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

**KHOA CƠ KHÍ CHẾ TẠO MÁY**

**BỘ MÔN CƠ ĐIỆN TỬ**

**🙡🙢✪🙠🙣**

****

**BÁO CÁO CUỐI KÌ**

**THỊ GIÁC MÁY**

**NHẬN DẠNG ĐỘNG VẬT HOANG DÃ ỨNG DỤNG XỬ LÝ ẢNH VÀ TRÍ TUỆ NHÂN TẠO**

**GVHD: TS. Nguyễn Văn Thái**

**MHP: MAVI332529\_22\_2\_05**

**Họ và tên: Võ Văn Thuận 20146084**

**Võ Tấn Thịnh 20146536**

**Lê Minh Tiên 20146196**

**Năm học: 2022 - 2023**

*Thành phố Hồ Chí Minh,28 tháng 5 năm 2023*

**I.GIỚI THIỆU TỔNG QUAN**

**1 Giới thiệu đề tài**

Động vật hoang dã đóng vai trò rất quan trọng trong môi trường sống của chúng ta. Chúng giúp duy trì sự cân bằng sinh thái và đóng góp vào chuỗi thức ăn tự nhiên, hỗ trợ cho việc duy trì sự phong phú và đa dạng của các loài trong hệ sinh thái. Tuy nhiên, động vật hoang dã đang đối mặt với nhiều thách thức trong môi trường sống của chúng. Một trong những vấn đề lớn nhất là vấn nạn săn bắn trái phép. Việc săn bắn trái phép động vật hoang dã không chỉ ảnh hưởng đến hệ sinh thái tự nhiên, mà còn gây ra nhiều tác động tiêu cực đến đời sống con người. Săn bắn trái phép động vật hoang dã có thể dẫn đến sự suy giảm của các loài động vật hoang dã, đẩy chúng vào tình trạng nguy cấp và làm mất đi sự đa dạng sinh học của khu vực đó. Bên cạnh đó, việc săn bắn trái phép còn có thể gây ra các tai nạn khiến con người bị thương hoặc thiệt mạng. Do đó, việc bảo vệ và bảo tồn động vật hoang dã là rất cần thiết. Chúng ta cần tăng cường giám sát và quản lý môi trường sống của các loài động vật hoang dã để giảm thiểu tác động tiêu cực từ con người. Ngoài ra, cũng cần xây dựng những chính sách và luật pháp hợp lý để kiềm chế và ngăn chặn hoạt động săn bắn trái phép động vật hoang dã. Chỉ có như vậy, chúng ta mới có thể bảo vệ được các loài động vật hoang dã và duy trì được hệ sinh thái tự nhiên trong tình trạng ổn định và cân bằng. Đề tài nhận diện động vật hoang dã sử dụng mạng neural tích chập (CNN) là một chủ đề rất thú vị trong lĩnh vực trí tuệ nhân tạo (AI). Đề tài xây dựng một hệ thống tự động phân loại hình ảnh động vật hoang dã từ các bức ảnh đầu vào. Để thực hiện điều này, CNN được sử dụng là một công cụ mạnh mẽ để học và trích xuất đặc trưng từ các hình ảnh quan sát được. Sau đó, việc huấn luyện mô hình để phân loại các loài động vật hoang dã dựa trên các đặc trưng đã học được sẽ được tiến hành.

**2. Lý do chọn đề tài**

Như đã đề cập ở trên, vấn nạn săn bắn trái phép động vật hoang dã đang ở mức báo động, ảnh hưởng xấu đến môi trường sinh sống và biểu đồ phân bố của động vật hoang dã trên thế giới bị thay đổi. Chính vì thế nên đề tài này rất quan trọng trong lĩnh vực bảo tồn động vật và môi trường. Việc nghiên cứu về vấn đề này có thể giúp đánh giá các nguy cơ và mối nguy hiểm tiềm ẩn của việc săn bắn động vật hoang dã. Giúp hiểu rõ hơn về tầm quan trọng của việc bảo tồn động vật hoang dã và môi trường. Các nhà khoa học và nhân viên bảo tồn động vật hoang dã có thể sử dụng các kết quả nghiên cứu này để phát triển các chiến lược bảo tồn hiệu quả hơn và ngăn chặn sự suy giảm của các loài động vật hoang dã. Có thể giúp tăng cường nhận thức của công chúng về việc bảo tồn động vật hoang dã và môi trường. Các kết quả nghiên cứu có thể được sử dụng để giáo dục và tuyên truyền cho các cộng đồng về việc bảo vệ các loài động vật hoang dã và môi trường sống của chúng.

**3. Mục tiêu nghiên cứu**

Kết quả của nghiên cứu này sẽ giúp cho việc nhận diện động vật hoang dã, môi trường sinh sống chủ yếu, đặc tính và số lượng trở nên dễ dàng và chính xác hơn, đồng thời có thể ứng dụng trong nhiều lĩnh vực khác nhau như bảo tồn động vật hoang dã, nghiên cứu sinh thái học và thăm dò địa hình. Tìm hiểu và đánh giá các nguy cơ, hậu quả và tiềm năng của việc săn bắn động vật hoang dã trên toàn cầu. Nhằm mục đích phát triển các chiến lược và chính sách bảo tồn động vật hoang dã hiệu quả hơn, từ đó ngăn chặn sự suy giảm và tuyệt chủng của các loài động vật hoang dã. Nâng cao kiến thức và kỹ năng trong lĩnh vực bảo tồn động vật và môi trường.

**4. Phương pháp nghiên cứu**

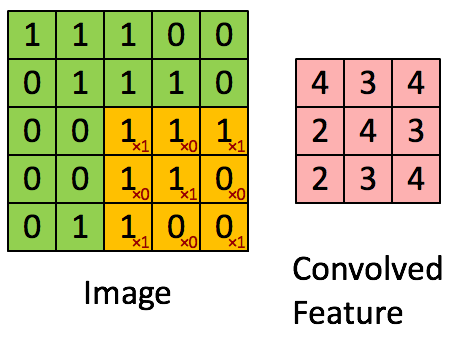
* Tìm hiểu về môi trường sống của động vật hoang dã và nạn săn bắn trái phép động vật hoang dã trên thế giới
* Tìm kiếm thông tin và dữ liệu hình ảnh phục vụ cho bài nghiên cứu từ Google hình ảnh và báo chí trên các nền tảng xã hội
* Xây dựng “Model trainning” và tiến hành chạy thử để kiểm chứng

**II.CƠ SỞ LÝ THUYẾT**

**1. Thuật toán CNN - Convolutional Neural Network là gì**

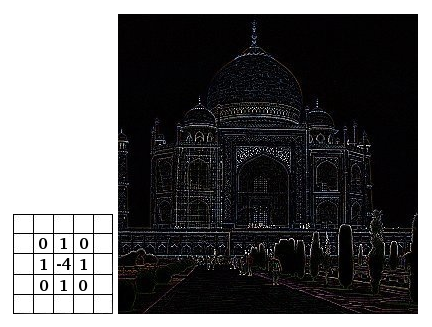
Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Được thiết kế để học các đặc trưng không gian từ các pixel ảnh hoặc khung hình video và phân tích cấu trúc không gian của các đối tượng trong hình ảnh. CNN bao gồm nhiều lớp, bao gồm lớp tích chập (convolutional layer), lớp kích hoạt phi tuyến (non-linear activation layer) và lớp tổng hợp (pooling layer). Lớp tích chập thực hiện việc tìm kiếm các đặc trưng có hình dạng nhất định trong các vùng không gian khác nhau của ảnh thông qua các bộ lọc. Lớp kích hoạt phi tuyến giúp kích hoạt các neuron trong mạng và tạo ra non-linearity. Cuối cùng, lớp tổng hợp giúp giảm kích thước của các bản đồ đặc trưng bằng cách áp dụng các phép toán như lấy giá trị lớn nhất hoặc trung bình trong các vùng không gian nhỏ. Các lớp này thường được sắp xếp theo một cấu trúc chồng lên nhau để hình thành một mô hình CNN hoàn chỉnh. CNN là một công cụ rất hiệu quả cho việc phân loại ảnh, phát hiện đối tượng và các nhiệm vụ xử lý ảnh khác.

**Convolutional** là một cửa sổ trượt (Sliding Windows) trên một ma trận (xem hình)

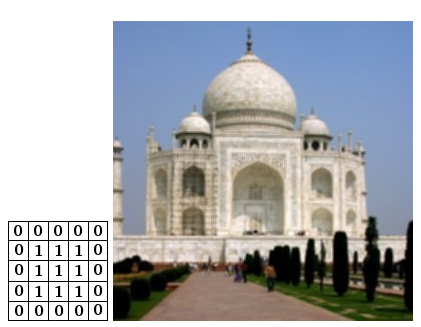


**Hình 2.1.** Ma trận Convolutional

Convolution là một phép toán trong đại số tuyến tính được sử dụng rộng rãi trong xử lý tín hiệu và xử lý hình ảnh. Nó được sử dụng để tìm kiếm các mẫu hoặc đặc trưng trong tín hiệu hoặc hình ảnh bằng cách thực hiện phép nhân giữa một kernel (hay còn gọi là filter hoặc bộ lọc) và một vùng của tín hiệu hoặc hình ảnh. Các lớp tích chập trong CNN thực hiện phép tích chập giữa kernel và một vùng của ảnh để tạo ra các bản đồ đặc trưng (feature maps). Trong hình ảnh ví dụ Hình 2.2, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước 5×5 và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột. Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3. Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhận ma trận Filter với ma trận ảnh 5×5 bên trái.



**Hình 2.2.** Hình ảnh trắng đen được số hóa



**Hình 2.3.** Hình ảnh được Convoled feature

**2. Cấu trục 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 CNNs, 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. Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Cấu trúc mạng của CNN thường bao gồm các lớp sau:

+ Lớp đầu vào (Input layer): Nhận dữ liệu đầu vào, thường là các hình ảnh.

+ Lớp tích chập (Convolutional layer): Thực hiện tích chập giữa các bộ lọc và đầu vào để tạo ra các bản đồ đặc trưng cho hình ảnh. Các bộ lọc này được học thông qua quá trình lan truyền ngược.

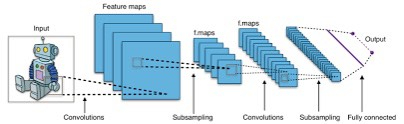
+ Lớp kích hoạt phi tuyến (Activation layer): Thực hiện phép toán phi tuyến tính trên các giá trị đầu ra của lớp tích chập để kích hoạt các neuron trong mạng.

+ Lớp tổng hợp (Pooling layer): Thực hiện giảm kích thước của các bản đồ đặc trưng bằng cách lấy giá trị lớn nhất hoặc trung bình trong các vùng không gian nhỏ.

+ Lớp đầy đủ kết nối (Fully connected layer): Thực hiện tính toán đầu ra cuối cùng bằng cách ghép các bản đồ đặc trưng lại với nhau và áp dụng các phép toán phi tuyến tính.

+ Lớp đầu ra (Output layer): Đưa ra dự đoán cuối cùng cho bài toán, thường là một vector xác suất cho các lớp của phân loại hoặc các giá trị dự đoán cho các bài toán hồi quy.

Trong quá trình huấn luyện mạng (traning) 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.



**Hình 2.4.** Cấu trúc mạng CNN

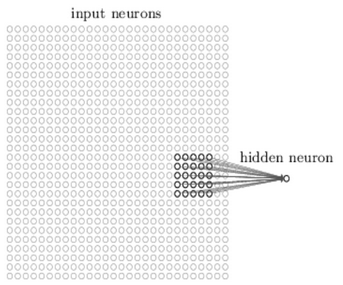
Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Pooling layer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

Mạng CNN sử dụng 3 ý tưởng cơ bản:

* **Các trường tiếp nhận cục bộ** (local receptive field)
* **Trọng số chia sẻ** (shared weights)
* **Tổng hợp** (pooling).

**3. Trường tiếp nhận cục bộ (local receptive field)**

Đầu vào của mạng CNN là một ảnh. Ví dụ như ảnh có kích thước 28×28 thì tương ứng đầu vào là một ma trận có 28×28 và giá trị mỗi điểm ảnh là một ô trong ma trận. Trong mô hình mạng ANN truyền thống thì chúng ta sẽ kết nối các neuron đầu vào vào tầng ảnh. Tuy nhiên trong CNN chúng ta chỉ kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5 tương ứng (28- 5 + 1) 24 điểm ảnh đầu vào. Mỗi một kết nối sẽ học một trọng số và mỗi neuron ẩn sẽ học một bias. Mỗi một vùng 5×5 đấy gọi là một trường tiếp nhận cục bộ.



**Hình 2.5.** Ma trận 28x28

Như vậy, local receptive field thích hợp cho việc phân tách dữ liệu ảnh, giúp chọn ra những vùng ảnh có giá trị nhất cho việc đánh giá phân lớp.

**4.Trọng số chia sẻ (shared weight and bias)**

Đầu tiên, các trọng số cho mỗi filter (kernel) phải giống nhau. Tất cả các nơ-ron trong lớp ẩn đầu sẽ phát hiện chính xác feature tương tự chỉ ở các vị trí khác nhau trong hình ảnh đầu vào. Chúng ta gọi việc map từ input layer sang hidden layer là một feature map. Tóm lại, một convolutional layer bao gồm các feature map khác nhau. Mỗi một feature map giúp detect một vài feature trong bức ảnh. Lợi ích lớn nhất của trọng số chia sẻ là giảm tối đa số lượng tham số trong mạng CNN.

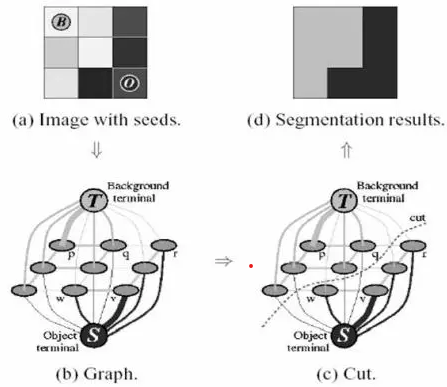
**5. Lớp tổng hợp (pooling layer)**

Lớp pooling thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron. Qua lớp Max Pooling thì số lượng neuron giảm đi phân nửa. Trong một mạng CNN có nhiều Feature Map nên mỗi Feature Map sẽ cho mỗi Max Pooling khác nhau. Có thể thấy rằng Max Pooling là cách hỏi xem trong các đặc trưng này thì đặc trưng nào là đặc trưng nhất. Ngoài Max Pooling còn có L2 Pooling. Cuối cùng đặt tất cả các lớp lại với nhau thành một CNN với đầu ra gồm các neuron với số lượng tùy bài toán.

**6. Thuật toán Grapcut**

Thuật toán GrabCut được thiết kế bởi Carsten Rother, Vladimir Kolmogorov & Andrew Blake từ Microsoft Research Cambridge, Vương quốc Anh. trong bài báo của họ, "GrabCut": khai thác nền trước tương tác bằng cách sử dụng cắt giảm đồ thị lặp lại. Một thuật toán rất cần thiết cho việc khai thác ảnh nền, tách nền đối với các hình ảnh chứa nọi dung cần xác định, và kết quả là GrabCut ra đời.

* Ý tưởng ở đây là tìm tiền cảnh và xóa nền.
* Chúng ta sẽ thấy được thuật toán GrabCut làm việc như thế nào, trích xuất ảnh nền trong hình ảnh.
* Chúng ta sẽ tạo ra một ứng dụng tương tác hình ảnh với thuật toán GrabCut cho việc tách nên các hình ảnh.



Trước khi có các mạng phân đoạn ngữ nghĩa/phiên bản/học sâu như Mask R-CNN, U-Net, v.v., GrabCut là phương pháp để phân đoạn chính xác tiền cảnh của hình ảnh từ hậu cảnh.

Thuật toán GrabCut hoạt động bằng cách:

Chấp nhận hình ảnh đầu vào bằng (1) hộp giới hạn chỉ định vị trí của đối tượng trong hình ảnh mà chúng tôi muốn phân đoạn hoặc (2) mặt nạ xấp xỉ phân đoạn

Lặp đi lặp lại thực hiện các bước sau:

Bước 1: Ước tính sự phân bố màu của tiền cảnh và hậu cảnh thông qua Mô hình hỗn hợp Gaussian (GMM)

Bước 2: Xây dựng trường ngẫu nhiên Markov trên nhãn pixel (nghĩa là nền trước so với nền)

Bước 3: Áp dụng tối ưu hóa cắt đồ thị để đi đến phân đoạn cuối cùng

**7. Các thư viện hỗ trợ lập trình**

**7.1 Pilow-xử lý ảnh**

Khi sử dụng Pilow, người dùng có thể mở và lưu hình ảnh, mà còn có thể

xử lý đặc điểm của hình ảnh, chẳng hạn như màu sắc, độ mờ, độ sáng, tối, ...

Pilow hỗ trợ xử lý nhiều tệp hình ảnh khác nhau như PDF, WebP, PCX,

PNG, JPEG, GIF, PSD, WebP, PCX, GIF, IM, EPS, ICO, BMP, và còn nhiều

hơn thế.

Với Piloư, bạn có thể dễ dàng tạo ra những ảnh thu nhỏ (thumbnails) cho

hình ảnh, những ảnh thu nhỏ này mang hầu hết đặc điểm của hình ảnh và gần

như không khác gì ảnh gốc ngoại trừ chúng được thu nhỏ lại.

Pillow hỗ trợ một bộ sưu tập các bộ lọc hình ảnh như – FIND\_EDGES,

DETAIL, SMOOTH, BLUR, CONTOUR, SHARPEN, SMOOTH\_MORE, etc.

**7.2 OpenCV- Xử lý hình ảnh**

OpenCV là một gói mô-đun hình ảnh lý tưởng cho phép bạn đọc và ghi,

thay đổi dữ liệu nhiều hình ảnh cùng một lúc.

Tạo ra thị giác máy tính cho phép bạn xây dựng lại, gián đoạn và thông

hiểu môi trường 3D từ môi trường 2D tương ứng của nó.

OpenCV được xử dùng nhiều trong nhận diện vật thể và hình ảnh được

thiết lập trước, chẳng hạn như khuôn mặt, động vật, cây cối, các vật thể di

chuyển, etc.

Bạn cũng có thể lưu và chụp bất kỳ khoảnh khắc nào của video và cũng có

thể phân tích các thuộc tính khác nhau của nó như chuyển động, nền, etc.

OpenCV tương thích với nhiều hệ điều hành như Windows, OS-X, Open

BSD và nhiều hệ điều hành khác.

**7.3 Keras-Deep Learning**

Keras là một thư viện Python mạnh mẽ. Nó có khả năng chạy trên Bộ công

cụ nhận thức của Microsoft (Microsoft Cognitive Toolkit), PaidML, TensorFlow và các nền tảng khác.

Thư viện Python này có nhiều cách triển khai từ các khối hình thành mạng

nơ-ron - chức năng, lớp, trình tối ưu hóa, các đối tượng và những thứ khác.

Keras cũng có nhiều công cụ hữu ích cho phép bạn làm việc với các hình

ảnh và văn bản khác nhau một cách dễ dàng.

Nó không chỉ hỗ trợ mạng nơ-ron mà còn cung cấp một môi trường hỗ trợ

đầy đủ cho các mạng nơ-ron tích hợp và tái hiện.

Sử dụng Keras, bạn có thể xây dựng các mô hình sâu cho điện thoại thông

minh - cả Android và iOS hoặc cả Java Virtual Machine.

**7.4 Numpy-xử lý mảng đa chiều, ma trận**

Numpy là một mô-đun mở rộng mã nguồn mở cho Python, cung cấp các

chức năng biên dịch nhanh cho các thao tác toán học và số, thậm chí là với

những ma trận và mảng có lượng dữ liệu khổng lồ. Bên cạnh đó các mô-đun

cung cấp một thư viện lớn các chức năng toán học cấp cao để hoạt động trên

các ma trận và mảng một cách dễ dàng và thuận tiện.

Numpy cung cấp những masked arrays đồng thời với mảng gốc. Nó cũng

đi kèm với các chức năng như thao tác với hình dạng logic, biến đổi Fourier rời

rạc, đại số tuyến tính tổng quát, và nhiều hơn nữa.

Gói mô-đun này cung cấp các công cụ hữu ích để tích hợp với các ngôn

ngữ lập trình khác. Chẳng hạn như C, C++, và ngôn ngữ lập trình Fortran.

Numpy cung cấp các chức năng tương đương với MATLAB. Cả hai đều

cho phép người dùng thao tác nhanh hơn.

**III. NỘI DUNG THỰC HIỆN**

**1. Datasets**

Sử dụng Google Colab để xây dựng mô hình trainning.

Trong nghiên cứu này, dữ liệu về về top 5 động vật hoang dã trên thế giới được lựa chọn từ Google hình ảnh, các trang báo tạp chí trên các nền tảng xã hội bao gồm 405 hình ảnh về 5 loài động vật với độ phân giải thay đổi từ 400pixel đến 2000pixel. Trong 405 hình ảnh này gồm 100 hình từ các đối tượng làm chuẩn (Validation) và 300 hình từ các đối tượng huấn luyện. Dữ liệu này được chia ra thành 3 thư mục nhỏ: train (dùng để trainning model), test (dùng để test model), validation (dùng để làm tiêu chuẩn, mức thẩm định cho training). Đường dẫn dữ liệu trên GoogleDrive:

<https://drive.google.com/drive/u/0/folders/1O8rW3jBG0aeJGt3R0va7In4mmhuv975J>

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Loài vật** | **Số lượng ảnh** | **Tổng** |
| **TRAIN** | BEAR | 60 | 300 |
| ELEPHANT | 60 |
| LION | 60 |
| PENGUIN | 60 |
| TIGER | 60 |
| **VALIDATION** | BEAR | 20 | 100 |
| ELEPHANT | 20 |
| LION | 20 |
| PENGUIN | 20 |
| TIGER | 20 |

**Bảng 3.1.** Datasets

**2. Xây dựng mô hình**

Bước 1: Liên kết Google Drive

|  |
| --- |
| %cd /content/drive/MyDrive/AI/CuoiKi\_1 |

Bước 2: Khai báo các thư viện cần thiết

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  import matplotlib.pyplot as plt  import os  import numpy as np  import tensorflow as tf  from keras.utils import np\_utils  from keras.models import Sequential  from keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D, Flatten  from tensorflow.keras.optimizers import SGD, RMSprop  from keras.callbacks import EarlyStopping, ModelCheckpoint  from sklearn.utils import validation  from sklearn import preprocessing  from tensorflow.keras.utils import to\_categorical  from tensorflow.keras.models import load\_model  from tensorflow.keras.utils import load\_img, img\_to\_array  from tensorflow.keras.preprocessing.image import ImageDataGenerator |

Bước 3: Lấy dữ liệu và mô tả dữ liệu

|  |
| --- |
| training=train\_data.flow\_from\_directory('/content/drive/MyDrive/AI/CuoiKi\_1/Train',target\_size=(256,256), batch\_size=32, class\_mode='categorical')  validation=train\_data.flow\_from\_directory('/content/drive/MyDrive/AI/CuoiKi\_1/Validation',target\_size=(256,256), batch\_size=32, class\_mode='categorical') |

Bước 4: Tăng cường dữ liệu (data augmentation)

Augmentation là kĩ thuật tạo ra dữ liệu training từ dữ liệu mà ta đang có. Việc này giúp chúng ta tạo ra nhiều hình ảnh hơn từ hình ảnh gốc.

|  |
| --- |
| train\_data=ImageDataGenerator(rescale=1./255, shear\_range=0.2, zoom\_range=0.2, horizontal\_flip=True) |

Found 300 images belonging to 5 classes.

Found 100 images belonging to 5 classes.

Kết quả ta được train\_set có 300 hình ảnh thuộc 5 lớp, và val\_set có 100 hình ảnh thuộc 2 lớp.

Bước 5: Xây dựn mô hình CNN

|  |
| --- |
| model=Sequential()  model.add(Conv2D(32,(3,3), activation='relu', kernel\_initializer='he\_uniform', padding='same',input\_shape=(256,256,3)))  model.add(Conv2D(32,(3,3), activation='relu', kernel\_initializer='he\_uniform', padding='same'))  model.add(MaxPooling2D(2,2))  model.add(Conv2D(64,(3,3), activation='relu', kernel\_initializer='he\_uniform', padding='same'))  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(Conv2D(128,(3,3), activation='relu', kernel\_initializer='he\_uniform', padding='same'))  model.add(MaxPooling2D(2,2))  model.add(Conv2D(256,(3,3), activation='relu', kernel\_initializer='he\_uniform', padding='same'))  model.add(Conv2D(256,(3,3), activation='relu', kernel\_initializer='he\_uniform', padding='same'))  model.add(MaxPooling2D(2,2))  model.add(Flatten())  model.add(Dense(256,activation='relu',kernel\_initializer='he\_uniform'))  model.add(Dropout(0.2))  model.add(Dense(5,activation='softmax'))  model.summary() |

Cấu trúc mô hình CNN được sử dụng trong đề tài cụ thể như sau:

* Lớp đầu tiên là lớp Convolution2D với dữ liệu đầu vào có kích thước là 256x256x3, sử dụng 32 filter. Lớp này có kích thước cửa sổ trượt Kernel\_size là 3x3 và bước trượt là 2x2
* Lớp kế tiếp vẫn là lớp Convolution2D, sử dụng 32 filter, cửa sổ trượt Kernel\_size 3x3 và bước trượt 2x2, lớp này được tăng cường thêm nhầm tăng độ chính xác cho cấu trúc.
* Tiếp theo là lớp Maxpooling2D, kích thước pool\_size là 2x2, lớp này có nhiệm vụ giảm kích thước ảnh tuy nhiên vẫn giữ nguyên các đặc trưng của ảnh, từ đó không làm thay đổi đặc tính dữ liệu.
* Lớp thứ ba là lớp Convolution2D với filter là 64, kích thước cửa sổ trượt Kernel\_size là 3x3, kích thước trượt là 2x2
* Lớp thứ tư là lớp bổ xung Convolution2D với filter là 64, kích thước cửa sổ trượt Kernel\_size là 3x3, kích thước trượt là 2x2
* Lớp thứ năm là lớp Maxpooling2D với kích thước pool\_size là 2x2
* Lớp thứ sáu là lớp Convolution2D với filter là 128, kích thước cửa sổ trượt Kernel\_size là 3x3, kích thước trượt là 2x2
* Lớp thứ bảy là lớp bổ xung Convolution2D với filter là 128, kích thước cửa sổ trượt Kernel\_size là 3x3, kích thước trượt là 2x2
* Lớp thứ tám là lớp Maxpooling2D với kích thước pool\_size là 2x2
* Lớp thứ chín là lớp Convolution2D với filter là 256, kích thước cửa sổ trượt Kernel\_size là 3x3, kích thước trượt là 2x2
* Lớp thứ mười là lớp bổ xung Convolution2D với filter là 256, kích thước cửa sổ trượt Kernel\_size là 3x3, kích thước trượt là 2x2
* Lớp thứ mười một là lớp Maxpooling2D với kích thước pool\_size là 2x2
* Tiếp theo là lớp Falten, lớp này có nhiệm vụ đưa tất cả dữ liệu về kích thước (1,1,Chiều sâu) hay nói cách khác, dữ liệu khi qua lớp này sẽ được trải phẳng và có dạng một chiều.

Các thông số Input, Ouput cụ thể được thể hiện trong bảng thông số mô hình dưới dây:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 256, 256, 32) 896

conv2d\_1 (Conv2D) (None, 256, 256, 32) 9248

max\_pooling2d (MaxPooling2D (None, 128, 128, 32) 0

)

conv2d\_2 (Conv2D) (None, 128, 128, 64) 18496

conv2d\_3 (Conv2D) (None, 128, 128, 64) 36928

max\_pooling2d\_1 (MaxPooling (None, 64, 64, 64) 0

2D)

conv2d\_4 (Conv2D) (None, 64, 64, 128) 73856

conv2d\_5 (Conv2D) (None, 64, 64, 128) 147584

max\_pooling2d\_2 (MaxPooling (None, 32, 32, 128) 0

2D)

conv2d\_6 (Conv2D) (None, 32, 32, 256) 295168

conv2d\_7 (Conv2D) (None, 32, 32, 256) 590080

max\_pooling2d\_3 (MaxPooling (None, 16, 16, 256) 0

2D)

flatten (Flatten) (None, 65536) 0

dense (Dense) (None, 256) 16777472

dropout (Dropout) (None, 256) 0

dense\_1 (Dense) (None, 5) 1285

=================================================================

Total params: 17,951,013

Trainable params: 17,951,013

Non-trainable params: 0

Bước 6: Huấn luyện mô hình

|  |
| --- |
| model.compile(optimizer='adam',loss='categorical\_crossentropy',metrics=['accuracy'])  history=model.fit(training,epochs=40,batch\_size=16,verbose=1, validation\_data=validation) |

Mô hình được huấn luyện với số lần lọc 40, ta được kết quả như sau:

Epoch 1/40

10/10 [==============================] - 153s 13s/step - loss: 10.8712 - accuracy: 0.2300 - val\_loss: 1.6116 - val\_accuracy: 0.2100

Epoch 2/40

10/10 [==============================] - 12s 1s/step - loss: 1.6105 - accuracy: 0.2067 - val\_loss: 1.5975 - val\_accuracy: 0.2000

Epoch 3/40

10/10 [==============================] - 13s 1s/step - loss: 1.5832 - accuracy: 0.2933 - val\_loss: 1.4186 - val\_accuracy: 0.4300

Epoch 4/40

10/10 [==============================] - 13s 1s/step - loss: 1.5141 - accuracy: 0.3200 - val\_loss: 1.3613 - val\_accuracy: 0.4200

Epoch 5/40

10/10 [==============================] - 11s 1s/step - loss: 1.3189 - accuracy: 0.3933 - val\_loss: 1.1771 - val\_accuracy: 0.4300

Epoch 6/40

10/10 [==============================] - 11s 1s/step - loss: 1.2578 - accuracy: 0.4433 - val\_loss: 1.1396 - val\_accuracy: 0.4900

Epoch 7/40

10/10 [==============================] - 12s 1s/step - loss: 1.1306 - accuracy: 0.5267 - val\_loss: 1.3348 - val\_accuracy: 0.4700

Epoch 8/40

10/10 [==============================] - 12s 1s/step - loss: 1.1555 - accuracy: 0.5433 - val\_loss: 1.0894 - val\_accuracy: 0.5200

Epoch 9/40

10/10 [==============================] - 13s 1s/step - loss: 1.1005 - accuracy: 0.5333 - val\_loss: 1.1432 - val\_accuracy: 0.5300

Epoch 10/40

10/10 [==============================] - 13s 1s/step - loss: 1.1284 - accuracy: 0.5200 - val\_loss: 1.0286 - val\_accuracy: 0.5700

Epoch 11/40

10/10 [==============================] - 11s 1s/step - loss: 1.0737 - accuracy: 0.5267 - val\_loss: 1.0305 - val\_accuracy: 0.6100

Epoch 12/40

10/10 [==============================] - 11s 1s/step - loss: 0.9747 - accuracy: 0.6033 - val\_loss: 1.0545 - val\_accuracy: 0.6100

Epoch 13/40

10/10 [==============================] - 12s 1s/step - loss: 0.9880 - accuracy: 0.6167 - val\_loss: 1.2675 - val\_accuracy: 0.4800

Epoch 14/40

10/10 [==============================] - 13s 1s/step - loss: 0.9949 - accuracy: 0.5833 - val\_loss: 1.1248 - val\_accuracy: 0.5100

Epoch 15/40

10/10 [==============================] - 12s 1s/step - loss: 0.8649 - accuracy: 0.6867 - val\_loss: 1.1486 - val\_accuracy: 0.6000

Epoch 16/40

10/10 [==============================] - 12s 1s/step - loss: 0.8881 - accuracy: 0.6667 - val\_loss: 0.9643 - val\_accuracy: 0.6100

Epoch 17/40

10/10 [==============================] - 13s 1s/step - loss: 0.7679 - accuracy: 0.6800 - val\_loss: 0.9897 - val\_accuracy: 0.6700

Epoch 18/40

10/10 [==============================] - 12s 1s/step - loss: 0.7932 - accuracy: 0.6967 - val\_loss: 0.9333 - val\_accuracy: 0.5700

Epoch 19/40

10/10 [==============================] - 12s 1s/step - loss: 0.7475 - accuracy: 0.7267 - val\_loss: 0.9263 - val\_accuracy: 0.6200

Epoch 20/40

10/10 [==============================] - 12s 1s/step - loss: 0.6322 - accuracy: 0.7367 - val\_loss: 1.5605 - val\_accuracy: 0.6600

Epoch 21/40

10/10 [==============================] - 12s 1s/step - loss: 0.6045 - accuracy: 0.7467 - val\_loss: 0.9176 - val\_accuracy: 0.7100

Epoch 22/40

10/10 [==============================] - 11s 1s/step - loss: 0.5057 - accuracy: 0.8133 - val\_loss: 0.8858 - val\_accuracy: 0.7200

Epoch 23/40

10/10 [==============================] - 11s 1s/step - loss: 0.4731 - accuracy: 0.8167 - val\_loss: 1.5580 - val\_accuracy: 0.5900

Epoch 24/40

10/10 [==============================] - 12s 1s/step - loss: 0.4392 - accuracy: 0.8233 - val\_loss: 1.4906 - val\_accuracy: 0.7200

Epoch 25/40

10/10 [==============================] - 12s 1s/step - loss: 0.4430 - accuracy: 0.8300 - val\_loss: 1.4934 - val\_accuracy: 0.6800

Epoch 26/40

10/10 [==============================] - 11s 1s/step - loss: 0.4446 - accuracy: 0.8433 - val\_loss: 2.8751 - val\_accuracy: 0.6400

Epoch 27/40

10/10 [==============================] - 11s 1s/step - loss: 0.4712 - accuracy: 0.8467 - val\_loss: 0.7917 - val\_accuracy: 0.7100

Epoch 28/40

10/10 [==============================] - 12s 1s/step - loss: 0.3205 - accuracy: 0.9000 - val\_loss: 1.4199 - val\_accuracy: 0.6400

Epoch 29/40

10/10 [==============================] - 12s 1s/step - loss: 0.4732 - accuracy: 0.8200 - val\_loss: 1.4028 - val\_accuracy: 0.6300

Epoch 30/40

10/10 [==============================] - 12s 1s/step - loss: 0.3438 - accuracy: 0.8633 - val\_loss: 1.3247 - val\_accuracy: 0.6900

Epoch 31/40

10/10 [==============================] - 12s 1s/step - loss: 0.3985 - accuracy: 0.8500 - val\_loss: 1.3478 - val\_accuracy: 0.6500

Epoch 32/40

10/10 [==============================] - 11s 1s/step - loss: 0.4757 - accuracy: 0.8167 - val\_loss: 0.9574 - val\_accuracy: 0.6900

Epoch 33/40

10/10 [==============================] - 11s 1s/step - loss: 0.3273 - accuracy: 0.8933 - val\_loss: 1.9159 - val\_accuracy: 0.7300

Epoch 34/40

10/10 [==============================] - 12s 1s/step - loss: 0.2648 - accuracy: 0.9133 - val\_loss: 1.4771 - val\_accuracy: 0.7100

Epoch 35/40

10/10 [==============================] - 13s 1s/step - loss: 0.2769 - accuracy: 0.8800 - val\_loss: 1.2959 - val\_accuracy: 0.7600

Epoch 36/40

10/10 [==============================] - 13s 1s/step - loss: 0.3017 - accuracy: 0.8933 - val\_loss: 1.5801 - val\_accuracy: 0.6800

Epoch 37/40

10/10 [==============================] - 11s 1s/step - loss: 0.2958 - accuracy: 0.9033 - val\_loss: 1.0057 - val\_accuracy: 0.7400

Epoch 38/40

10/10 [==============================] - 11s 1s/step - loss: 0.1931 - accuracy: 0.9300 - val\_loss: 1.4956 - val\_accuracy: 0.6200

Epoch 39/40

10/10 [==============================] - 12s 1s/step - loss: 0.3035 - accuracy: 0.9133 - val\_loss: 0.9758 - val\_accuracy: 0.7400

Epoch 40/40

10/10 [==============================] - 11s 1s/step - loss: 0.1727 - accuracy: 0.9400 - val\_loss: 1.7555 - val\_accuracy: 0.6700

Từ kết quả mô hình huấn luyện, ta có được đồ thị độ chính xác theo số lần lọc:

Score=model.evaluate(training,verbose=0)

print('Train Loss', Score[0])

print('Train Accuracy', Score[1])

Train Loss 0.15415412187576294

Train Accuracy 0.9433333277702332

accuracy = history.history['accuracy']

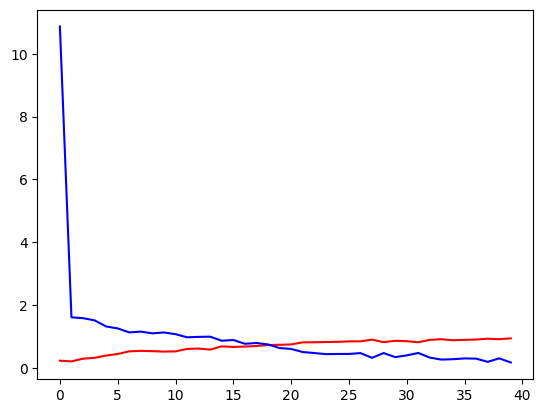
loss = history.history['loss']

epochs = range(len(accuracy))

plt.plot(epochs,accuracy,'r',label='Training accuracy')

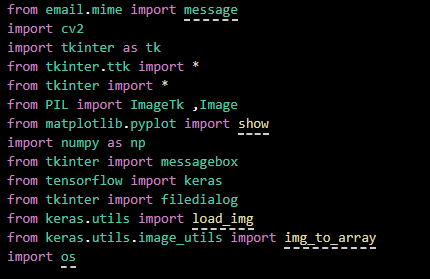
plt.plot(epochs,loss,'b',label='Training loss')

plt.show()



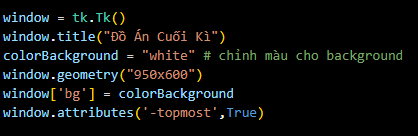
**3. Giải thích code giao diện**

**3.1 Khai báo các thư viện cần thiết**

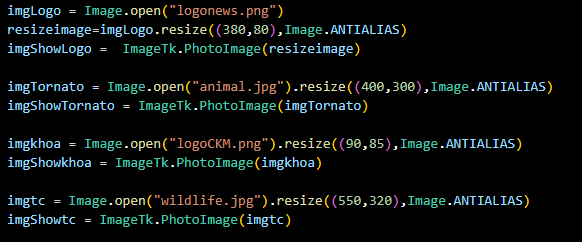
****

**3.2 Xây dựng giao diện**

- Tạo nên cữa sổ hiển thị giao diện

****

- Load các hình ảnh cần thiết để hiển thị

****

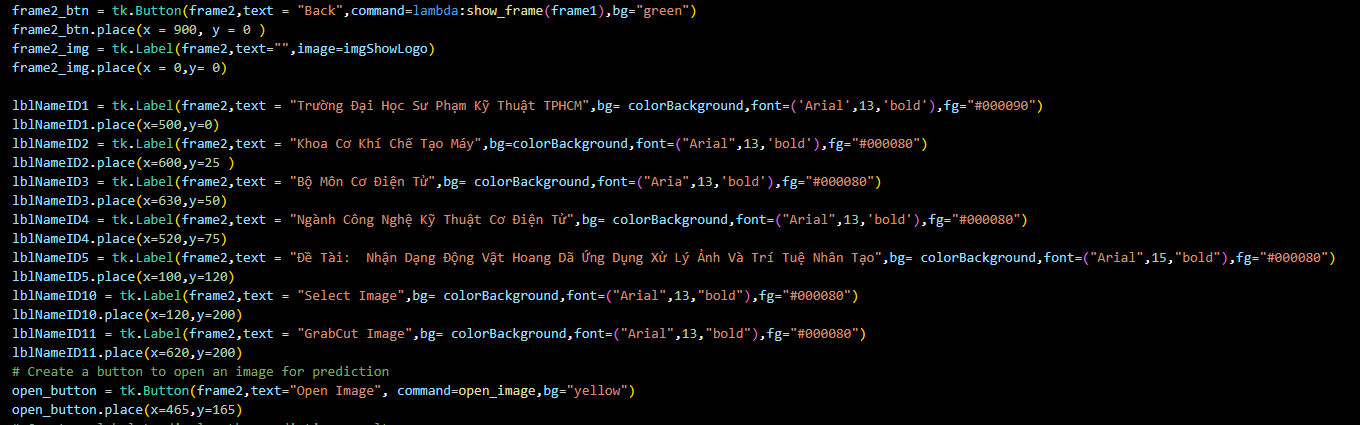
- Tạo các label trên cữa sổ giao diện khi chạy chương trình

****

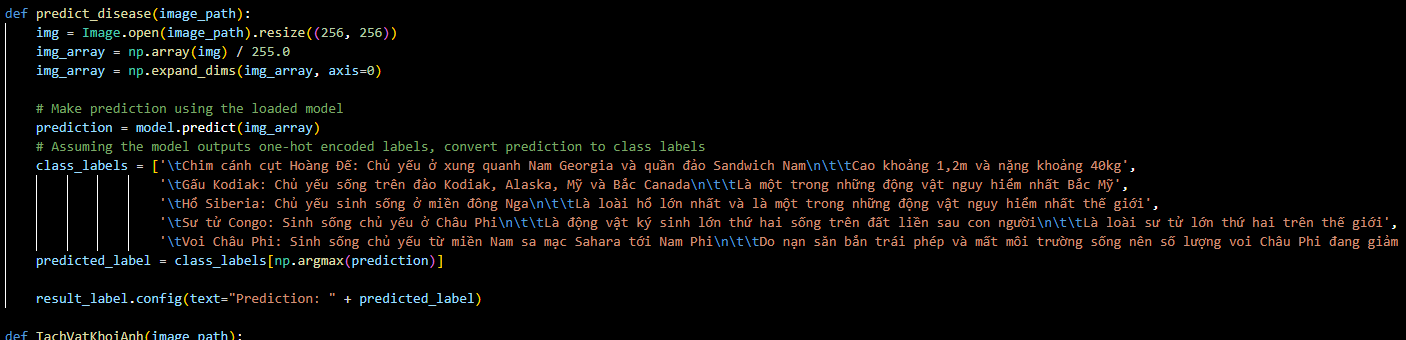
- Tạo các label hiển thị sau khi nhấn nút Login

****

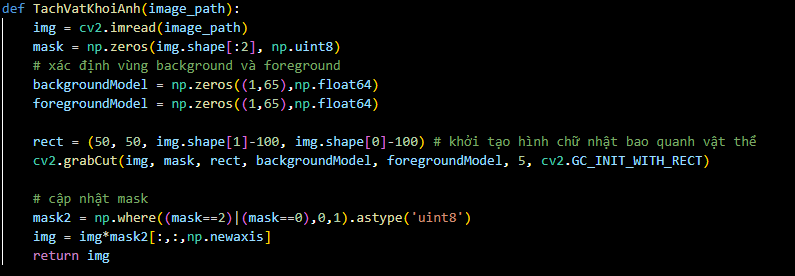
- Tạo các label để hiển thị kết quả sau khi nhấn Processing

****

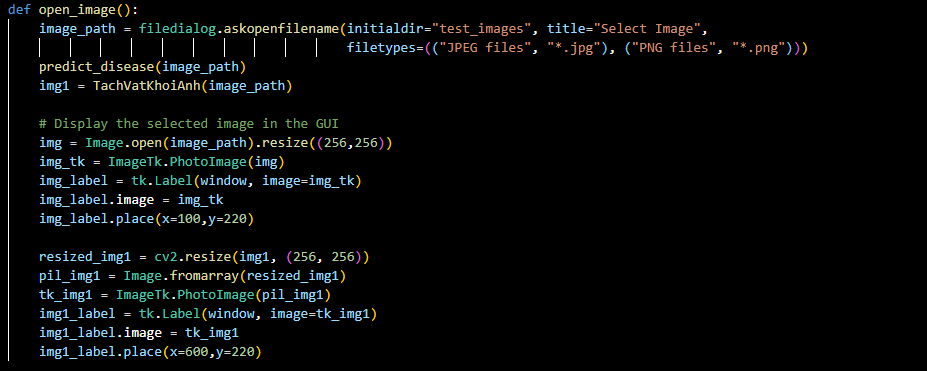
- Tiến hành nhận diện Image

****

- Xóa phong nền và hiển thị hình ảnh con vật sử dụng Grapcut

****

- Load hình ảnh để nhận dạng

****

**IV. KẾT QUẢ ĐẠT ĐƯỢC**

**1.Giao diện**

Bước 1: Sau khi chạy chương trình sẽ hiện ra giao diện đăng nhập và nhấn nút Login



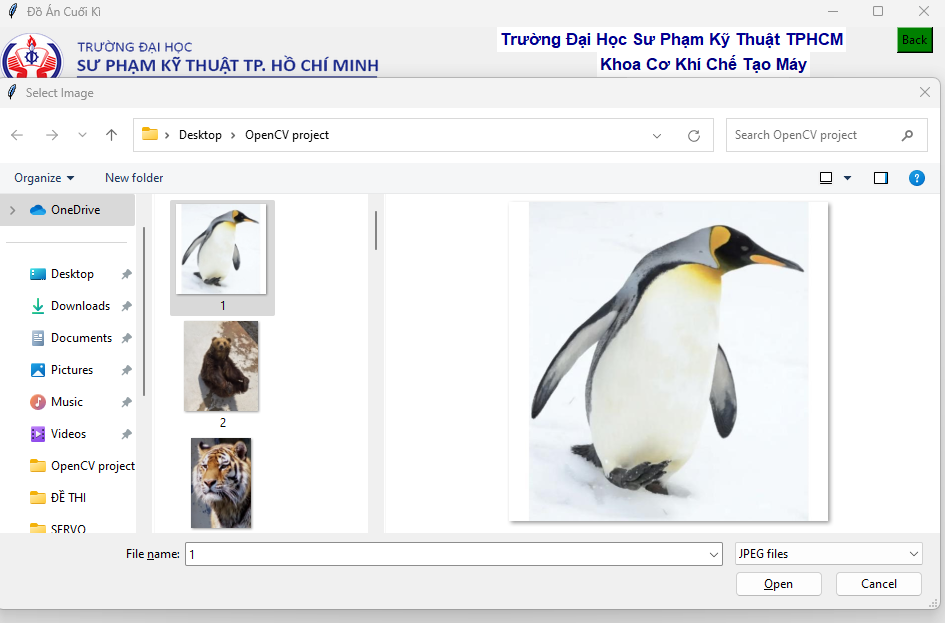
Bước 2: Nhấn Nút Processing để vào cữa sổ nhận dạng



Bước 3: Hiển thị của sổ để nhận dạng



Bước 4: Nhấn nút Open Image để load ảnh từ máy tính



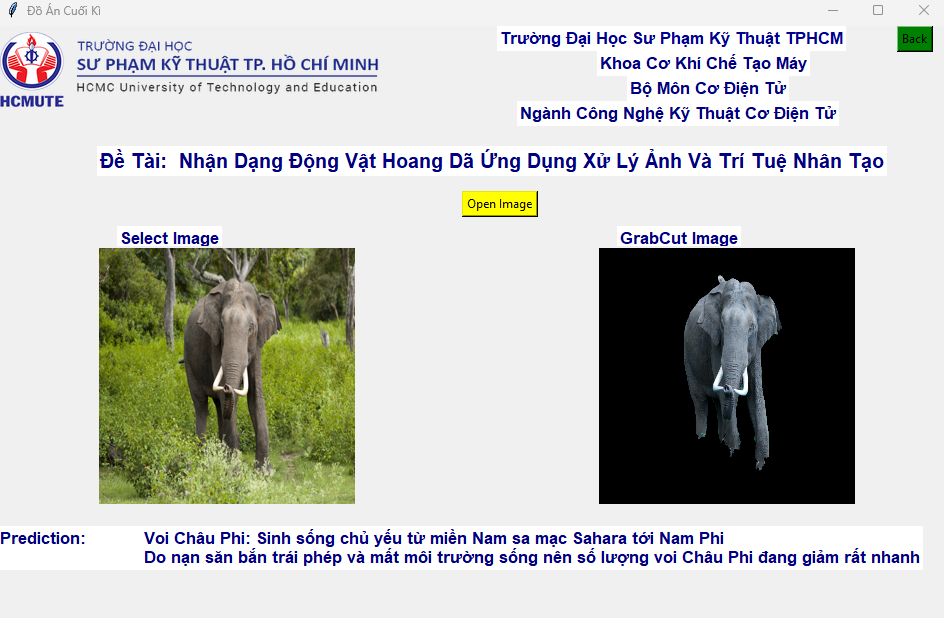
Bước 5: Hiển thị kết quả sau khi nhận dạng



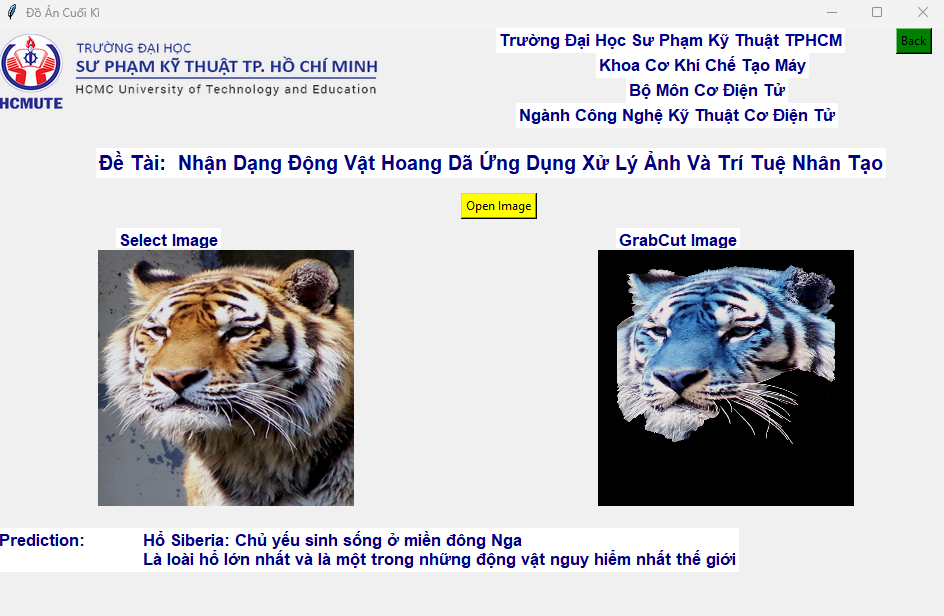
**2. Kết quả**

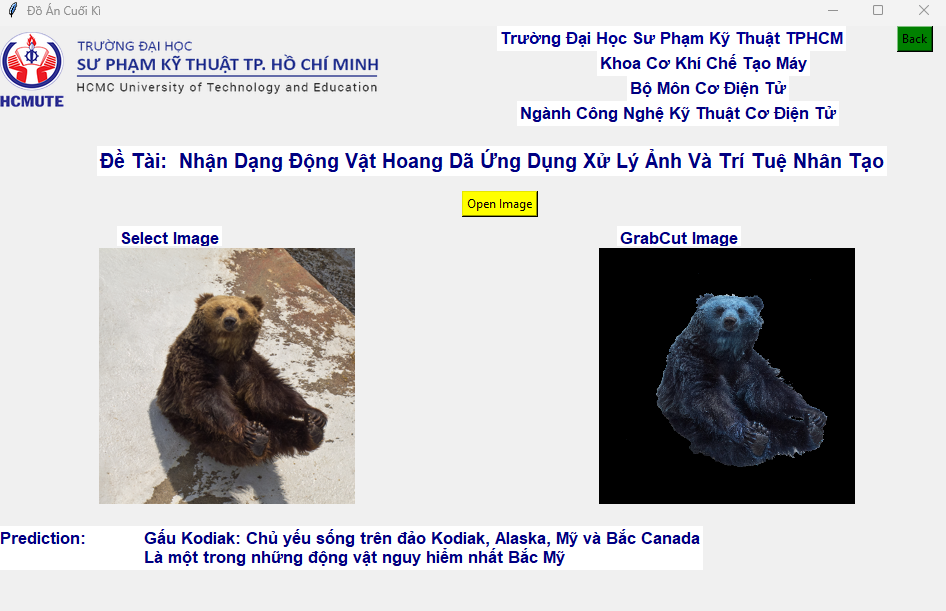
Dự đoán gần như chính xác các loài động vật hoang dã một cách nhanh chóng.











**Nhận xét:**

* Ưu Điểm: Khả năng nhận diện chính xác hình ảnh tương đối cao, mình cũng có thể ứng dụng nó vào trong các lĩnh vực khác như: Nhận dạng và phân loại rác thải, nhận dạng các loại trái cây hư…
* Nhược điểm: Nhưng đôi lúc cũng có một số ảnh nhận diện không chính xác dẫn đến đưa ra kết quả nhận dạng sai. Do đó, cần phải training cho nó nhiều dữ liệu học hơn thêm lượng ảnh đầu vào.

TÀI LIỆU THAM KHẢO

1. Chim cánh cụt Hoàng Đế, link truy cập:

<https://vi.wikipedia.org/wiki/C%C3%A1nh_c%E1%BB%A5t_ho%C3%A0ng_%C4%91%E1%BA%BF>

1. Gấu Kodiak, link truy cập:

<https://vi.wikipedia.org/wiki/G%E1%BA%A5u_Kodiak>

1. Hổ Siberia, link truy cập:

<https://vi.wikipedia.org/wiki/H%E1%BB%95_Siberia>

1. Sư tử Congo, link truy cập:

<https://vi.wikipedia.org/wiki/S%C6%B0_t%E1%BB%AD>

[5]. Voi Châu Phi, link truy cập:

<https://vi.wikipedia.org/wiki/Voi_%C4%91%E1%BB%93ng_c%E1%BB%8F_ch%C3%A2u_Phi>

[6]. TOPDev, Thuật toán CNN – Convolutional Neural Network, Link truy cập: <https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network>

[7] Thư viện Numpy

<https://viblo.asia/p/gioi-thieu-ve-numpy-mot-thu-vien-chu-yeu-phuc-vu-cho-khoa-hoc-may-tinh-cua-python-maGK7kz9Kj2>

[8] Thư viện OpenCV

<https://kienthucphanmem.com/tin-tuc/opencv-la-gi/>

[9] Thư viện Pilow

<https://viblo.asia/p/huong-dan-su-dung-thu-vien-pillow-de-xu-ly-hinh-anh-trong-python-cho-nguoi-moi-bat-dau-3Q75wm4MZWb>