TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Phát triển phần mềm

Nhận dạng ảnh dùng AI trên PYTHON

GVHD: Từ Lãng Phiêu
Sinh viên thực hiện: Thủy Ngọc Mai Thy - 3120410521
Nguyễn Thị Phương Thùy - 3120410515

TP. HỒ CHÍ MINH, THÁNG 4/2024

Mục lục

1 Giới thiệu về đề tài

1.1 Giới thiệu chung đề tài

- Deep Learning là một thuật toán dựa trên một số ý tưởng não bộ tới việc tiếp thu nhiều tầng biểu đạt, cả cụ thể lẫn trừu tượng, qua đó làm rõ nghĩa của các loại dữ liệu. Deep Learning được ứng dụng nhiều trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên.
- Hiện nay rất nhiều các bài toán nhận dạng, phân loại sử dụng deep learning để giải quyết do deep learning có thể giải quyết các bài toán với số lượng, kích thước đầu vào lớn với hiệu năng cũng như độ chính xác vượt trội so với các phương pháp phân lớp truyền thống.
- Convolutional Neutal Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được nhưng hệ thống thông minh với độ chính xác cao. Trong bài báo cáo này, chúng em tập trung nghiên cứu về "Mạng Nơ-ron nhân tạo" cũng như ý tưởng phân lớp ảnh dựa trên mô hình CNNs. Và áp dụng để xây dựng ứng dụng phân lớp ảnh "airplane", "automobile", "bird", "cat", "deer", "frog", "horse", "ship" và "truck".

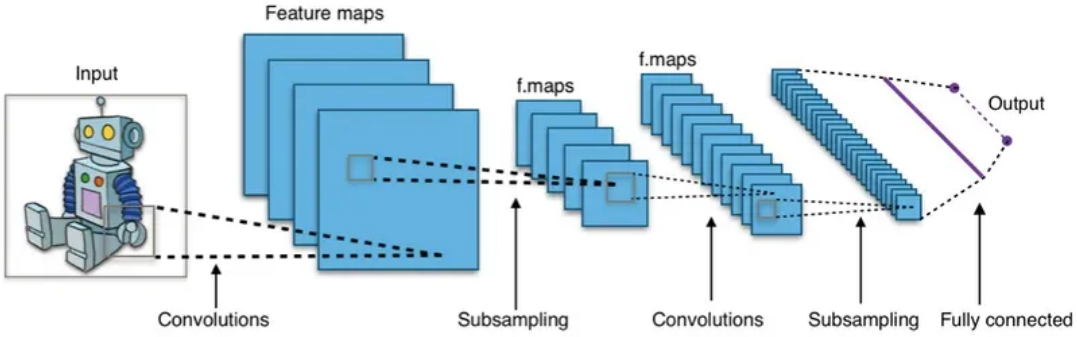
1.2 Mục đích của đề tài

- Xây dựng website nhận diện hình ảnh với các lớp ảnh "airplane", "automobile", "bird", "cat", "deer", "frog", "horse", "ship" và "truck".
- Hiểu và biết cách sử dụng ngôn ngữ lập trình Python.
- Tìm hiểu về thư viện Keras.
- Tìm hiểu phương pháp nhận dạng hình ảnh - Convolutional Neutal Network (CNN).

2 Tìm hiểu mạng nơ-ron tích chập

2.1 Mạng nơ-ron tích chập - CNN

- CNN là viết tắt của Convolutional Neural Network hay còn được gọi là CNNS mạng nơ-ron tích chập, là một trong những mô hình Deep Learning cực kỳ tiên tiến, bởi chúng cho phép bạn xây dựng những hệ thống có độ chính xác cao và thông minh. Nhờ khả năng đó, CNN có rất nhiều ứng dụng, đặc biệt là những bài toán cần nhận dạng vật thể (object) trong ảnh.
- CNN vô cùng quan trọng để tạo nên những hệ thống nhận diện thông minh với độ chính xác cao trong thời đại công nghệ ngày nay. Lý do cụ thể vì sao CNN đặc biệt phát huy hiệu quả trong việc nhận dạng (detection), chúng ta sẽ tìm hiểu kỹ hơn ngay dưới đây.



Hình 1: Mạng nơ-ron tích chập - CNN

2.2 Convolutional là gì?

- Convolutional là một loại cửa sổ dạng trượt nằm trên một ma trận. Các convolutional layer sẽ chứa các parameter có khả năng tự học, qua đó sẽ điều chỉnh và tìm ra cách lấy những thông tin chính xác nhất mà không cần chọn feature.
- Lúc này, convolution hay tích chập đóng vai trò là nhân các phần tử thuộc ma trận. Sliding Window, hay được gọi là kernel, filter hoặc feature detect, là loại ma trận có kích thước nhỏ.

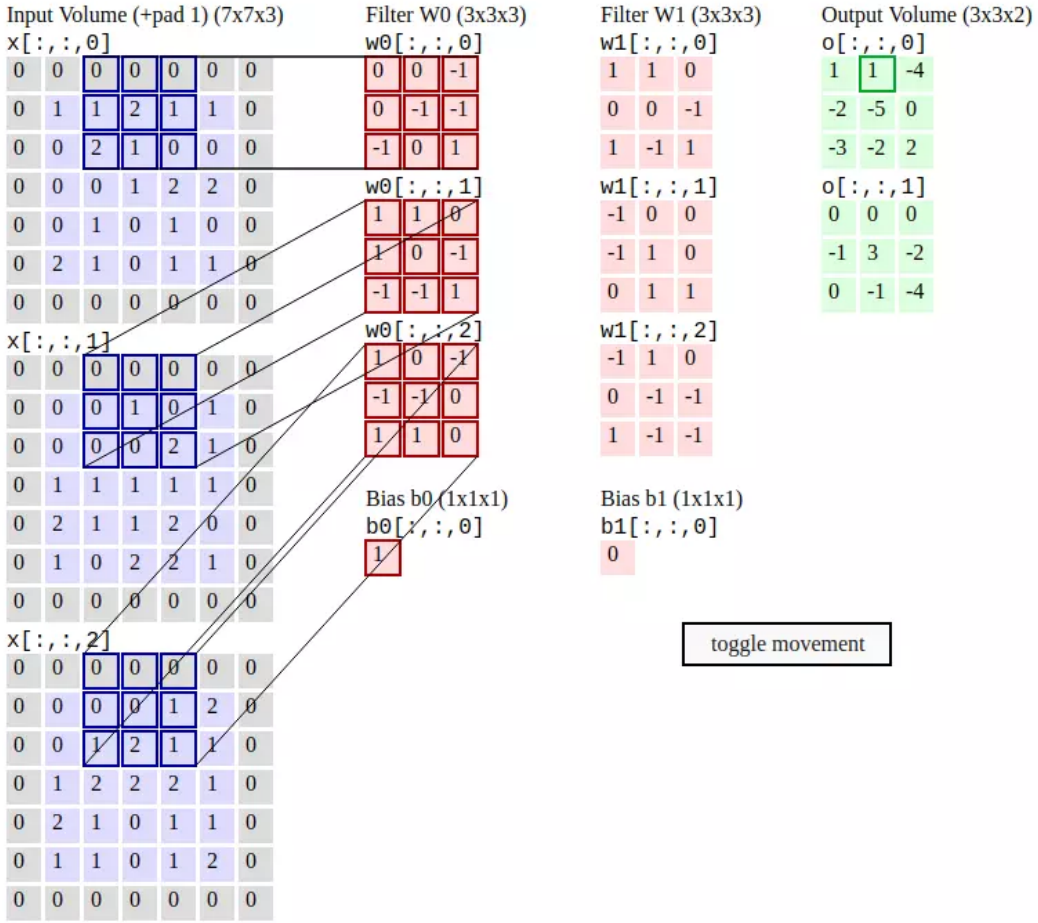
2.3 Các lớp cơ bản của mạng CNN:

2.3.1 Convolutional layer

Đây chính là lớp đóng vai trò mấu chốt của CNN, khi layer này đảm nhiệm việc thực hiện mọi tính toán để trích xuất ra những đặc tính nổi bật của tấm hình đầu vào. Stride, padding, filter map, feature map là những yếu tố quan trọng nhất của convolutional layer.

- Cơ chế của CNN là tạo ra các filter áp dụng vào từng vùng hình ảnh. Các filter map này được gọi là ma trận 3 chiều, bên trong chứa các parameter dưới dạng những con số.
- Stride là sự dịch chuyển filter map theo pixel dựa trên giá trị từ trái sang phải.

- Padding: Là các giá trị 0 được thêm cùng lớp input.
- Feature map: Sau mỗi lần quét, một quá trình tính toán sẽ được thực hiện. Feature map sẽ thể hiện kết quả sau mỗi lần filter map quét qua input.



Hình 2: Convolutional layer

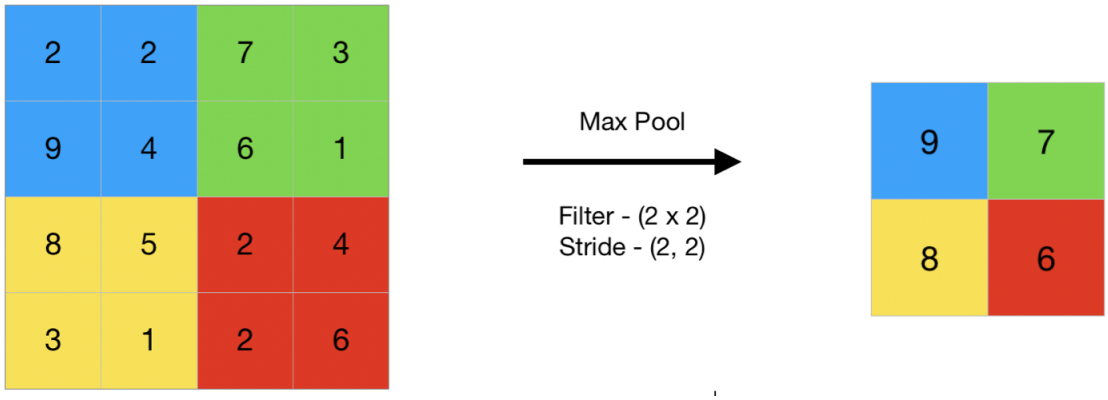
2.3.2 ReLU layer

ReLU (Rectified Linear Unit) layer hay còn gọi là hàm activation (hàm kích hoạt), hàm này thường được mô tả như một cách để kích hoạt neuron trong mạng. ReLU layer thực hiện vai trò này bằng cách xử lý giá trị đầu vào và sinh ra một đầu ra tương ứng.

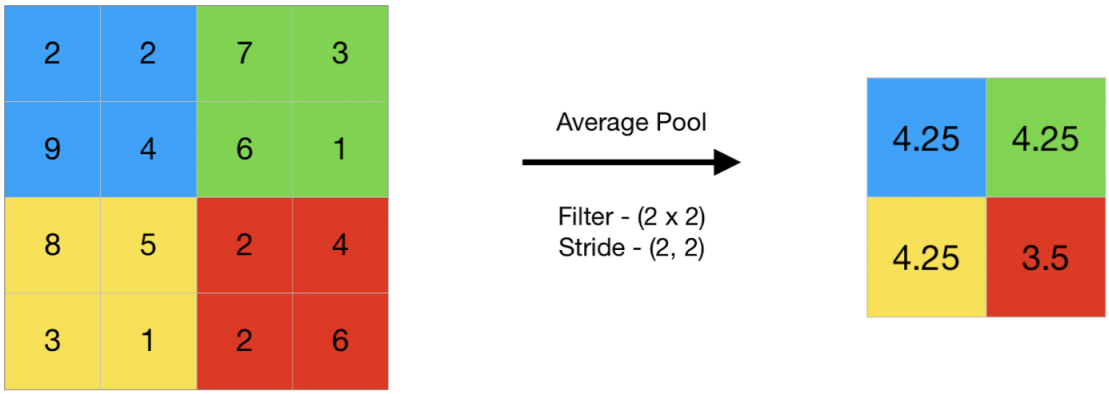
- ReLU layer thực hiện một phép toán đơn giản là chuyển đổi tất cả các giá trị âm thành 0 và giữ nguyên các giá trị dương.
- Mục tiêu của ReLU là giữ lại sự phi tuyến tính trong mạng nơ-ron, giúp nó học các đặc trưng phức tạp hơn.

2.3.3 Pooling layer

Khi nhận phải đầu vào quá lớn, các lớp pooling layer sẽ được xếp giữa những lớp Convolutional layer nhằm mục đích giảm parameter. Pooling layer được chia thành 2 loại phổ biến là max pooling và average.



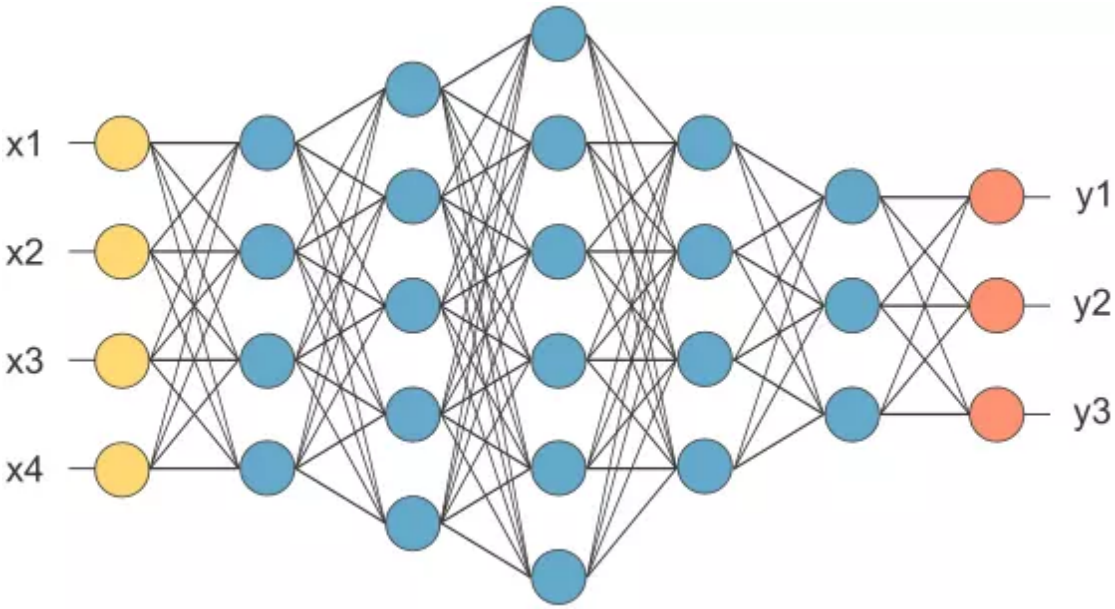
Hình 3: Pooling Layer - Max Pooling



Hình 4: Pooling Layer - Average

2.3.4 Fully connected layer

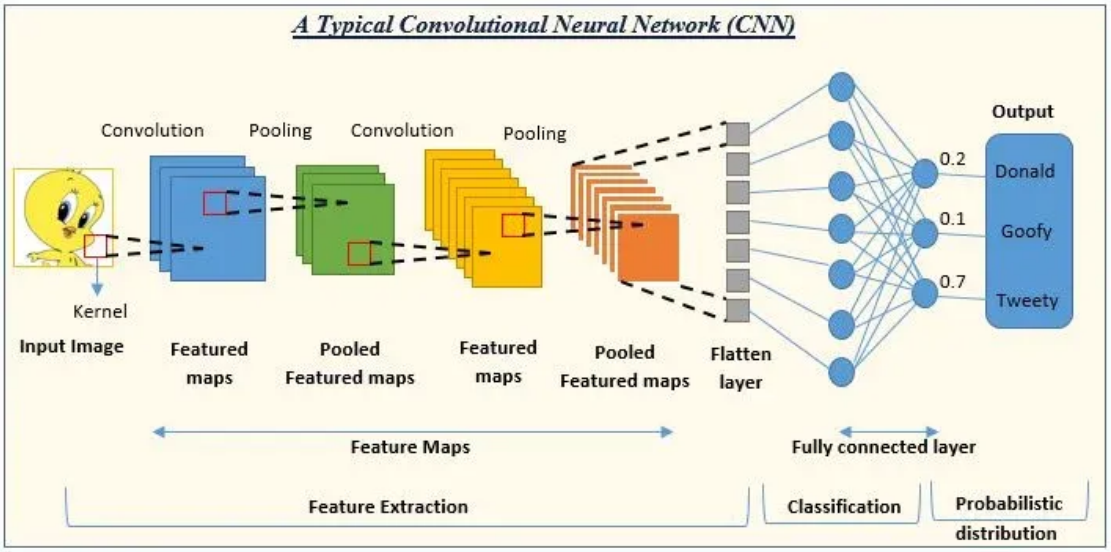
Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh, tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron. Với fully connected layer được kết hợp các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra (softmax và sigmoid là hai hàm kích hoạt thường dùng để chuyển đổi giá trị đầu ra của một lớp thành xác suất).



Hình 5: Fully connected layer

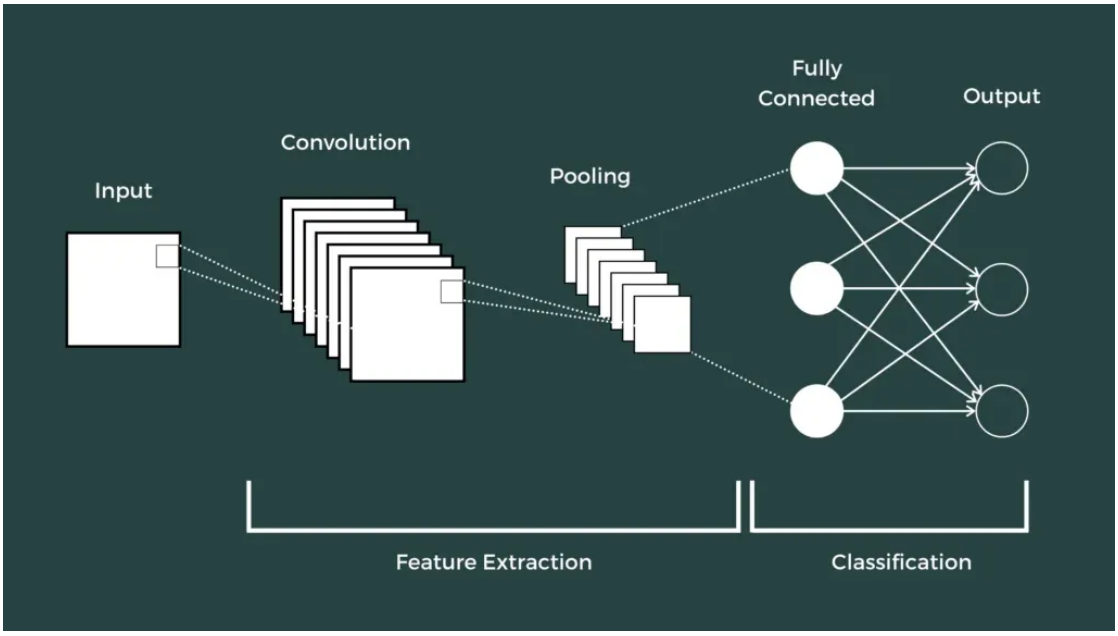
3 Cấu trúc của mạng CNN:

- Mạng CNN là một trong những tập hợp của lớp Convolution được chồng lên nhau. Mạng CNN còn sử dụng các hàm nonlinear activation (như ReLU và tanh) nhằm kích hoạt trọng số trong node. Khi đã thông qua hàm, lớp này sẽ thu được trọng số trong các node và tạo ra nhiều thông tin trừu tượng hơn cho các lớp kế cận.
- Đặc điểm mô hình CNN có 2 khía cạnh cần phải đặc biệt lưu ý là tính bất biến và tính kết hợp, do đó độ chính xác hoàn toàn có thể bị ảnh hưởng nếu có cùng một đối tượng được chiếu theo nhiều phương diện khác biệt. Với các loại chuyển dịch, co giãn và quay, người ta sẽ sử dụng pooli layer và làm bất biến những tính chất này. Từ đó, CNN sẽ cho ra kết quả có độ chính xác ứng với từng loại mô hình.
- Pooling layer giúp tạo nên tính bất biến đối với phép dịch chuyển, phép co giãn và phép quay. Trong khi đó, tính kết hợp cục bộ sẽ thể hiện các cấp độ biểu diễn, thông tin từ mức độ thấp đến cao, cùng độ trừu tượng thông qua convolution từ các filter. Dựa trên cơ chế convolution, một mô hình sẽ liên kết được các layer với nhau.
- Với cơ chế này, layer tiếp theo sẽ là kết quả được tạo ra từ convolution thuộc layer kế trước. Điều này đảm bảo bạn có được kết nối cục bộ hiệu quả nhất. Mỗi nơ-ron sinh ra ở lớp tiếp theo từ kết quả filter sẽ áp đặt lên vùng ảnh cục bộ của nơ-ron tương ứng trước đó. Cũng có một số layer khác như pooling/subsampling layer được dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).
- Suốt quá trình huấn luyện, CNN sẽ tự động học hỏi các giá trị thông qua lớp filter với model là cách thức người dùng thực hiện. Điều này khá giống với cách bộ não con người nhận diện những vật thể trong tự nhiên.



Hình 6: Cấu trúc của mạng CNN

- Một cấu trúc cơ bản nhất của CNN sẽ bao gồm 3 phần chủ yếu:
  - Local receptive field (trường cục bộ): Nhiệm vụ của trường cục bộ là phân tách và lọc dữ liệu cũng như thông tin ảnh, sau đó chọn ra các vùng ảnh có giá trị sử dụng cao nhất.
  - Shared weights and bias (trọng số chia sẻ): Trong mạng CNN, thành phần này có tác dụng giảm thiểu tối đa lượng tham số có tác dụng lớn. Trong mỗi convolution sẽ chứa nhiều feature map khác nhau, mỗi feature lại có khả năng giúp nhận diện một số feature trong ảnh.
  - Pooling layer: Pooling layer là lớp cuối cùng, với khả năng đơn giản hóa thông tin đầu ra. Khi đã hoàn tất tính toán và quét qua các lớp, pooling layer sẽ được tạo ra nhằm mục đích lược bớt các thông tin không cần thiết và tối ưu đầu ra. Điều này giúp người dùng nhận được kết quả đúng với yêu cầu hay mong muốn.



Hình 7: CNN được ứng dụng rất rộng rãi

4 Thư viện Keras:

- Keras là một thư viện mã nguồn mở được sử dụng rộng rãi trong lĩnh vực deep learning (học sâu) và mạng nơ-ron. Nó được thiết kế để giúp các nhà phát triển xây dựng và thử nghiệm các mô hình học sâu một cách dễ dàng và linh hoạt.
- Mục tiêu chính của Keras là cung cấp một API đơn giản, trực quan và dễ sử dụng, giúp người dùng tập trung vào việc xây dựng mô hình mà không cần lo lắng về chi tiết kỹ thuật phức tạp.
- Một trong những điểm mạnh của Keras là khả năng tích hợp với các framework học sâu khác như TensorFlow và Theano, cho phép người dùng tận dụng sức mạnh của cả hai thế giới. Đây chính là điều giúp Keras trở thành một lựa chọn hàng đầu cho các dự án học sâu từ phân loại ảnh, dự đoán chuỗi thời gian, đến các ứng dụng trong lĩnh vực ngôn ngữ tự nhiên và nhiều lĩnh vực khác.

5 Xây dựng ứng dụng:

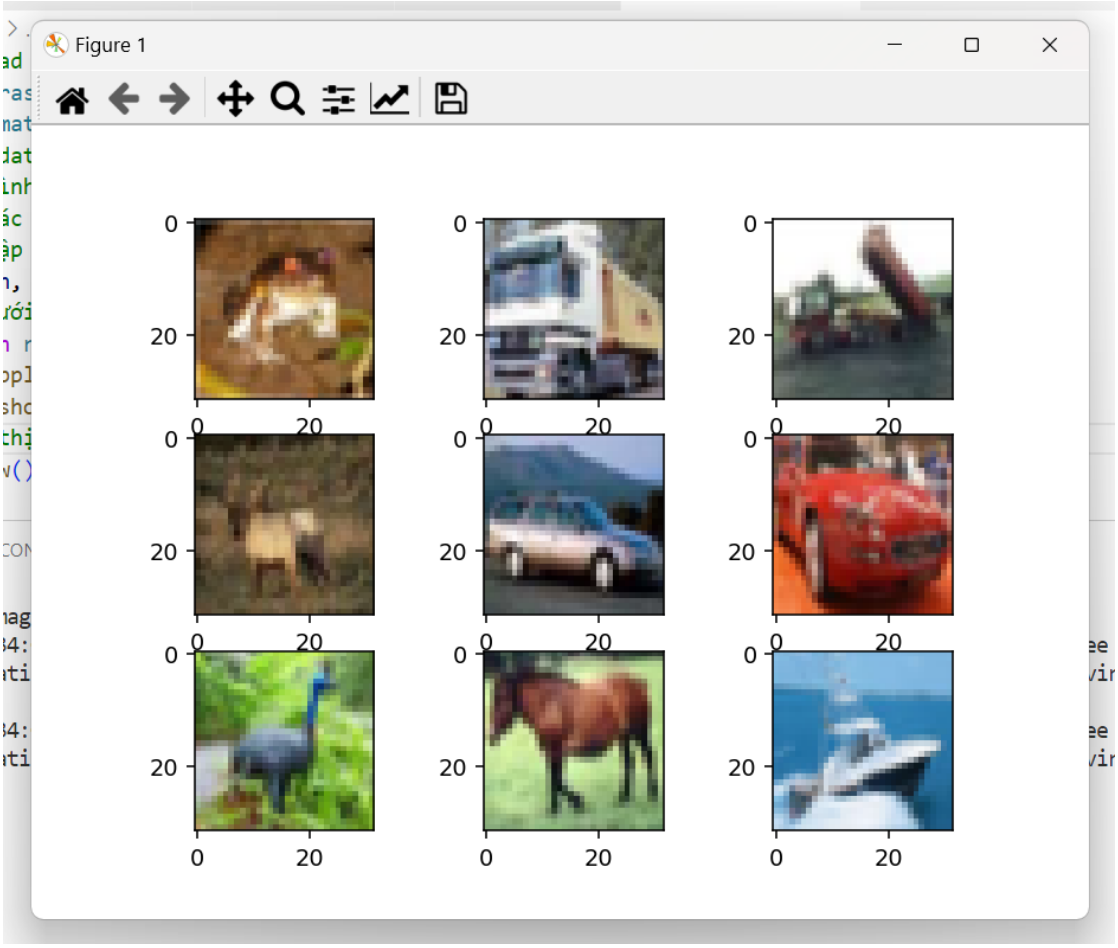
5.1 Bộ dữ liệu CIFAR-10 trong Keras

- Keras tự động tải xuống các bộ dữ liệu tiêu chuẩn như CIFAR-10 và lưu trữ chúng trong thư mục /.keras/datasets bằng hàm cifar10.load_data().
- Tập dữ liệu được lưu trữ dưới dạng tập huấn luyện và tập kiểm tra đã được chọn lọc, sẵn sàng để sử dụng trong Keras. Mỗi hình ảnh được biểu diễn dưới dạng ma trận ba chiều, với các kích thước là đỏ, lục, lam, chiều rộng và chiều cao. Chúng ta có thể vẽ hình ảnh trực tiếp bằng matplotlib.

```
# Plot ad hoc CIFAR10 instances
from keras.datasets import cifar10
import matplotlib.pyplot as plt
# load data
# Mỗi hình ảnh được biểu diễn dưới dạng ma trận ba chiều,
# với các kích thước là đỏ, lục, lam, chiều rộng và chiều cao
# tải tập dữ liệu lên 2 mảng train và test
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
# Tạo lưới hình ảnh 3x3
for i in range(0, 9):
 plt.subplot(330 + 1 + i)
 plt.imshow(X_train[i])
# hiển thị hình ảnh
plt.show()
```

Hình 8: Tải dữ liệu CIFAR-10

- Việc chạy mã sẽ tạo ra một ô ảnh 3×3. Các hình ảnh đã được phóng to từ kích thước nhỏ 32×32 nhưng bạn có thể thấy rõ xe tải, ngựa và ô tô. Bạn cũng có thể thấy một số biến dạng ở một số hình ảnh bị ép ở tỷ lệ khung hình vuông.



Hình 9: Mẫu nhỏ hình ảnh CIFAR-10

5.2 Mô hình CNN trong CIFAR-10:

- Vấn đề CIFAR-10 được giải quyết tốt nhất bằng cách sử dụng mạng nơ ron tích chập (CNN).
- Các lớp và hàm cần thiết:

```
# CNN model for the CIFAR-10 Dataset
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import MaxNorm
from keras.optimizers import SGD
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.utils import to_categorical
```

Hình 10: Các lớp và hàm trong keras cần sử dụng

- Tải tập dữ liệu CIFAR-10:

```
# load data
# Các giá trị pixel nằm trong khoảng từ 0 đến 255
# cho mỗi kênh màu đỏ, xanh lục và xanh lam.
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

Hình 11: Tải tập dữ liệu CIFAR-10

- Các giá trị đầu vào được hiểu rõ, vì vậy có thể dễ dàng chuẩn hóa về khoảng từ 0 đến 1 bằng cách chia mỗi giá trị cho giá trị quan sát tối đa, tức là 255.
- Các biến đầu ra được định nghĩa là một vectơ số nguyên từ 0 đến 1 cho mỗi lớp.

```
# normalize inputs from 0-255 to 0.0-1.0
# dữ liệu được tải dưới dạng int
# => phải chuyển nó sang float để thực hiện phép chia.
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
```

Hình 12: Chuẩn hóa giá trị đầu vào về khoảng từ 0 đến 1

- Sử dụng mã hóa one-hot để chuyển đổi chúng thành một ma trận nhị phân để phân loại mô hình tốt nhất. Với vấn đề này, có mười lớp, vì vậy ta mong đợi ma trận nhị phân có độ rộng là 10.

```
# chuyển đổi về dạng one-hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]
```

Hình 13: Chuyển đổi về ma trận nhị phân

- Sử dụng một cấu trúc với hai lớp convolutional, tiếp theo là max pooling và làm phẳng mạng ra thành các lớp fully connected để thực hiện dự đoán.
- Tuy nhiên, để cải thiện độ chính xác của mô hình, chúng ta sẽ tạo một mạng lưới sâu hơn, mẫu này sẽ được lặp lại ba lần với 32, 64 và 128 feature maps. Hiệu ứng là số lượng feature maps tăng dần với kích thước nhỏ hơn do sự xuất hiện của các lớp max pooling. Cuối cùng, một lớp Dense lớn hơn sẽ được sử dụng ở đầu ra của mạng trong một nỗ lực để chuyển đổi tốt hơn số lượng lớn feature maps thành các giá trị lớp.
- Cấu trúc mạng có thể được tóm tắt như sau:

```
# Tạo model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3)
                                    , activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D())
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D())
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(learning_rate=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.summary()
```

Hình 14: Tạo và biên dịch một mô hình mạng neural convolutional (CNN)

- Bạn có thể huấn luyện mô hình này với 25 vòng lặp (epochs) và kích thước batch là 32.
- Sau khi mô hình được huấn luyện, ta đánh giá nó trên tập dữ liệu kiểm tra và in ra độ chính xác phân loại.

```
# Huấn luyện mô hình
model.fit(X_train, y_train, validation_data=(X_test, y_test),
          epochs=epochs, batch_size=32)
model.save("model1_cifar_10epoch.h5")
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Hình 15: Các lớp và hàm trong keras cần sử dụng

→ Kết quả, cấu trúc mạng được tóm tắt để xác nhận rằng thiết kế đã được triển khai chính xác.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
dropout (Dropout)	(None, 32, 32, 32)	0
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18,496
dropout_1 (Dropout)	(None, 16, 16, 64)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_4 (Conv2D)	(None, 8, 8, 128)	73,856
dropout_2 (Dropout)	(None, 8, 8, 128)	0
conv2d_5 (Conv2D)	(None, 8, 8, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dropout_3 (Dropout)	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dropout_4 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524,800
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5,130

Total params: 2,915,114 (11.12 MB)
Trainable params: 2,915,114 (11.12 MB)
Non-trainable params: 0 (0.00 B)

Hình 16: Tổng quan về kiến trúc của mô hình sau khi compile

- Độ chính xác phân loại và tổn thất được in ra sau mỗi vòng lặp trên cả tập dữ liệu huấn luyện và kiểm tra.

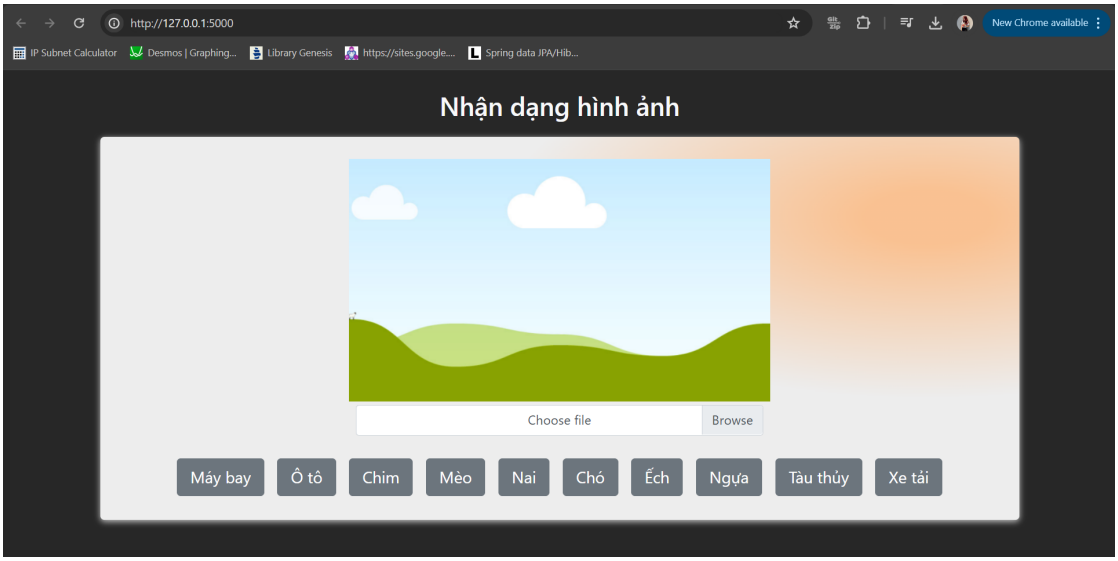
```
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 188s 86ms/step - accuracy: 0.7638 - loss: 0.6703 - val_accuracy: 0.7525 - val_loss: 0.7214
Epoch 10/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 131s 84ms/step - accuracy: 0.7781 - loss: 0.6352 - val_accuracy: 0.7428 - val_loss: 0.7416
Epoch 11/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 129s 82ms/step - accuracy: 0.7893 - loss: 0.5980 - val_accuracy: 0.7616 - val_loss: 0.6961
Epoch 12/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 135s 86ms/step - accuracy: 0.7930 - loss: 0.5871 - val_accuracy: 0.7779 - val_loss: 0.6654
Epoch 13/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 175s 107ms/step - accuracy: 0.8040 - loss: 0.5573 - val_accuracy: 0.7683 - val_loss: 0.6750
Epoch 14/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 177s 113ms/step - accuracy: 0.8102 - loss: 0.5400 - val_accuracy: 0.7650 - val_loss: 0.6956
Epoch 15/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 140s 90ms/step - accuracy: 0.8181 - loss: 0.5288 - val_accuracy: 0.7651 - val_loss: 0.6877
Epoch 16/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 136s 87ms/step - accuracy: 0.8146 - loss: 0.5347 - val_accuracy: 0.7661 - val_loss: 0.6904
Epoch 17/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 130s 83ms/step - accuracy: 0.8196 - loss: 0.5185 - val_accuracy: 0.7638 - val_loss: 0.6974
Epoch 18/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 130s 83ms/step - accuracy: 0.8221 - loss: 0.5180 - val_accuracy: 0.7640 - val_loss: 0.6977
Epoch 19/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 132s 84ms/step - accuracy: 0.8271 - loss: 0.5050 - val_accuracy: 0.7545 - val_loss: 0.7333
Epoch 20/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 140s 83ms/step - accuracy: 0.8234 - loss: 0.5111 - val_accuracy: 0.7710 - val_loss: 0.6858
Epoch 21/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 143s 83ms/step - accuracy: 0.8208 - loss: 0.5143 - val_accuracy: 0.7463 - val_loss: 0.7465
Epoch 22/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 131s 84ms/step - accuracy: 0.8256 - loss: 0.5061 - val_accuracy: 0.7358 - val_loss: 0.7856
Epoch 23/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 130s 83ms/step - accuracy: 0.8249 - loss: 0.5070 - val_accuracy: 0.7578 - val_loss: 0.7357
Epoch 24/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 143s 83ms/step - accuracy: 0.8234 - loss: 0.5167 - val_accuracy: 0.7706 - val_loss: 0.6906
Epoch 25/25
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 133s 85ms/step - accuracy: 0.8230 - loss: 0.5193 - val_accuracy: 0.7563 - val_loss: 0.7391
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file f
e Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
Accuracy: 75.63%
```

Hình 17: Kết quả hiệu suất mô hình sau khi compile

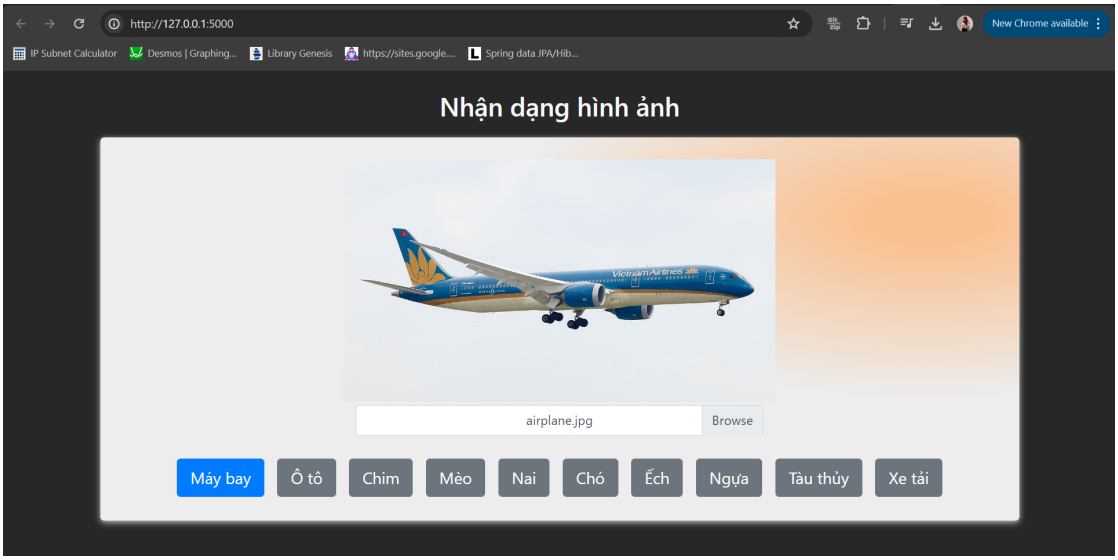
→ Ước tính độ chính xác phân loại cho mô hình cuối cùng là 75.63%.

5.3 Kết quả xây dựng website:

- Giao diện website nhận dạng hình ảnh:



- Giao diện kết quả nhận dạng hình ảnh tải lên:



Tài liệu

[1] Trần Đức Trung. (2020). Tìm hiểu về Convolutional Neural Network và làm một ví dụ nhỏ về phân loại ảnh. Truy cập từ: "**link: https://viblo.asia/p/tim-hieu-ve-convolutional-neural-network-va-lam-mot-vi-du-nho-ve-phan-loai-anh**", lần truy cập cuối: 10/04/2024.

[2] Nguyễn Thanh Tuấn. (2019). Bài 7: Giới thiệu keras và bài toán phân loại ảnh. Truy cập từ: "**link: https://nttuan8.com/bai-7-gioi-thieu-keras-va-bai-toan-phan-loai-anh**", lần truy cập cuối: 11/04/2024.

[3] Nguyễn Thanh Tuấn. (2019). Học sâu với Phân loại hình ảnh CIFAR-10. Truy cập từ: "**link: https://ichi.pro/vi/hoc-sau-voi-phan-loai-hinh-anh-cifar-10**", lần truy cập cuối: 11/04/2024.