VEHICLE CLASSIFICATION

SVTH: Diệp Khải Hoàn **Môn: Trí tuệ nhân tạo**

MSSV: 20146090

Bài báo cáo tập trung vào việc sử dụng CNN để phân loại phương tiện giao thông. CNN là một phương pháp được áp dụng rộng rãi trong thị giác máy tính và nhận dạng hình ảnh. Bằng cách xây dựng mô hình CNN và huấn luyện trên tập dữ liệu phương tiện giao thông, chúng ta có thể đạt được độ chính xác cao trong việc phân loại các loại xe. Việc ứng dụng AI sử dụng CNN trong phân loại phương tiện giao thông hứa hẹn mang lại nhiều tiềm năng trong việc tăng cường an ninh, quản lý giao thông và xây dựng các thành phố thông minh.

1. Introduction

Đoạn code này được viết để phân loại các phương tiện giao thông khác nhau (cụ thể là 7). Các hình ảnh của các phương tiện này được lưu trữ trong một thư mục trên Google Drive. Sử dụng thư viện Keras và mạng nơ-ron tích chập (CNN) để thực hiện phân loại này. Bộ dữ liệu bao gồm 11976 hình ảnh, và đoạn code này chia nhỏ bộ dữ liệu thành tập huấn luyện và tập kiểm tra. Tập huấn luyện bao gồm 10778 hình ảnh, và tập kiểm tra bao gồm 1198 hình ảnh. Mô hình được huấn luyện bằng tập huấn luyện và được đánh giá bằng tập kiểm tra. Độ chính xác đạt được bởi mô hình được in ra ở cuối đoạn code. Cuối cùng, đoạn code này sử dụng mô hình đã được huấn luyện để phân loại một số phương tiện với hình ảnh được lấy từ máy.

2. Methodology

Đoạn code sử dụng mô hình mạng nơ-ron tích chập (CNN) để nhận diện cụ thể 7 phương tiện khác nhau. Tập dữ liệu được sử dụng trong đoạn code này bao gồm các hình ảnh của các 7 phương tiện đó. Chương trình được viết bằng ngôn ngữ lập trình Python và sử dụng một số thư viện, bao gồm NumPy, Keras, Scikit-learn và Matplotlib,…

Bước đầu tiên là tải các hình ảnh và nhãn tương ứng của chúng. Các hình ảnh được thay đổi kích thước thành 90x120 và chuyển đổi thành mảng NumPy. Các hình ảnh và nhãn của chúng được lưu vào một tệp để sử dụng sau này.

Tiếp theo, tập dữ liệu được chia thành các tập huấn luyện và kiểm tra bằng cách sử dụng hàm train_test_split từ thư viện Scikit-learn. Tập huấn luyện bao gồm 10778 hình ảnh, trong khi tập kiểm tra chứa 1198 hình ảnh.

Mô hình CNN trong đoạn mã sử dụng thư viện Keras được xây dựng theo tuần tự. Nó bao gồm các lớp Conv2D để trích xuất đặc trưng từ hình ảnh, các lớp MaxPooling2D để giảm kích thước và các lớp Dropout để tránh overfitting. Sau đó, các lớp Dense được sử dụng để kết nối và phân loại các loại phương tiện khác nhau.

Xây dựng giao diện với thư viện Tkinter của python.

Thực hiện phân loại một vài phương tiện dựa vào hình ảnh đã có sẵn.

3. Implementation

- Import các thư viện và module cần thiết. Đoạn code sử dụng các thư viện như numpy, keras, matplotlib và google.colab để load ảnh từ Google Drive, chuyển đổi ảnh thành các mảng numpy và trực quan hóa kết quả dự đoán của mô hình.

```
from os import listdir
from numpy import asarray
from numpy import save
from keras.utils import load_img, img_to_array
import matplotlib.pyplot as plt
from google.colab import drive
```

- Mount Google Drive vào colab. Mount Google Drive vào colab để có thể truy cập và đọc file ảnh từ Drive.

```
drive.mount('/content/drive')
```

• Load ảnh và gán nhãn. Đoạn code trên sử dụng hàm listdir() để lấy danh sách các file trong thư mục folder. Từ tên file, gán nhãn cho loại hoa tương ứng (từ 0 đến 4). Load ảnh từ file và chuyển đổi thành mảng numpy. Cuối cùng, lưu các mảng ảnh và nhãn vào các file photos.npy và labels.npy và copy các file này vào thư mục trên Google Drive.

```
folder = '/content/drive/My Drive/Colab Notebooks/data_AI/'
photos, labels = list(), list()
for file in listdir(folder):
  if file.startswith('bicycle'):
    output = 0
  elif file.startswith('boat'):
    output = 1
  elif file.startswith('car'):
    output = 2
  elif file.startswith('motorbike'):
    output = 3
  elif file.startswith('airplane'):
    output = 4
  elif file.startswith('train'):
    output = 5
  elif file.startswith('truck'):
    output = 6
  else:
    continue
  photo = load_img(folder + file, target_size=(90, 120))
  photo = img_to_array(photo)
  photos.append(photo)
```

```
        labels.append(output)
    photos = asarray(photos)
    labels = asarray(labels)
    save('photos_AI.npy', photos)
    save('labels_AI.npy', labels)
    !cp photos_AI.npy "/content/drive/My Drive/Colab Notebooks"
    !cp labels_AI.npy "/content/drive/My Drive/Colab Notebooks"
```

- Chia dữ liệu thành tập train và test. Sử dụng hàm train_test_split() từ sklearn để chia dữ liệu thành tập train và test với tỷ lệ test_size=0.1 (10% dữ liệu làm tập test).

```
    from sklearn.model_selection import train_test_split
    import numpy as np
    photos = np.load('/content/drive/My Drive/Colab Notebooks/photos_AI.npy')
    labels = np.load('/content/drive/My Drive/Colab Notebooks/labels_AI.npy')
    x_train, x_test, y_train, y_test = train_test_split(photos, labels, test_size=0.1)
```

- Định hình lại các kích thước ảnh và chuẩn hóa dữ liệu:

```
    x_train = x_train.reshape(10778, 90, 120, 3)
    x_test = x_test.reshape(1198, 90, 120, 3)
    x_train = x_train.astype('float32') / 255
    x_test = x_test.astype('float32') / 255
```

- Xác định các loại hoa theo tên tương ứng với từng hình ảnh.

```
    labels_vehicle = ['bicycle','boat','car','motorbike','airplane','train','truck']
```

- Chuyển đổi nhãn từ dạng số sang dạng nhị phân bằng phương thức to_categorical.

```
    from keras.utils import to_categorical
```

```
y_train = to_categorical(y_train, 7)
y_test = to_categorical(y_test, 7)
```

- Mô hình này sử dụng các lớp Conv2D để trích xuất đặc trưng từ ảnh, MaxPooling2D để giảm kích thước đầu vào, Dropout để tránh overfitting, và Dense để thực hiện phân loại ảnh vào 7 lớp khác nhau.

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, LeakyReLU, Dropout
model = Sequential()

model.add(Conv2D(32, kernel_size=3, activation='relu', input_shape=(90,120,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(7, activation='softmax'))
```

```
model.summary()
```

- Biên dịch mô hình bằng phương pháp compile và huấn luyện mô hình với tập dữ liệu huấn luyện và đánh giá với tập dữ liệu kiểm tra.

```
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, verbose=1, mode='max')
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=12, epochs=100, verbose=1)
test_loss,test_acc=model.evaluate(x_test,y_test)
print('test_acc:',test_acc)
print('test_loss:',test_loss)
```

- Lưu mô hình đã huấn luyện vào tệp tin VEHICLE.h5.

```
model.save('/content/drive/My Drive/Colab Notebooks/VEHICLE.h5')
```

- Sau khi đã có được VEHICLE.h5 ta tiến hành lưu về và thực hiện phân loại trên giao diện Tkinter, hình ảnh trước khi được phân loại sẽ được chuyển đổi sao cho đúng chuẩn đầu vào của mô hình. Tương tự vậy, đây là đoạn code trên Collab dùng để load hình ảnh từ driver và chuyển nó thành mảng numpy và phân loại, in ra kết quả:

```
img = load_img('/content/drive/My Drive/Colab Notebooks/oto.jpg',target_size=(90,120))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1, 90, 120, 3)
img = img.astype('float32') / 255
result = labels_vehicle[np.argmax(model.predict(img))]
print(result)
```

- Còn dưới đây là đoạn code giao diện khi đã có được file VEHICLE.h5:

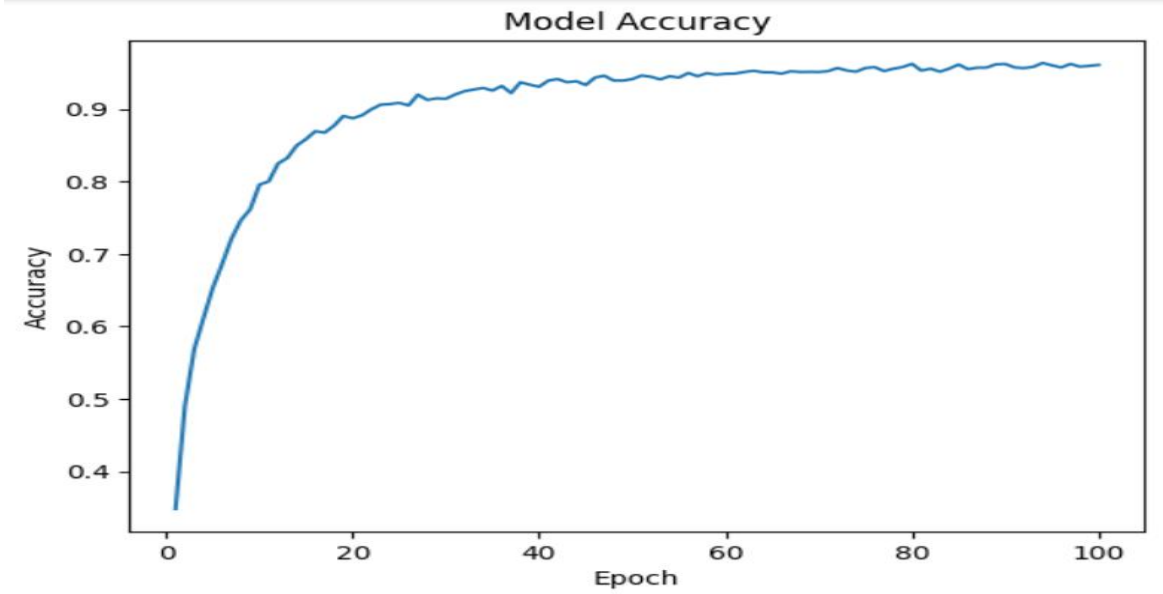
```
    import tkinter as tk
    from PIL import Image, ImageTk
    from tkinter import filedialog
    from keras.utils import img_to_array, load_img
    import numpy as np
    from tensorflow import keras

    model = keras.models.load_model('VEHICLE.h5')
    labels_vehicle = ['bicycle', 'boat', 'car', 'motorbike', 'airplane', 'train',
'truck']

    classified_images_bicycle = []
    classified_images_boat = []
    classified_images_car = []
    classified_images_motorbike = []
    classified_images_airplane = []
    classified_images_train = []
    classified_images_truck = []

    def open_image():
        global file_path
        file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.jpg")])
        if file_path:
            current_image = Image.open(file_path)
            current_image = current_image.resize((400, 300))
            image_tk = ImageTk.PhotoImage(current_image)

            image_label.configure(image=image_tk)
            image_label.image = image_tk

            classify_button.configure(state=tk.NORMAL)

    def view_classified_images(list_classified, name):
        if list_classified:
            new_window = tk.Toplevel(window)
            new_window.title("Classified Images: " + str(name))
            new_window.geometry("800x600")

            row_count = 0
            col_count = 0
            for image_path in list_classified:
                classified_image = Image.open(image_path)
```

```
                classified_image = classified_image.resize((200, 150))
                classified_image_tk = ImageTk.PhotoImage(classified_image)

                image_label = tk.Label(new_window, image=classified_image_tk)

                image_label.image = classified_image_tk
                image_label.grid(row=row_count, column=col_count, padx=10, pady=10)

                col_count += 1
                if col_count > 3:
                    col_count = 0
                    row_count += 1
        else:
            tk.messagebox.showinfo("No Classified Images", "There are no
classified images.")

    def create_button_view(name_vehicle, classified_list, x, y):
        view_button = tk.Button(window, text=name_vehicle, command=lambda:
view_classified_images(classified_list, name_vehicle),width=10,height=2)
        view_button.place(x=x, y=y)

    def vehicle_classification():
        global result
        global file_path
        global accuracy
        current_image_resized = load_img(file_path, target_size=(90, 120))
        current_image_array = img_to_array(current_image_resized)
        current_image_array = current_image_array.reshape(1, 90, 120, 3)
        current_image_array = current_image_array.astype('float32') / 255

        classified = model.predict(current_image_array)
        classified_class_index = np.argmax(classified)
        result = labels_vehicle[ classified_class_index]
        accuracy =classified[0][classified_class_index] * 100


        comment_textbox.delete(1.0, tk.END)
        comment_textbox.insert(tk.END, result)

        comment_textbox1.delete(1.0, tk.END)
        comment_textbox1.insert(tk.END, accuracy)

        if result == 'bicycle':
            classified_images_bicycle.append(file_path)
        elif result == 'boat':
```

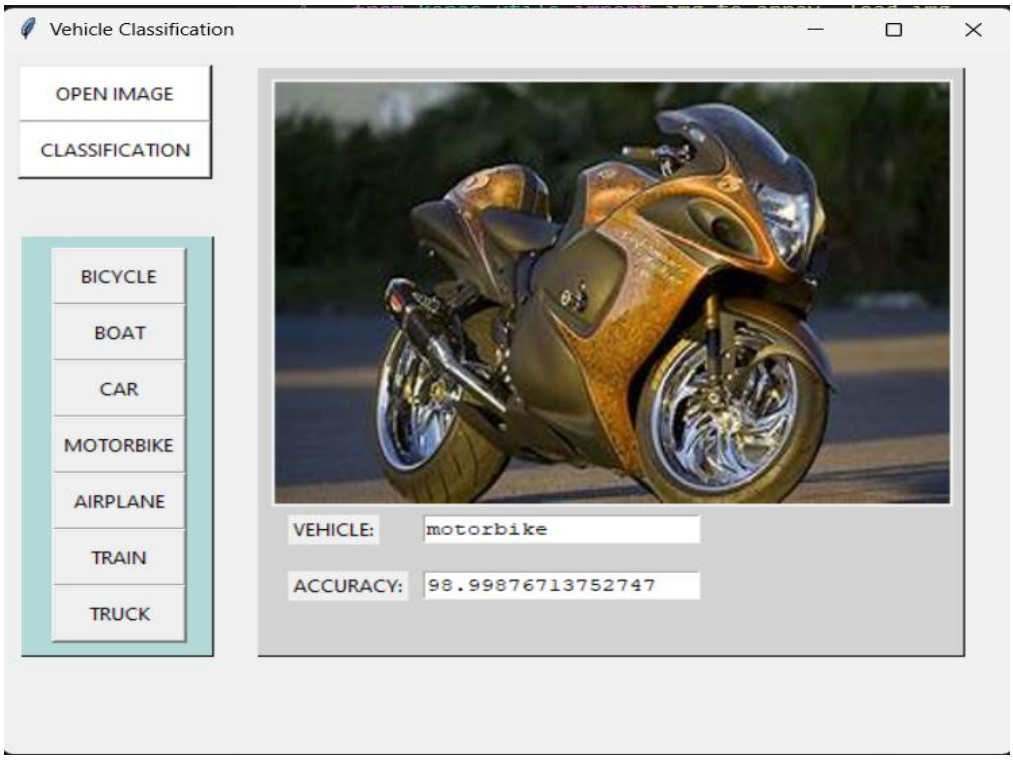
```
            classified_images_boat.append(file_path)
        elif result == 'car':
            classified_images_car.append(file_path)
        elif result == 'motorbike':
            classified_images_motorbike.append(file_path)
        elif result == 'airplane':
            classified_images_airplane.append(file_path)
        elif result == 'train':
            classified_images_train.append(file_path)
        elif result == 'truck':
            classified_images_truck.append(file_path)

    def open_enter(e):
        open_button.config(bg="red", fg="white")

    def open_leave(e):
        open_button.config(bg="white", fg="black")

    def classify_enter(e):
        classify_button.config(bg="red", fg="white")

    def classify_leave(e):
        classify_button.config(bg="white", fg="black")

    window = tk.Tk(screenName="VEHICLE")
    window.geometry("600x500")
    window.title("Vehicle Classification")

    open_button = tk.Button(window, text="OPEN IMAGE",
command=open_image,width=15,height=2)
    open_button.place(x=10, y=10)
    open_button.bind("<Enter>", open_enter)
    open_button.bind("<Leave>", open_leave)

    classify_button = tk.Button(window, text="CLASSIFICATION",
command=vehicle_classification, state=tk.DISABLED,width=15,height=2)
    classify_button.place(x=10, y=50)
    classify_button.bind("<Enter>", classify_enter)
    classify_button.bind("<Leave>", classify_leave)

    #tạo nền cho các nút ấn
    canvas = tk.Canvas(window, width=115, height=300)
    canvas.place(x=10,y=130)
    canvas.create_rectangle(0, 0, 115, 300, fill="#b2d8d8")
```

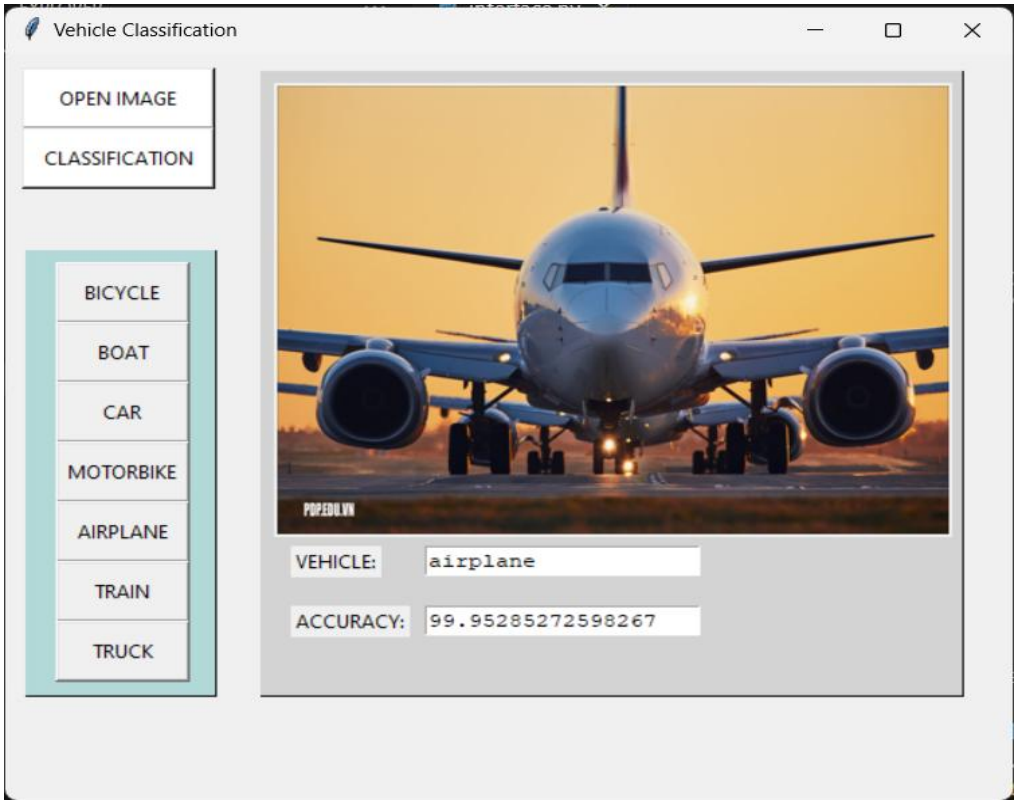
```
    create_button_view('BICYCLE', classified_images_bicycle, 30, 140)
    create_button_view('BOAT', classified_images_boat, 30, 180)
    create_button_view('CAR', classified_images_car, 30, 220)
    create_button_view('MOTORBIKE', classified_images_motorbike, 30, 260)
    create_button_view('AIRPLANE', classified_images_airplane, 30, 300)
    create_button_view('TRAIN', classified_images_train, 30, 340)
    create_button_view('TRUCK', classified_images_truck, 30, 380)

    canvas = tk.Canvas(window, width=420, height=420)
    canvas.place(x=150,y=10)
    canvas.create_rectangle(0, 0, 420, 420, fill="lightgray")

    image_label = tk.Label(window)
    image_label.place(x=160,y=20)

    label = tk.Label(window, text="VEHICLE:")
    label.place(x=170,y=330)

    comment_textbox = tk.Text(window, width=20, height=1)
    comment_textbox.place(x=250, y=330)

    label = tk.Label(window, text="ACCURACY:")
    label.place(x=170,y=370)

    comment_textbox1 = tk.Text(window, width=20, height=1)
    comment_textbox1.place(x=250, y=370)

    window.mainloop()
```

4. Results




```
test_acc: 0.7320533990859985
test_loss: 1.6241074800491333
```





5. Conclusion

Trong đoạn code trên, một mô hình học máy được xây dựng để phân loại phương tiện giao thông. Dữ liệu được sử dụng là một tập hình ảnh của các phương tiện được lưu trữ trong thư mục trên Google Drive. Dữ liệu được chia thành tập huấn luyện và tập kiểm tra và mô hình được xây dựng bằng cách sử dụng mạng nơ-ron tích chập (CNN). Mô hình đạt được độ chính xác tạm ổn trên tập kiểm tra, chỉ ra rằng nó có thể phân loại các loại phương tiện khác nhau với độ chính xác cao. Với việc sử dụng các thư viện như Keras và NumPy, mô hình được xây dựng và huấn luyện một cách hiệu quả. Do đó, đoạn code này là một ví dụ tuyệt vời về cách sử dụng Deep learning để giải quyết các vấn đề trong lĩnh vực thị giác máy. Tuy nhiên, khi phân loại vẫn còn một số nhầm lẫn nguyên nhân do tập dữ liệu tự tạo chưa chuẩn xác, một vài phương tiện có phần đầu thật sự gần giống nhau nên có thể tạo ra nhầm lẫn, do đó cần phải tạo nên kho dữ liệu rộng hơn, chính xác hơn, bên cạnh đó cần áp dụng thêm một vài thuật toán như lật ảnh, tăng độ tương phản,…để khiến cho kho dữ liệu ngày càng phong phú.

MÃ QR GITHUB:

Link youtube:

https://www.youtube.com/watch?v=CZpqPz6yAvA&ab_channel=Kh%E1%BA%A3iHo%C3%A0nDi%E1%BB%87p

Link data:

https://drive.google.com/drive/folders/174crkQQsDOGyCOAxlrE5XD9Lic_ZY-h-?usp=sharing