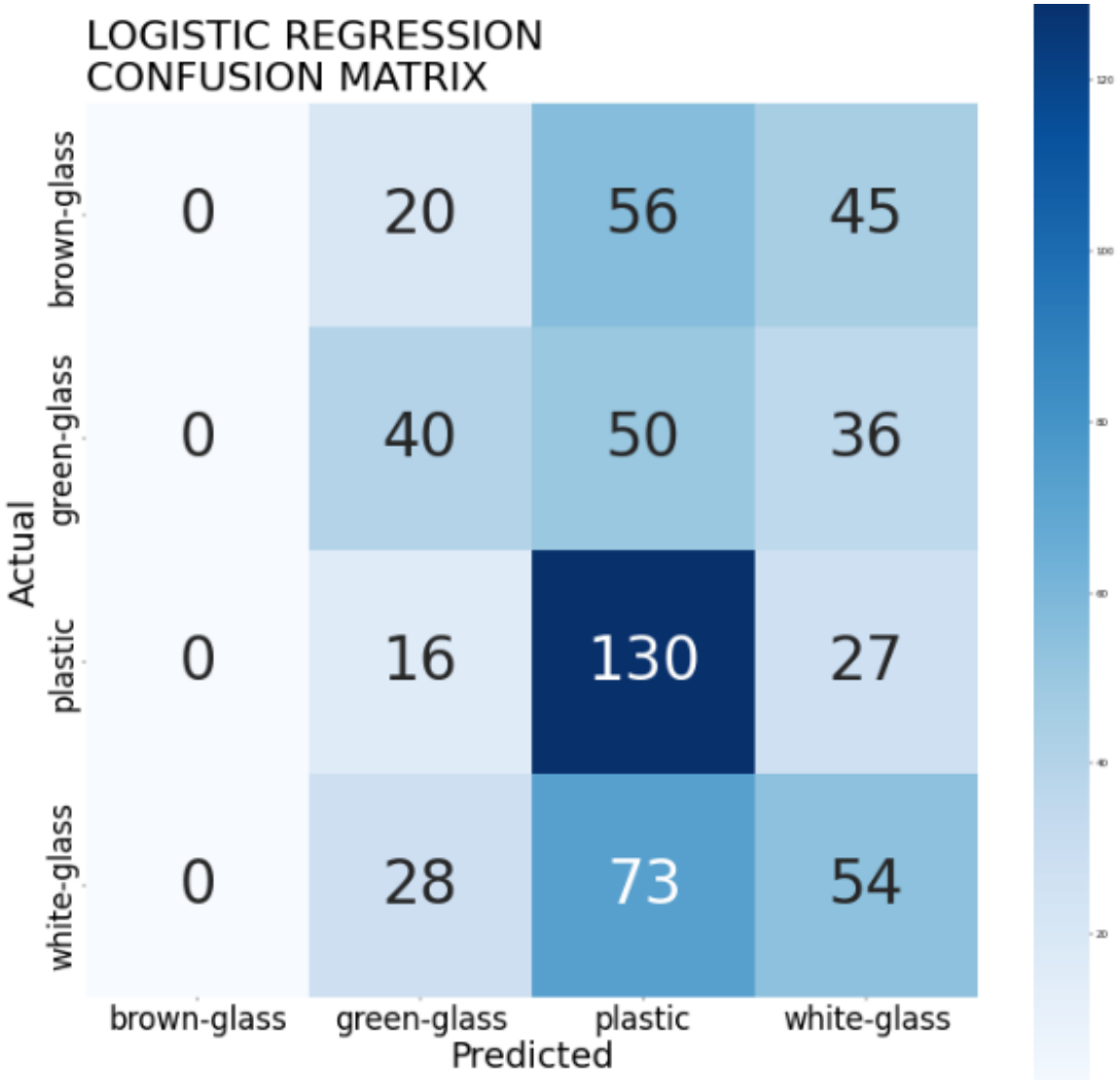
- VGG19_base_model = tensorflow.keras.applications.VGG19(weights='imagenet', include_top=False, input_shape=X_train_r.shape[1:]): Tạo một mô hình VGG19 sử dụng trọng số được đào tạo trước từ ImageNet, loại bỏ lớp Fully Connected (FC) đầu ra (include_top=False), và xác định hình dạng của dữ liệu đầu vào bằng X_train_r.shape[1:].
- for layer in VGG19_base_model.layers: và layer.trainable = False: Đóng băng các lớp convolutional của mô hình VGG19 để giữ cho trọng số của chúng không được cập nhật trong quá trình huấn luyện.
- x = VGG19_base_model.output: Lấy đầu ra của mô hình VGG19_base_model.
- x = Flatten()(x): Thêm một lớp Flatten để chuyển đổi tensor đầu ra từ dạng ma trận thành dạng vector phẳng để sử dụng trong các lớp kết nối đầy đủ.
- x = Dense(512, activation='relu')(x): Thêm một lớp kết nối đầy đủ với 512 đơn vị và hàm kích hoạt ReLU.
- x = Dense(256, activation='elu',name = '2D_layer')(x): Thêm một lớp kết nối đầy đủ khác với 256 đơn vị và hàm kích hoạt ELU, có tên là '2D_layer'.
- predictions = Dense(4, activation='softmax')(x): Thêm một lớp kết nối đầy đủ cuối cùng với 4 đơn vị và hàm kích hoạt softmax để dự đoán xác suất của 4 lớp đầu ra.
- VGG19 = Model(inputs=VGG19_base_model.input, outputs=predictions): Tạo một đối tượng mô hình mới, sử dụng đầu vào từ VGG19_base_model và đầu ra từ lớp dự đoán.
- VGG19.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[f1_m]): Biên dịch mô hình với tối ưu hóa Adam, hàm mất mát là 'categorical_crossentropy', và sử dụng metric là f1_m.



Hình 3.14. Chực quan dữ liệu cho model VGG19

3.4. Đánh giá mô hình

- **Đánh giá mô hình Logistic Regression**

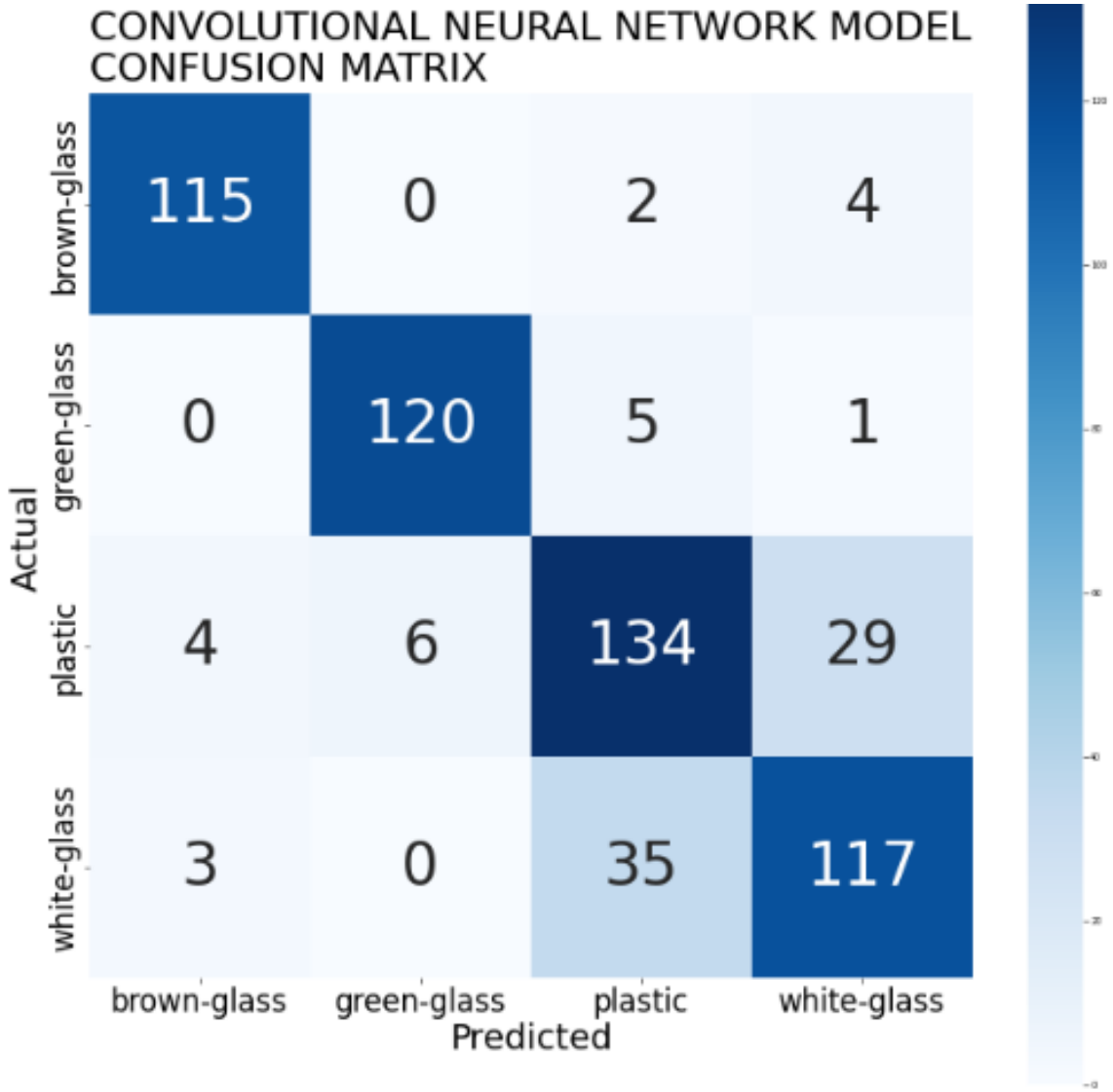


Hình 3.15. Số lần dự đoán đúng sai của mô hình hồi quy logistic

Biểu đồ trên chỉ ra được số lần dự đoán chính xác và đoán sai của mô hình hồi quy logistic. Như hình trên mô hình đã dự đoán đúng 130 lần với plastic và dự đoán sai tới 50 lần với ảnh thủy tinh xanh, 73 lần với ảnh thủy tinh trắng.

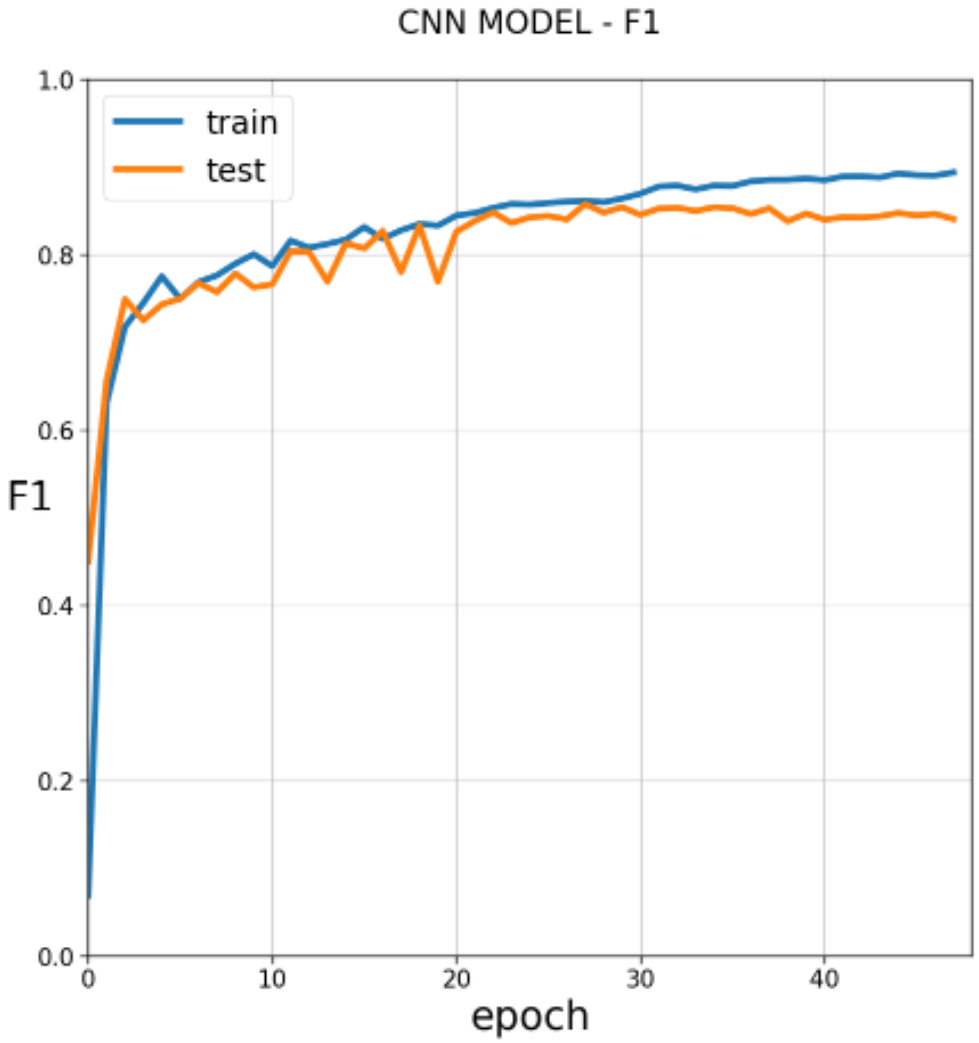
Biểu đồ này mô tả không được tốt. với cột thủy tinh nâu lại không có cho ra dự đoán và những cột khác dự đoán đúng và sai đều rất ít. Vậy em nên sử dụng thư viện sklearn.metrics để đưa ra F1 score là 0.332. độ chính xác đưa ra rất thấp.

- **Mô hình CNN**



Hình 3.16. Số lần dự đoán đúng sai của mô hình CNN

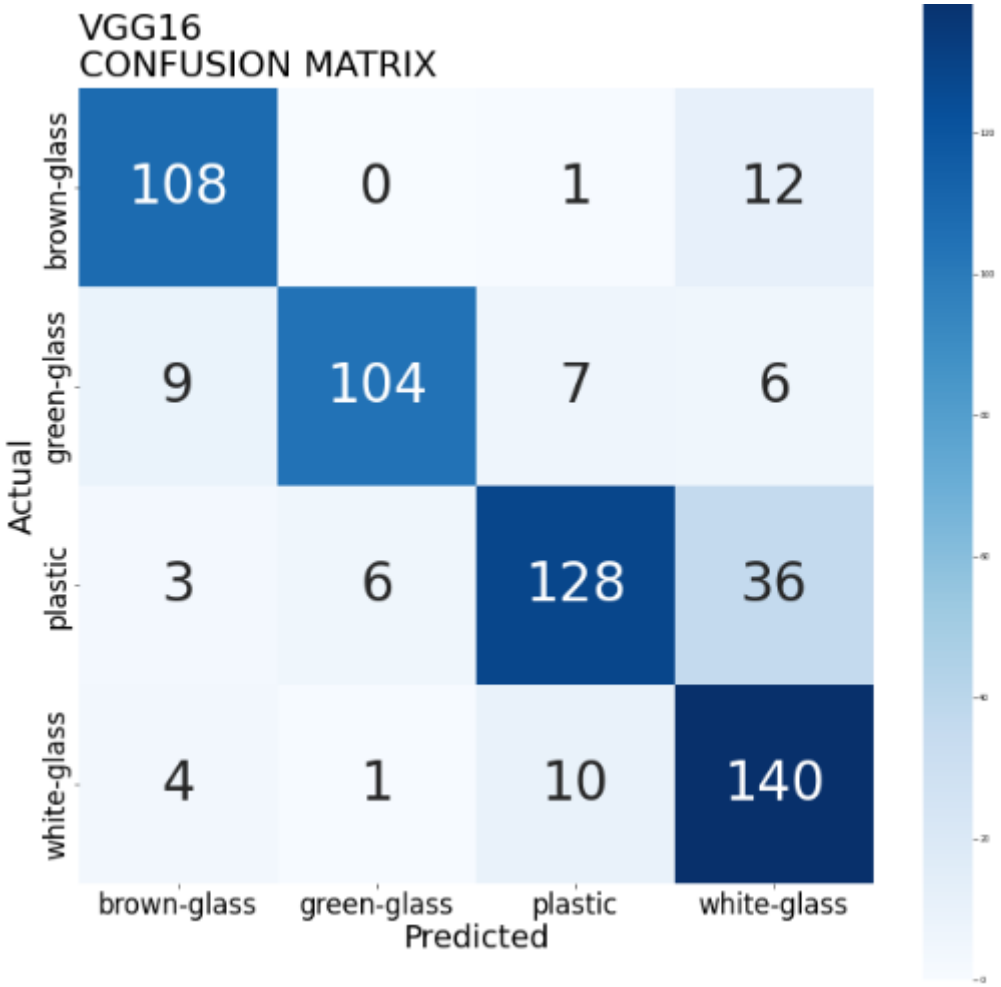
Biểu đồ trên chỉ ra được số lần dự đoán chính xác và đoán sai của mô hình CNN. Như hình trên mô hình đã dự đoán đúng 117 lần với thủy tinh trắng và dự đoán sai tới 29 lần với nhựa, 4 lần với ảnh thủy tinh nâu.



Hình 3.17. Biểu đồ đường CNN model

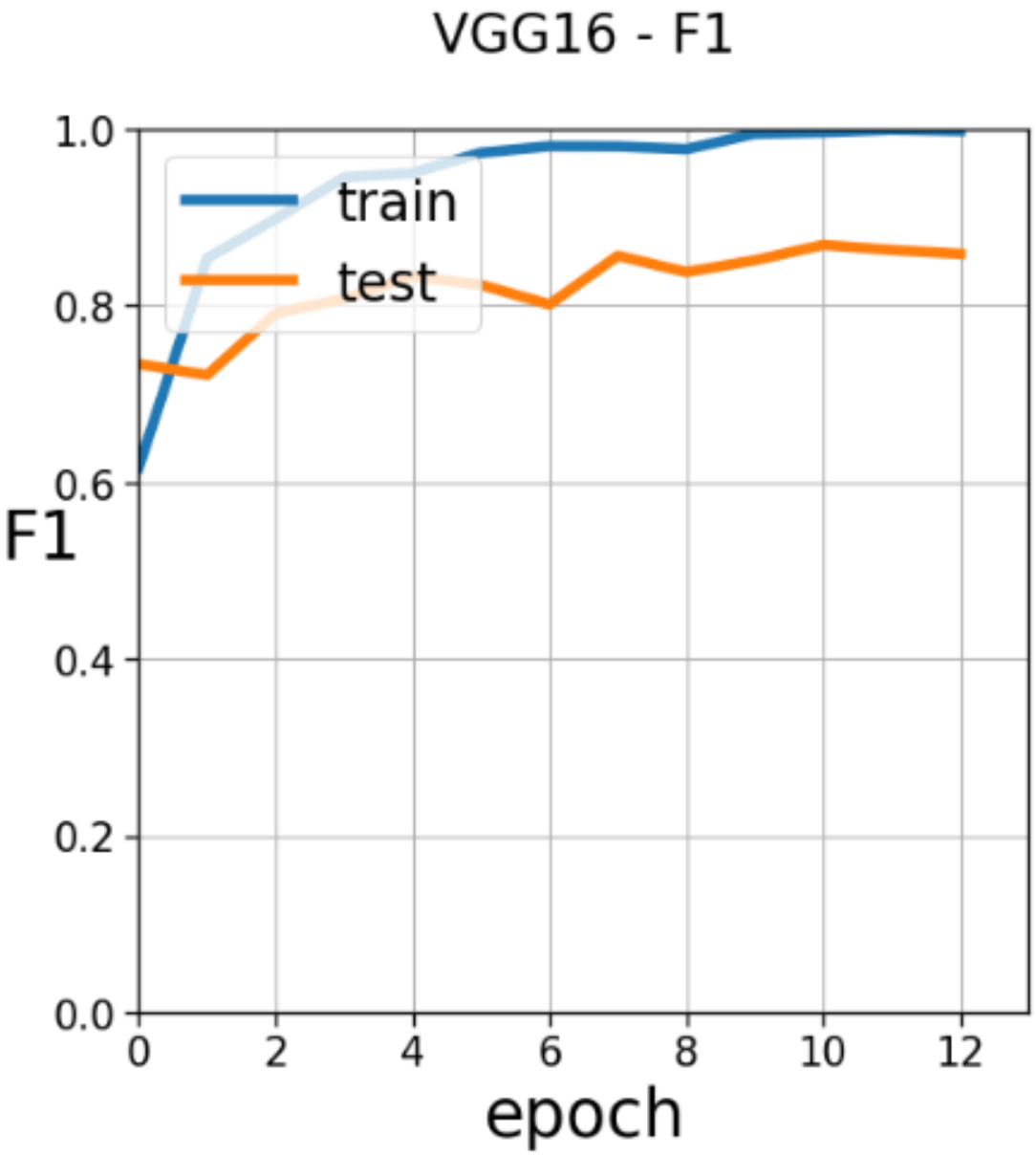
Biểu đồ đường này cho thấy được độ chính xác của hai tập train và test. Đường màu xanh dương biểu thị cho bộ tập trainning và đường cam biểu thị cho bộ dữ liệu tập test. Tập train đạt được 0.893 và tập test đạt được 0.840.

- **Mô hình VGG16**



Hình 3.18. Số lần dự đoán đúng sai của mô hình VGG16

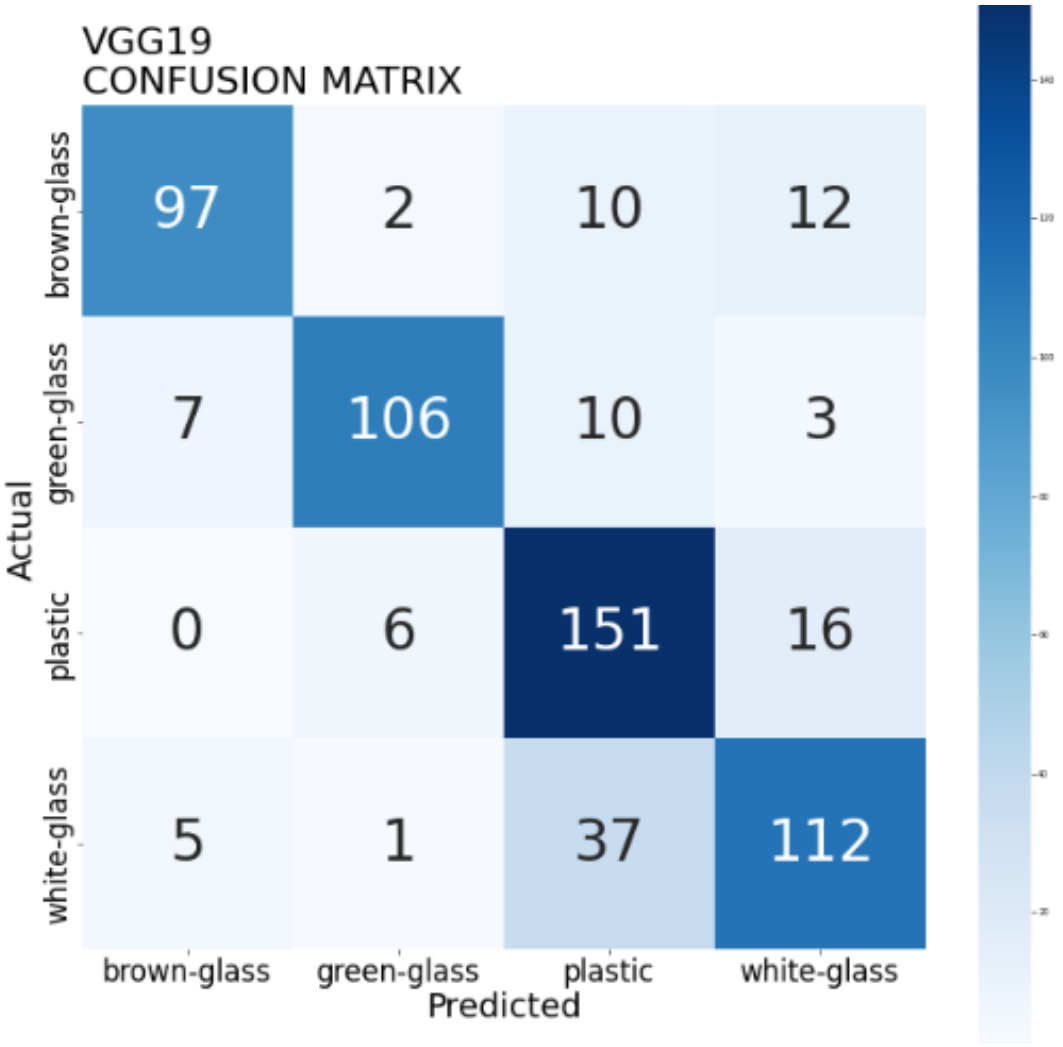
Biểu đồ trên chỉ ra được số lần dự đoán chính xác và đoán sai của mô hình VGG16. Như hình trên mô hình đã dự đoán đúng 104 lần với thủy tinh xanh và dự đoán sai tới 6 lần với nhựa, 0 lần với ảnh thủy tinh nâu.



Hình 3.19. Biểu đồ đường VGG16 model

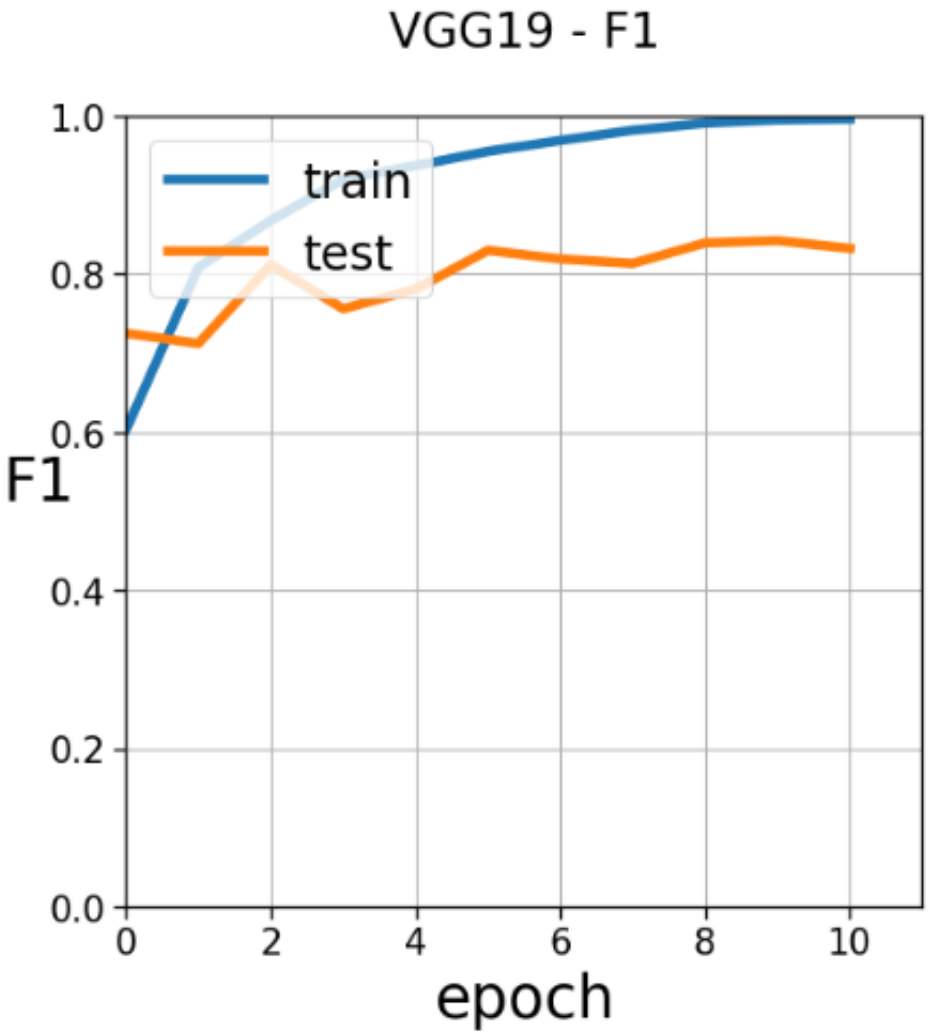
Tập train đạt được 0.998 và tập test đạt được 0.858

- **Mô hình VGG19**



Hình 3.20. Số lần dự đoán đúng sai của mô hình VGG19

Biểu đồ trên chỉ ra được số lần dự đoán chính xác và đoán sai của mô hình VGG16. Như hình trên mô hình đã dự đoán đúng 106 lần với thủy tinh xanh và dự đoán sai tới 6 lần với nhựa, 2 lần với ảnh thủy tinh nâu.



Hình 3.21. Biểu đồ đường VGG19 model

Tập train đạt được 0.996 và tập test đạt được 0.832

3.5. So sánh các mô hình

Mỗi một mô hình đều sẽ có các điểm mạnh chính của nó sẽ tùy thuộc vào mục đích của đề tài hay dự án của được sử dụng tới. vậy em đưa ra bảng so sánh điểm mạnh và hạn chế của mỗi một mô hình như sau.

Bảng so sánh chung bốn loại model

Model	Logistic	CNN	VGG16	VGG19
Đặc điểm chính	- Phù hợp cho dữ liệu có cấu trúc đơn giản, không có hình ảnh. - Đơn giản, dễ hiểu và dễ triển khai. - Hiệu quả với dữ liệu có tính cấu trúc đơn giản.	- Phù hợp cho việc xử lý dữ liệu hình ảnh, với khả năng học các đặc trưng cấp cao - Hiệu suất tốt trên dữ liệu hình ảnh. - Có khả năng học các đặc trưng phức tạp. - Chia sẻ trọng số giữa các vùng không gian giúp giảm lượng tham số.	- Mạng nơ ron sâu với 16 lớp, thường được sử dụng để trích xuất đặc trưng trong hình ảnh. - Hiệu suất tốt trên nhiều bài toán - Dễ triển khai do tính đồng nhất của kiến trúc	- Mạng nơ ron sâu với 19 lớp, có thể cung cấp hiệu suất tốt hơn VGG16 nhưng có thể yêu cầu nhiều tài nguyên tính toán hơn - Đạt được hiệu suất cao hơn VGG16 - Đa dạng hóa và độ sâu của mạng có thể giúp trích xuất các đặc trưng phức tạp
Ưu điểm	- Đơn giản, dễ hiểu và dễ triển khai. - Hiệu quả với dữ liệu có tính cấu trúc đơn giản.	- Hiệu suất tốt trên dữ liệu hình ảnh - Có khả năng học các đặc trưng phức tạp. - Chia sẻ trọng số giữa các vùng không gian giúp giảm lượng tham số	- Hiệu suất tốt trên nhiều bài toán - Dễ triển khai do tính đồng nhất của kiến trúc	- Đạt được hiệu suất cao hơn VGG16 - Đa dạng hóa và độ sâu của mạng có thể giúp trích xuất các đặc trưng phức tạp

Nhược điểm	- Không thể mô hình các quan hệ phức tạp trong dữ liệu hình ảnh. - Khả năng tránh bị overfitting khi có nhiều đặc trưng.	- Đòi hỏi nhiều dữ liệu huấn luyện. - Thời gian và tài nguyên tính toán lớn. - Dễ bị overfitting nếu không có đủ dữ liệu hoặc kỹ thuật regularization.	- Có thể quá phức tạp cho các bài toán nhỏ hoặc có ít dữ liệu - Cần nhiều tài nguyên tính toán và bộ nhớ	- Tăng độ phức tạp và tài nguyên tính toán so với VGG16 - Cần nhiều dữ liệu huấn luyện để tránh overfitting - Dễ gặp hiện tượng overfitting đặc biệt khi không có kỹ thuật regularization

Các mô hình sau đây đã được đào tạo và thử nghiệm:

- Hồi quy logistic (mô hình cơ sở): F1 trên bộ xác thực: 0.339
- Mạng nơ ron tích chập (CNN): F1 trên bộ xác thực: 0,840
- VGG16: F1 trên bộ xác thực: 0,858
- VGG19: F1 trên bộ xác thực: 0,832

Mô hình cuối cùng được chọn cho dự án này là VGG16 vì nó thể hiện điểm tốt hơn một chút và cân bằng các đặc điểm khác mặc dù CNN và VGG19 có hiệu suất tương tự nhau và có thể là mô hình được lựa chọn tùy thuộc vào việc tinh chỉnh.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận

Kết quả của đồ án nghiên cứu và xây dựng hệ thống phân loại chai thủy tinh và nhựa đã cung cấp cái nhìn sâu sắc về khả năng ứng dụng của các mô hình học máy hiện đại trong lĩnh vực phân loại vật liệu. Em đã thực hiện thành công ba mô hình phân loại chính, bao gồm Logistic Regression, Convolutional Neural Network (CNN), và hai phiên bản của mạng VGG16 và VGG19.

Kết quả thử nghiệm đã chỉ ra rằng mô hình CNN với kiến trúc phức tạp và khả năng học cấu trúc không gian tốt hơn so với mô hình Logistic Regression trong việc phân loại chai thủy tinh và nhựa. Đặc biệt, các mô hình VGG16 và VGG19, được huấn luyện trên dữ liệu hình ảnh lớn, đã đem lại kết quả ấn tượng, với độ chính xác cao và khả năng tổng quát hóa tốt.

Đặc biệt với mô hình VGG16 đã cho thấy sự vượt trội trong quá trình huấn luyện. Mô hình VGG19 cũng gần bằng với VGG16 nên mô hình VGG19 cũng có thể sử dụng thay thế cho VGG16.

Có một lưu ý rằng hiệu suất của mỗi mô hình phụ thuộc phần lớn vào chất lượng và khối lượng dữ liệu huấn luyện. Điều này đề xuất một hướng phát triển tiềm năng là tăng cường thu thập dữ liệu và cải thiện quá trình tiền xử lý để cải thiện hiệu suất của các mô hình học máy trong tương lai.

Hướng phát triển

Sau khi đã chọn được mô hình phù hợp và có kết quả em nhận thấy đề tài vẫn có thể phát triển lên được nhiều hơn.

- Tối ưu hóa mô hình: dù đã chọn được một mô hình, việc tối ưu hóa các siêu tham số và kiến trúc của nó vẫn có thể cải thiện hiệu suất của hệ thống phân loại. Bạn có thể thử nghiệm và điều chỉnh các thông số như tốc độ học, kích thước kernel, số lượng lớp và nút ẩn, để tối ưu hóa hiệu suất của mô hình.
- Dữ liệu mở rộng: tăng cường dữ liệu huấn luyện có thể giúp cải thiện khả năng tổng quát hóa của mô hình. Bạn có thể thực hiện việc thu thập thêm dữ liệu hoặc

sử dụng kỹ thuật như tăng cường ảnh (image augmentation) để tạo ra các biến thể của dữ liệu huấn luyện.

- Kiểm tra độ tin cậy: xây dựng các phương pháp kiểm tra độ tin cậy (confidence estimation) để đánh giá mức độ chính xác của mô hình trong việc phân loại mỗi mẫu dữ liệu. Điều này có thể giúp xác định các trường hợp mà mô hình không chắc chắn, từ đó cải thiện độ tin cậy của hệ thống.
- Phát triển ứng dụng thực tế: sau khi mô hình đã được tinh chỉnh và kiểm tra kỹ lưỡng, bạn có thể phát triển một ứng dụng thực tế cho việc phân loại chai thủy tinh và nhựa. Điều này có thể bao gồm việc tích hợp mô hình vào một ứng dụng di động hoặc hệ thống tự động.
- Nghiên cứu thêm về lĩnh vực: nghiên cứu thêm về các phương pháp và công nghệ mới trong lĩnh vực phân loại vật liệu có thể mang lại những cải tiến đáng kể cho đề tài của bạn. Các hướng nghiên cứu có thể bao gồm việc sử dụng mạng nơ-ron tái chế (recurrent neural networks) hoặc mạng nơ-ron hồi quy (recursive neural networks) để xử lý dữ liệu chuỗi hoặc dữ liệu không cấu trúc.

TÀI LIỆU THAM KHẢO

Link data: https://www.kaggle.com/datasets/mostafaabla/garbage-classification

Tài liệu tiếng anh:

- Học thuật:
    1. Very Deep Convolutional Networks for Large-Scale Image Recognition bởi Karen Simonyan và Andrew Zisserman, được trình bày tại ICLR 2015.
- Trực tuyến:
    2. https://d2l.ai/chapter_convolutional-neural-networks/lenet.html(truy cập 09/04/2024)
    3. https://www.tensorflow.org/tutorials/images/cnn?hl=vi (truy cập 14/04/2024)
    4. https://medium.com/hitchhikers-guide-to-deep-learning/ (truy cập 10/05/2024)
    5. https://medium.com/hitchhikers-guide-to-deep-learning/10-introduction-to-deep-learning-with-computer-vision-types-of-convolutions-atrous-convolutions (truy cập 12/05/2024)
    6. https://neurohive.io/en/popular-networks/vgg16/(truy cập 12/05/2024)

Tài liệu tiếng Việt:

- Học thuật:
    7. MẠNG NƠ-RON TÍCH CHẬP (CNN) VÀ MẠNG VGG16 bởi Vien Thanh Nha , Tiep Sy Minh Phung , Pham Thanh Cong , Nguyen Thanh Tung , Le Dinh Phu Cuong, Được trình bày trên trí khoa học tháng 5 năm 2022.
- Trực tuyến:
    1. https://machinelearningcoban.com/2017/01/27/logisticregression/ (truy cập 09/04/2024)
    8. https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2(truy cập 09/04/2024)