

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH**

හමුදා



HCMUTE

BÁO CÁO CUỐI KÌ

PHÂN LOẠI 15 LOẠI GỖ (ẢNH GIẢI PHẪU GỖ)

GVHD: PGS. TS Nguyễn Trường Thịnh

MHP: ARIN337629_06

Họ và tên: Đoàn Đức Hiếu

MSSV: 19146333

Năm học: Học kì 2 2021 - 2022

TP. Hồ Chí Minh, tháng 06 năm 2022

LỜI CẢM ƠN

Trên chặng đường thành công của mỗi chúng ta, chắc chắn chúng ta sẽ luôn cần tới những người dẫn đường và những người thầy. Người mà đã có kinh nghiệm từng trải, có thể hỗ trợ chúng ta ở hầu hết các tình huống khó khăn. Và với đề tài cuối kỳ của em lần này cũng không ngoại lệ. Để có thể hoàn thành một cách trọn vẹn đề tài cuối kỳ môn Trí tuệ nhân tạo lần này dựa trên một phần sự tham khảo, đúc kết kinh nghiệm từ các tác giả của các trường đại học, tổ chức nghiên cứu đi trước, phần còn lại dựa trên sự nỗ lực hết sức của chính bản thân mình.

Trước hết, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến **Thầy Nguyễn Trường Thịnh**, là giảng viên giảng dạy môn *Trí tuệ nhân tạo* trong học kỳ này. Thầy đã luôn tạo điều kiện cả về vật chất lẫn tinh thần để hỗ trợ em trong suốt quá trình thực hiện nghiên cứu và hoàn thành đề tài. Tuy rằng em đã cố gắng hoàn thiện đề tài một cách tốt nhất có thể, nhưng chắc chắn sẽ không thể tránh khỏi những thiếu sót, sơ suất trong quá trình làm việc. Em kính mong quý Thầy, Cô trong bộ môn, các bạn bè có những ý kiến đóng góp chân thành, thẳng thắn để em có thể tiếp tục hoàn thiện đề tài trong tương lai.

Một lần nữa, em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày ... tháng 6 năm 2022

Tác giả

MỤC LỤC

MỞ ĐẦU	1
Giới thiệu.....	1
Lí do chọn đề tài:.....	2
Mục tiêu của đề tài:.....	2
Phương pháp nghiên cứu	2
CHƯƠNG I: TỔNG QUAN	2
1. Khái niệm về giải phẫu gỗ.....	2
2. Cách lấy ảnh giải phẫu gỗ	3
3. Phương pháp sử dụng.....	4
CHƯƠNG II: : THƯ VIỆN VÀ CÔNG NGHỆ SỬ DỤNG.....	5
1. Thư viện Open CV	5
2. Machine learning với phương pháp CNN	6
2.1 Machine learning là gì ?	6
2.2 Thuật toán CNN – Convolutional Neural Network.....	6
CHƯƠNG III: XÂY DỰNG MÔ HÌNH PHÂN LOẠI GỖ DÙNG CONVOLUTIONAL NEURAL NETWORK (CNN)9	
1. Chuẩn bị dữ liệu.....	9
2. Quá trình training	10
CHƯƠNG IV: : THỰC NGHIỆM.....	16
1. Code.....	16
2. Kết quả: trên colab.....	16
3. Nhận diện trên app bằng python:	18
CHƯƠNG V: ĐÁNH GIÁ KẾT QUẢ	28
1. Đánh giá kết quả.....	28
2. Hướng phát triển	28
KẾT LUẬN	29
TÀI LIỆU THAM KHẢO.....	30

MỞ ĐẦU

Giới thiệu

Xác định tên các loài gỗ là công việc quan trọng cần thiết đối với hầu hết các hoạt động mua bán giao dịch gỗ, chẳng hạn như việc xuất nhập khẩu gỗ. Nó được sử dụng để đánh giá giá trị của gỗ trên thị trường thương mại, và việc kiểm tra gỗ tại hải quan để giám sát gỗ và ngăn chặn việc buôn bán gỗ bất hợp pháp, và nó rất hữu ích cho việc bảo vệ thực vật. Tuy nhiên, hầu hết các bộ dữ liệu của thí nghiệm đều phải trải qua nhiều công đoạn, rất khó để có được dữ liệu vì dữ liệu cần rất nhiều công đoạn chuẩn bị, chẳng hạn như ngâm và cắt gỗ sẽ mất hơn một tuần. Hơn nữa, thiết bị kính hiển vi là một dụng cụ tinh vi, đắt, không thể mang đi và không thể sử dụng bên ngoài. Để giải quyết những khó khăn này, chúng ta cố gắng tìm ra một phương pháp mới, có thể được áp dụng bên ngoài với kính lúp x10 nhỏ và dễ mang theo.

Mạng nơron chuyển đổi (CNN) là một trong những phương pháp được sử dụng phổ biến nhất trong việc phân loại các đối tượng đối tượng hiện nay. Nó cần tập dữ liệu lớn, nhưng những yêu cầu về dữ liệu này thì không dễ đáp ứng. Việc triển khai học tập cho các mô hình trên từng miền cụ thể cho nhiều miền mục tiêu và nó có thể thúc đẩy sự phát triển học tập của các mô hình. Việc học có thể làm giảm số lượng và loại dữ liệu cần thiết trong dữ liệu, nó sẽ sử dụng thông tin trước đó để tối ưu hóa nội dung học tập theo mô hình. Học những kiến thức đã học trong trường hợp có đủ dữ liệu sang một môi trường mới với lượng dữ liệu nhỏ và nó có thể tìm ra mối quan hệ giữa dữ liệu lớn và dữ liệu nhỏ, đồng thời chuyển kiến thức từ dữ liệu lớn sang dữ liệu nhỏ. Học chuyển giao trong không gian có ứng dụng rộng rãi. Thông qua đào tạo và học hỏi dữ liệu, em đã thiết lập một model cho nó học và sau đó chuyển nó sang nhận dạng hình ảnh gỗ. Bằng cách này, em có thể nhận biết chính xác các loài gỗ dựa trên hình ảnh gỗ trong trường hợp một lượng nhỏ dữ liệu.

Lí do chọn đề tài:

Nhận dạng gỗ là một mối quan tâm lớn đối với các nước nhiệt đới với nguồn tài nguyên rừng phong phú, do đó có nhu cầu cao về các hệ thống nhận dạng gỗ, để giải quyết vấn đề đa dạng sinh học trên diện rộng. Có rất nhiều nhu cầu tại chỗ khác nhau để nhận dạng gỗ, chẳng hạn như bảo tồn các loài đặc hữu, điều chỉnh việc buôn bán gỗ khai thác bất hợp pháp gỗ, và sàng lọc các loại gỗ gian lận.

Bên cạnh đó, trên thực tế không thể đào tạo một số lượng công nhân để đi nhận dạng ảnh giải phẫu gỗ. Việc nhận dạng gỗ đòi hỏi kiến thức chuyên môn cao về giải phẫu gỗ và cần có kinh nghiệm lâu năm. Do đó, ngay cả khi bỏ ra nhiều tiền và thời gian, việc đào tạo công nhân lành nghề vẫn có những giới hạn nhất định.

Mục tiêu của đề tài:

Dự án tập trung vào các mục tiêu như sau

- Phân loại được 15 loại gỗ quý hiếm
- Có thể xuất ra tên của mỗi loại gỗ trên ảnh
- Đưa mô hình lên web-app để mọi người dễ sử dụng và nhận biết dễ dàng

Phương pháp nghiên cứu

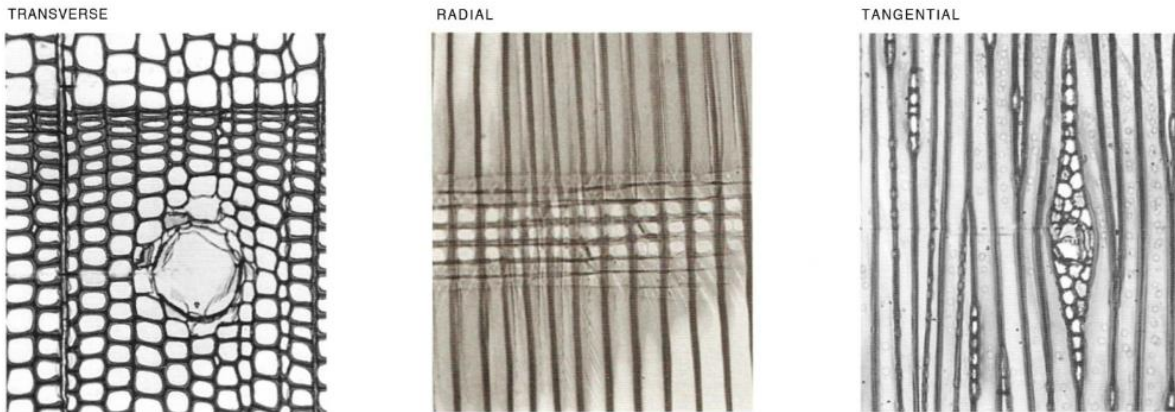
- Tìm hiểu lý thuyết về nhận dạng giải phẫu gỗ sử dụng AI
- Tìm hiểu các phương pháp hiện có thông qua Github, Kaggle
- Xây dựng “Model training” và tiến hành chạy thử để kiểm chứng

CHƯƠNG I: TỔNG QUAN

1. Khái niệm về giải phẫu gỗ

Gỗ là một ma trận ba chiều xốp, hút ẩm, liên kết với nhau của xenlulo, hemixenlulo và lignin. Nói một cách dễ hiểu, cây có thể được mô tả như một bó mạch, thành của nó bao gồm xenlulo được dán lại với nhau bằng lignin. Các mạch hoặc tế bào này vận chuyển thức ăn và các chất thải qua cây. Các tế bào mới phát triển xung quanh chu vi của cây, tạo thành một vòng ngay trong vỏ cây. Bên trong vỏ cây và bao gồm phần lớn thân cây là gỗ, có đặc điểm là có nhiều vòng sinh trưởng sắp xếp đồng tâm xung quanh

trung tâm. Các bộ phận của thân cây là sự tích tụ của vô số tế bào. Tế bào là đơn vị cấu trúc cơ bản của vật chất thực vật. Mỗi tế bào bao gồm một thành tế bào bên ngoài bao quanh một khoang tế bào bên trong. Tế bào gỗ dài hơn chiều rộng và chủ yếu hướng song song với trục dài của thân và cành. Hình 1 cho thấy cấu trúc tế bào của gỗ thông, nhìn nó từ các mặt cắt ngang khác nhau.



Hình 1: Các phần mỏng của cây thông trắng phía đông được phóng đại 100 lần cho thấy cấu trúc tế bào của mẫu vật.

2. Cách lấy ảnh giải phẫu gỗ

Bề mặt tế bào của các khối được cắt bằng dao hoặc mài bằng giấy nhám để lộ rõ đặc điểm biểu đồ giải phẫu. Hình ảnh tỷ lệ vĩ mô có thể được chụp trực tiếp từ các khối gỗ bằng máy ảnh kỹ thuật số hoặc kính hiển vi soi nổi. Trong khi đó, để chụp ảnh kính hiển vi, phải chuẩn bị các lam kính hiển vi của các mẫu gỗ thông qua các quy trình tiêu chuẩn dẫn đến làm mềm, cắt tìng, nhuộm, khử nước và gắn kết hi chụp ảnh, chất lượng của ảnh thu được có thể khác nhau tùy thuộc vào điều kiện ánh sáng. Các mô-đun hình ảnh được trang bị hệ thống quang học đã được sử dụng để điều khiển ánh sáng một cách đồng và các kỹ thuật xử lý hình ảnh như làm mờ được áp dụng để bình thường hóa độ sáng của hình ảnh được chụp. Hình ảnh vĩ mô và hình lập thể được ưa thích trong các nghiên cứu nhằm phát triển các hệ thống có thể triển khai. Các tính năng ở cấp độ hiển vi được trích xuất từ các hình ảnh hiển vi cho phép giải phẫu thông qua các quy trình tiêu chuẩn dẫn đến làm mềm, cắt tìng, nhuộm, khử nước và gắn kết. Tất cả các loại hình ảnh gỗ có thể được sử dụng làm dữ liệu để nhận dạng. Các loại hình ảnh thường được sử dụng nhất là hình ảnh macro, hình ảnh chụp cắt lớp vi tính tia X, hình lập thể, và hình ảnh vi thể. Hình ảnh kính hiển vi là hình ảnh được chụp không cần

phóng đại bằng máy ảnh kỹ thuật số thông thường. Ảnh lập thể thường là hình ảnh được chụp ở độ phóng đại ống kính cầm tay (x10), nhưng có thể sử dụng độ phóng đại cao hơn tùy thuộc vào mục đích.

3. Phương pháp sử dụng

Train có giám sát là một kỹ thuật ML sử dụng một cặp hình ảnh và nhãn của nó làm dữ liệu đầu vào. Tất cả là các mô hình phân loại từng class học, các hình ảnh được gắn nhãn để xác định các quy tắc phân loại class và sau đó phân loại dữ liệu truy vấn dựa trên các quy tắc. Ngược lại, trong học tập không giám sát, mô hình tự khám phá ra thông tin chưa biết bằng cách học dữ liệu không được gắn nhãn. Classification nói chung là một nhiệm vụ của học có giám sát và phân cụm nói chung là một nhiệm vụ của học không có giám sát. Để giải đáp các nhu cầu trong lĩnh vực, nhiều phương pháp khác nhau đã được đề xuất, chẳng hạn như phép đo khối phổ, quang phổ cận hồng ngoại, đồng vị ổn định và các phương pháp dựa trên DNA. Tuy nhiên, các cách tiếp cận này có những hạn chế thực tế như một công cụ hỗ trợ hoặc thay thế việc kiểm tra bằng mắt do chi phí tương đối cao và sự phức tạp về thủ tục của chúng. Đây là nơi mà các kỹ thuật nhận dạng dựa trên CV (Computer vision) và mô hình ML có thể rất quan trọng. Rõ ràng, hệ thống nhận dạng gồ tự động là cấp thiết cần thiết và hệ thống nhận dạng gồ dựa trên CV đã nổi lên như một hệ thống đầy hứa hẹn. Phân loại hình ảnh được chia theo phương pháp luận thành ML thông thường và học sâu (DL), cả hai đều là dạng AI. Trong ML conventional, trích xuất đối tượng, quá trình trích xuất các tính năng quan trọng từ hình ảnh (còn được gọi là tính năng kết nối) và phân loại, quá trình tìm hiểu các tính năng được trích xuất và phân loại hình ảnh truy vấn, được hình thành độc lập. Đầu tiên, tất cả các hình ảnh trong tập dữ liệu đều được xử lý trước bằng nhiều kỹ thuật xử lý hình ảnh khác nhau để chuyển đổi chúng thành một dạng có thể được sử dụng bởi một thuật toán cụ thể để trích xuất các tính năng. Sau đó 15 tập dữ liệu được tách thành các tập huấn luyện và kiểm tra, và các tính năng được trích xuất từ các hình ảnh tập huấn luyện bằng cách sử dụng tính năng thuật toán. Thiết lập mô hình phân loại phân loại bằng cách học các tính năng được trích xuất. Cuối cùng, các hình ảnh của bộ thử nghiệm được đưa vào và mô hình phân loại, trả về các lớp được dự đoán của mỗi hình ảnh và sau đó thành việc nhận dạng.

CHƯƠNG II: : THƯ VIỆN VÀ CÔNG NGHỆ SỬ DỤNG

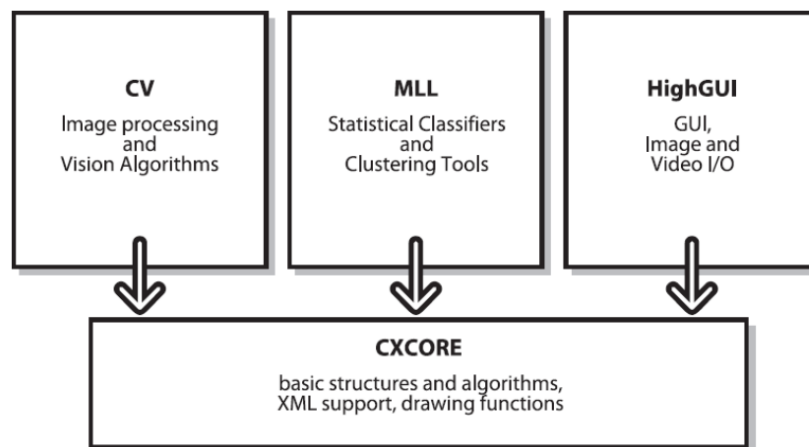
1. Thư viện Open CV

OpenCV (Open Computer Vision) là một thư viện mã nguồn mở hàng đầu cho xử lý về thị giác máy tính, machine learning, xử lý ảnh. OpenCV được viết bằng C/C++, vì vậy có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực. OpenCV có các interface cho C/C++, Python Java vì vậy hỗ trợ được cho Window, Linux, MacOS lẫn Android, iOS OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần.

OpenCV có rất nhiều ứng dụng:

- Nhận dạng ảnh
- Xử lý hình ảnh
- Phục hồi hình ảnh/video
- Thực tế ảo
- Các ứng dụng khác

OpenCV có thể được cấu trúc rộng rãi thành năm thành phần chính, bốn trong số đó được thể hiện trong hình. Thành phần CV chủ yếu chứa các thuật toán xử lý hình ảnh cơ bản và thị giác máy tính cấp cao hơn; MLL thư viện học máy bao gồm nhiều bộ phân loại thống kê cũng như các công cụ phân cụm. Thành phần HighGUI chứa các quy trình I / O với các chức năng lưu trữ, tải video và hình ảnh, trong khi CXCore chứa tất cả các cấu trúc và nội dung dữ liệu cơ bản.



2. Machine learning với phương pháp CNN

2.1 Machine learning là gì ?

Machine learning là một lĩnh vực con của Trí tuệ nhân tạo (Artificial Intelligence) sử dụng các thuật toán cho phép máy tính có thể học từ dữ liệu để thực hiện các công việc thay vì được lập trình một cách rõ ràng. Mục tiêu của học máy là biến dữ liệu thành thông tin. Sau khi học được từ việc thu thập dữ liệu, người dùng muốn một máy có thể trả lời bất kỳ câu hỏi nào về dữ liệu:

- Các dữ liệu khác tương tự với dữ liệu đã cho là gì?
- Loại quảng cáo nào sẽ ảnh hưởng đến người dùng?
- Thường có một thành phần chi phí, vì vậy câu hỏi có thể trở thành:

Trong số rất nhiều sản phẩm mà chúng ta có thể kiếm tiền, sản phẩm nào có nhiều khả năng được người dùng mua nhất nếu quảng cáo được hiển thị cho sản phẩm đó?

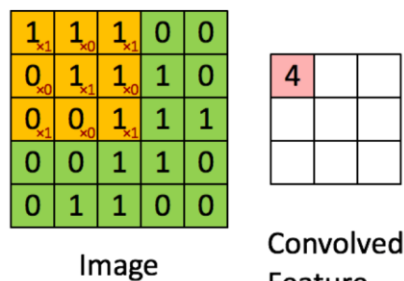
Máy học chuyển đổi dữ liệu thành thông tin bằng cách phát hiện các quy tắc hoặc mẫu từ dữ liệu đó?

2.2 Thuật toán CNN – Convolutional Neural Network

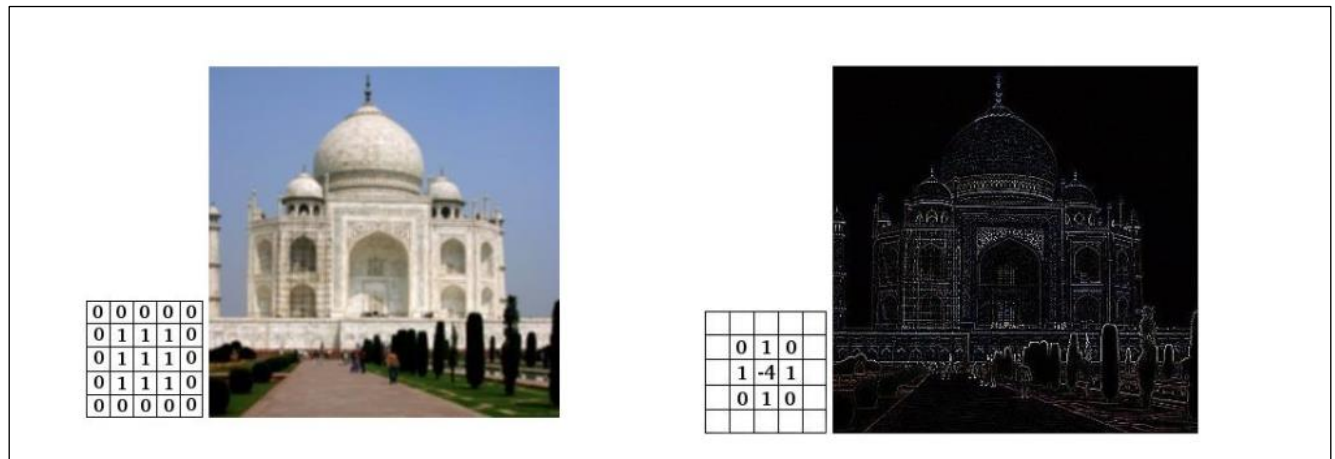
Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

a) *Convolutional* là gì?

Là một cửa sổ trượt (Sliding Windows) trên một ma trận như mô tả hình dưới:



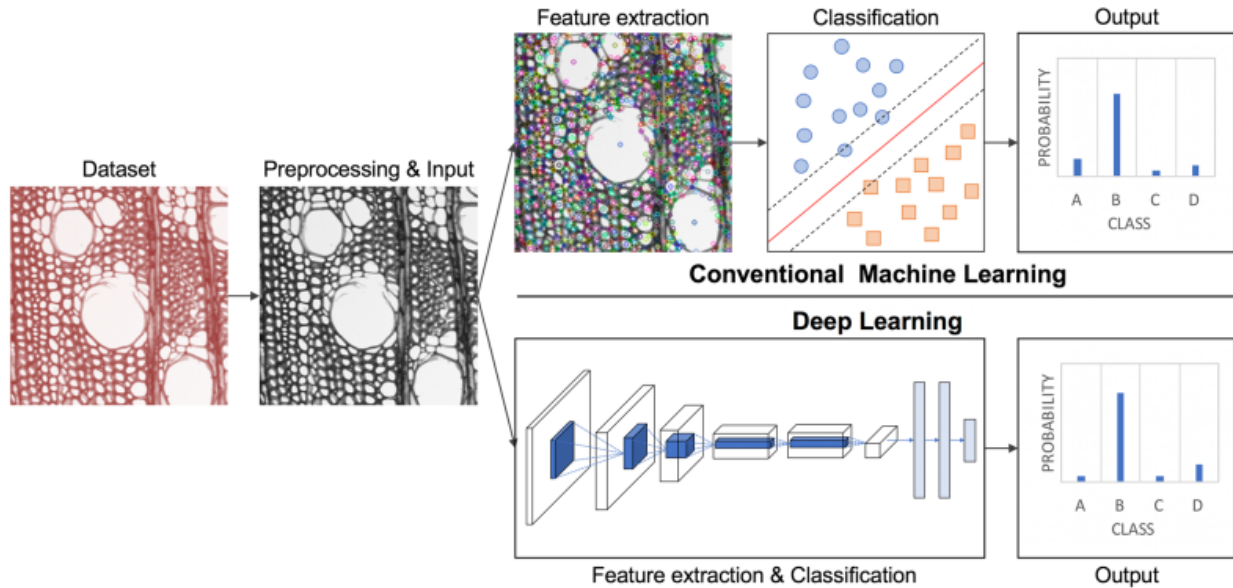
Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên phải. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhân ma trận Filter với ma trận ảnh 5×5 bên trái



b) Cấu trúc của mạng CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo. Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó. Trong quá trình huấn luyện mạng (training) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Trong mô hình CNN có 2 khía cạnh cần quan tâm là **tính bất biến** (Location Invariance) và **tính kết hợp** (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

c) Chọn tham số cho CNN


















- Số các convolution layer: càng nhiều các convolution layer thì performance càng được cải thiện. Sau khoảng 3 hoặc 4 layer, các tác động được giảm một cách đáng kể.
- Filter size: thường filter theo size 5×5 hoặc 3×3
- Pooling size: thường là 2×2 hoặc 4×4 cho ảnh đầu vào lớn
- Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt nhất.

CHƯƠNG III: XÂY DỰNG MÔ HÌNH PHÂN LOẠI GỖ DỪNG CONVOLUTIONAL NEURAL NETWORK (CNN)

1. Chuẩn bị dữ liệu

Dữ liệu chuẩn bị cho việc train và test. Trong đó dữ liệu được chia thành 2 thư mục gồm train và test. Thư mục train có 15 thư mục (11118 ảnh) tương ứng với 15 loại ảnh giải phẫu gỗ khác nhau. Thư mục test có 15 thư mục (2465 ảnh) tương ứng với 15 loại ảnh giải phẫu gỗ khác nhau.

Hình ảnh file dữ liệu đầu vào

 Test	6/13/2022 1:25 PM	File folder
 Train	6/11/2022 7:18 AM	File folder
 Anh_dao	6/11/2022 7:30 AM	File folder
 Bach_dang_nk	6/11/2022 7:31 AM	File folder
 Cao_su	6/11/2022 7:32 AM	File folder
 Cho_chi	6/11/2022 7:32 AM	File folder
 Dang_huong_viet	6/11/2022 7:33 AM	File folder
 Hoang_dan	6/11/2022 7:33 AM	File folder
 Iroko	6/11/2022 7:41 AM	File folder
 Lat_hoa	6/11/2022 7:35 AM	File folder
 Lim	6/11/2022 7:36 AM	File folder
 Mit_mat	6/11/2022 7:37 AM	File folder
 Muong_den	6/11/2022 7:37 AM	File folder
 Sa_moc_dau	6/11/2022 7:37 AM	File folder
 Tau_mat	6/11/2022 7:38 AM	File folder
 Thuy_tung	6/11/2022 7:38 AM	File folder
 Trai_li	6/11/2022 7:38 AM	File folder

Sau đó đưa dữ liệu train và test lên drive để thuận tiện trong việc chạy chương trình trên colaboraty:



Drive của tôi > AI > Data9-1 > Train ▾

Thư mục		Tên ↑
Anh_dao	Bach_dang_nk	Cao_su
Cho_chi	Dang_huong_viet	Hoang_dan
Iroko	Lat_hoa	Lim
Mit_mat	Muong_den	Sa_moc_dau
Tau_mat	Thuy_tung	Trai_li

2. Quá trình train

Quá trình training được thực hiện hoàn toàn trên colab nhằm đẩy nhanh tốc độ train (colab hỗ trợ CPU).

Kết nối google drive:

Do dữ liệu đầu vào không quá lớn nên dữ liệu được up lên google drive sẽ thuận tiện cho việc dùng lại sau này.

```
] from google.colab import drive  
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

Đưa dữ liệu vào trong biến

Đưa dữ liệu từ google drive vào trong biến x_train và y_train, sau đó chuyển đổi dữ liệu và lưu các biến lên lại google drive

```

#Đưa dữ liệu vào biến
import os
import numpy as np
from numpy import save
from keras.utils import np_utils
from keras.preprocessing import image
from keras.preprocessing.image import load_img, img_to_array, array_to_img, ImageDataGenerator
# Đưa đường dẫn lấy ảnh
dir_folder = '/content/drive/MyDrive/AI/Data9-1/Train'
x_train = []
y_train = []
labels = []
count = 0
img_height = 150
img_width = 150
for i in os.listdir(dir_folder):
    path = os.path.join(dir_folder, i)
    labels.append(str(i))
    for j in os.listdir(path):
        path_img = os.path.join(path, j)
        img = load_img(path_img, target_size=(img_height, img_width))
        img = img_to_array(img)
        img = img.reshape(img_height, img_width, 3)
        img = img.astype('float32')
        img = img/255
        x_train.append(img)
        y_train.append(count)

```

```

x_train = np.asarray(x_train)
y_train = np.asarray(y_train)
y_train = np_utils.to_categorical(y_train)
print(x_train.shape, y_train.shape)
#Luu data len drive
save('/content/drive/MyDrive/Colab Notebooks/x_train.npy', x_train)
save('/content/drive/MyDrive/Colab Notebooks/y_train.npy', y_train)

```

```
(11118, 150, 150, 3) (11118, 15)
```

Load lại dữ liệu

Load dữ liệu x_train và y_train lại từ google drive, sau đó thực hiện chia dữ liệu ra (lấy 10% của dữ liệu train làm dữ liệu test lại của model).

```
# Chia du lieu de train va test lai
import numpy as np
from sklearn.model_selection import train_test_split
x_train = np.load("/content/drive/MyDrive/Colab Notebooks/x_train.npy" )
y_train = np.load("/content/drive/MyDrive/Colab Notebooks/y_train.npy")
x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.1, random_state = 70) #
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
print(y_test[:5])

(10006, 150, 150, 3) (10006, 15)
(1112, 150, 150, 3) (1112, 15)
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Xây dựng model:

a) Mô tả cấu trúc model

Dùng hàm Sequential() của keras để tạo model tuần tự .

Feature extraction: bao gồm 3 lớp Conv2D-MaxPooling2D

- Conv2D-MaxPooling2D lớp thứ 1 : Hàm Conv2D dùng hàm kích hoạt relu, kernel có kích thước 3x3 tạo ra output có 32 feature map có kích thước bằng với kích thước ảnh đầu vào. Sau đó qua hàm MaxPooling2D cho ra output có kích thước giảm đi một phần hai (150x150x32).

- Conv2D-MaxPooling2D lớp thứ 2: Hàm Conv2D dùng hàm kích hoạt relu, kernel có kích thước 3x3 tạo ra output có 64 feature map có kích thước bằng với kích thước output của hàm MaxPooling2D lớp thứ 1. Sau đó qua hàm MaxPooling2D cho ra output có kích thước giảm đi một phần hai (75x75x64).

- Conv2D-MaxPooling2D lớp thứ 3: Hàm Conv2D dùng hàm kích hoạt relu, kernel có kích thước 3x3 tạo ra output có 128 feature map có kích thước bằng với kích thước output của hàm MaxPooling2D lớp thứ 2. Sau đó qua hàm MaxPooling2D cho ra output có kích thước giảm đi một phần hai (37x37x32).

Classification: Gồm có Flatten (), 3 lớp Dense

- Lớp Flatten(): Output cuối cùng ta nhận được từ Feature extraction là 32 ảnh có kích thước 37x37 chồng lên nhau . Dùng hàm này có tác dụng tách rời thành một dãy node có chiều dọc.

- Dense thứ 1: có tác dụng tạo lớp ẩn có 128 node dùng hàm kích hoạt relu.
- Dense thứ 2: có tác dụng tạo lớp ẩn có 128 node dùng hàm kích hoạt relu.
- Dense thứ 3: dùng softmax đánh giá phần trăm tỉ suất của đối tượng.

```
# tạo model
from keras.backend import dropout
from tensorflow.keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
from keras.models import Sequential
from keras.layers import Dense, Activation, Conv2D, MaxPooling2D, Flatten
model = Sequential()
model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform',
                 padding='same',
                 input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform',
                 padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform',
                 padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(15, activation='softmax'))
model.summary()
```

b) Các thông số của model

Yêu cầu ảnh đầu vào 150x150x32 và sẽ có 15 đầu ra

Model: "sequential"		
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 150, 150, 32)	896
max_pooling2d (MaxPooling2D)	(None, 75, 75, 32)	0
conv2d_1 (Conv2D)	(None, 75, 75, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 37, 37, 64)	0
conv2d_2 (Conv2D)	(None, 37, 37, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 18, 18, 128)	0
flatten (Flatten)	(None, 41472)	0
dense (Dense)	(None, 128)	5308544
dense_1 (Dense)	(None, 128)	16512
dense_2 (Dense)	(None, 15)	1935
=====		
Total params: 5,420,239		
Trainable params: 5,420,239		
Non-trainable params: 0		

Trainning và kiểm tra

Checkpoint là thời điểm sẽ lấy model có độ chính xác và giá trị sai số thấp nhất.

Train cho model học 600 lần.

a) Trainning

```
#training
checkpoint = ModelCheckpoint('best_model_wood1.h5',      # model filename
                             monitor='val_loss',         # quantity to monitor
                             verbose=1,                 # verbosity - 0 or 1
                             save_best_only=True,        # The latest best model will not be overwritten
                             mode='auto')

history = model.fit(x_train, y_train,
                    epochs=600,
                    validation_data=(x_test, y_test),
                    callbacks=[checkpoint],
                    verbose=1)
model.save('model_wood1.h5')
```

b) Kiểm tra độ chính xác model

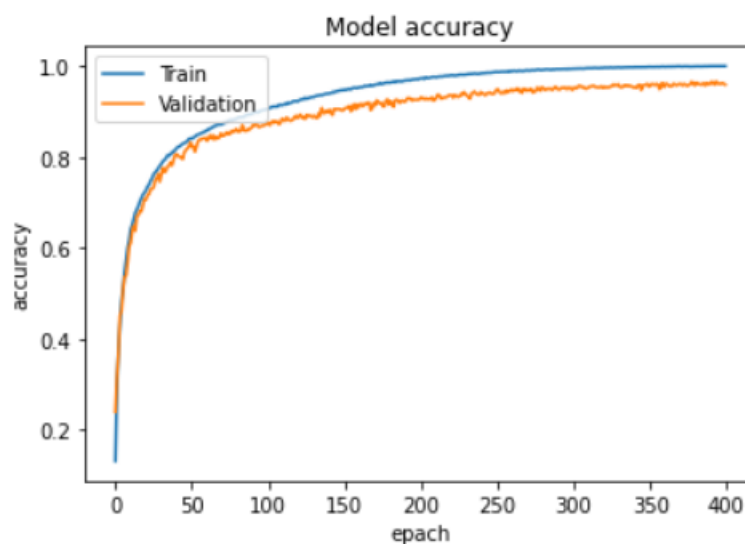
Sau khi train hoàn tất ta được độ chính xác tốt nhất là 96,22% với độ sai số là 0.12. Tải model về và đưa lại lên drive.

```
#load data
from keras.models import load_model
model = load_model('/content/drive/MyDrive/Colab Notebooks/best_model.h5')
#Kiểm tra chương trình dữ liệu
score = model.evaluate(x_test, y_test, verbose = 0)
print('Test loss:',score[0])
print('Test accuracy', score[1])
```

```
Test loss: 0.12122871726751328
Test accuracy 0.9622302055358887
```

c) Vẽ đồ thị

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'],loc='upper left')
plt.show
plt.show()
```



CHƯƠNG IV: : THỰC NGHIỆM

1. Code

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import numpy as np
import matplotlib.pyplot as plt

classes = ["Bach_dang_nk", "Cao_su", "Dang_huong_viet", "Cho_chi", "Anh_dao",
           "Hoang_dan", "Tau_mat", "Lim", "Iroko", "Mit_mat",
           "Muong_den", "Lat_hoa", "Sa_moc_dau", "Trai_li", "Thuy_tung"]

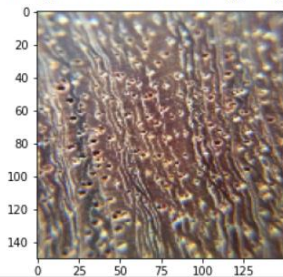
img = load_img('/content/drive/MyDrive/AI/Data9-1/Train/Dang_huong_viet/Dang_huong_viet-1056.jpg', target_size=(150,150))
# load model
model = load_model('/content/drive/MyDrive/Colab Notebooks/best_model.h5')

plt.imshow(img)
imgRe = img_to_array(img)
imgRe = imgRe.reshape(1,150,150,3)
imgRe = imgRe.astype('float32')
imgRe /= 255
y_pred = model.predict(imgRe)
score = y_pred.max()
score = round(score*100,2)
y_classes = [np.argmax(element) for element in y_pred]
print('The type of woods is: ',classes[y_classes[0]],": Accuracy: ",score, "%")
```

2. Kết quả: trên colaboratory

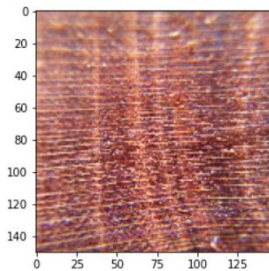
Load ảnh : Dang_huong_viet-1056.jpg

The type of woods is: Dang_huong_viet : Accuracy: 97.98 %



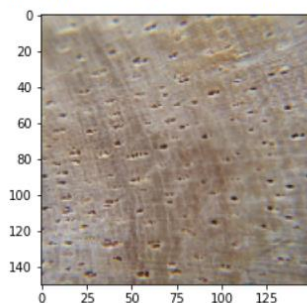
Load ảnh: Anh_Dao-1059.jpg

The type of woods is: Anh_dao : Accuracy: 99.96 %



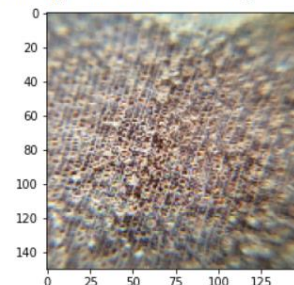
Load ảnh: Cao_su-1005.jpg

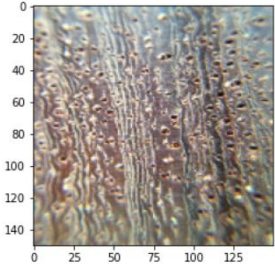
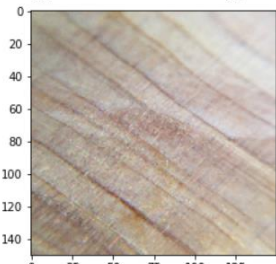
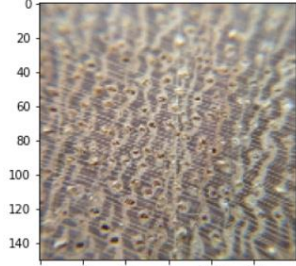
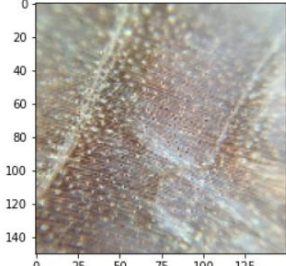
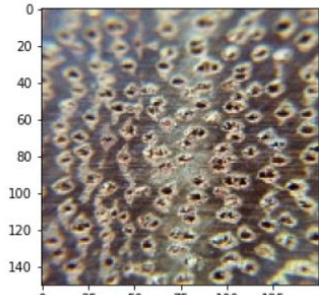
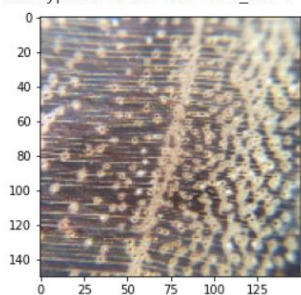
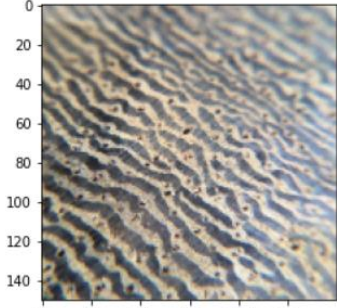
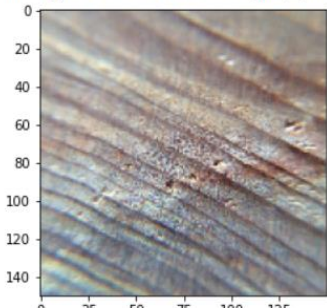
The type of woods is: Cao_su : Accuracy: 99.83 %

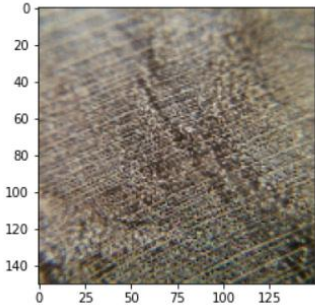
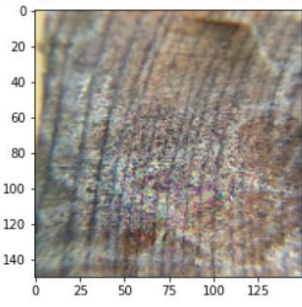
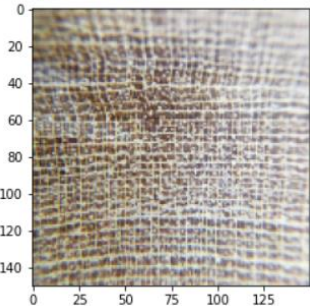


Load ảnh: Cho_chi-1016.jpg

The type of woods is: Cho_chi : Accuracy: 95.37 %



<p>Load ảnh: Dang_huong_viet-1034.jpg</p> <p>The type of woods is: Dang_huong_viet : Accuracy: 89.67 %</p> 	<p>Load ảnh: Hoang_dan-1017.jpg</p> <p>The type of woods is: Hoang_dan : Accuracy: 99.87 %</p> 
<p>Load ảnh: Iroko-1021.jpg</p> <p>The type of woods is: Iroko : Accuracy: 94.65 %</p> 	<p>Load ảnh: Lat_hoa-1009.jpg</p> <p>The type of woods is: Lat_hoa : Accuracy: 71.63 %</p> 
<p>Load ảnh: Lim-1095.jpg</p> <p>The type of woods is: Lim : Accuracy: 99.98 %</p> 	<p>Load ảnh: Mit_mat-1015.jpg</p> <p>The type of woods is: Mit_mat : Accuracy: 99.53 %</p> 
<p>Load ảnh: Muong_den-1020.jpg</p> <p>The type of woods is: Muong_den : Accuracy: 99.99 %</p> 	<p>Load ảnh: Sa_moc_dau-1010.jpg</p> <p>The type of woods is: Sa_moc_dau : Accuracy: 92.17 %</p> 
<p>Load ảnh: Tau_mat-1025.jpg</p>	<p>Load ảnh: Thuy_tung-1016.jpg</p>

<p>The type of woods is: Tau_mat : Accuracy: 75.29 %</p> 	<p>The type of woods is: Thuy_tung : Accuracy: 99.95 %</p> 
<p>Load ảnh: Trai_li-1096.jpg</p> <p>The type of woods is: Trai_li : Accuracy: 99.53 %</p> 	

3. Nhận diện trên app bằng python:

a) Giới thiệu về tkinter

Tk là một bộ công cụ widget đa nền tảng, mã nguồn mở cho phép các nhà phát triển tạo GUI cơ bản bằng cách sử dụng phát triển dựa trên widget. Tk ban đầu được phát hành vào năm 1991 với thư viện gốc được viết bằng C. Kể từ đó một số Language Binding đã được phát hành, bao gồm: Haskell, Ruby, Perl và Python.

Liên kết Python cho Tk được gọi là Tkinter và nó sẽ là thư viện mà chúng tôi sẽ tập trung vào trong phần giới thiệu này. Tkinter kể từ đó đã trở thành tiêu chuẩn để ta GUI bằng Python tuy nhiên có rất nhiều lựa chọn thay thế bao gồm: PyQt, PySide, PyGame, wxPython và PyGTK.



b) Code trên python

```
from cProfile import label
from pickle import FROZENSET
from tkinter import *
from tkinter import filedialog
import os
import tkinter as tk
from PIL import Image, ImageTk
from matplotlib.image import thumbnail
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
import cv2
import numpy as np

model = load_model('C:/Users/ADMIN/Desktop/AI/Cuoi_ki/Data9-1/best_model.h5')

classes = ["Bach_dang_nk", "Cao_su", "Dang_huong_viet", "Cho_chi", "Anh_dao",
           "Hoang_dan", "Tau_mat", "Lim", "Iroko", "Mit_mat",
           "Muong_den", "Lat_hoa", "Sa_moc_dau", "Trai_li", "Thuy_tung"]

def showimage():
    global fln
    fln = filedialog.askopenfilename(initialdir=os.getcwd(),
                                     title="Select Image File",
                                     filetypes=(("JPG File", ".jpg"),
                                                ("PNG File", ".png"),
                                                ("ALL Files", ".")))

    img = Image.open(fln)
    img.thumbnail((300, 300))
    img = ImageTk.PhotoImage(img)
    lbl3.configure(image=img)
    lbl3.image = img

def recognize():
    global lbl1
    global lbl2
    img_path = fln
    img = plt.imread(img_path)
    print('Input image shape is ', img.shape)
    img = cv2.resize(img, (150, 150))
    print('the resized image has shape ', img.shape)
    plt.axis('off')
    plt.imshow(img)

    img = np.expand_dims(img, axis=0)
    print('image shape after expanding dimensions is ', img.shape)
    y_pred = model.predict(img)
    score = y_pred.max()
    score = round(score*100, 2)
    y_classes = [np.argmax(element) for element in y_pred]
    print('the shape of prediction is ', y_pred.shape)
```

```

print(f'the image is predicted as being {classes[y_classes[0]]} with a probability of {score} %')
lb11 = Label(root, text = f"Nhận Diện: {classes[y_classes[0]]}" , fg= "green", font=("Arial", 20))
lb11.pack(pady= 20)
lb12 = Label(root, text = f"Độ chính xác: {score} %" , fg= "blue", font=("Arial", 20))
lb12.pack(pady= 20)
return

def clear():
    lb11.after(1000, lb11.destroy())
    lb12.after(1000, lb12.destroy())
    return

root = Tk()
root.title("Wood Recognition")
root.geometry("600x600")

frm = Frame(root)
frm.pack(side=BOTTOM, padx=15, pady=15)

lb1 = Label(root,
            text = "Wood Recognition",
            fg= "red",
            font=("Arial", 30))
lb1.pack(padx=10, pady= 10)

lb13 = Label(root)
lb13.pack()

btn1 = Button(frm, text = "Browser Image", command= showimage)
btn1.pack(side=tk.LEFT, padx= 15)

btn2 = Button(frm, text = "Recognize", command= recognize)
btn2.pack(side=tk.LEFT, padx= 15, pady = 10)

btn3 = Button(frm, text = "Clear", command= clear )
btn3.pack(side=tk.LEFT, padx=20)

btn4 = Button(frm, text = "EXIT", command=lambda: exit())
btn4.pack(side=tk.LEFT, padx=20)

root.mainloop()

```

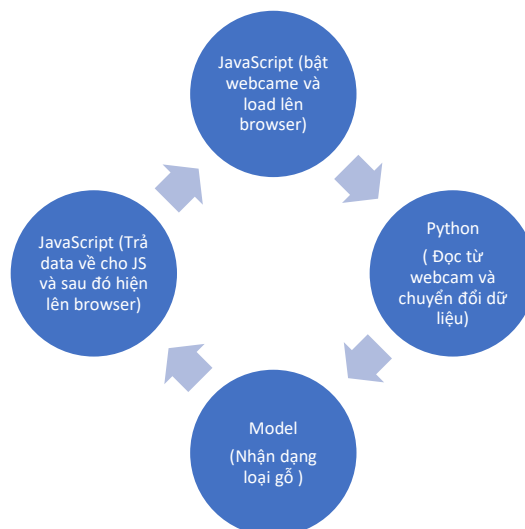
c) Kết quả:



4. Nhận diện bằng camera trên colaboratory

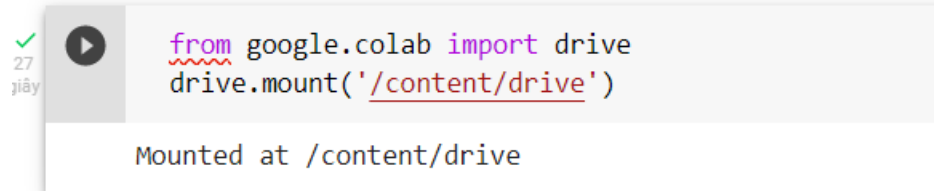
a) Quy trình thực hiện

- Sơ đồ thực hiện nhận dạng trên webcam



b) Code

- Kết nối với google drive để lấy dữ liệu, model đã được lưu



A screenshot of a Google Colab code cell. On the left, there is a green checkmark and the text '27 giây' (27 seconds). The code cell contains the following Python code: `from google.colab import drive` and `drive.mount('/content/drive')`. Below the code, a message states 'Mounted at /content/drive'.

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

- JavaScript để bật webcam và load lên browser

```
from IPython.display import display, Javascript, Image
# JavaScript to properly create our live video stream using our webcam as input

def video_stream():
    js = Javascript('''
        var video;
        var div = null;
        var stream;
        var captureCanvas;
        var imgElement;
        var labelElement;

        var pendingResolve = null;
        var shutdown = false;

        function removeDom() {
            stream.getVideoTracks()[0].stop();
            video.remove();
            div.remove();
            video = null;
            div = null;
            stream = null;
            imgElement = null;
            captureCanvas = null;
            labelElement = null;
        }

        function onAnimationFrame() {
            if (!shutdown) {
                window.requestAnimationFrame(onAnimationFrame);
            }
            if (pendingResolve) {
                var result = "";
                if (!shutdown) {
                    captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
                    result = captureCanvas.toDataURL('image/jpeg', 0.8)
                }
                var lp = pendingResolve;
                pendingResolve = null;
                lp(result);
            }
        }
    ''')
    display(js)
    Image()
```

```

async function createDom() {
  if (div !== null) {
    return stream;
  }

  div = document.createElement('div');
  div.style.border = '2px solid black';
  div.style.padding = '3px';
  div.style.width = '100%';
  div.style.maxWidth = '600px';
  document.body.appendChild(div);

  const modelOut = document.createElement('div');
  modelOut.innerHTML = "<span>Status:</span>";
  labelElement = document.createElement('span');
  labelElement.innerText = 'No data';
  labelElement.style.fontWeight = 'bold';
  modelOut.appendChild(labelElement);
  div.appendChild(modelOut);

  video = document.createElement('video');
  video.style.display = 'block';
  video.width = div.clientWidth - 6;
  video.setAttribute('playsinline', '');
  video.onclick = () => { shutdown = true; };
  stream = await navigator.mediaDevices.getUserMedia(
    {video: { facingMode: "environment" }});
  div.appendChild(video);

  imgElement = document.createElement('img');
  imgElement.style.position = 'absolute';
  imgElement.style.zIndex = 1;
  imgElement.onclick = () => { shutdown = true; };
  div.appendChild(imgElement);

  const instruction = document.createElement('div');
  instruction.innerHTML =
    '<span style="color: red; font-weight: bold;">' +
    'Bấm vào video để dừng</span>';
  div.appendChild(instruction);
  instruction.onclick = () => { shutdown = true; };

  video.srcObject = stream;
  await video.play();

  captureCanvas = document.createElement('canvas');
  captureCanvas.width = 640; //video.videoWidth;
  captureCanvas.height = 480; //video.videoHeight;
  window.requestAnimationFrame(onAnimationFrame);

  return stream;
}

```

```

async function stream_frame(label, imgData) {
  if (shutdown) {
    removeDom();
    shutdown = false;
    return '';
  }

  var preCreate = Date.now();
  stream = await createDom();

  var preShow = Date.now();
  if (label !== '') {
    labelElement.innerHTML = label;
  }

```

```

  if (imgData !== '') {
    var videoRect = video.getClientRects()[0];
    imgElement.style.top = videoRect.top + "px";
    imgElement.style.left = videoRect.left + "px";
    imgElement.style.width = videoRect.width + "px";
    imgElement.style.height = videoRect.height + "px";
    imgElement.src = imgData;
  }

  var preCapture = Date.now();
  var result = await new Promise(function(resolve, reject) {
    pendingResolve = resolve;
  });
  shutdown = false;

  return {'create': preShow - preCreate,
        'show': preCapture - preShow,
        'capture': Date.now() - preCapture,
        'img': result};
}
''')
display(js)

def video_frame(label, bbox):
  data = eval_js('stream_frame("{}","{}").format(label, bbox)')
  return data

```

- Tiến hành đọc dữ liệu từ webcam người dùng và trả về cho python trên colab

```

) # function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
    """
    Params:
        js_reply: JavaScript object containing image from webcam
    Returns:
        img: OpenCV BGR image
    """

```

```

# decode base64 image
image_bytes = b64decode(js_reply.split(',')[1])
# convert bytes to numpy array
jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
# decode numpy array into OpenCV BGR image
img = cv2.imdecode(jpg_as_np, flags=1)

return img

# function to convert OpenCV Rectangle bounding box image
# into base64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
    """
    Params:
        bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
    Returns:
        bytes: Base64 image byte string
    """

# convert array into PIL image
bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
iobuf = io.BytesIO()
# format bbox into png for return
bbox_PIL.save(iobuf, format='png')
# format return string
bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue())), 'utf-8'))

return bbox_bytes

```

- Nhận diện gỗ trên colab

```

[9] %cd /content
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import numpy as np
import PIL
import io
import cv2
from keras.models import load_model

# start streaming video from webcam
video_stream()
# label for video
label_html = 'Đang lấy hình ảnh...'
# initialize bounding box to empty
bbox = ''
count = 0

```

```
# Load model
model_file_path = "/content/drive/MyDrive/Colab Notebooks/best_model.h5"
model = load_model(model_file_path)
```

```
classes = ["Bach_dang_nk", "Cao_su", "Dang_huong_viet", "Cho_chi", "Anh_dao",
           "Hoang_dan", "Tau_mat", "Lim", "Iroko", "Mit_mat",
           "Muong_den", "Lat_hoa", "Sa_moc_dau", "Trai_li", "Thuy_tung"]
```

```
while True:
```

```
    # Đọc ảnh trả về từ JS
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break
```

```
    # convert JS response to OpenCV Image
    frame = js_to_image(js_reply["img"])
```

```
    # Resize để đưa vào model
    frame_p = cv2.resize(frame, dsize=(150,150))
    tensor = np.expand_dims(frame_p, axis=0)
```

```
    # Feed vào mạng
    y_pred = model.predict(tensor)
    score = y_pred.max()
    score = round(score*100,2)
    class_id = np.argmax(y_pred)
    class_name = classes[class_id]
```

```
    # Vẽ lên một ảnh để tạo nữa overlay
```

```
    # create transparent overlay for bounding box
    bbox_array = np.zeros([480,640,4], dtype=np.uint8)
```

```
    bbox_array = cv2.putText(bbox_array, "{}".format(class_name),
                             (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                             (0, 255,0), 2)
```

```
bbox_array = cv2.putText(bbox_array, "{}".format(score),
                          (400, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                          (0, 255,0), 2)
```

```
bbox_array[:, :, 3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
```

```
# convert overlay of bbox into bytes
```

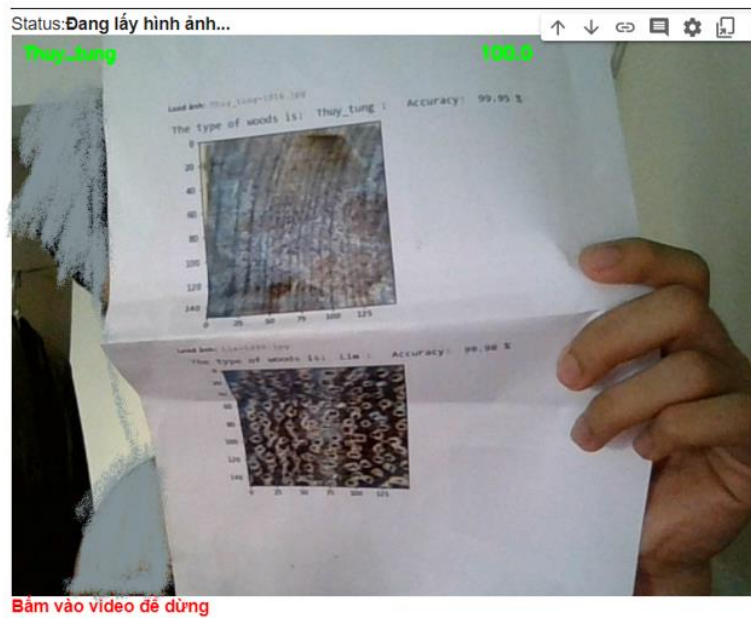
```
bbox_bytes = bbox_to_bytes(bbox_array)
```

```
# update bbox so next frame gets new overlay
```

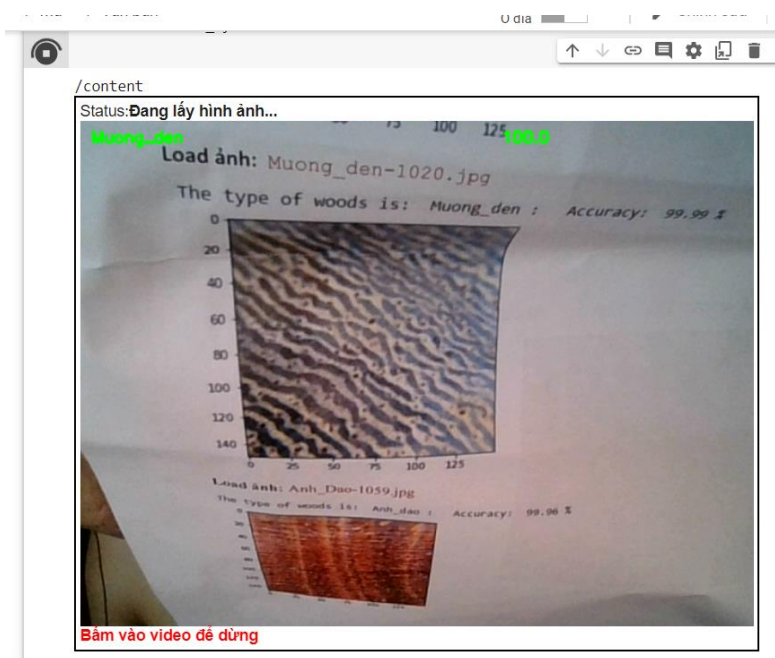
```
bbox = bbox_bytes
```

c) Kết quả

- Nhận dạng gỗ Thuy_tung trên colab bằng webcam



- Nhận dạng gỗ Muong den colab bằng webcam



CHƯƠNG V: ĐÁNH GIÁ KẾT QUẢ

1. Đánh giá kết quả

Theo như kết quả độ chính xác khi tạo ra file model đạt 96,2%. Trong quá trình thử nghiệm độ chính xác của dự án, em nhận thấy hệ thống có những mặt đạt được và chưa đạt được như sau:

Những mặt đạt được:

- Hệ thống nhận dạng đúng được các loại gỗ mà mình đã training. Có thể hiện được phần trăm độ chính xác so với vật mẫu.
- Có sử dụng app trên máy tính để nhận dạng.
- Có sử dụng webcam để nhận dạng

Những mặt chưa đạt được:

- Quá trình lấy mẫu mới để trải gặp nhiều khó khăn.
- Nhận dạng phân loại đôi lúc còn sai sót do có điều kiện tác động bên ngoài như ánh sáng, độ ẩm,..
- Khi vật mẫu dính 1 vật gì đó thì sẽ giảm đi độ chính xác rất nhiều.
- Do không có vật mẫu. Chỉ xác định thông qua ảnh đã photo nên vẫn còn nhiều sai sót.
- Mặc dù nhận dạng được trên webcam nhưng độ chính xác lại không cao. Quá trình nhận dạng còn gặp nhiều sai sót

2. Hướng phát triển

- Trong thời gian tới, em sẽ tiếp tục cải tiến sản phẩm về vấn đề trên bề mặt gỗ có thêm vật bị dính nhưng vẫn đảm bảo độ chính xác của hệ thống.
- Thêm các tính năng tự động tìm kiếm tài liệu, tự học của AI.
- Thiết kế giao diện tương tác người dùng thu hút và đẹp hơn.

KẾT LUẬN

Qua quá trình thực hiện đề tài, em tuy gặp không ít khó khăn và trở ngại khi thực hiện đề tài nhưng cuối cùng cũng đã đạt được kết quả khả quan và đó cũng là động lực để em tiếp tục hoàn thiện, phát triển đề tài hơn nữa.

Em đã phần nào ứng dụng được những kiến thức đã học vào việc làm một dự án cụ thể, những kỹ năng rất quan trọng với người kỹ sư tương lai. Đồng thời, em cũng đã củng cố thêm cách giải quyết các vấn đề khó khăn trong quá trình làm dự án.

TÀI LIỆU THAM KHẢO

- [1]. PGS. TS. Nguyễn Trường Thịnh, Giáo trình trí tuệ nhân tạo (2021)
- [2]. Mì AI, Xây dựng web app, triển khai model, Link truy cập:
<https://miai.vn/2021/12/29/xay-dung-web-app-trien-khai-model-trong-1-phut-voi-streamlit-mi-ai/>
- [3]. Mì AI, Xây dựng realtime trên colab, Link truy cập:
https://github.com/thangnch/MIAI_Colab_Realtime
- [4]. Viblo, *Tìm hiểu về OpenCV*, Link truy cập:
<https://viblo.asia/p/tim-hieu-ve-opencv-Do754NrXZM6>
- [5]. Ichi, *Giới thiệu về Tkinter*, Link truy cập:
<https://ichi.pro/vi/gui-cho-python-gioi-thieu-ve-tkinter-207369661260093>
- [6]. TOPDev, Thuật toán CNN – Convolutional Neural Network, Link truy cập:
<https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>
- [7]. Review, Tài liệu về nhận dạng gỗ (Computer vision-based wood identification and its expansion and contribution potentials in wood science: A review), Link truy cập:
<https://plantmethods.biomedcentral.com/articles/10.1186/s13007-021-00746-1>
- [8]. Article, Tài liệu về nhận dạng gỗ (Wood Defect Detection Based on Depth Extreme Learning Machine), Link truy cập:
https://www.researchgate.net/publication/346412523_Wood_Defect_Detection_Based_on_Depth_Extreme_Learning_Machine
- [9]. Wood Research, Tài liệu về nhận dạng gỗ (WOOD SPECIES IDENTIFICATION BASED ON AN ENSEMBLE OF DEEP CONVOLUTION NEURAL NETWORKS), Link truy cập:
https://www.researchgate.net/publication/346412523_Wood_Defect_Detection_Based_on_Depth_Extreme_Learning_Machine
- [10]. Review Article, Tài liệu về nhận dạng gỗ (Wood Recognition and Quality Imaging Inspection Systems), Link truy cập:
https://www.researchgate.net/publication/345013180_Wood_Recognition_and_Quality_Imaging_Inspection_Systems