TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

MÔN HỌC: TRÍ TUỆ NHÂN TẠO

BÁO CÁO CUỐI KÌ

Ngành: Cơ Điện Tử

GVHD: PGS.TS Nguyễn Trường Thịnh

SVTH: Nguyễn Quốc Huy

MSSV: 20146341

Lớp: ST6

Thành phố Hồ Chí Minh, tháng 5 năm 2023

I. LÝ THUYẾT

1.Tìm hiểu về cấu trúc mạng CNN là gì?

Định nghĩa CNN là gì?

CNN là tên viết tắt của từ Convolutional Neural Network (hay còn gọi là CNNs_mang nơ ron tích chập). Đây là một trong những mô hình Deep Learning vô cùng tiên tiến. CNN sẽ cho phép bạn xây dựng các hệ thống thông minh với độ chính xác vô cùng cao. Hiện nay, CNN được ứng dụng rất nhiều trong những bài toán nhận dạng object trong ảnh. Và kiến thức cụ thể về CNN đã được lý giải như sau:

Convolutional

Đây là một loại cửa sổ dạng trượt nằm trên một ma trận. Những convolutional layer sẽ có các parameter được học để điều chỉnh và lấy ra những thông tin chính xác nhất mà không cần phải chọn feature. Convolution hay tích chập chính là nhân các phần tử trong ma trận. Sliding Window còn được gọi là kernel, filter hoặc feature detect và là loại ma trận có kích thước nhỏ.

Feature

Feature là đặc điểm, các CNN sẽ so sánh hình ảnh dựa theo từng mảnh và những mảnh này được gọi là Feature. Thay vì phải khớp các bức ảnh lại với nhau thì CNN sẽ nhìn ra sự tương động khi tìm kiếm thô các Feature khớp với nhau bằng 2 hình ảnh tốt hơn. Mỗi Feature được xem là một hình ảnh mini có nghĩa chúng là những mảng 2 chiều nhỏ. Các Feature này đều tương ứng với các khía cạnh nào đó của hình ảnh và chúng có thể khớp lại với nhau.

2. Các lớp cơ bản của mạng CNN là gì?

CNN bao gồm những phần lớp cơ bản là:

Convolutional layer

Đây là lớp quan trọng nhất của CNN, lớp này có nhiệm vụ thực hiện mọi tính toán. Những yếu tố quan trọng của một convolutional layer là: stride, padding, filter map, feature map.

+ CNN sử dụng các filter để áp dụng vào vùng của hình ảnh. Những filter map này được gọi là ma trận 3 chiều, mà bên trong nó là các con số và chúng là parameter.

+ Stride có nghĩa là khi bạn dịch chuyển filter map theo pixel dựa vào giá trị trừ trái sang phải. Và sự chuyển dịch này chính là Stride.

+ Padding: Là các giá trị 0 được thêm vào với lớp input.

+ Feature map: Nó thể hiện kết quả của mỗi lần filter map quét qua input. Sau mỗi lần quét sẽ xảy ra quá trình tính toán.

Relu Layer

Relu layer là hàm kích hoạt trong neural network và hàm này còn được gọi là activation function. Hàm kích hoạt có tác dụng mô phỏng các neuron có tỷ lệ truyền xung qua axon. Trong activation function thì nó còn có hàm nghĩa là: Relu, Leaky, Tanh, Sigmoid, Maxout,...Hiện nay, hàm relu được dùng phổ biến và vô cùng thông dụng.

Nó được sử dụng nhiều cho các nhu cầu huấn luyện mạng neuron thì relu mang lại rất nhiều ưu điểm nổi bật như: việc tính toán sẽ trở nên nhanh hơn,... Quá trình sử dụng relu, chúng ta cần lưu ý đến vấn đề tùy chỉnh các learning rate và theo dõi dead unit. Những lớp relu layer đã được sử dụng sau khi filter map được tính ra và áp dụng hàm relu lên những giá trị của filter map.

Pooling layer

Khi đầu vào quá lớn, những lớp pooling layer sẽ được xếp vào giữa giữa những lớp Convolutional layer để làm giảm parameter. Hiện nay, pooling layer có 2 loại chủ yếu là: max pooling và average.

Fully connected layer

Lớp này có nhiệm vụ đưa ra kết quả sau khi lớp convolutional layer và pooling layer đã nhận được ảnh truyền. Lúc này, ta thu được kết quả là model đã đọc được thông tin của ảnh và để liên kết chúng cũng như cho ra nhiều output hơn thì ta sử dụng fully connected layer.

Ngoài ra, nếu như fully connected layer có được giữ liệu hình ảnh thì chúng sẽ chuyển nó thành mục chưa được phân chia chất lượng. Cái này khá giống với phiếu bầu rồi chúng sẽ đánh giá để bầu chọn ra hình ảnh có chất lượng cao nhất.

3.Cấu trúc của mạng CNN là gì?

Mang CNN là một trong những tập hợp của lớp Convolution bị chồng lên nhau cũng như sử dụng hàm nonlinear activation như ReLU và tanh để kích hoạt trọng số trong node. Lớp này sau khi thông qua hàm thì sẽ được trọng số trong các node. Những lớp này sau khi đã thông qua hàm kích hoạt thì có thể tạo ra những thông tin trừu tượng hơn cho những lớp tiếp theo.

Trong mô hình CNN có tính bất biến và tích kết hợp. Nếu như bạn có cùng một đối tượng mà lại chiếu theo nhiều góc độ khác nhau thì độ chính xác có thể sẽ bị ảnh hưởng. Với chuyển dịch, quay và co giãn thì pooling layer sẽ được sử dụng để giúp làm bất biến những tính chất này. Vì vậy, CNN sẽ đưa ra kết quả có độ chính xác tương ứng ở từng mô hình.

Trong đó, pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển, phép co dãn và phép quay. Còn tính kết hợp cục bộ sẽ cho bạn thấy những cấp độ biểu diễn, thông tin từ thấp đến mức độ cao với độ trừu tượng thông qua convolution từ các filter. Mô hình CNN có các layer liên kết được với nhau dựa vào cơ chế convolution.

Những layer tiếp theo sẽ là kết quả từ những convolution từ layer trước đó, vì thế mà bạn sẽ có các kết nối cục bộ phù hợp nhất. Vậy, mỗi neuron ở lớp sinh ra tiếp theo từ kết quả filter sẽ áp đặt lên vùng ảnh cục bộ của một neuron có trước đó. Trong khi huấn luyện mạng, CNN sẽ tự động học hỏi các giá trị thông qua lớp filter dựa vào cách thức mà người dùng thực hiện.

II. ĐỀ TÀI : ANIMAL CLASSIFICATION WITH NEURAL NETWORKS

1. Giới thiệu

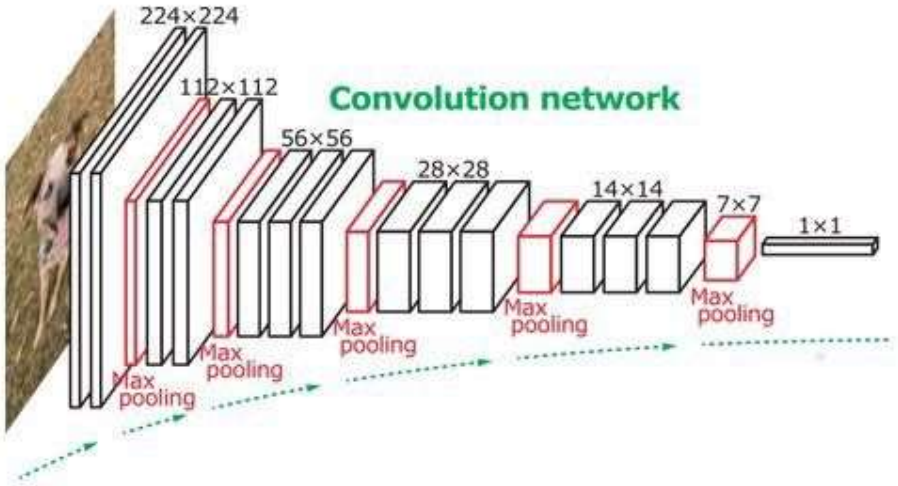
Xây dựng mô hình học máy sử dụng CNN để nhận diện các loài động vật cụ thể với các sinh vật biển . Với yêu cầu đề bài thì em sử dụng hình ảnh các loài sinh vật biển khác nhau để dự đoán là: Bạch Tuộc, Cá, Cá Heo, Cá Mập, Cá Ngựa, Cá Voi, Cua, Rái Cá, Sao Biển và Tôm Biển . Thu thập dữ liệu bằng cách lấy ảnh trên google.

Các tập dữ liệu thu thập:

+ train4 : 80%
+ test4 : 10%

2. Phương pháp luận

Bài toán sử dụng mô hình CNN huấn luyện tập dữ liệu để nhận diện các loài sinh vật biển . Mạng nơron tích tụ CNN là một loại mạng nơron nhân tạo sử dụng chủ yếu để nhận dạng và xử lý ảnh do khả năng nhận dạng các mẫu trong ảnh.



3. Thực hiện

Để mà thực hiện thì chúng ta phải thu thập hình ảnh các loài sinh vật các loài động vật biển . Rồi đưa hình ảnh vào các folder là: train4, test4. Sau đó up Google driver. Rồi vào google colab thực hiện các bước sau:

3.1 Khai báo các thư viện cần sử dụng

```
#importing libraries

from os import listdir
from numpy import asarray, save
from keras.utils import load_img
from keras.utils import img_to_array
```

3.2 Kết nối Google Driver để đọc và lưu dữ liệu

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

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

3.3 Đọc dữ liệu train và gán nhán dữ liệu

Lưu ý là phải copy đúng đường dẫn trên google drive để đọc được dữ liệu chính xác.

```
# đoạn mã này được sử dụng để tải các tệp ảnh và nhãn tương ứng từ thư mục trên Google Drive
# chuẩn bị chúng cho việc huấn luyện mô hình học máy.
folder = '/content/drive/MyDrive/PROJECT_CUOI_KI_AI/SEA_ANIMAL/'
photos, labels = list(), list()
for file in listdir(folder):
  output= 0.0
  if file.startswith('Bach_tuoc'):
    output= 1.0
  if file.startswith('Ca'):
    output= 2.0
  if file.startswith('Ca_heo'):
    output= 3.0
  if file.startswith('Ca_map'):
    output= 4.0
  if file.startswith('Ca_ngua'):
    output= 5.0
  if file.startswith('Ca_voi'):
    output= 6.0
  if file.startswith('cua'):
    output= 7.0
  if file.startswith('Rai_ca'):
    output= 8.0
  if file.startswith('Sao_bien'):
    output= 9.0
  if file.startswith('Tom'):
    output= 10.0
  photo = load_img(folder + file)
  photo= img_to_array(photo)

  photo = load_img(folder + file, target_size= (40,40))
  photo= img_to_array(photo)

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

3.4 Lưu trữ các mảng numpy và labels

```
# lưu trữ các mảng numpy photos và labels
photos= asarray(photos)
labels= asarray(labels)
print(photos.shape, labels.shape)
save('/content/drive/MyDrive/PROJECT_CUOI_KI_AI/ANIMAL/ANIMAL_photos.npy', photos)
save('/content/drive/MyDrive/PROJECT_CUOI_KI_AI/ANIMAL/ANIMAL_labels.npy', labels)
```

```
(5455, 40, 40, 3) (5455,)
```

3.5 Xây dựng mô hình

```
from keras.layers import LeakyReLU

model = Sequential()

model.add(Conv2D(32, kernel_size = (3,3), activation = 'linear', input_shape = (40, 40, 3), padding= 'same')) # tích chập 32 lần mỗi lần 3 hà
model.add(LeakyReLU(alpha = 0.1))
model.add(MaxPooling2D((2,2), padding = 'same'))

model.add(Conv2D(64, (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(128, (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(256, (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(512, (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(1024, (3,3), activation = 'linear', input_shape = (40, 40, 3), padding = 'same'))
model.add(LeakyReLU(alpha = 0.1))
model.add(MaxPooling2D((2,2), padding = 'same'))
```

3.6 Đưa vào bộ ANN, bộ ANN để phân loại

```
#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(LeakyReLU(alpha = 0.1))
model.add(Dense(classes, activation = 'softmax'))

model.summary()
```

3.7 Thiết lập lập thông số và huấn luyện mô hình

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

```
Epoch 1/100
81/81 [==============================] - 7s 40ms/step - loss: 1.9371 - accuracy: 0.2431 - val_loss: 1.7737 - val_accuracy: 0.3326
Epoch 2/100
81/81 [==============================] - 2s 25ms/step - loss: 1.6742 - accuracy: 0.3486 - val_loss: 1.5553 - val_accuracy: 0.3679
Epoch 3/100
81/81 [==============================] - 2s 23ms/step - loss: 1.5452 - accuracy: 0.3978 - val_loss: 1.4326 - val_accuracy: 0.4495
Epoch 4/100
81/81 [==============================] - 2s 21ms/step - loss: 1.4144 - accuracy: 0.4505 - val_loss: 1.4019 - val_accuracy: 0.4495
Epoch 5/100
81/81 [==============================] - 2s 21ms/step - loss: 1.3838 - accuracy: 0.4746 - val_loss: 1.2191 - val_accuracy: 0.5256
Epoch 6/100
81/81 [==============================] - 2s 21ms/step - loss: 1.2440 - accuracy: 0.5165 - val_loss: 1.1575 - val_accuracy: 0.5356
Epoch 7/100
81/81 [==============================] - 2s 24ms/step - loss: 1.1215 - accuracy: 0.5721 - val_loss: 0.9530 - val_accuracy: 0.6336
Epoch 8/100
81/81 [==============================] - 2s 23ms/step - loss: 1.0253 - accuracy: 0.6110 - val_loss: 0.8657 - val_accuracy: 0.6689
Epoch 9/100
81/81 [==============================] - 2s 25ms/step - loss: 0.9208 - accuracy: 0.6431 - val_loss: 0.8045 - val_accuracy: 0.6923
Epoch 10/100
81/81 [==============================] - 2s 21ms/step - loss: 0.8115 - accuracy: 0.6838 - val_loss: 0.7194 - val_accuracy: 0.7164
Epoch 11/100
81/81 [==============================] - 2s 21ms/step - loss: 0.6803 - accuracy: 0.7380 - val_loss: 0.5027 - val_accuracy: 0.8125
Epoch 12/100
81/81 [==============================] - 2s 21ms/step - loss: 0.5067 - accuracy: 0.8038 - val_loss: 0.3990 - val_accuracy: 0.8509
Epoch 13/100
81/81 [==============================] - 2s 24ms/step - loss: 0.4591 - accuracy: 0.8254 - val_loss: 0.5105 - val_accuracy: 0.8061
Epoch 14/100
81/81 [==============================] - 2s 24ms/step - loss: 0.3800 - accuracy: 0.8557 - val_loss: 0.2813 - val_accuracy: 0.8881
Epoch 15/100
81/81 [==============================] - 2s 24ms/step - loss: 0.3407 - accuracy: 0.8705 - val_loss: 0.3015 - val_accuracy: 0.8852
```

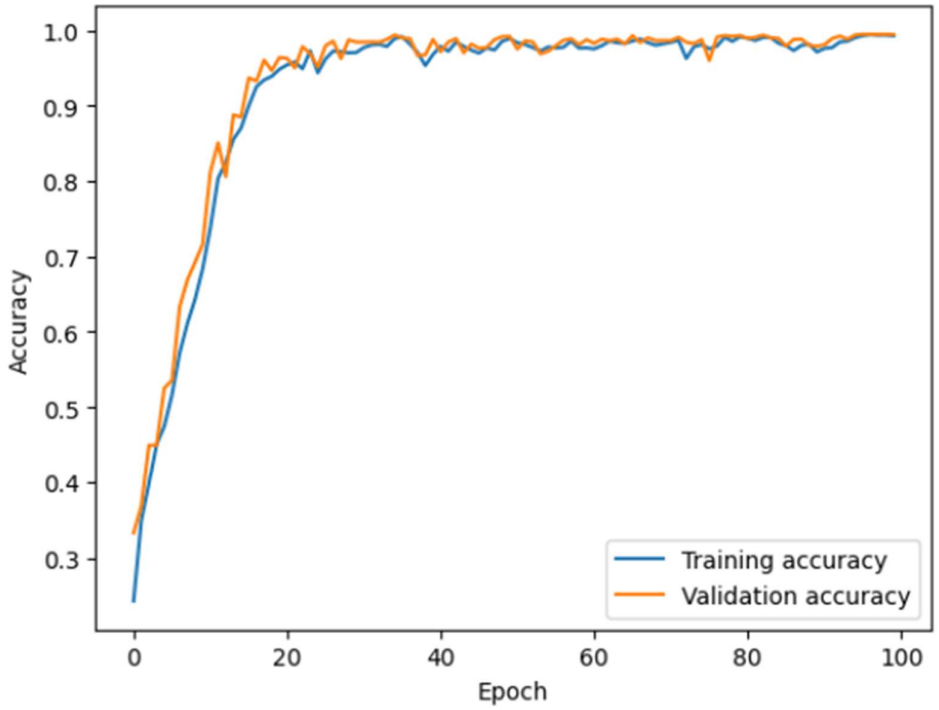
3.8 Vẽ đồ thị sau khi huấn luyện mô hình

Vẽ đồ thị để chúng ta có thể đánh giá rõ được độ chính xác của mô hình

```
#Đoạn code này đánh giá độ chính xác của mô hình
test_loss, test_acc = model.evaluate(test_x, test_y)
print('Test accuracy:', test_acc)
```

```
9/9 [==============================] - 0s 7ms/step - loss: 5.1223 - accuracy: 0.5110
Test accuracy: 0.5110294222831726
```

```
# plot the training and validation accuracy over epochs
import matplotlib.pyplot as plt
plt.plot(train.history['accuracy'], label='Training accuracy')
plt.plot(train.history['val_accuracy'], label = 'Validation accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



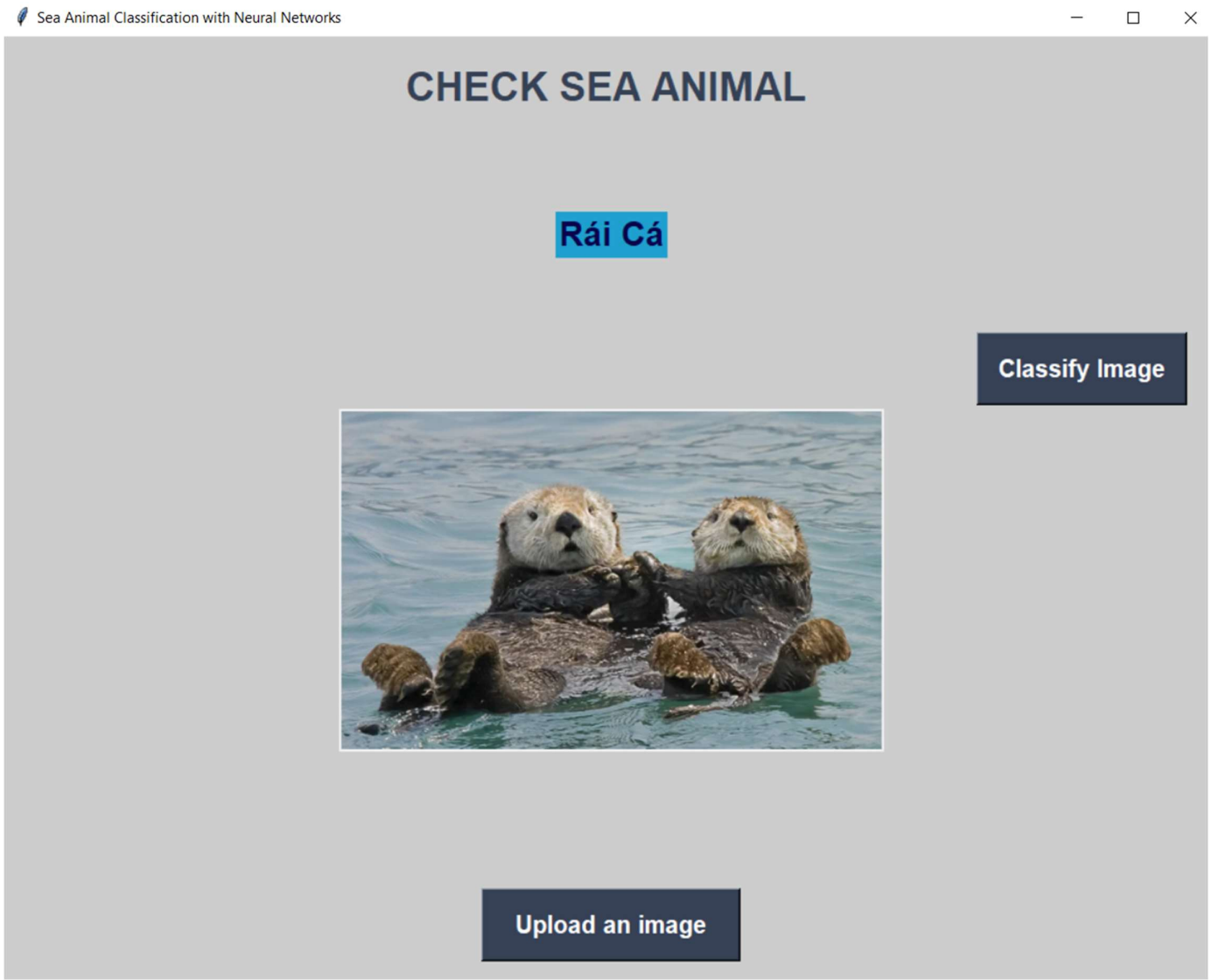
3.9 Lưu mô hình đã huấn luyện

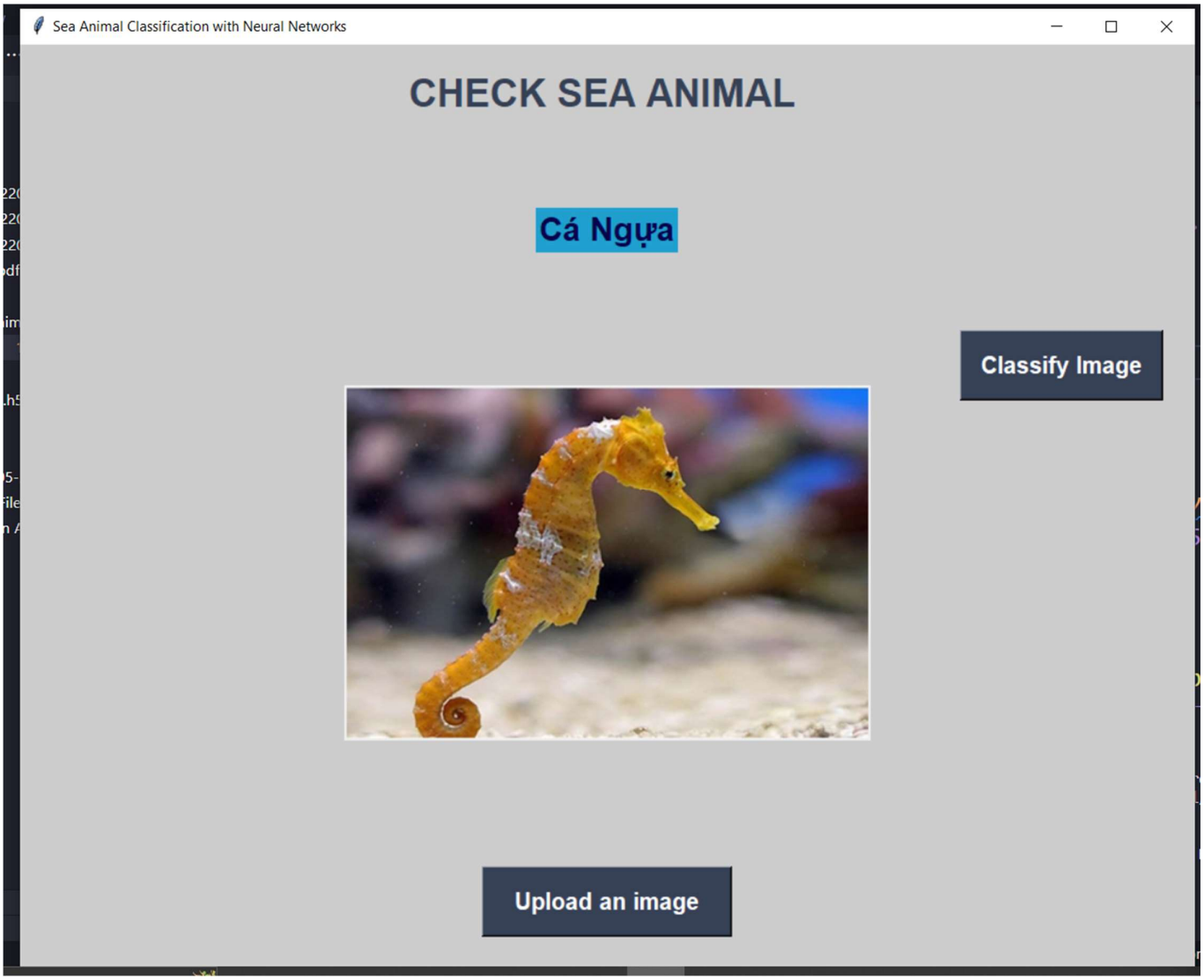
```
model.save('/content/drive/MyDrive/PROJECT_CUOI_KI_AI/model4_animal.h5')
```

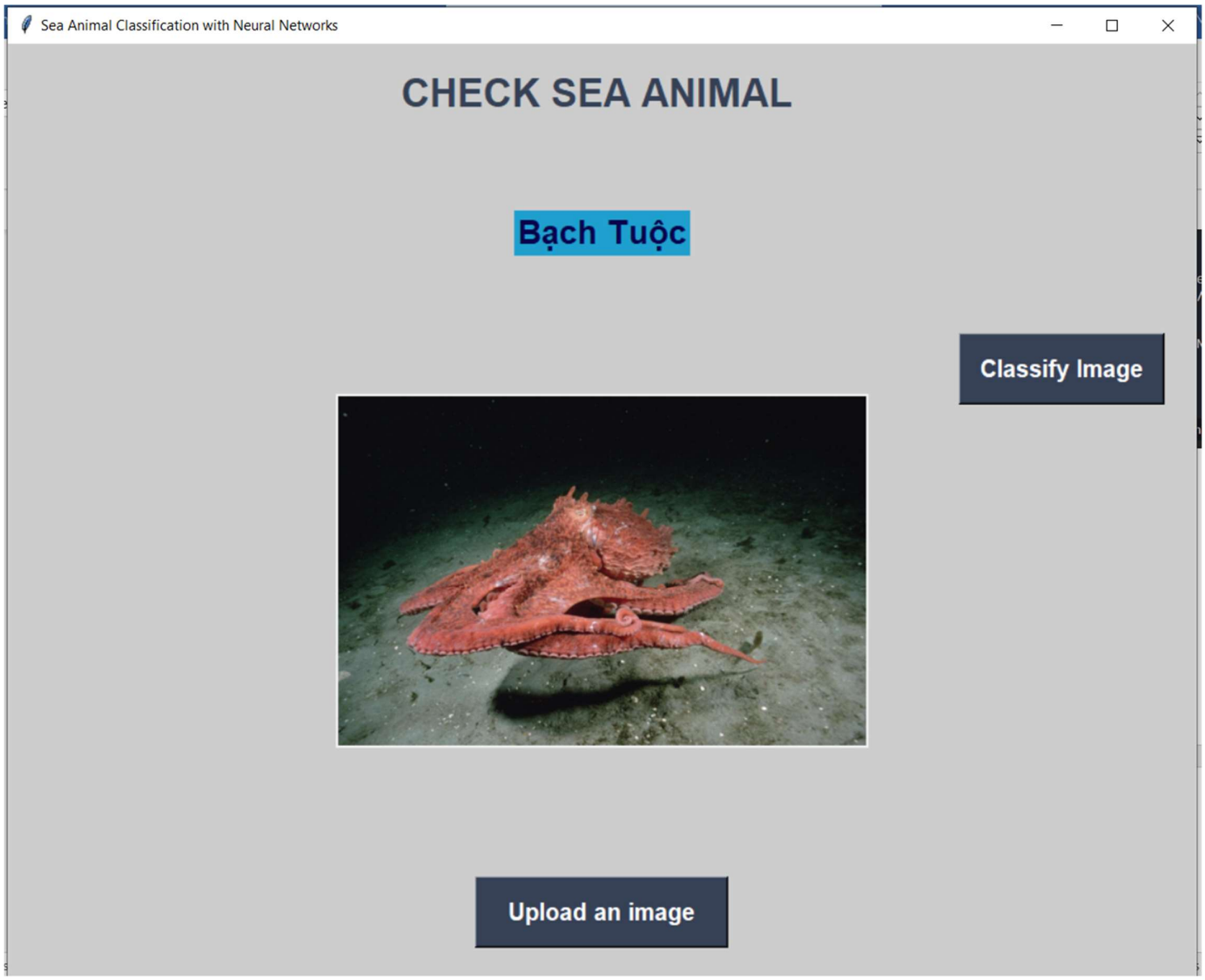
```
from tensorflow import keras
model = keras.models.load_model('/content/drive/MyDrive/PROJECT_CUOI_KI_AI/model3_animal.h5/')
```

3.10 Sử dụng mô hình

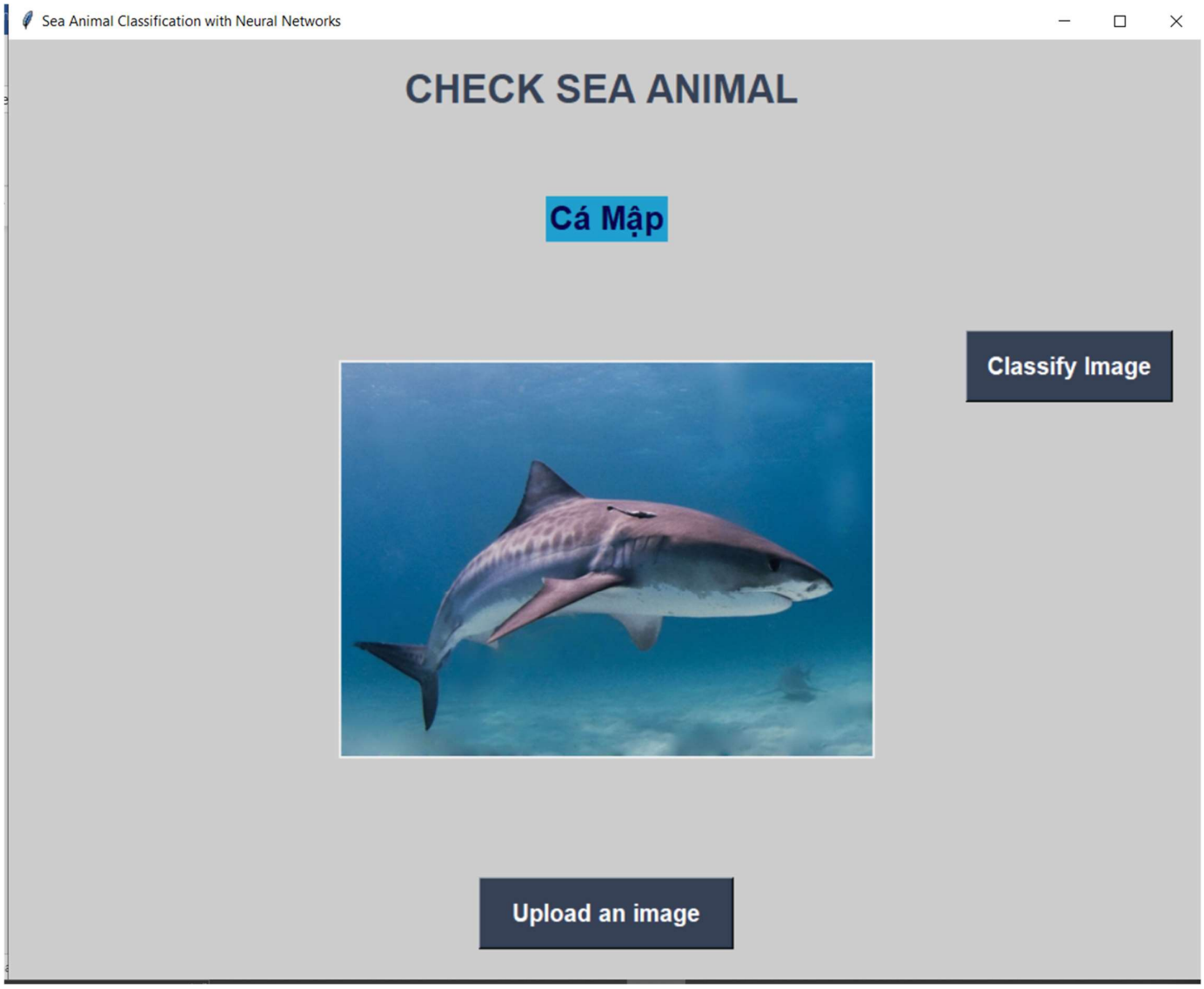
- Sử dụng mô hình bằng cách load model lên trên phần mềm giao diện Tkinter và dùng giao diện để sử dụng model.

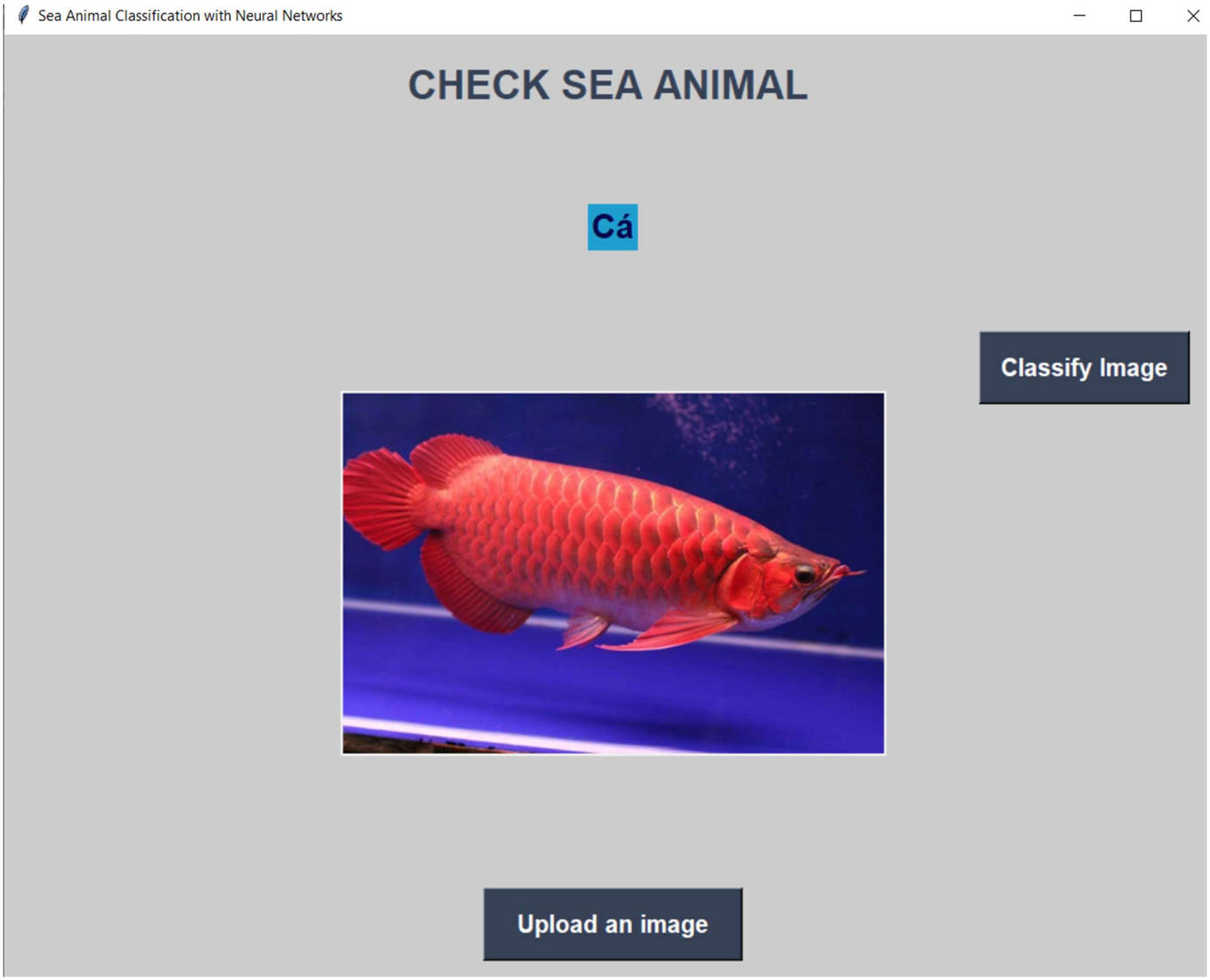


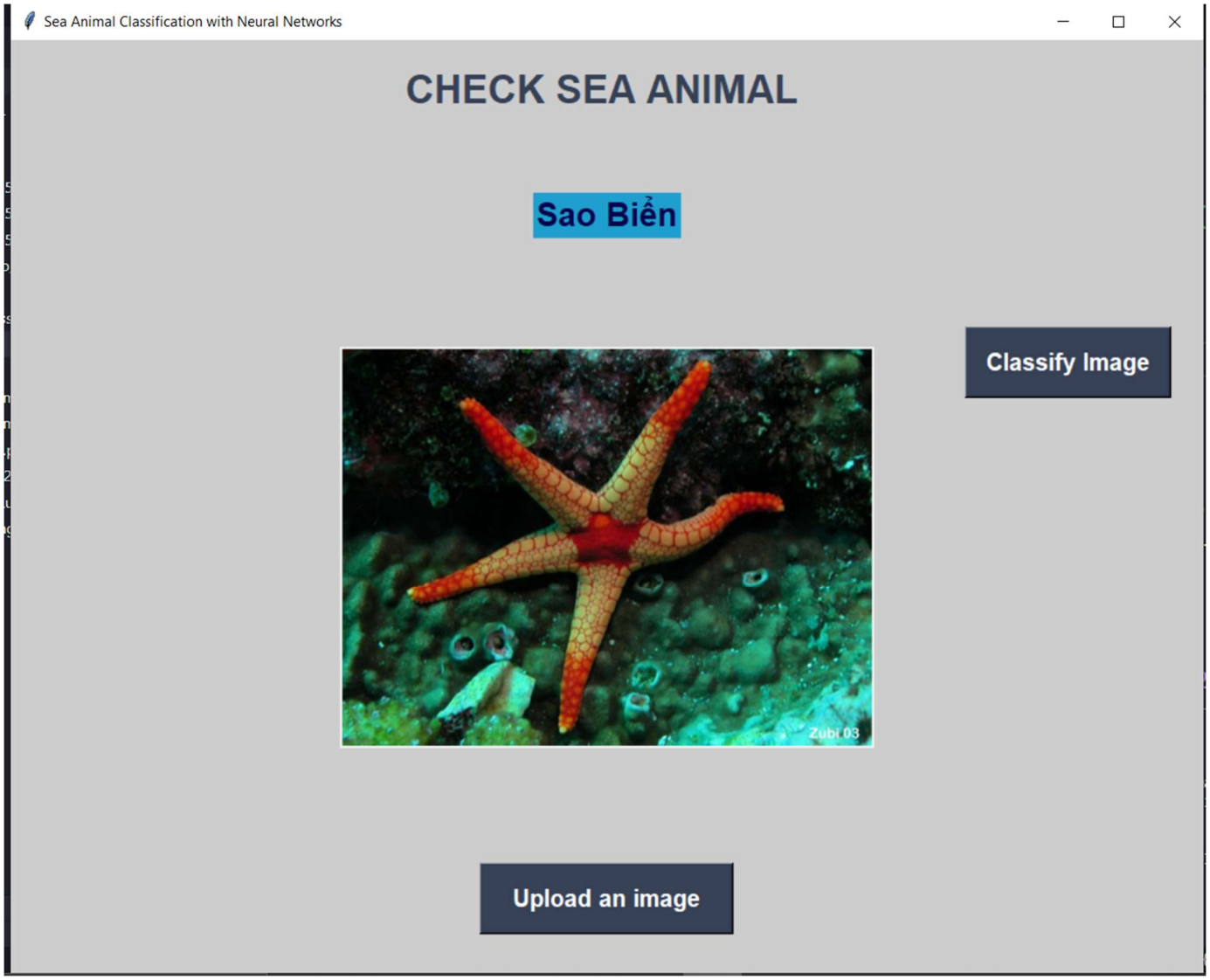

Sea Animal Classification with Neural Networks
CHECK SEA ANIMAL
Cá Ngựa
Classify Image
Upload an image

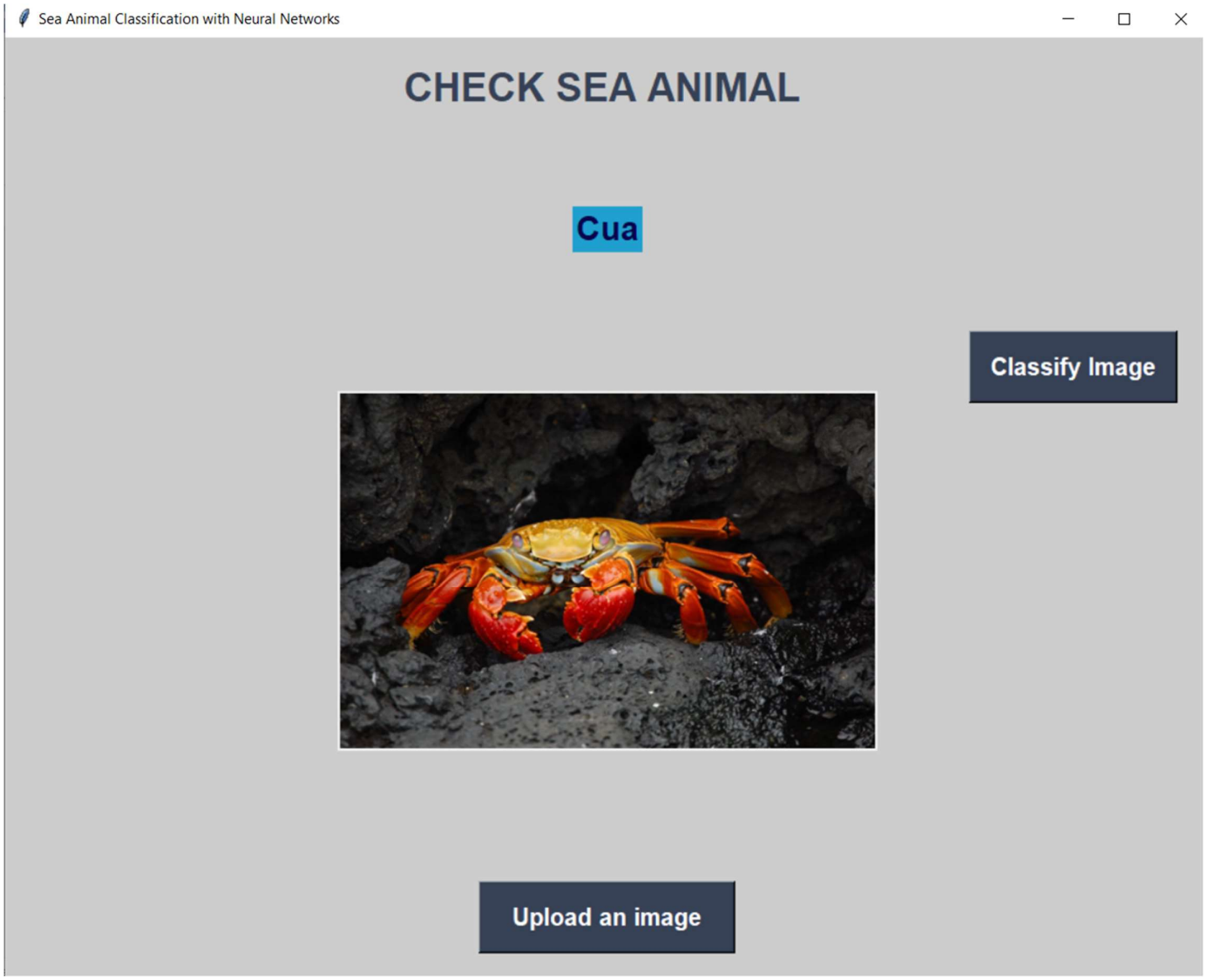

Sea Animal Classification with Neural Networks
CHECK SEA ANIMAL
Bạch Tuộc
Classify Image
Upload an image

-


Sea Animal Classification with Neural Networks
CHECK SEA ANIMAL
Cá Mập
Classify Image
Upload an image


Sea Animal Classification with Neural Networks
CHECK SEA ANIMAL
Cá
Classify Image
Upload an image


Sea Animal Classification with Neural Networks
CHECK SEA ANIMAL
Sao Biển
Classify Image
Upload an image


Sea Animal Classification with Neural Networks
CHECK SEA ANIMAL
Cua
Classify Image
Upload an image

4. Kết quả

Kết quả dự đoán trong tập dữ liệu test:

	Bạch Tuộc	Cá	Rái Cá	Cá Ngựa	Cua	Cá Mập
Dự đoán đúng	10	11	13	12	13	11
Dự đoán sai	1	1	1	1	1	1
Tỷ lệ chính xác(%)	90.9	91.7	92.9	92.3	92.9	91.7

5. Kết luận

Độ chính xác trung bình của mô hình val_accuracy = 98.77%

- Nguyên nhân dẫn đến độ chính xác như vậy là:

+ Một số loài có màu sắc và hình dạng gần giống nhau
+ Số lượng ảnh còn ít, góc chụp của hình chưa được nhiều
+ Các thông số vòng lặp, lớp CNN

- Phương pháp khắc phục:

+ Tăng số lượng hình ảnh của các loại động vật nhất là những loài có hình dạng khá giống nhau
+ Thay đổi nhiều thông số vòng lặp, CNN và cho chạy thử nghiệm nhiều lần
+ Chọn ảnh rõ ràng, đúng đối tượng