BỘ GIÁO DỤC VÀ ĐÀO TẠO

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

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

BỘ MÔN CƠ ĐIỆN TỬ

BÁO CÁO CUỐI KỲ

TRÍ TUỆ NHÂN TẠO

GVHD: **PGS. TS. Nguyễn Trường Thịnh**

SVTH: **Lê Phan Văn Việt**

MSSV:**19146302**

Thủ Đức, thứ hai ngày 20 tháng 6 năm 2022

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Chữ ký giảng viên

Mục lục

Danh mục từ viết tắt

CNN - Convolutional Neural Networks

AI – Trí tuệ nhân tạo

Phụ lục hình ảnh

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1 Lý do chọn đề tài

Rắn là tên gọi chung để chỉ một nhóm các loài động vật bò sát ăn thịt, từng có chân và thân hình tròn dài (hình trụ), thuộc phân bộ Serpentes. Phần lớn các loài rắn không có nọc độc, còn những loài nào có nọc độc thì chủ yếu sử dụng nó vào việc giết chết hay khuất phục con mồi thay vì để phòng vệ. Một số loài có nọc độc mạnh tới mức đủ gây ra vết thương đau nhức hay gây tử vong cho con người. Các loài rắn không nọc độc hoặc là nuốt sống con mồi hoặc là giết nó bằng cách quấn và vặn xiết. (1)

Nọc độc của các loài rắn đa phần không ảnh hưởng đến con người, chỉ gây đau nhứt hoặc mẫn đỏ,.. Tuy nhiên nọc độc của rắn là nguyên nhân gây ra cái chết cho NNNN người hàng năm theo thống kê của WHO. Có nhiều nguyên nhân dẫn đến việc rắn cắn người bị thương, rắn cắn chết người, có thể là do tai nạn, do bản tính hung hãn của loài rắn đó nhưng đa phần là do chúng tự vệ khi bị con người tấn công. Ở Việt Nam, hầu hết chúng ta khi tháy rắn ở ngoài tự nhiên thì đa phần sẽ sỡ hãi và cố gắng đuổi chúng đi hoặc giết chúng để tránh bị cắn nhưng vô tình khiến chúng nổi điên hay đơn thuần là chỉ muốn tự vệ, đó là một phản xạ tự nhiên. Chỉ có một số loài rắn có bản tính hung hãn (rất ít) mới chủ động tấn công con người, còn lại đa phần chúng không cắn người nếu không bị tấn công. Bên cạnh đó việc săn bắt rắn để phục vụ các nhu cầu ăn uống, y học,… cũng là nguyên nhân dẫn đến các trường hợp đáng tiếc. Rượu ngâm "rắn" cũng là một trong những nguyên nhân gây ngộ độc hay tử vong do nọc độc.

Vậy vấn đề ở đây đặt ra là nguyên nhân nào là nguyên nhân cốt lõi ? Thật ra, nguyên nhân chính là do sự thiếu hiểu biết của chúng ta về các loài rắn. Vì không biết phân biệt đâu là rắn độc và đâu là rắn không độc mà chúng ta không xác định được cần phải làm gì khi tình cờ bắt gặp một con rắn, cũng vì vậy mà người ta mới bị ngộ độc khi ngâm rượu rắn có độc,…

Từ những nguyên nhân và vấn đề đã đề cập ở trên, sinh viên quyết định chọn đề tài : ‘ PHÂN LOẠI RẮN ĐỘC HAY KHÔNG ĐỘC ….. SỬ DỤNG APP ANDROID REAL TIME’ :

1.1 Mục tiêu nghiên cứu

Xây dựng được tập dữ liệu dùng để huấn luyện cho mô hình

Xây dựng mô hình phân loại rắn độc hay không độc và phân loại 8 loài rắn sử dụng CNN

Triên khai mô hình lên thiết bị di động

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu về trí tuệ nhân tạo (AI)

Trí Tuệ Nhân Tạo - AI (Artificial Intelligence) hoặc trí thông minh nhân tạo là công nghệ mô phỏng các quá trình suy nghĩ và học tập của con người cho máy móc, đặc biệt là hệ thống máy tính. Trí tuệ nhân tạo này do con người lập trình ra với mục đích tự động hóa các hành vi thông minh như con người, từ đó cắt giảm bớt nhân công là con người và có tính chuẩn xác cao hơn. Cụ thể, trí tuệ nhân tạo giúp máy tính có được những trí tuệ của con người như: biết suy nghĩ và lập luận để giải quyết vấn đề, biết giao tiếp do hiểu ngôn ngữ, tiếng nói, biết học và tự thích nghi, …

Theo dòng chảy của cuộc cách mạng 4.0, trí tuệ nhân tạo ngày càng được phổ biến và ứng dụng rộng rãi trong mọi lĩnh vực của cuộc sống, mặc dù được John McCarthy – nhà khoa học máy tính người Mỹ đề cập lần đầu tiên vào những năm 1950 nhưng đến ngày nay thuật ngữ trí tuệ nhân tạo mới thực sự được biết đến rộng rãi và được các “ông lớn” của làng công nghệ chạy đua phát triển.

AI là công nghệ sử dụng đến kỹ thuật số có khả năng thực hiện những nhiệm vụ mà bình thường phải cần tới trí thông minh của con người, được xem là phổ biến nhất. Đặc

trưng của công nghệ AI là năng lực "tự học" của máy tính, do đó có thể tự phán đoán, phân tích trước các dữ liệu mới mà không cần sự hỗ trợ của con người, đồng thời có khả năng xử lý dữ liệu với số lượng rất lớn và tốc độ cao. Hiện mỗi ngày trên toàn cầu có khoảng 2,2 tỷ Gb dữ liệu mới (tương đương 165.000 tỷ trang tài liệu) được tạo ra và được các công ty, như Google, Twitter, Facebook, Amazon, Baidu, Weibo, Tencent hay Alibaba thu thập để tạo thành "dữ liệu lớn" (big data). Trí tuệ nhân tạo là một lĩnh vực liên quan đến chuyên ngành khoa học máy tính và công nghệ thông tin, bản chất của trí tuệ nhân tạo vẫn do con người làm ra, họ xây dựng các thuật toán, lập trình bằng các công cụ phần mềm công nghệ thông tin, giúp các máy tính có thể tự động xử lý các hành vi thông minh như con người.

Trí tuệ nhân tạo có khả năng tự thích nghi, tự học và tự phát triển, tự đưa ra các lập luận để giải quyết vấn đề, có thể giao tiếp như người…tất cả là do AI được cài một cơ sở dữ liệu lớn, được lập trình trên cơ sở dữ liệu đó và tái lập trình trên cơ sở dữ liệu mới sinh ra. Cứ như vậy cấu trúc của AI luôn luôn thay đổi và thích nghi trong điều kiện và hoàn cảnh mới. Dự báo đến năm 2030 của công ty kiểm toán và tư vấn tài chính PwC, GDP toàn cầu có thể tăng trưởng thêm 14% từ sự hỗ trợ của trí tuệ nhân tạo, AI đã xuất hiện trong nhiều ngành, từ cung cấp dịch vụ mua sắm ảo và ngân hàng trực tuyến đến giảm chi phí đầu tư trong sản xuất và hợp lý hóa chẩn đoán trong chăm sóc sức khỏe. AI đã thúc đẩy hầu hết các ngành công nghiệp tiến lên và thay đổi cuộc sống của nhiều người.

Nhìn chung, đây là một ngành học rất rộng, bao gồm các yếu tố tâm lý học, khoa học máy tính và kỹ thuật. Một số ví dụ phổ biến về AI có thể kể đến ô tô tự lái, phần mềm dịch thuật tự động, trợ lý ảo trên điện thoại hay đối thủ ảo khi chơi trò chơi trên điện thoại.

2.2 Giới thiệu về Deep learning

Deep Learning là tập hợp con của Machine Learning và nó có tác dụng hỗ trợ cho máy tính tự huấn luyện chính nó để có thể thực hiện mọi tác vụ tương tự như con người. Điều này chính là giúp máy tính bắt chước con người cách học hỏi và suy nghĩ.

Học máy (machine learning, ML) là một tập con của trí tuệ nhân tạo. Machine learning là một lĩnh vực nhỏ trong khoa học máy tính, có khả năng tự học hỏi dựa trên dữ liệu được đưa vào mà không cần phải được lập trình cụ thể.

Những năm gần đây, sự phát triển của các hệ thống tính toán cùng lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn đã giúp machine learning tiến thêm một bước dài. Một lĩnh vực mới được ra đời được gọi là học sâu (deep learning, DL). Deep learning đã giúp máy tính thực thi những việc vào mười năm trước tưởng chừng là không thể: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết, giao tiếp với con người, chuyển đổi ngôn ngữ, hay thậm chí cả sáng tác văn thơ và âm nhạc.

Các hệ thống của Deep Learning có khả năng cải thiện được những hiệu suất của chúng với quyền truy cập vào dữ liệu sẽ được nhiều hơn. Deep Learning có hỗ trợ cho việc dịch ngôn ngữ, phân loại các hình ảnh, nhận dạng giọng nói. Chính vì thế, nó có thể được ứng dụng để giải quyết mọi nhu cầu cần nhận dạng mẫu mà không cần đến sự can thiệp của con người

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNN) là một trong những mô hình deep learning phổ biến nhất và có ảnh hưởng nhiều nhất trong cộng đồng Computer Vision. CNN được dùng trong trong nhiều bài toán như nhân dạng ảnh, phân tích video, ảnh MRI, hoặc cho bài các bài của lĩnh vự xử lý ngôn ngữ tự nhiên,và hầu hết đều giải quyết tốt các bài toán này.

CNN có lịch sử khá lâu đời. Kiến trúc gốc của mô hình CNN được giới thiệu bởi một nhà khoa học máy tính người Nhật vào năm 1980. Sau đó, năm 1998, Yan LeCun lần đầu huấn luyện mô hình CNN với thuật toán backpropagation cho bài toán nhận dạng chữ viết tay. Tuy nhiên, mãi đến năm 2012, khi một nhà khoa học máy tính người Ukraine Alex Krizhevsky (đệ của Geoffrey Hinton) xây dựng mô hình CNN (AlexNet) và sử dụng GPU để tăng tốc quá trình huấn luyện deep nets để đạt được top 1 trong cuộc thi Computer

Vision thường niên ImageNet với độ lỗi phân lớp top 5 giảm hơn 10% so với những mô hình truyền thống trước đó, đã tạo nên làn sóng mãnh mẽ sử dụng deep CNN với sự hỗ trợ của GPU để giải quyết càng nhiều các vấn đề trong Computer Vision.

Cấu trúc của CNN: Mạng CNN là tập hợp nhiều lớp Convolutional chồng lên nhau, sử dụng các hàm Nonlinear Activation và tanh để kích hoạt các trọng số trong các node. Ở mỗi lớp CNN, sau khi được các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho những lớp tiếp theo. Mỗi Layer kết tiếp sẽ là kết quả Convolution từ Layer trước đó nên chúng ta có được các kết nối cục bộ.

Thông qua quá trình huấn luyện mạng, các lớp Layer CNN tự động học các giá trị được thể hiện qua các lớp Filter.

2.4 Các phương pháp đánh giá mô hình – Performance metrics

2.4.1 Learning curve

Dùng để đánh giá được kết quả quá tình học tập và kết quả học tập. Từ đó cho ta biết được mô hình có bị overfitting hay ungderfitting và quá trình mô hình được train như thế nào.

2.5 Confusion matrix

Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix.

Confusion matrix là một kỹ thuật đánh giá hiệu năng của mô hình cho các bài toán phân lớp. Confusion matrix là một ma trận thể hiện số lượng điểm dữ liệu thuộc vào một class và được dự đoán thuộc vào class.

2.6 Positive/Negative

Nhược điểm của Accuracy là chỉ cho ta biết độ chính xác khi dự báo của mô hình, nhưng không thể hiện mô hình đang dự đoán sai như thế nào, vì vậy chúng ta cần một

phương pháp đánh giá khác – Confusion Matrix. Confusion matrix là một kỹ thuật đánh giá hiệu năng của mô hình cho các bài toán phân lớp. Confusion matrix là một ma trận thể hiện số lượng điểm dữ liệu thuộc vào một class và được dự đoán thuộc vào class. Trong những bài toán này, người ta thường định nghĩa lớp dữ liệu quan trọng hơn cần được xác định đúng là lớp Positive (P-dương tính), lớp còn lại được gọi là Negative (N-âm tính). Ta định nghĩa True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) dựa trên confusion matrix.

- True Positive (TP): đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Positive (dự đoán đúng).
- True Negative (TN): đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Negative (dự đoán đúng).
- False Positive (FP): đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Positive (dự đoán sai).
- False Negative (FN): đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Negative (dự đoán sai).

2.7 Percision

Precision trả lời cho câu hỏi trong các trường hợp được dự báo là positive thì có bao nhiêu trường hợp là đúng. Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive.

Công thức tính Precision:

$$\text{Precision} = \frac{\text{TP}}{\text{total predicted positive}} = \frac{\text{TP}}{\text{TP+FP}}$$

2.8 Recall

Recall giúp chúng ta biết được có bao nhiêu điểm dữ liệu thực sự ở lớp Positive được mô hình phân lớp đúng trong mọi điểm dữ liệu thực sự ở lớp Positive. Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive.

Công thức tính Recall:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

2.9 F1-score

F1-Score là trung bình điều hòa giữa precision và recall. Do đó nó đại diện hơn trong việc đánh giá độ chính xác trên đồng thời precision và recall. Một mô hình tốt khi cả Precision và Recall đều cao, thể hiện cho mô hình ít phân loại nhầm giữa các lớp cũng như tỉ lệ bỏ sót các đối tượng thuộc lớp cần quan tâm là thấp. Tuy nhiên, hai giá trị Precision và Recall thường không cân bằng với nhau (giá trị này tăng thì giá trị kia thường có xu hướng giảm). Để đánh giá cùng lúc cả Precision và Recall, ta sử dụng F1-Score.

Công thức F1-score:

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

2.10 Các thư viện sử dụng trong đề tài

2.10.1 Thư viện Scikit-learn

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập. Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

2.10.2 Thư viện Tensorflow

Tensorflow là một thư viện mã nguồn mở cung cấp khả năng xử lí tính toán số học dựa trên biểu đồ mô tả sự thay đổi của dữ liệu, trong đó các node là các phép tính toán học còn các cạnh biểu thị luồng dữ liệu.

Một chương trình Tensorflow được chia thành hai phần chính. Phần thứ nhất là xây dựng mô hình tính toán (được gọi là construction phase), phần thứ hai là chạy mô hình vừa mới xây dựng (được gọi là execution phase).

2.10.3 Thư viện Keras

Keras là một open source cho Neural Network được viết bởi ngôn ngữ Python. Nó là một library được phát triển vào năm 2005 bởi Francois Chollet, là một kỹ sư nghiên cứu Deep Learning. Keras là một thư viện tương đối dễ sử dụng đối với người mới bắt đầu. Nó cung cấp các hàm số cần thiết với cú pháp đơn giản.

Một số ưu điểm của Keras:

- Dễ sử dụng, xây dựng model nhanh.
- Run được trên cả CPU và GPU.
- Hỗ trợ xây dựng CNN , RNN hoặc cả hai.

2.10.4 Thư viện OpenCV (CV2)

OpenCV là một thư viện mã nguồn mở hàng đầu cho lĩnh vực thị giác máy tính (Computer Vision), lĩnh vực học máy (Machine learning), xử lý ảnh và các tính năng tăng tốc GPU trong hoạt động thời gian thực.

OpenCV có thể được sử dụng ở hầu hết các ngôn ngữ như C/C++, Python, Java… và các hệ điều hành khác nhau như Windows, Linux, OS, Android, iOS… Ngoài ra có thể sử dụng card đồ họa GPU để xử lý nhằm tăng tốc độ xử lý.

OpenCV là dự án bắt đầu bởi hãng Intel vào năm 1999 bởi Gary Bradsky và ra mắt lần đầu tiên vào năm 2000.

Các ứng dụng của OpenCV

OpenCV được sử dụng trong các ứng dụng:

- Hình ảnh Street Views.
- Kiểm tra và giám sát tự động Robot và xe tự động.
- Phân tích hình ảnh y tế.

- Tìm kiếm và phục hồi hình ảnh/ video phim – cấu trúc 3D từ chuyển động.
- Nghệ thuật sắp đặt tương tác.

2.11 Real-time

TensorFlow Lite là phiên bản rút gọn của TensorFlow. Đây chính là phiên bản dùng để hỗ trợ triển khai mô hình học máy trên ứng dụng di động. Đặc điểm của TensorFlow Lite

- Nhanh hơn, do nó cho phép thực hiện machine learning ngay trên device với độ trễ thấp.
- Tốn ít dung lượng nên khá tốt cho mobile. TensorFlow Lite cũng hỗ trợ cảm biến gia tốc của thiết bị android với Android Neural Networks API

CHƯƠNG 4. NỘI DUNG ĐỀ TÀI

4.1 Mô tả đề tài

Hiện nay có một số loài rắn sống gần môi trường sống của con người đặc biệt là vùng quê. Việc không biết rắn độc hay không dễ khiến con người gặp nguy hiểm đến tính mạng khi tiếp với rắn độc. Trong đề tài nghiên cứu này, em đã phân loại rắn độc hay không độc bằng mô hình sử dụng thuật toán CNN (Mạng nơ-ron tích chập). Và cũng trong đề tài này em cũng phân loại 8 loại rắn. Cung cấp thông tin về loại rắn đó mức độ độc tính và giúp bác sĩ có thể kịp thời tìm ra huyết thanh cứu người. Để triển khai mô hình này, em đã sử dụng cơ sở dữ liệu được xây dựng bởi trình thu thập ảnh từ tổ hợp một số nguồn khác nhau với 9200 và 1800 hình ảnh tương ứng cho mỗi mô hình làm dữ liệu huấn luyện. Công việc bao gồm làm giàu dữ liệu, tiêu chuẩn hoá kích thước hình ảnh. Sau đó, mô hình được đào tạo và sau đó được kiểm tra để xác minh độ tin cậy của mô hình. Độ chính xác của mô hình rắn độc hay không độc là 64% và độ chính xác của mô hình còn lại là 60%. Để mô hình có tính linh động cao và thuận lợi thì mô hình được triển khai trên thiết bị di động có thể cho kết quả từ hình chụp hoặc ở chế độ realtime trên camera.

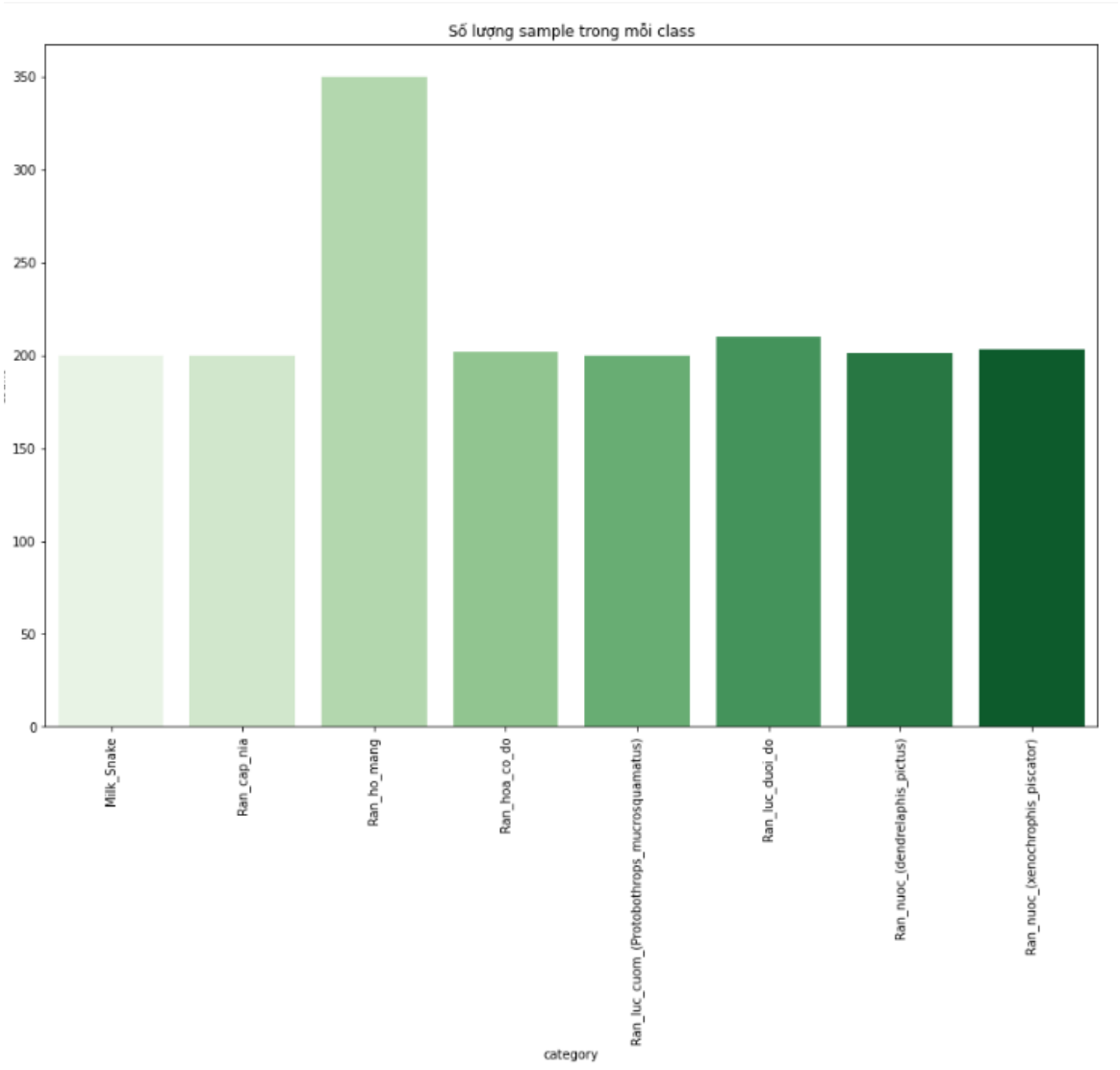
4.2 Giới thiệu về datasets

4.2.1 Dataset Phân loại rắn độc và không độc

Tập dữ liệu dùng để huấn luyện hai mô hình được em tự thu thập chủ yêu từ google image và một số trang web khác. Dữ liệu được gộp . Tập dữ liệu bao gồm 9200 chia đều cho 2 lớp: rắn độc và rắn không độc. Tất cả dữ liệu hình ảnh tỷ lệ 1:1 và độ phân giải lớn nhất là 1417 pixel, thấp nhất là 130 pixel.

Show số lượng hình của mỗi lớp

```
plt.figure(figsize=(8, 4))
sns.countplot(x="category", data=df, palette='Greens')
plt.xticks(rotation=90)
plt.title('Số lượng sample trong mỗi class')
plt.show()
```



Hình 4.2.1 Biểu đồ số lượng sample trong mỗi lớp

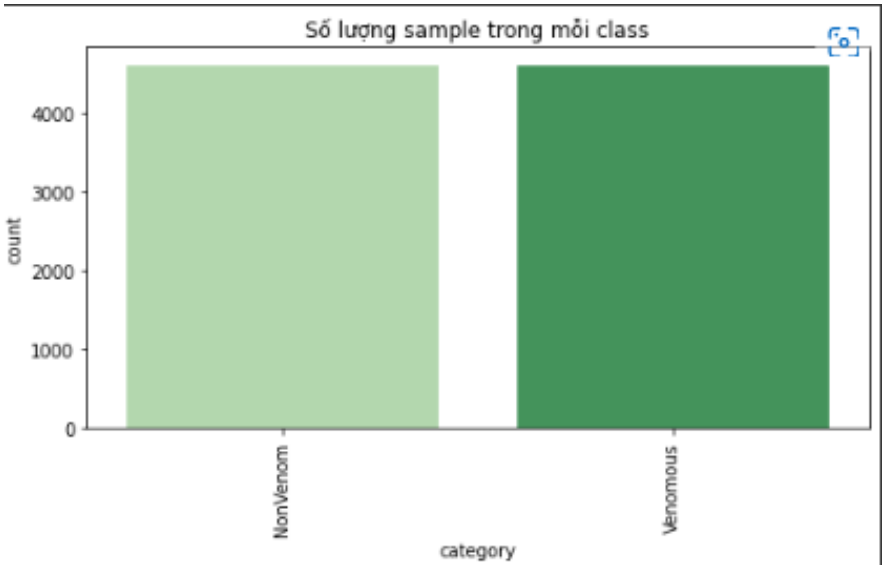
4.2.2 Dataset Phân loại 8 loài rắn

Tập dữ liệu bao gồm 8 loại rắn: Milk snake, rắn cạp nia, rắn hổ mang, rắn lục đuôi đỏ, rắn cỏ hoa cổ đỏ, rắn lục cườm, rắn nước (Dendrelaphis pictus) và rắn nước (xenochrophis piscator). Tập dữ liệu dùng để huấn luyện mô hình được em tự thu thập chủ yêu từ google image và một số trang web khác. Tập dữ liệu bao gồm gần 1800 Các lớp có số lượng dữ liệu nằm ở khoảng 200 tấm. Số lượng dữ liệu nhiều nhất là Rắn hổ mang với

350 tấm. Tất cả dữ liệu hình ảnh tỷ lệ 1:1 và độ phân giải lớn nhất là 1417 pixel, thấp nhất là 130 pixel

Show số lượng hình của mỗi lớp

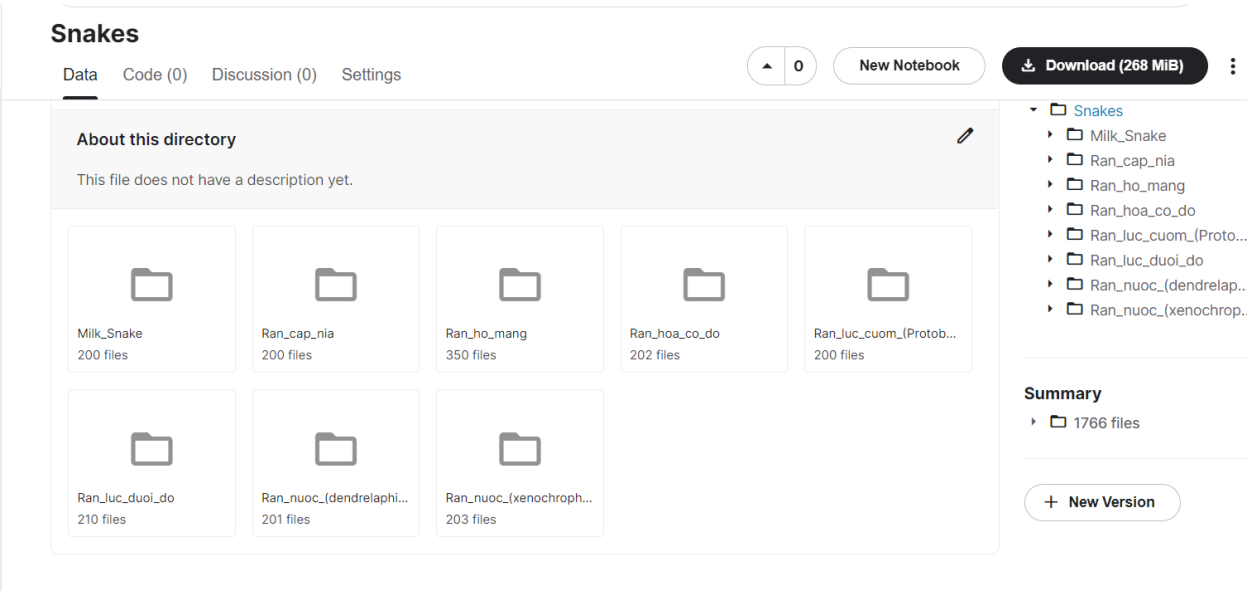
```
plt.figure(figsize=(8, 4))
sns.countplot(x="category", data=df, palette='Greens')
plt.xticks(rotation=90)
plt.title('Số lượng sample trong mỗi class')
plt.show()
```



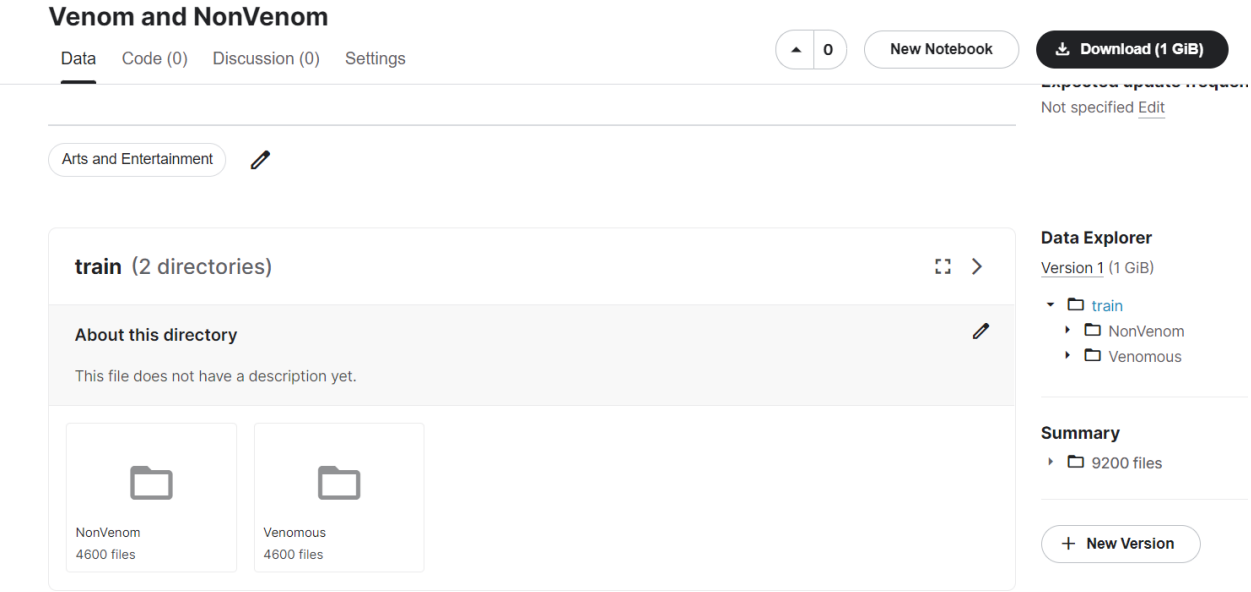
Hình 4.2.2 Biểu đồ số lượng sample trong mỗi lớp

4.2.3 Nơi lưu trữ dataset

Hiện tại toàn bộ dataset của 2 mô hình phân loại rắn độc hay không độc và phân loại 8 loài rắn được lưu trên Kaggle để thuận tiện tải vào Colab huấn luyện



Hình 4.2.3 Kaggle phân loại rắn độc hay không độc



Hình 4.2.4 Kaggle phân loại 8 loài rắn

Tải dữ liệu từ KAGGLE bằng cách lấy API token để tạo đường dẫn sau đó tải data về thông qua API command rồi giải nén vào thư mục đã tạo sẵn

```
import os
import json
kaggleAPIToken = {"username":"lephanvanviet","key":"5ce69bf8b6c7991dd0cdf4c0731e2f39"}
with open('/content/kaggle/kaggle.json', 'w') as file:
```

```
    json.dump(kaggleAPIToken, file)
!chmod 600 /content/kaggle/kaggle.json
!sudo mkdir ~/.kaggle
!cp /content/kaggle/kaggle.json ~/.kaggle/kaggle.json
```

Chỉ tới thư mục muốn tải dataset với mô hình phân loại rắn độc hay không độc

```
!kaggle datasets download -d lephanvanviet/venom-and-nonvenom
```

Chỉ tới thư mục muốn tải dataset với mô hình phân loại 8 loài rắn

```
!kaggle datasets download -d lephanvanviet/snakes
```

4.3 Xử lý dataset

Đầu tiên càn chia tập dữ liệu ra làm 3 phần đầu tiên là tập train tập validation và tập test với tỷ lệ là 8:1:1

```
train_df, test_df = train_test_split(df, test_size=0.1, random_state=1)
train_df, val_df = train_test_split(train_df, test_size=0.1, random_state=
1)
```

Để mô hình có kết quả tốt hơn đầu tiên cần chuẩn hoá nó về một kích thước duy nhất ở mô hình phân loại rắn độc hay không độc 100x100 và mô hình phân loại 8 loại rắn là 50x50. Tăng cường dữ liệu là một kỹ thuật giúp làm dày tập dữ liệu lên từ đó giúp mô hình có thể học được thêm nhiều đặc trưng của từng lớp cho ra độ chính xác cao hơn. Kỹ thuật này vô cùng cần thiết với mô hình phân loại 8 loài rắn vì số lượng hình của mỗi lớp là khá thấp . Các tham số tăng cường dữ liệu

Hàm	Thông số
horizontal_flip	True
vertical_flip	True
Rescale	1. / 255
rotation_range	45
brightness_range	0.5 - 1.0
zoom_range	0.5 - 1.0
width_shift_range	0.2
height_shift_range	0.2

```
aug = ImageDataGenerator(horizontal_flip=True,
```

```
    vertical_flip=True,
    validation_split=0.1,
    rescale=1./255,
    rotation_range=45,
    brightness_range= [0.5, 1.0],
    zoom_range=[0.5,1.0],
    width_shift_range = 0.2,
    height_shift_range = 0.2)
```

Những tham số trên chỉ áp dụng cho tập train và tập validation. Còn tập test sẽ chỉ Rescale 1. /255

```
aug = ImageDataGenerator(rescale=1./255)
```

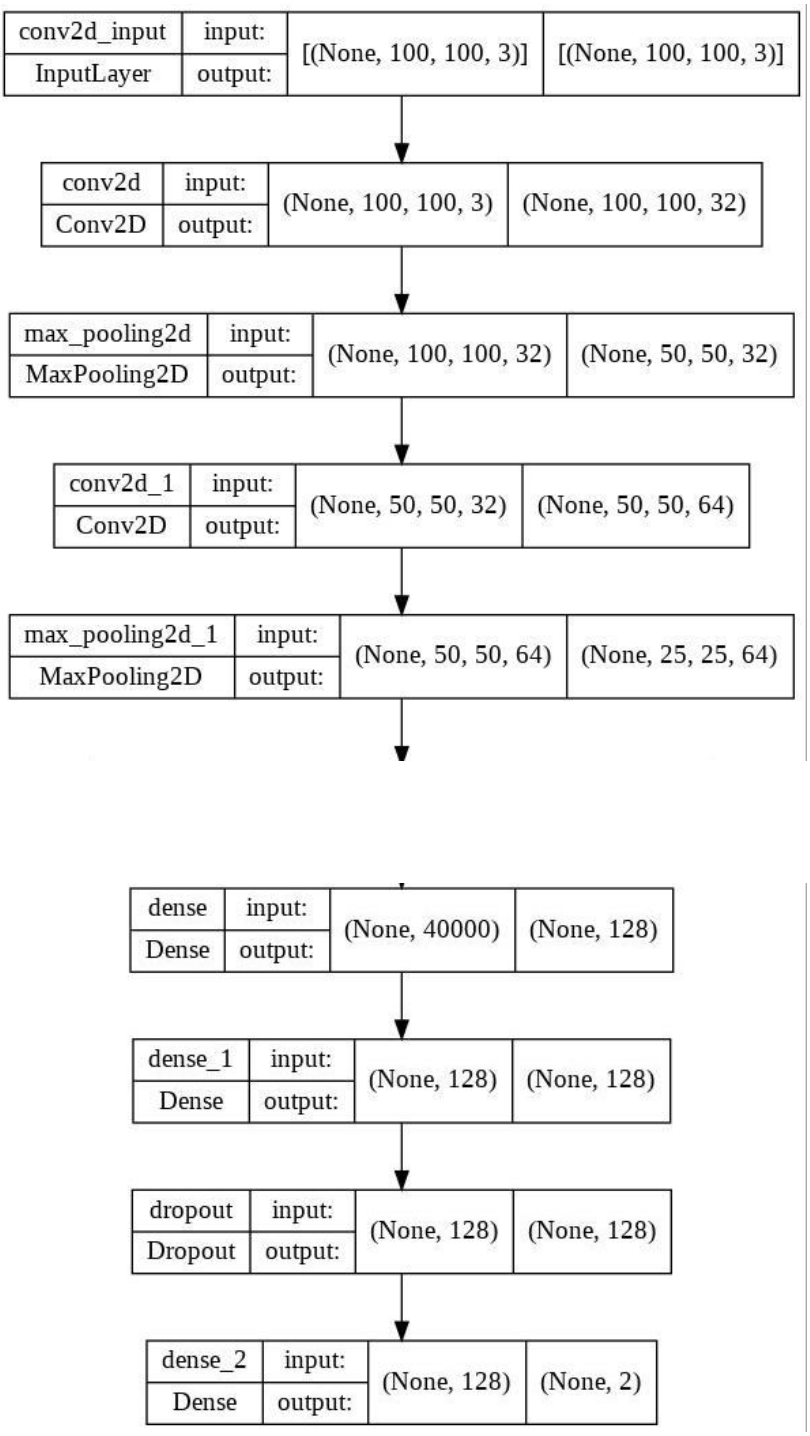
Khi ảnh được tải lên đưa vào để tăng cường dữ liệu sẽ được chuyển về dải màu RGB, và kích thước như được đề cập ở trên.

4.4 Xây dựng CNN và Huấn luyện mô hình

Mô hình Phân loại rắn độc hay không độc sử dụng CNN có thông số đầu vào là (100,100,3) và đâu ra là 2 lớp. Số lớp tích chập là 4 và số lớp đầy đủ là 3. Mỗi lớp tích chập có lớp tổng hợp riêng và lớp chuẩn hóa riêng có kích thước đồng nhất là 2x2 và bước là 2. Ngoài ra, tất cả lớp tích chập đều áp dụng hàm "Relu" để khử tuyến tính và có đều có kernel = 3x3. Khả năng mỗi hình ảnh nhận dạng chính xác cho mỗi mẫu được tính toán dựa trên bộ phân loại "sigmoid"

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uni
form', padding='same', input_shape=(100, 100, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uni
form', padding='same'))
model.add(MaxPooling2D((2, 2)))


model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.3))
```

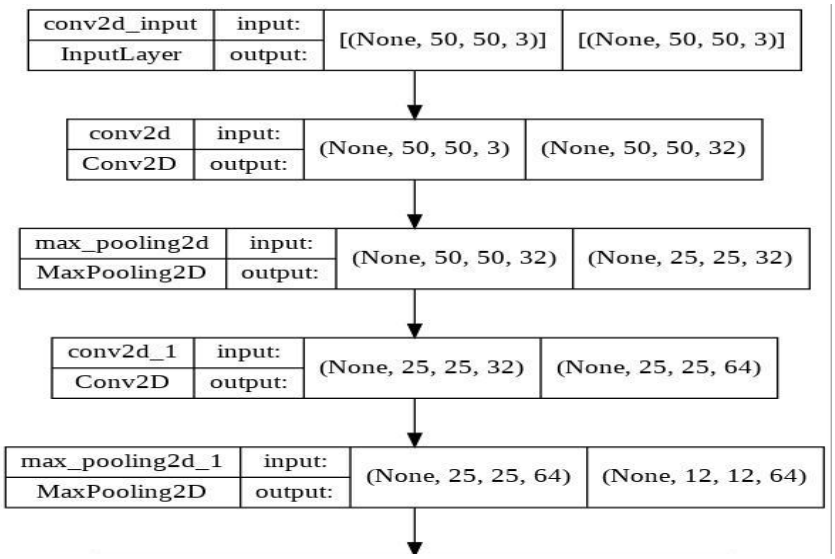
```
model.add(Dense(2,activation='sigmoid'))
```

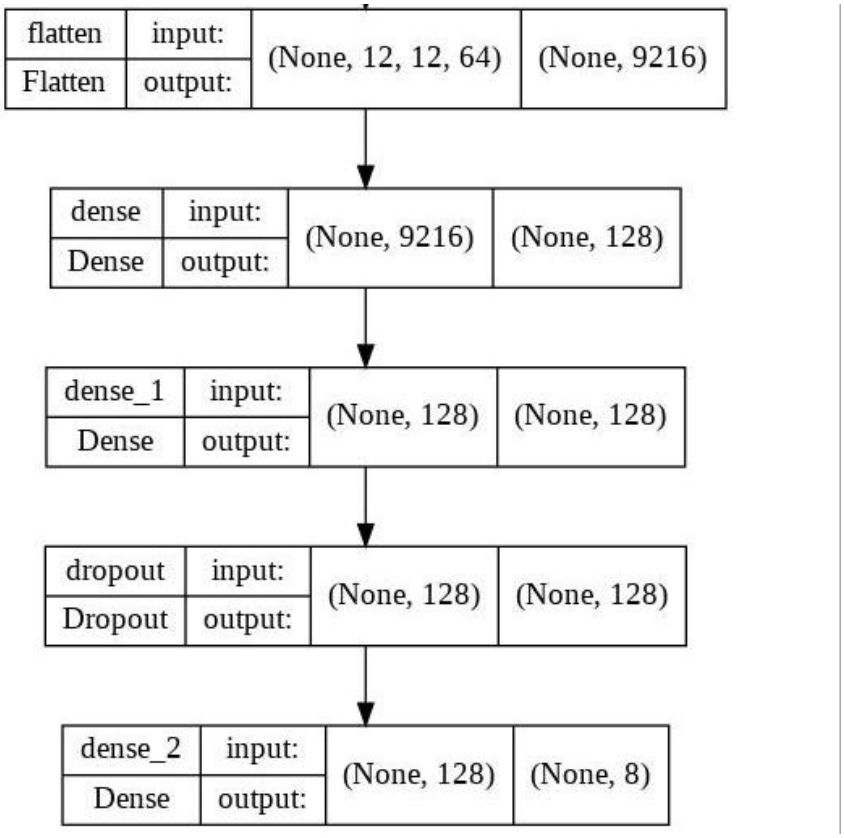


Hình 4.4.1 Lưu đồ mạng CNN mô hình phân loại rắn độc hay không độc

Mô hình Phân loại rắn độc hay không độc sử dụng CNN có thông số đầu vào là (50,50,3) và đâu ra là 8 lớp. Số lớp tích chập là 4 và số lớp đầy đủ là 3. Mỗi lớp tích chập có lớp tổng hợp riêng và lớp chuẩn hóa riêng có kích thước đồng nhất là 2x2 và bước là 2. Ngoài ra, tất cả lớp tích chập đều áp dụng hàm "Relu" để khử tuyến tính và có đều có kernel = 3x3. Khả năng mỗi hình ảnh nhận dạng chính xác cho mỗi mẫu được tính toán dựa trên bộ phân loại "softmax"

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uni
form', padding='same', input_shape=(50, 50, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uni
form', padding='same'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(8,activation='softmax'))
```





Hình 4.4.2 Lưu đồ mạng CNN mô hình phân loại 8 loại rắn

Cả hai mô hình đều dùng chọn thuật toán huấn luyện là adam, đánh sử dụng accurancy.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[
'accuracy'])
```

Sử dụng thuật toán earlyStopping để tìm ra điểm hội tụ tránh overfitting. Tham số để dừng earlyStopping là loss_validation. Thuật toán sẽ lưu file h5 có tham số loss_validation thấp nhất. Nếu sau 10 lần chạy mà thâm số loss_validation không thấp hơn nữa sẽ cho dừng huấn luyện mô hình

```
callbacks_list = [ModelCheckpoint(filepath='model.h5',monitor='val_loss',
save_best_only=True, verbose=1),EarlyStopping(monitor='val_loss', patience
=10,verbose=1)]
```

Mô hình được huấn luyện với tập dữ liệu test và validation sau khi được tăng cường và chuẩn hoá. Mô hình sẽ được huận luyện qua tất cả dữ liệu 100 lần. Nhưng ở đây sẽ có thể không chạy hết 100 lần mà nó có thể dừng khi earlyStopping được kích hoạt.

4.5 Đánh giá mô hình

Đánh gía mô hình với độ chính xác là "Accurancy" của mô hình với tập Validation. Mô hình phân loại rắn độc hay không độc cho độ chính xác là 64% và mô hình phân loại 8 loài rắn là 60%.

```
2/2 [==============================] - 1s 196ms/step - loss: 1.1937 - accuracy: 0.6013
Test loss: 1.1937
Test accuracy 0.6013
```

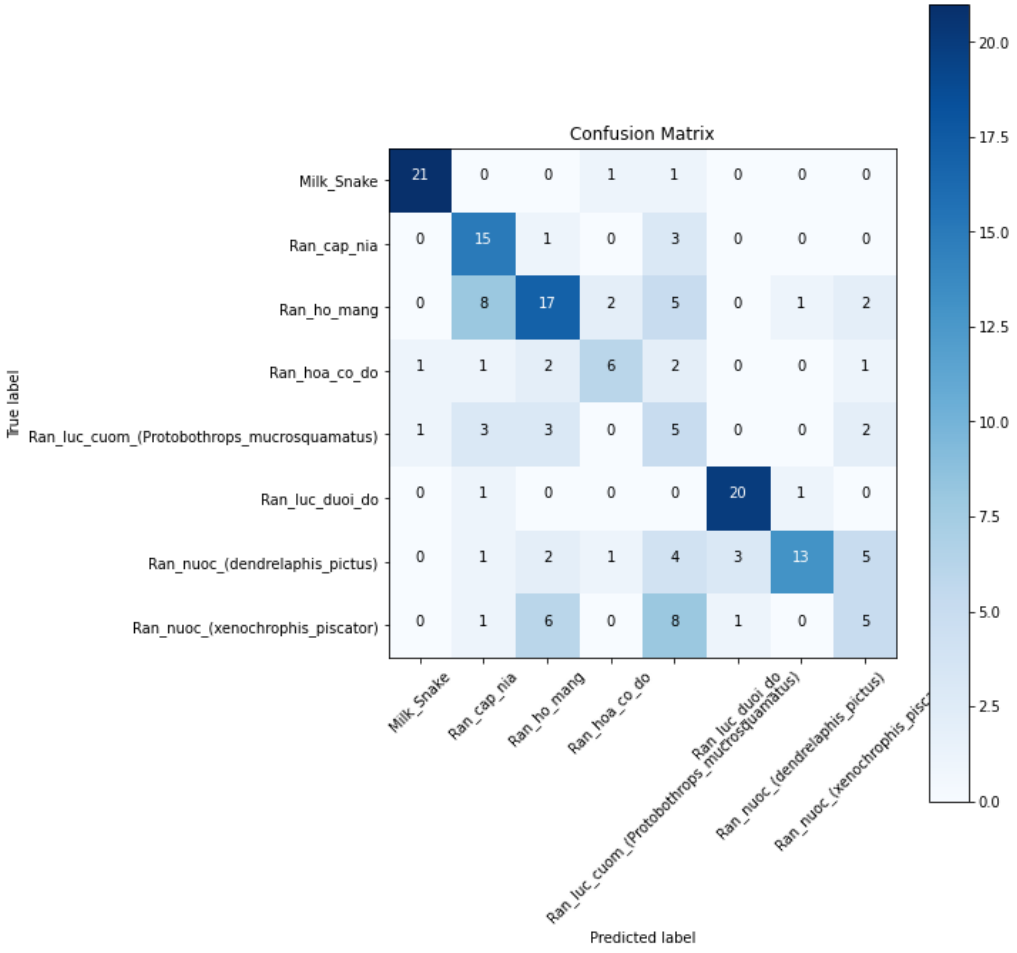
Hình 4.5.1 Kết quả accurancy của mô hình phân loại rắn độc hay không độc

```
7/7 [==============================] - 8s 1s/step - loss: 0.6203 - accuracy: 0.6365
Test loss: 0.6203
Test accuracy 0.6365
```

Hình 4.5.2 Kết quả accurancy của mô hình phân loại 8 loài rắn

Vẽ đồ thì learning curve. Thông qua đồ thị ta thấy được accurancy_trainning và val_accurancy luôn bám sát nhau ở cả 2 mô hình mô hình không có hiện tượng overfitting

Ma trận nhầm lẫn ở mô hình phân loại rắn độc hay không độc thì số lượng ảnh nhận nhầm của 3 loài là rắn hoa cổ đỏ, rắn lục cườm và rắn nước Xenochrophis là khá cao. Còn ở rắn Milk snake và rắn lục đuôi đỏ thì tỷ lệ nhận nhầm thấp.



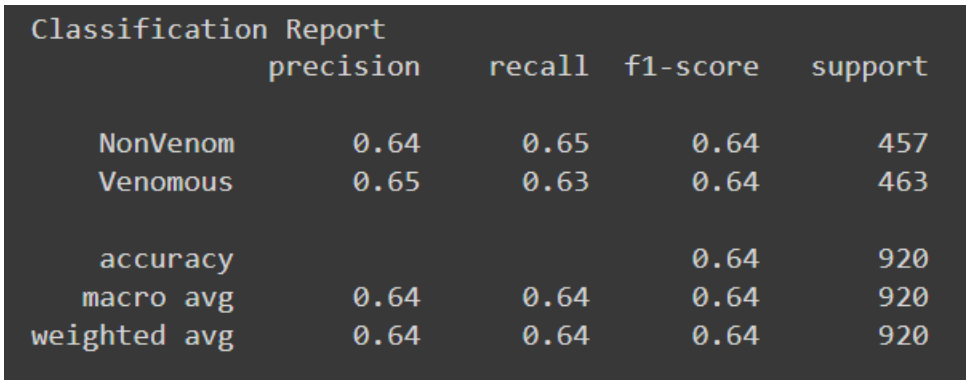
Hình 4.5.3 ma trận nhầm lẫn mô hình phân loại 8 loài rắn

Precision, recall, f1 score của mô hình

Classification Report

	precision	recall	f1-score	support
Milk_Snake	0.91	0.91	0.91	23
Ran_cap_nia	0.50	0.79	0.61	19
Ran_ho_mang	0.55	0.49	0.52	35
Ran_hoa_co_do	0.60	0.46	0.52	13
Ran_luc_cuom_(Protobothrops_mucrosquamatus)	0.18	0.36	0.24	14
Ran_luc_duoi_do	0.83	0.91	0.87	22
Ran_nuoc_(dendrelaphis_pictus)	0.87	0.45	0.59	29
Ran_nuoc_(xenochrophis_piscator)	0.33	0.24	0.28	21
accuracy			0.58	176
macro avg	0.60	0.58	0.57	176
weighted avg	0.63	0.58	0.58	176

Hình 4.5.4 Bảng kết quả F1 score của mô hình phân loại rắn độc hay không độc



Classification Report

	precision	recall	f1-score	support
NonVenom	0.64	0.65	0.64	457
Venomous	0.65	0.63	0.64	463
accuracy			0.64	920
macro avg	0.64	0.64	0.64	920
weighted avg	0.64	0.64	0.64	920

Hình 4.5.5 Bảng kết quả F1 score của mô hình phân loại 8 loài rắn

4.6 Triển khai mô hình lên mobile APP

Để triển khai mô hình sử dụng thuật toán CNN cần chuyển đổi file h5 sang file tflite để có thể sử dụng kết quả huấn luyện mô hình trên TensorFlow Lite. Sau khi chuyển dổi file xong thì nhúng file đó vào trong Android Studio. Ở đây vì cả hai đều thuộc sở hữu của Google nên trên Android Studio có hỗ trợ trực tiếp việc nhúng file tflite vào ứng dụng.

Khi chuyển đổi sang file tflite cần chú ý đến dung lượng file vì Android Studio chỉ hỗ trợ file dưới 200Mb. Khi file lớn hơn thì cần sử hàm `tf.lite.Optimize.DEFAULT` để giảm dung lượng file.

```
from keras.models import load_model
import tensorflow as tf

model = load_model("model.h5")
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
with open('modelClassi.tflite','wb') as f:
  f.write(tflite_model)
```

File tflite khi được đưa lên Android Studio

Kotlin Java

```
try {
    ModelVenom model = ModelVenom.newInstance(context);

    // Creates inputs for reference.
    TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 200, 200, 3}, DataType.FLOAT32);
    inputFeature0.loadBuffer(byteBuffer);

    // Runs model inference and gets result.
    ModelVenom.Outputs outputs = model.process(inputFeature0);
    TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

    // Releases model resources if no longer used.
    model.close();
} catch (IOException e) {
    // TODO Handle the exception
}
```

Ứng dụng điện thoại sau khi được nhúng file tflite có thể phân loại rắn độc hay không độc và phân loại 8 loại rắn trực tiếp thông qua camera hoặc có thể chụp hình lấy ảnh phân loại. Trước khi tiến hành phân loại thì ảnh cần được xử lý chuyển về dạng Bitmap. Ở dạng Bitmap cho phép chuẩn hoá hình ảnh để phù hợp với mô hình (Chuyển đổi kích thước ảnh, chuyển đổi dải màu RGB)

```
ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * size * size * 3);
byteBuffer.order(ByteOrder.nativeOrder());

int[] intValues = new int[size * size];
img.getPixels(intValues, offset: 0, img.getWidth(), x: 0, y: 0, img.getWidth(), img.getHeight());
int pixel = 0;
//iterate over each pixel and extract R, G, and B values. Add those values individually to the byte buffer.
for(int i = 0; i < size; i ++){
    for(int j = 0; j < size; j++){
        int val = intValues[pixel++]; // RGB
        byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f / 255));
        byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f / 255));
        byteBuffer.putFloat((val & 0xFF) * (1.f / 255));
    }
}
```

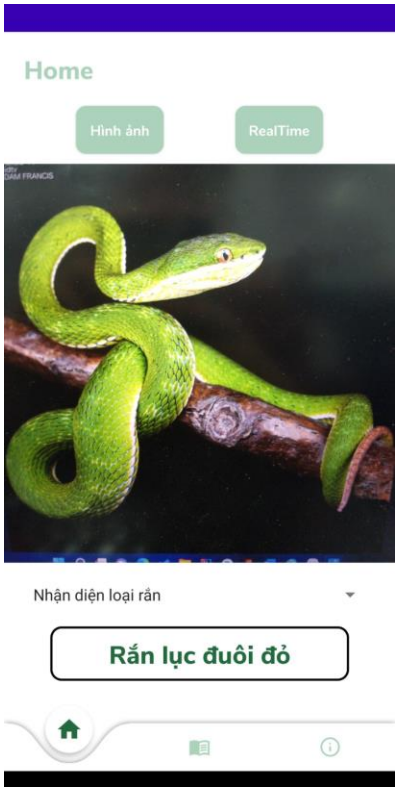
Ứng dụng được xây dựng

Việc nhận diện realTime ở ứng dụng điện thoại thông qua thư viện cameraX sẽ lấy được preview camera. Nhận được file ảnh ImageProxy. Để có thể xử lý ảnh đầu vào ta còn chuyển đổi file ảnh trên thành dạng BitMap

```
private Bitmap toBitmap(Image imgage) {
    Image.Plane[] planes = imgage.getPlanes();
    ByteBuffer yBuffer = planes[0].getBuffer();
    ByteBuffer uBuffer = planes[1].getBuffer();
    ByteBuffer vBuffer = planes[2].getBuffer();

    int ySize = yBuffer.remaining();
    int uSize = uBuffer.remaining();
    int vSize = vBuffer.remaining();

    byte[] nv21 = new byte[ySize + uSize + vSize];
    //U and V are swapped
    yBuffer.get(nv21,  offset: 0, ySize);
    vBuffer.get(nv21, ySize, vSize);
    uBuffer.get(nv21,  offset: ySize + vSize, uSize);

    YuvImage yuvImage = new YuvImage(nv21, ImageFormat.NV21, imgage.getWidth(), imgage.getHeight(),  strides: null);
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    yuvImage.compressToJpeg(new Rect( left: 0,  top: 0, yuvImage.getWidth(), yuvImage.getHeight()),  quality: 100, out);

    byte[] imageBytes = out.toByteArray();
    return BitmapFactory.decodeByteArray(imageBytes,  offset: 0, imageBytes.length);
}
```

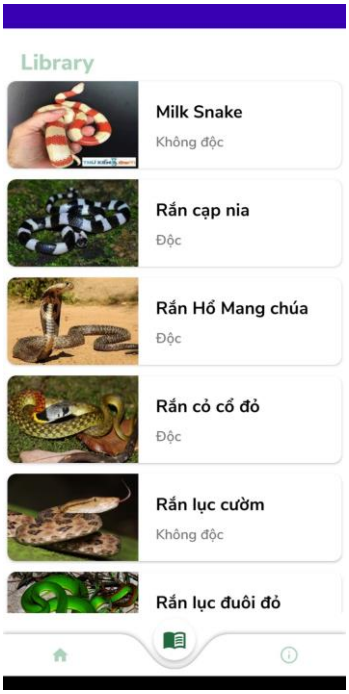
Giao diện ứng dụng điện thoại gồm 3 phần đầu tiên là

Trang chủ dùng để chụp ảnh và chạy realtime cho kết quả phân loại



Hình 4.6.1 Trang chủ app android

Thư viện chứa thông tin của các loại rắn



Hình 4.6.2 Thư viện app android

Thông tin ứng dụng

Hình 4.6.3 Trang thông tin app android

4.7 Kết quả

Sau khi huấn luyện mô hình độ chính xác thu được ở mô hình phân loại rắn độc hay không độc là 64% và mô hình phân loại 8 loại rắn là 60%. Cả 2 mô hình đều không bị overfitting

Mô hình được triển khai trên ứng dụng điện thoại có thể phân loại dựa vào ảnh trực tiếp hoặc có thể phân loại trực tiếp trên camera.

4.8 Kết luận

Mô hình hoạt động tốt trên ứng dụng. Nhưng về độ chính xác cần cải thiện để có thể phân loại tốt hơn. Thông qua bảng F1 score để bổ sung thêm dữ liệu cho các lớp có phần trăm thấp và tập dữ liệu cần được xử lý mất cân đối. Thay đổi một vài thông số trong CNN để cải thiện.

Trong tương lai khi độ chính xác được cải thiện thì mô hình cần được huấn luyện để có thể phân loại ở khoảng cách xa hơn tránh nguy hiểm cho người sử dụng, và thêm mô hình phân loại mức độ độc tính.

Tài liệu tham khảo

1. “ Tổng quan về AI- Artifical intelligence”, Charlie, https://insights.magestore.com/posts/tong-quan-ai-artificial-intelligence
2. “Imbalanced Multiclass Datasets”, Tô Đức Thắng, https://viblo.asia/p/imbalanced-multiclass-datasets-Do754dmQ5M6
3. “ Convolutional Neural Network là gì vậy? phương pháp chọn tham số cho Convolutional Neural Network chuẩn chỉnh”, Hỏi đáp, https://hocdauthau.com/convolutional-neural-network-la-gi-cach-chon-tham-so-cho-convolutional-neural-network-chuan-chinh/
4. “Snakebite envenoming”, WHO, https://www.who.int
5. “Sử dụng TensorFlow Lite Library để nhận diện Object”, Lê Minh Quang https://viblo.asia/p/android-tensorflow-lite-machine-learning-example-924lJxXmKPM