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

TRÍ TUỆ NHÂN TẠO

NHẬN DIỆN HÌNH ẢNH VÀ TÊN CỦA MỘT SỐ LOÀI ĐỘNG VẬT BẰNG AI

GVHD: PGS. TS Nguyễn Trường Thịnh

MHP: ARIN337629_08

Họ và tên: Nguyễn Thanh Tùng

MSSV: 20146186

Năm học: Học kì 2 2023 – 2024

Thành phố Hồ Chí Minh, tháng 5 năm 2023

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN

1.1. Giới thiệu về vấn đề bảo vệ động vật hiện nay

Động vật hoang dã là một bộ phận không thể thiếu trong lớp sinh quyển mà con người chúng ta đang có mặt. Động vật hoang dã được ví như một tài nguyên quý giá thúc đẩy sự phát triển toàn diện của xã hội, là một mắc xích quan trọng cho chuỗi chuyển hóa sinh học đang diễn ra.

Sự tồn tại của thế giới động vật tác động không nhỏ đến sự cân bằng hệ sinh thái, duy trì môi trường sống trong lành cho con người. Trong khi đó, hiện nay trên thế giới đang đứng trước nguy cơ mất đi rất nhiều loài sinh vật quý do tác động chính từ con người. Đã có các quốc gia, tổ chức lên án và áp dụng những biện pháp nhằm bảo vệ động vật hoang dã và môi trường sống của chúng. Bài thuyết trình về bảo vệ động vật hoang dã sẽ giúp các bạn có cái nhìn cụ thể, toàn diện về ý nghĩa, vai trò của động vật hoang dã, nguy cơ mà chúng đang đối mặt đồng thời bài viết cũng đưa ra một số biện pháp đang được sử dụng để bảo vệ động vật hoang dã.

Các biện pháp như tăng cường phòng chống vi phạm, thực hiện chính sách pháp luật về bảo tồn đa dạng sinh học, bổ sung nguồn lực cho công tác bảo tồn, chống săn bắt, tiếp tục đẩy mạnh hợp tác quốc tế và sử dụng truyền thông làm công cụ thay đổi hành vi của cộng đồng xã hội. Việc bảo vệ các loài động vật hoang dã, đặc biệt là những loài đang gặp nguy cơ tuyệt chủng, là rất cấp thiết và cần được thực hiện kịp thời và hiệu quả.

1.2. Lý do chọn đề tài

Trên thế giới hiện nay có hơn 1556 loài đang tuyệt chủng và đứng trước nguy cơ biến mất khỏi danh sách động vật hoang dã. Nguyên nhân chủ yếu dẫn đến sự tuyệt chủng này là do săn bắt trái phép của con người. Sự bùng nổ dân số cũng như sự phát triển của các công cụ lao động đã khiến con người tàn phá thiên nhiên để mở rộng của sống. Rừng ngày càng thu hẹp, môi trường sống của động vật bị ô nhiễm…cũng là nguyên nhân hàng đầu.

Ở Việt Nam là một trong những nơi có sự đang dạng về động vật quý hiếm. Tuy nhiên sự đa dạng này đang dần bị phá hủy. Các loài động vật quý hiếm như chim, sếu, hươu, voi..bị săn bắt trái

phép cùng với diện tích rừng ngày một thu hẹp đã khiến cho rất nhiều loài đứng trước nguy cơ tuyệt chủng.

Để phát triển một công nghệ mà có thể nhận diện động vật nhằm quan sát và bảo vệ động vật tốt hơn thì việc nhận diện động vật thông qua hình ảnh là rất quan trọng trong các nghiên cứu sinh học và bảo tồn động vật hoang dã. Nếu chúng ta có thể nhận diện và định danh các loài động vật một cách chính xác, chúng ta có thể giúp ích cho việc bảo vệ chúng và duy trì môi trường sống của chúng.

Ngoài ra, việc nhận diện động vật cũng có thể giúp cho công tác giáo dục về động vật và tăng cường sự quan tâm của mọi người đối với việc bảo vệ môi trường. Người ta sẽ dễ dàng hình dung và hiểu rõ hơn về các loài động vật khi có thể nhận biết được chúng.

Cuối cùng, việc phát triển các công nghệ nhận diện hình ảnh và tên của động vật cũng giúp chúng ta có thể phát triển các ứng dụng thực tế trong đời sống hàng ngày, chẳng hạn như ứng dụng hỗ trợ vào việc phân loại thực phẩm, phát hiện người nghiện ma túy, và các ứng dụng trong lĩnh vực y tế. Đó cũng là lí do em chọn đề tài: "**Nhận diện hình ảnh và tên của một số loài động vật**".

1.3. Mục tiêu nghiên cứu

- Xây dựng mô hình dự đoán hình ảnh và tên của một số loài động vật bằng giải pháp CNN
- Đưa mô hình lên webapp để mọi người dễ sử dụng

1.4. Phương pháp nghiên cứu

- Tìm hiểu tên và hình ảnh của các loài động vật, tiến hành lấy dữ liệu
- Tìm hiểu các phương pháp hiện có thông qua Github, Kagle
- Xây dựng "Model trainning" và tiến hành chạy thử để kiểm chứng

1.5. Nội dung báo cáo

Bài báo cáo tập trung vào việc nghiên cứu phương pháp tối ưu để tạo model train bằng CNN sao cho độ chính xác đạt cao nhất. Từ đó đưa ra chính xác hình ảnh và tên của loài động vật mà chúng ta cần nhận diện. Model tạo ra sẽ được đưa lên webapp để mọi người dễ dàng sử dụng.

Bài báo cáo này được xây dựng với … chương, bao gồm:

CHƯƠNG 1: TỔNG QUAN

CHƯƠNG 2: CƠ SỞ LÍ THUYẾT

CHƯƠNG 3: ỨNG DỤNG

CHƯƠNG 4: KẾT LUẬN

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

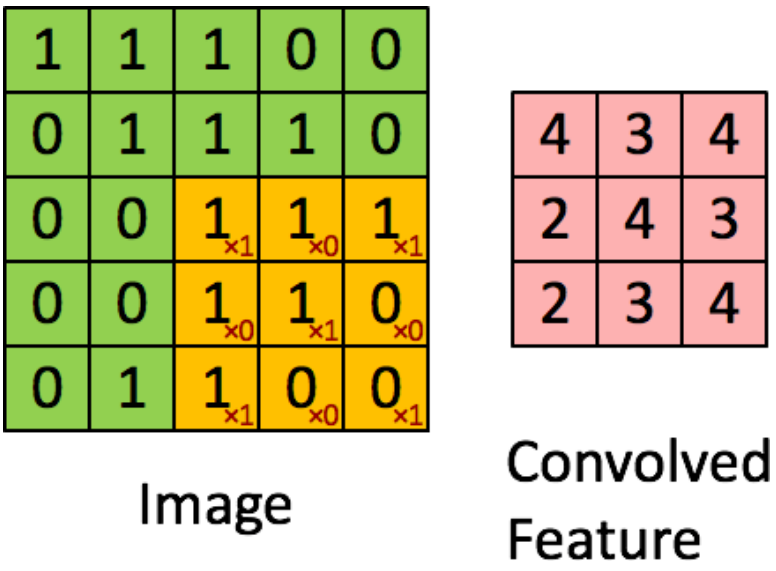
2.1. Thuật toán CNN - Convolutional Neural Network

2.1.1. 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. Để tìm hiểu tại sao thuật toán nàyđược sử dụng rộng rãi cho việc nhận dạng (detection), chúng ta hãy cùng tìm hiểu về thuật toán này.

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



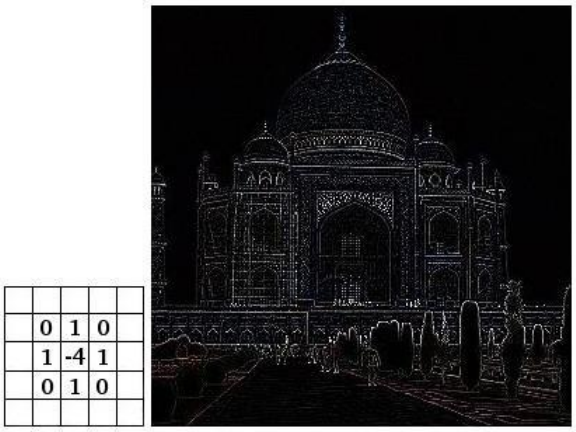
Hình 2.1. Ma trận Convolutional

Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

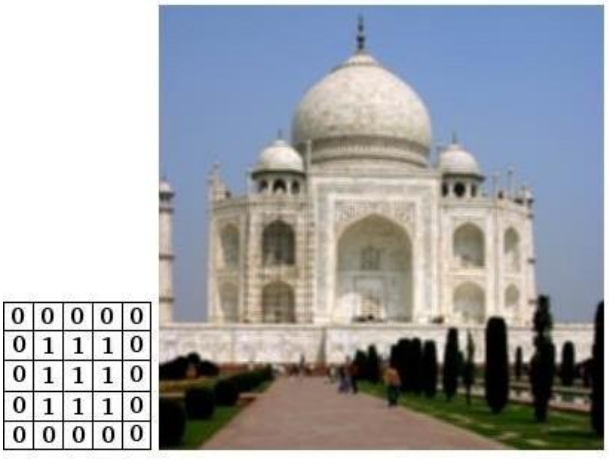
Trong hình ảnh ví dụ Hình 2.2, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước 5×5 và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểmcủa dòng và cột.

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

2.1.2. 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 và tanh 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.

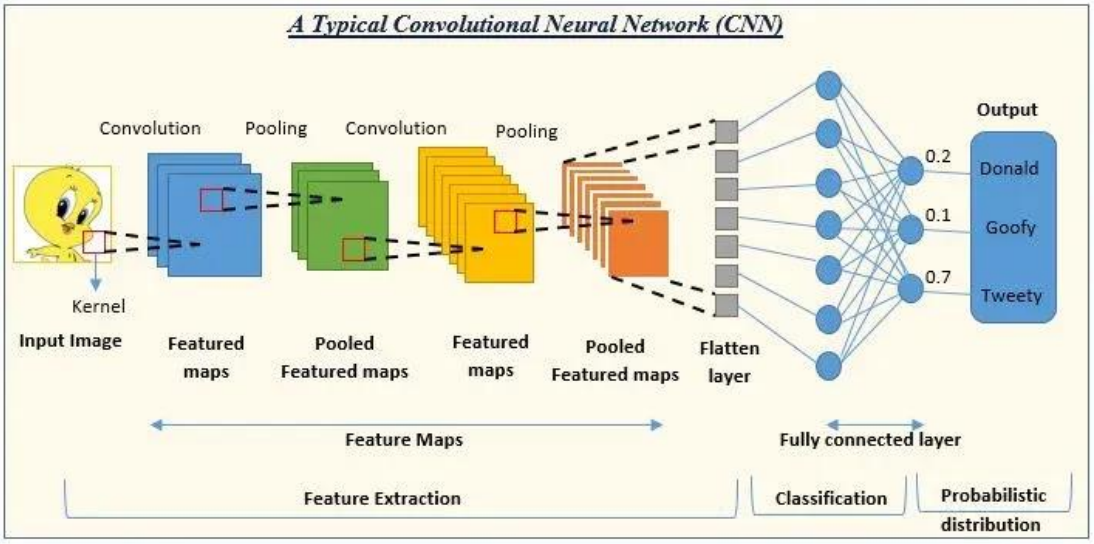
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

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.

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).

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 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.

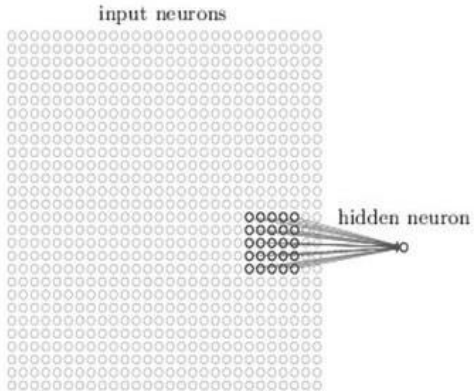
Mạng CNN sử dụng 3 ý tưởng cơ bản:

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

2.1.2.1. 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 không làm như vậy mà 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.

2.1.2.2. 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.

2.1.2.3. 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.

Như vậy qua lớp Max Pooling thì số lượng neuron giảm đi phân nửa. Trong một mạng CNN có nhiều Feature Map nên mỗi Feature Map chúng ta sẽ cho mỗi Max Pooling khác nhau. Chúng ta 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 ta đặt tất cả các lớp lại với nhau thành một CNN với đầu ra gồm các neuron với số lượng tùy bài toán.

2.2. Thư viện Tensorflow

Tensorflow là một thư viện mã nguồn mở phục vụ cho hoạt động Machine Learning. Nó được xây dựng và phát triển bởi chính Google. Trong quy trình phát triển một phần mềm bất kỳ đòi hỏi rất nhiều đoạn mã cũng như thuật toán được triển khai. Thuật toán vừa để phân tích, tổng hợp dữ liệu vừa là nền tảng để phần mềm có thể khởi chạy. Tuy nhiên chương trình càng lớn thì khối lượng phép toán càng nhiều. Cách tính toán thủ công không thể đảm bảo hiệu suất như mong muốn được. Vì thế Tensorflow xuất hiện như một chương trình hỗ trợ tính toán bằng cách tiếp cận mạnh mẽ các phép tính và bài toán phức tạp. Nhờ có Tensorflow, người dùng có thể đơn giản hóa toán học thông qua các đồ thị luồng dữ liệu tổng hợp. Ngoài ra, không thể không kể đến các ứng dụng của Tensorflow như:

- Tích hợp sẵn rất nhiều các thư viện machine learning
- Có khả năng tương thích và mở rộng tốt. Được Google phát triển cho machine learning phục vụ cả nghiên cứu lẫn xây dựng các ứng dụng thực tế

Dù đem đến nhiều lợi ích nhưng thực ra nguyên lý hoạt động của Tensorflow khá đơn giản. Về cơ bản Tensorflow sẽ giúp người dùng tạo ra các biểu đồ luồng dữ liệu hoặc những cấu trúc mô tả. Đây cũng là lý do tại sao Tensorflow được coi như là một framework. Những khung sườn này sẽ hướng dẫn dữ liệu làm cách nào để đi qua một biểu đồ hoặc một series nodes đang được xử lý. Lúc này, mỗi nodes sẽ đại diện cho một hoạt động toán học cần xử lý. Còn mỗi kết nối hoặc mỗi edge sẽ được coi như một tensor hoặc một mảng dữ liệu đa chiều.

Như đã nêu ở phần trên, Tensorflow vốn được phát triển từ Python. Vì thế bản thân Tensorflow cũng là một ứng dụng Python. Còn các nodes và tensor trong Tensorflow là những đối tượng thuộc Python. Điều này giúp ích rất nhiều cho lập trình viên. Python vốn là một hệ thống dễ sử dụng, nó cho phép các đối tượng trừu tượng bậc cao có thể dễ dàng kết hợp với nhau. Chính nhờ sự giúp đỡ đắc lực này, quá trình phát triển phần mềm được đơn giản hóa đi rất nhiều.

2.3. Thư viện Keras

Keras chạy trên các thư viện máy mã nguồn mở như TensorFlow, Theano hoặc Bộ công cụ nhận thức (CNTK). Theano là một thư viện python được sử dụng cho các tác vụ tính toán số nhanh. TensorFlow là thư viện toán học biểu tượng nổi tiếng nhất được sử dụng để tạo mạng nơ-ron và mô hình học sâu. TensorFlow rất linh hoạt và lợi ích chính là tính toán phân tán. CNTK là khung học sâu được phát triển bởi Microsoft. Nó sử dụng các thư viện như Python, C #, C ++ hoặc các bộ công cụ học máy độc lập. Theano và TensorFlow là những thư viện rất mạnh nhưng khó hiểu để tạo mạng nơ-ron.

Keras dựa trên cấu trúc tối thiểu, cung cấp một cách dễ dàng và dễ dàng để tạo các mô hình học sâu dựa trên TensorFlow hoặc Theano. Keras được thiết kế để xác định nhanh các mô hình học sâu. Chà, Keras là một lựa chọn tối ưu cho các ứng dụng học sâu.

Keras tận dụng các kỹ thuật tối ưu hóa khác nhau để làm cho API mạng thần kinh cấp cao dễ dàng hơn và hiệu quả hơn. Nó hỗ trợ các tính năng sau:

- API nhất quán, đơn giản và có thể mở rộng.
- Cấu trúc tối thiểu - dễ dàng đạt được kết quả mà không cần rườm rà.
- Hỗ trợ nhiều nền tảng và backend.
- Thân thiện với người dùng chạy trên cả CPU và GPU.

- Khả năng mở rộng tính toán cao.

Keras năng động, mạnh mẽ và có những ưu điểm sau

- Cộng động lớn hỗ trợ
- Dễ dàng để kiểm tra.
- Mạng nơ-ron Keras được viết bằng Python giúp mọi thứ đơn giản hơn.

Keras hỗ trợ cả mạng convolution và recurrent.

2.4. Thư viện Scikit-learn (sklearn)

Scikit-learn (sklearn) là một thư viện rất mạnh mẽ, giúp mang các thuật toán học máy (machine learning) vào trong một hệ thống. Thư viện này tích hợp rất nhiều thuật toán hiện đại và cổ điển giúp bạn vừa học vừa tiến hành đưa ra các giải pháp hữu ích cho bài toán của bạn một cách đơn giản. 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.

Để cài đặt scikit-learn trước tiên phải cài thư viện SciPy (Scientific Python). Những thành phần gồm:

- Numpy: Gói thư viện xử lý dãy số và ma trận nhiều chiều
- SciPy: Gói các hàm tính toán logic khoa học
- Matplotlib: Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
- IPython: Notebook dùng để tương tác trực quan với Python
- SymPy: Gói thư viện các kí tự toán học
- Pandas: Xử lý, phân tích dữ liệu dưới dạng bảng

Những thư viện mở rộng của SciPy thường được đặt tên dạng SciKits. Như thư viện này là gói các lớp, hàm sử dụng trong thuật toán học máy thì được đặt tên là scikit-learn.

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.

Mặc dù được viết cho Python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của C để tăng hiệu suất làm việc. Ví dụ như: Numpy (Tính toán ma trận), LAPACK, LibSVM và Cython.

Thư viện tập trung vào việc mô hình hóa dữ liệu. Nó không tập trung vào việc truyền tải dữ liệu, biến đổi hay tổng hợp dữ liệu. Những công việc này dành cho thư viện Numpy và Pandas.

2.5. Streamlit - A faster way to build and share data apps

Streamlit là một open-source Python lib, nó giúp ta dễ dàng tạo một web app cho MachineLearning và Data Science. Đối với một người có rất ít kiến thức về HTML, CSS, JavaScript thì đây có lẽ là công cụ rất phù hợp với mình để demo các sản phẩm AI. Chúng ta có thể test cục bộ được, sau đó muốn deploy lên internet có thể dùng Heroku, Streamlit hay Ngrok đều được.

Streamlit cho phép người dùng sử dụng các lệnh đơn giản, dễ dàng thực hiện. Kết hợp Streamlit và GitHub cho phép một hệ sinh thái vô cùng phong phú và đa dạng ứng dụng hữu ích, từ trang tổng quan đến mạng sâu và hơn thế nữa.

Model sau khi train xong được đưa lên Github, và sử dụng các lệnh của Streamlit để hiển thị thành webapp, điều này khá đơn giản và tiết kiệm thời gian. Link webapp dễ dàng được truy cập bởi những ai có link, đây là một điều khá tiện lợi.

Để cài đặt:

```
pip install streamlit
```

Sau khi cài đặt xong có thể chạy ngay câu lệnh sau, nó đưa chúng ta tới tab với đường link http://localhost:8501/

```
streamlit hello
```

Sau khi cài xong bạn có thể chạy ứng dụng với cú pháp sau:

```
streamlit run app.py
```

Để Desploy webapp streamlit chúng ta chỉ cần đưa lên github và liên kết chúng lại, quá trình desploy sẽ xử lý và đưa ra link web để sử dụng.

CHƯƠNG 3. ỨNG DỤNG

3.1. Xây dựng mô hình – Trainning model

Em sử dụng Google Colab để xây dựng mô hình trainning.

3.1.1. Datasets

Trong nghiên cứu này, em sử dụng dữ liệu về hình ảnh và tên của một số loài động vật được em tự tạo dữ liệu bằng cách lên Google tải ảnh của các con vật em muốn nhận diện sau đó gắn nhãn tên cho từng loài động. Dữ liệu này được chia ra thành 3 thư mục nhỏ: train (dùng để trainning model), test (dùng để test model), val (validations).

Bảng 3.1. Datasets

	Loại hình	Số lượng ảnh
train	Tên và hình của 10 loài động vật	1010
test	Tên và hình của 10 loài động vật	10
val	Tên và hình của 10 loài động vật	10

3.1.2. Xây dựng mô hình

Bước 1: Liên kết Google Drive

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

Bước 2: Khai báo các thư viện cần thiết

```
from os import listdir
from numpy import asarray, save
from keras.utils import load_img
from keras.utils import img_to_array
from keras.utils import to_categorical
from keras.models import Sequential,Model
from keras.layers import
Dense,Flatten,Dropout,Conv2D,MaxPooling2D,Normalization,Input
from keras.optimizers import Adam
from keras import losses
```

```
from keras.layers import LeakyReLU
from keras.losses import categorical_crossentropy
import matplotlib.pyplot as plt
import numpy as np
from os import listdir
from keras.utils import load_img
from keras.utils.image_utils import img_to_array
```

Bước 3: Lấy dữ liệu và mô tả dữ liệu

```
folder = '/content/drive/MyDrive/FILE_ANH_AI/Animals/'
photos, labels = list(), list()
for file in listdir(folder):
  output= 0.0
  if file.startswith('cat'):
    output= 1.0
  if file.startswith('cow'):
    output= 2.0
  if file.startswith('dog'):
    output= 3.0
  if file.startswith('duck'):
    output= 4.0
  if file.startswith('elephant'):
    output= 5.0
  if file.startswith('horse'):
    output= 6.0
  if file.startswith('monkey'):
    output= 7.0
  if file.startswith('rabbit'):
    output= 8.0
  if file.startswith('sheep'):
    output= 9.0
  if file.startswith('snake'):
    output= 10.0
  photo = load_img(folder + file, target_size= (40,40))
  photo= img_to_array(photo)

  photos.append(photo)
   labels.append(output)
```

```
photos= asarray(photos)
labels= asarray(labels)
print(photos.shape, labels.shape)
save('/content/drive/MyDrive/npy_Data/animals_photos.npy', photos)
save('/content/drive/MyDrive/npy_Data/animals_labels.npy', labels)
```

```
(1008, 40, 40, 3) (1008,)
```

```
# Split data into train & test
split_index = int(0.05 * len(photos))
test_x, test_y = photos[:split_index], labels[:split_index]
train_x, train_y = photos[split_index:], labels[split_index:]
```

In [46]:

```
print(test_x.shape, train_x.shape)
```

```
(50, 40, 40, 3) (958, 40, 40, 3)
```

```
train_x = train_x.reshape((958, 40, 40, 3)) #sửa lại cho giống như ma trận hàng trêntrên
train_x = train_x.astype('float32')/255

test_x = test_x.reshape((50, 40, 40, 3))
test_x = test_x.astype('float32')/255

from keras.utils import to_categorical
train_y = to_categorical(train_y,11) #bao nhiu output thì sửa lạilại
test_y = to_categorical(test_y,11)
from keras.models import Sequential,Model
from keras.layers import
Dense,Flatten,Dropout,Conv2D,MaxPooling2D,Normalization,Input
from keras.optimizers import Adam
from keras import losses
loss = losses
batch_size = 64
epochs = 5
classes = 11 # bao nhiêu output thì sửa lại
```

Bước 3: Xây dựn mô hình CNN

```
from keras.layers import LeakyReLU
model = Sequential()
model.add(Conv2D(32, kernel_size = (3,3), activation = 'linear', input_shape
= (40, 40, 3), padding= 'same'))
model.add(LeakyReLU(alpha = 0.1))

model.add(MaxPooling2D((2,2), padding = 'same'))
model.add(Conv2D(64, (3,3), activation = 'linear', padding = 'same'))
model.add(Conv2D(64, (3,3), activation = 'linear', padding = 'same'))
model.add(LeakyReLU(alpha = 0.1))

model.add(MaxPooling2D((2,2), padding = 'same'))
model.add(Conv2D(128, (3,3), activation = 'linear', padding = 'same'))
model.add(LeakyReLU(alpha = 0.1))

model.add(MaxPooling2D((2,2), padding = 'same'))
model.add(Conv2D(256, (3,3), activation = 'linear', padding = 'same'))
model.add(LeakyReLU(alpha = 0.1))

model.add(MaxPooling2D((2,2), padding = 'same'))
model.add(Conv2D(512, (3,3), activation = 'linear', padding = 'same'))
model.add(LeakyReLU(alpha = 0.1))

model.add(MaxPooling2D((2,2), padding = 'same'))
model.add(Conv2D(1024, (3,3), activation = 'linear', padding = 'same'))
model.add(LeakyReLU(alpha = 0.1))
```

```
#Đưa vào ANN, bộ ANN để phân loại:
from keras.losses import categorical_crossentropy
model.add(Flatten())
model.add(Dense(1024, activation = 'linear'))

model.add(Dense(classes, activation = 'softmax'))
model.summary()
```

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

- Thêm một layer Conv2D với 32 filters, kernel_size là (3,3), activation là 'linear', và padding là 'same' để đảm bảo kích thước của đầu ra không thay đổi. input_shape chỉ định kích thước của ảnh đầu vào là (40, 40, 3), trong đó 3 là số kênh (RGB) của ảnh.
- Thêm layer MaxPooling2D với kích thước pool_size = (2,2), và padding = 'same'. Layer này giúp giảm kích thước của đầu ra, giảm chi phí tính toán và giảm khả năng overfitting.
- Thêm layer Conv2D thứ hai với 64 filters, kernel_size = (3,3), activation = 'linear', và padding = 'same'.
- Thêm layer Conv2D thứ ba với 64 filters, kernel_size = (3,3), activation = 'linear', và padding = 'same'.
- Thêm layer MaxPooling2D thứ hai với kích thước pool_size = (2,2), và padding = 'same'.
- Thêm layer Conv2D thứ tư với 128 filters, kernel_size = (3,3), activation = 'linear', và padding = 'same'.
- Thêm layer MaxPooling2D thứ ba với kích thước pool_size = (2,2), và padding = 'same'.
- Thêm layer Conv2D thứ năm với 256 filters, kernel_size = (3,3), activation = 'linear', và padding = 'same'.
- Thêm layer MaxPooling2D thứ tư với kích thước pool_size = (2,2), và padding = 'same'.
- Thêm layer Conv2D thứ sáu với 512 filters, kernel_size = (3,3), activation = 'linear', và padding = 'same'.
- Thêm layer MaxPooling2D thứ năm với kích thước pool_size = (2,2), và padding = 'same'.
- Thêm layer Conv2D thứ bảy với 1024 filters, kernel_size = (3,3), activation = 'linear', và padding = 'same'.
- Thêm layer kích hoạt LeakyReLU với giá trị alpha = 0.1.
- Mô hình hoàn tất trích xuất đặc trưng của ảnh đầu vào và trả về đầu ra với kích thước (2,2,1024).

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

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_21 (Conv2D)          (None, 40, 40, 32)        896

 leaky_re_lu_18 (LeakyReLU)  (None, 40, 40, 32)        0

 max_pooling2d_15 (MaxPoolin  (None, 20, 20, 32)       0
 g2D)

 conv2d_22 (Conv2D)          (None, 20, 20, 64)        18496

 conv2d_23 (Conv2D)          (None, 20, 20, 64)        36928
```

```
 leaky_re_lu_19 (LeakyReLU)  (None, 20, 20, 64)        0

 max_pooling2d_16 (MaxPoolin  (None, 10, 10, 64)       0
 g2D)

 conv2d_24 (Conv2D)          (None, 10, 10, 128)       73856

 leaky_re_lu_20 (LeakyReLU)  (None, 10, 10, 128)       0

 max_pooling2d_17 (MaxPoolin  (None, 5, 5, 128)        0
 g2D)

 conv2d_25 (Conv2D)          (None, 5, 5, 256)         295168

 leaky_re_lu_21 (LeakyReLU)  (None, 5, 5, 256)         0

 max_pooling2d_18 (MaxPoolin  (None, 3, 3, 256)        0
 g2D)

 conv2d_26 (Conv2D)          (None, 3, 3, 512)         1180160

 leaky_re_lu_22 (LeakyReLU)  (None, 3, 3, 512)         0

 max_pooling2d_19 (MaxPoolin  (None, 2, 2, 512)        0
 g2D)

 conv2d_27 (Conv2D)          (None, 2, 2, 1024)        4719616

 leaky_re_lu_23 (LeakyReLU)  (None, 2, 2, 1024)        0

 flatten_3 (Flatten)         (None, 4096)              0

 dense_6 (Dense)             (None, 1024)              4195328

 dense_7 (Dense)             (None, 11)                11275

=================================================================
Total params: 10,531,723
Trainable params: 10,531,723
Non-trainable params: 0
```

Bước 4: Huấn luyện mô hình và test độ chính xác

```
#Compile:
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics
= ['accuracy'])
train = model.fit(train_x, train_y, batch_size= batch_size, epochs= epochs,
verbose= 1)
```

Mô hình được huấn luyện với số lần lọc 5, ta được kết quả như sau:

```
Epoch 1/5
15/15 [==============================] - 4s 23ms/step - loss: 0.1023 - accuracy: 0.9864
Epoch 2/5
15/15 [==============================] - 0s 18ms/step - loss: 0.1228 - accuracy: 0.9697
Epoch 3/5
15/15 [==============================] - 0s 18ms/step - loss: 0.2131 - accuracy: 0.9530
Epoch 4/5
15/15 [==============================] - 0s 18ms/step - loss: 0.0778 - accuracy: 0.9770
Epoch 5/5
15/15 [==============================] - 0s 18ms/step - loss: 0.0471 - accuracy: 0.9916
```

Độ chính xác

```
# Evaluate model
test_loss, test_acc = model.evaluate(test_x, test_y)
print('Test accuracy:', test_acc)

2/2 [==============================] - 0s 11ms/step - loss: 10.9134 -
accuracy: 0.1600
Test accuracy: 0.1599999964237213
```

Bước 5: Lưu model

```
model.save('/content/drive/MyDrive/Colab Notebooks/Data/animals.h5')
```

3.2. Xây dựng GUI với

Fima là một thư viện Python cho phép bạn tạo giao diện người dùng (GUI) để tương tác với các ứng dụng và script Python của mình. Fima cung cấp giao diện trực quan cho người dùng thông qua các phần tử như cửa sổ, nút bấm, trường văn bản và các phần tử khác.
Fima là thư viện mã nguồn mở, được cài đặt bằng Python và được phát triển để làm cho việc tạo giao diện và tương tác với các ứng dụng Python dễ dàng hơn. Với Fima, người dùng có thể tạo ra các giao diện đơn giản và nhanh chóng, trong đó người dùng có thể nhập dữ liệu, chọn từ một loạt các tùy chọn và xem kết quả được hiển thị trực quan trên giao diện.
Các ứng dụng của Fima có thể được sử dụng trong nhiều môi trường khác nhau, bao gồm các ứng dụng máy tính để bàn, ứng dụng web và các ứng dụng di động. Fima hỗ trợ các hệ điều hành khác

nhau bao gồm Windows, MacOS và Linux, giúp cho việc phát triển nhanh chóng và tiết kiệm thời gian.

Trong đề tài này, mô hình sau khi em đã train được lưu dưới dạng file .h5, model sẽ được load lên GUI trong python, tiến hành xử lí hình ảnh theo các thông số đầu vào như đã trainning ở trên và đưa ra dự đoán, GUI sẽ hiện ra lựa chọn hình ảnh muốn dự đoán, có các nút nhấn thao tác để nhấn chọn ảnh,dự đoán và sẽ xuất ra kết quả dự đoán giúp người dùng dễ dàng sử dụng hơn.

CHƯƠNG 4. KẾT LUẬN

4.1. Kết quả đạt được

4.1.1. Mô hình chuẩn đoán

Mô hình CNN chuẩn đoán hình ảnh và tên động vật với độ chính xác 85%, cho ra kết quả chuẩn đoán khá chính xác, phân biệt được hình ảnh và tên của các loài động vật khác nhau.

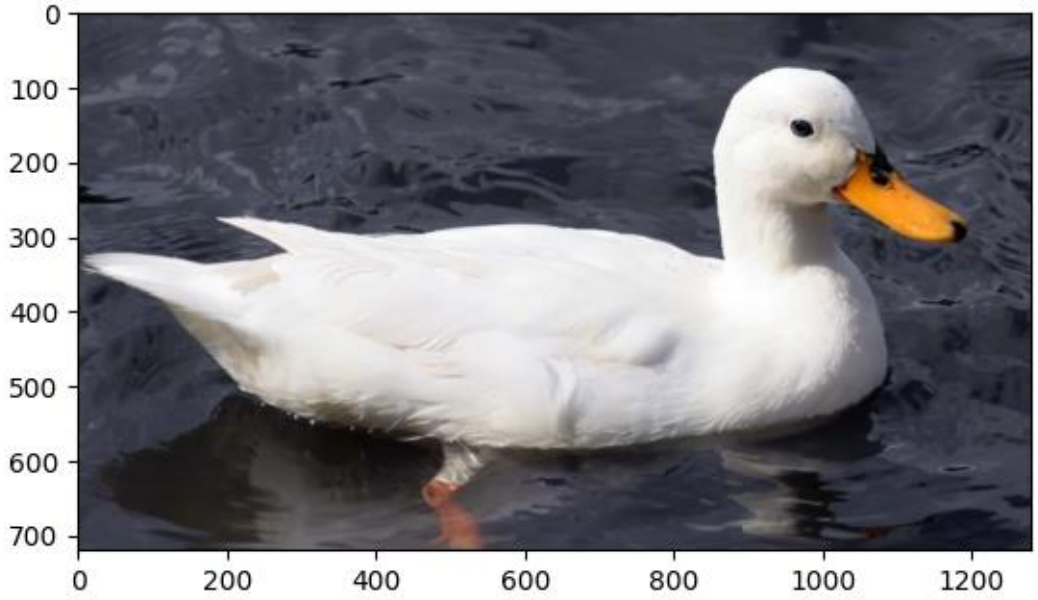
Sử dụng google colab để chạy dự đoán, chúng ta có một số kết quả sau:

```
import matplotlib.pyplot as plt
import numpy as np
from os import listdir
from keras.utils import load_img
from keras.utils.image_utils import img_to_array
folder = '/content/drive/MyDrive/FILE_ANH_AI/Test_animals/'
for file in listdir(folder):
  photo = load_img(folder  +  file)
  plt.imshow(photo)

  photo = load_img(folder +   file, target_size = (40, 40))
  photo=img_to_array(photo)
  photo=photo.astype('float32')
  photo=photo/255
  photo=np.expand_dims(photo,axis=0)
  result=(model.predict(photo).argmax())
  class_name=['','cat', 'cow', 'dog', 'duck', 'elephant', 'horse', 'monkey',
'rabbit', 'sheep', 'snake']
  print(class_name[result])
  plt.show()
```

```
1/1 [==============================] - 0s 163ms/step
cat
```

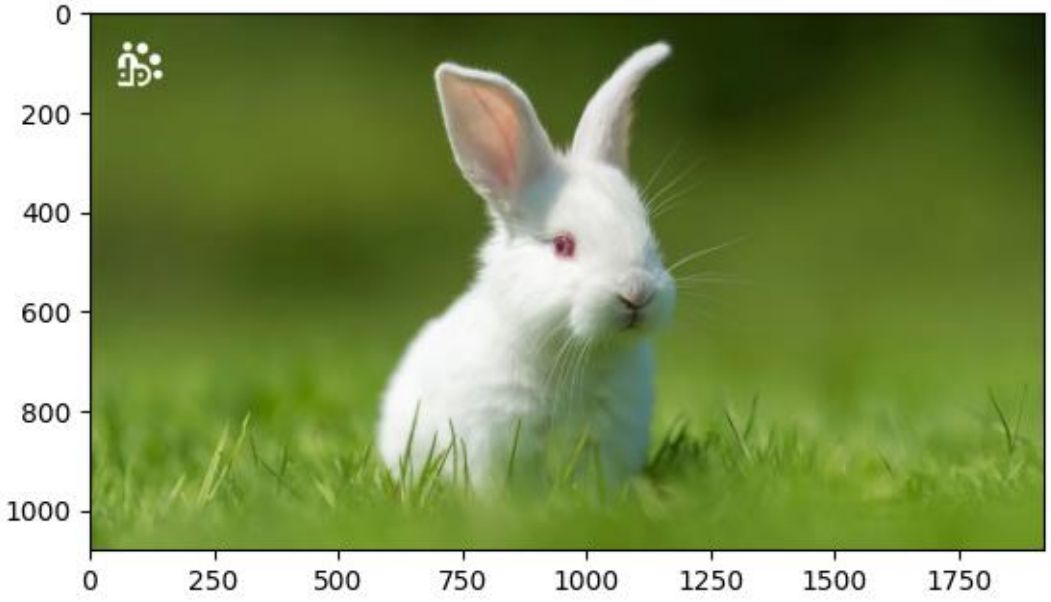
```
1/1 [==============================] - 0s 23ms/step
duck
```



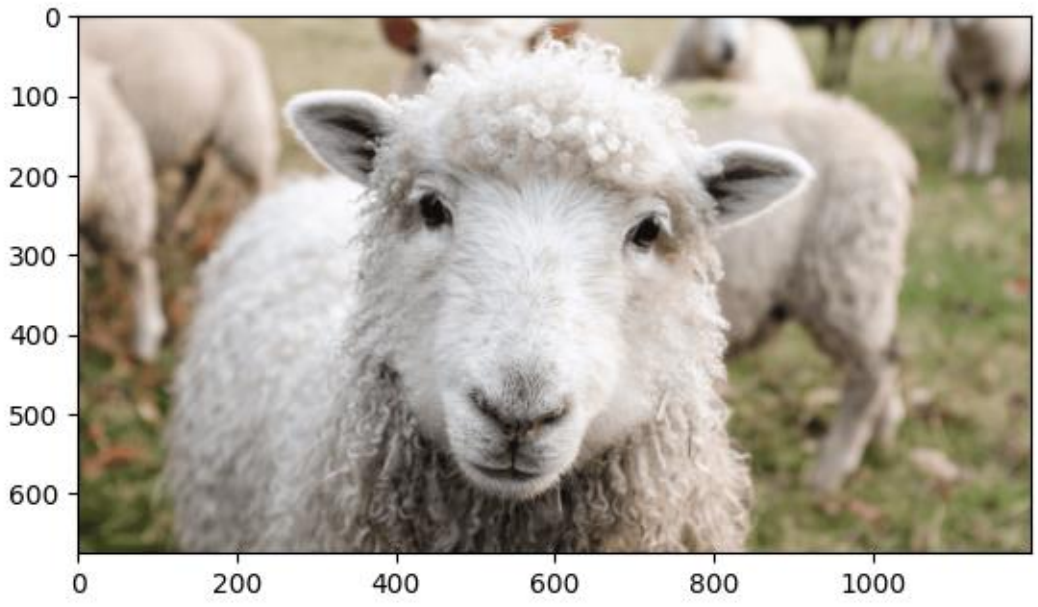

```
1/1 [==============================] - 0s 25ms/step
horse
```


```
1/1 [==============================] - 0s 21ms/step
snake
```

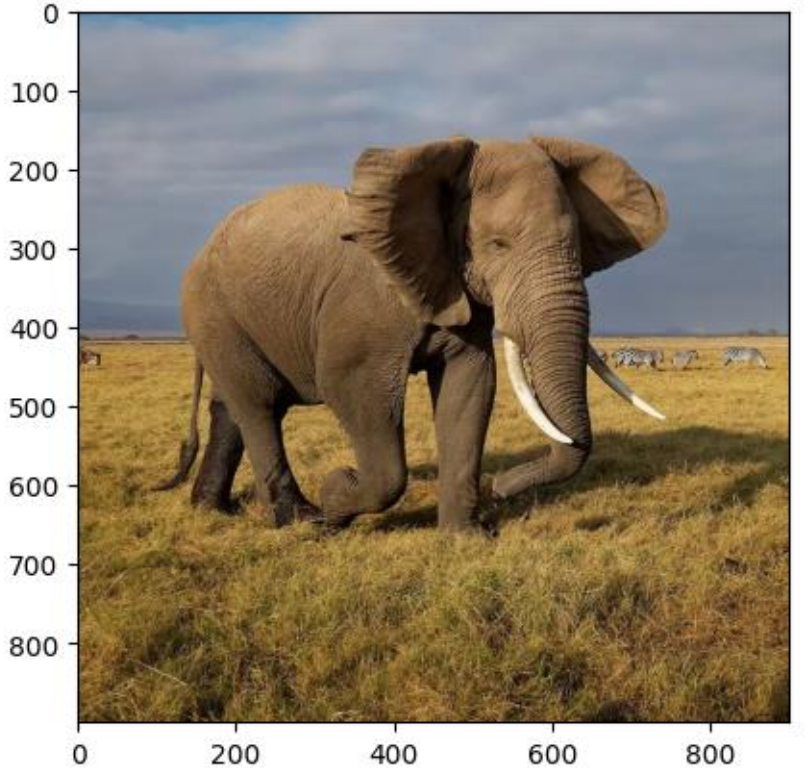

```
1/1 [==============================] - 0s 22ms/step
dog
```



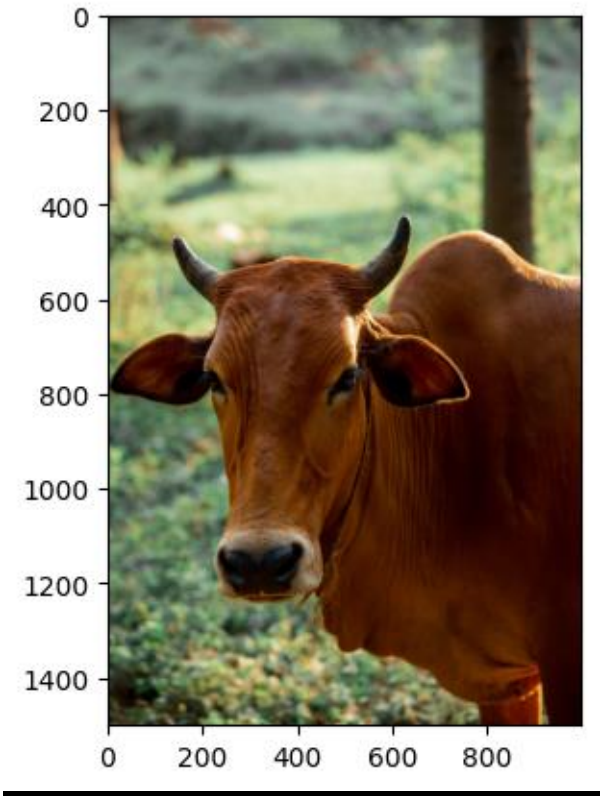

```
1/1 [==============================] - 0s 24ms/step
sheep
```



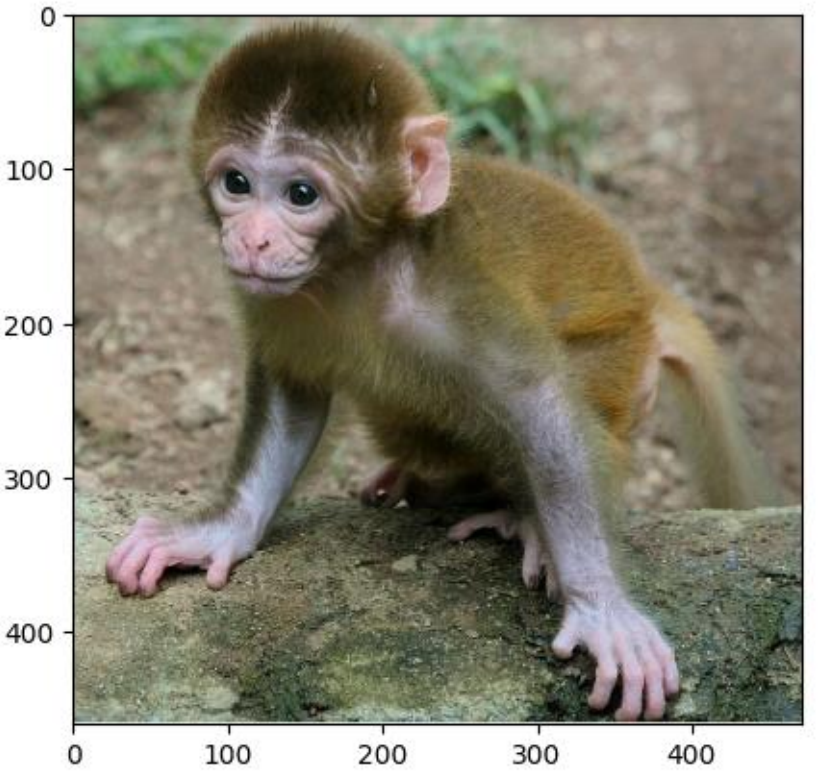

```
1/1 [==============================] - 0s 22ms/step
elephant
```



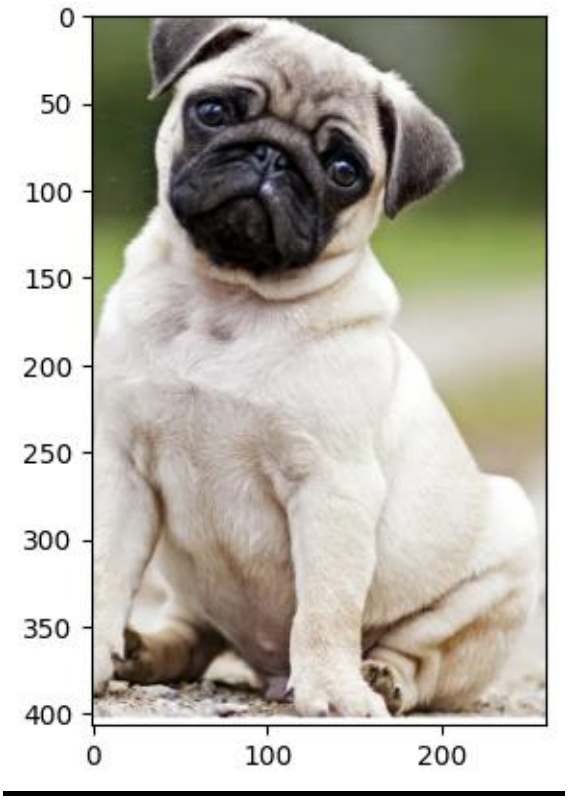

```
1/1 [==============================] - 0s 40ms/step
cow
```



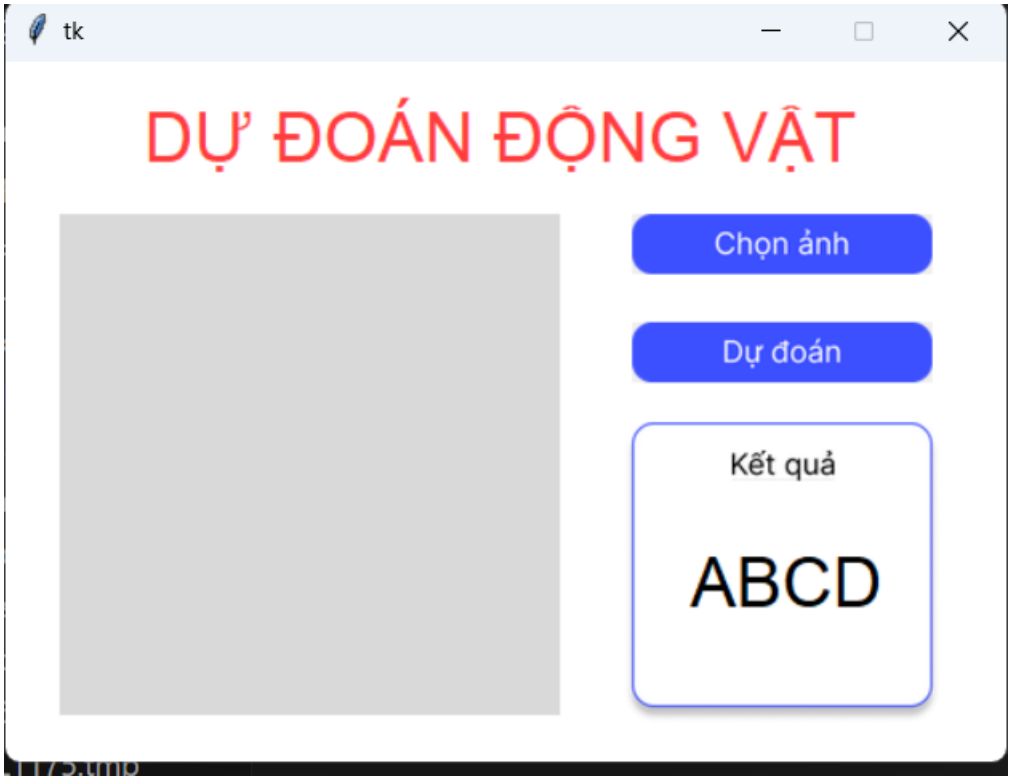

```
1/1 [==============================] - 0s 28ms/step
monkey
```




```
1/1 [==============================] - 0s 35ms/step
cat
```

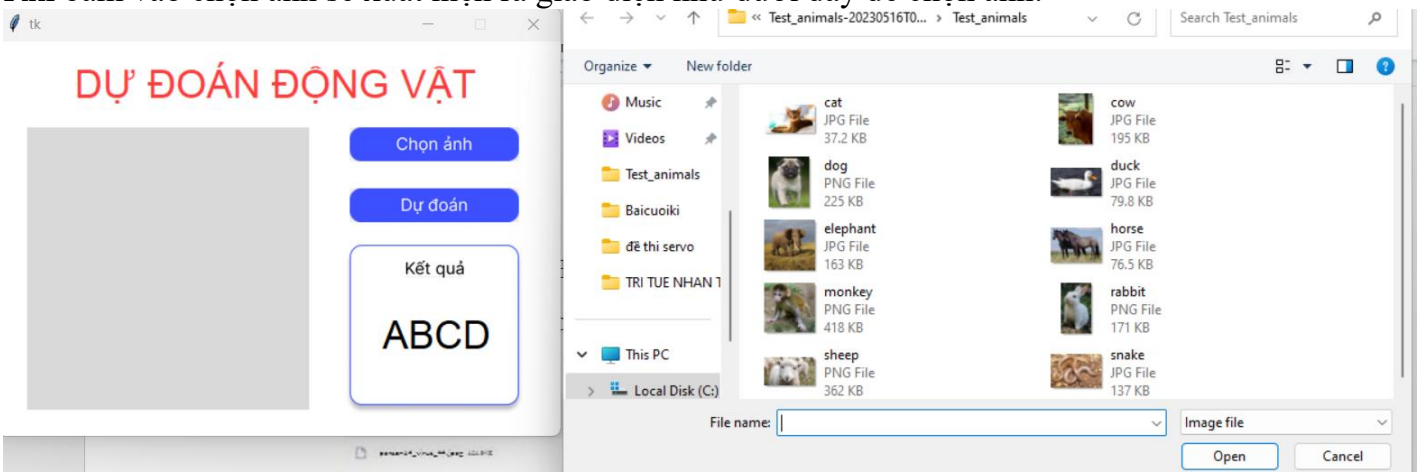

0
50
100
150
200
250
300
350
400
0
100
200

4.1.2. GUI

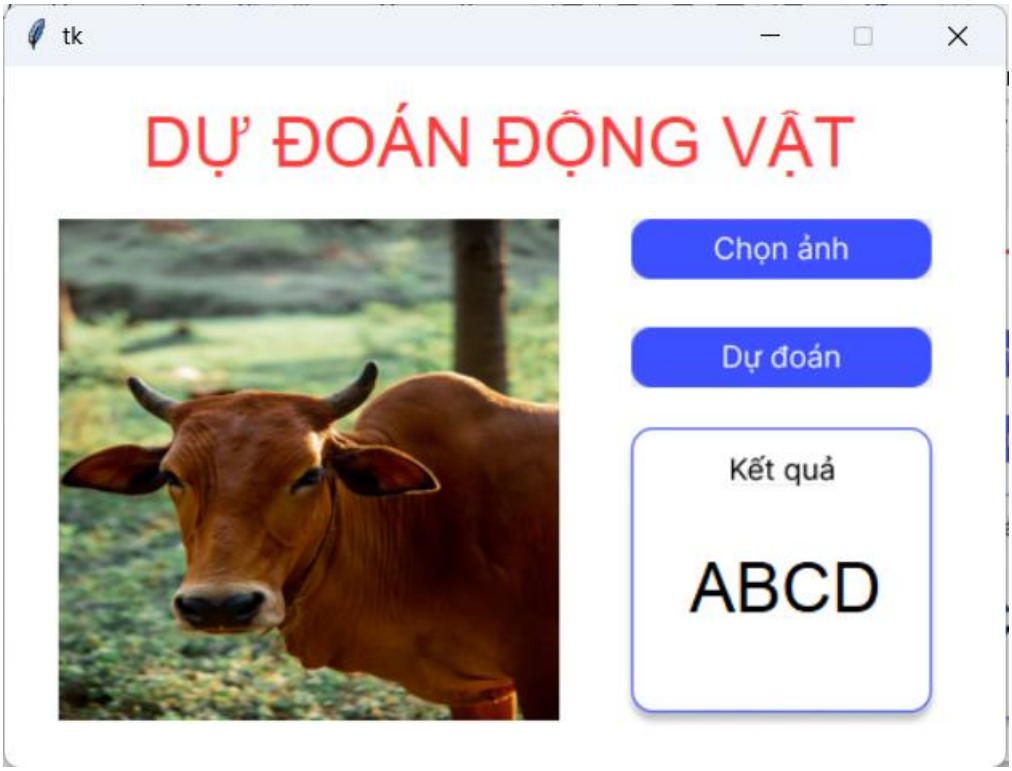


Giao diện của GUI có các thành phần như sau: một nút nhấn để chọn ảnh cần test, một khung để hiển thị hình ảnh, một nút dự đoán khi bấm vào thì sẽ chạy hàm dự đoạn, một khung chứa kết quả tên của động vật.

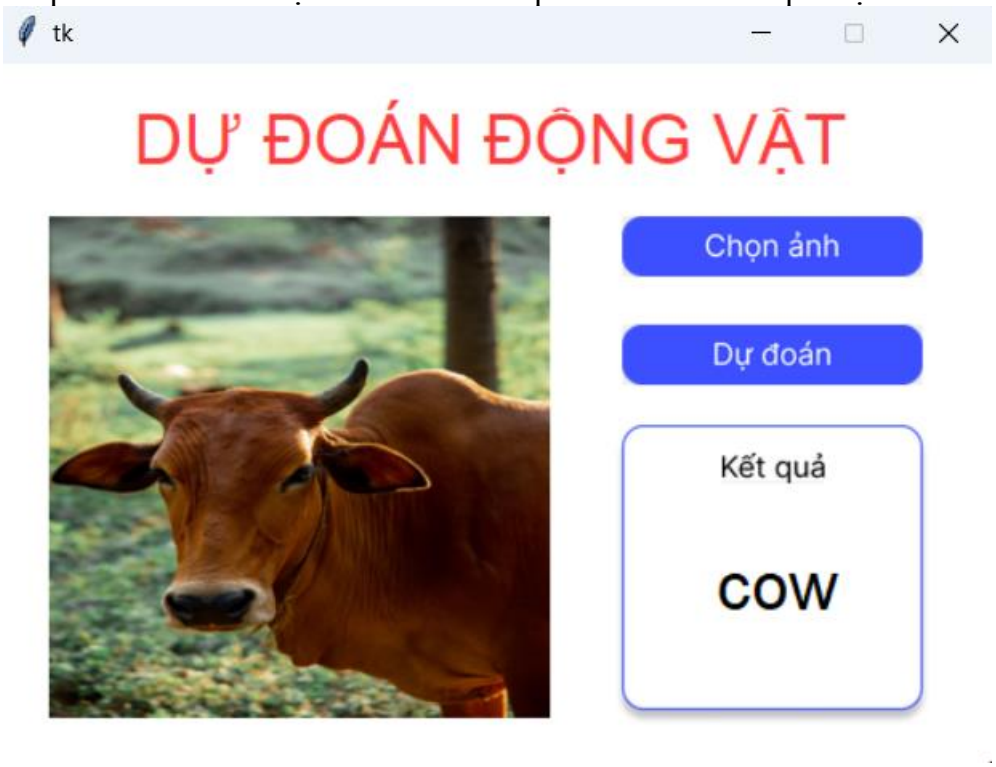
Khi bấm vào chọn ảnh sẽ xuất hiện ra giao diện như dưới đây để chọn ảnh:



Khi chúng ta chọn ảnh thì hình ảnh sẽ hiển thị lên khung ảnh:



Tiếp theo nhấn nút dự đoán thì ô kết quả sẽ xuất ra kết quả dự đoán:



4.2. Nhược điểm

Mô hình tạo ra với độ chính xác tương đối cao nhưng vẫn còn chuẩn đoán sai một số hình ảnh (Chủ yếu hình ảnh chưa cắt sát còn phông nền nên độ chính xác không cao).

Mô hình chưa kết hợp với chạy thời gian thực nên tính ứng dụng chưa cao.

4.3. Hướng phát triển

Trong tương lai, em sẽ tiếp tục phát triển huấn luyện mô hình để nâng độ chính xác lên cao hơn. Bên cạnh đó, em sẽ tiếp tục hoàn thiện GUI , kết hợp việc phát triển các ứng dụng thực tế trong đời sống hàng ngày, chẳng hạn như ứng dụng hỗ trợ vào việc phân loại thực phẩm, phát hiện người nghiện ma túy, và các ứng dụng trong lĩnh vực y tế.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1].Thiết kết giao diện GUI bằng Figma trên wed, Link : https://www.figma.com/file/p2IKLx1SsJopHeZELrd8qq/Untitled?type=design&node-id=0-1&t=Uj0iYB0F9Eo4QPzi-0

[2]. Trần Văn Huy, Introduction to Streamlit, Link truy cập: https://huytranvan2010.github.io/Introduction-to-streamlit/

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

[4]Hướng dẫn tạo GUI bằng Figma, Link truy cập: https://vietnix.vn/figma-la-gi/

PHỤ LỤC

1. Link Datasets:
   https://drive.google.com/drive/folders/1XRmNpa0DoD8qPAUanh0ZABbq3Ag3YyjU
2. Link Github(model và train):
   https://github.com/NguyenthanhTung14/TONGHOPCACFILEAI
3. Link GUI làm trên Figma:
   https://www.figma.com/file/p2IKLx1SsJopHeZELrd8qq/Untitled?type=design&node-id=0-1&t=1tVXsWsNqjNU6H73-0
4. Link Youtube: https://youtu.be/4D64ieHbbEw
5. Link file animals.h5: https://drive.google.com/drive/search?q=animals.h5