BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

A blue diamond shaped sign with a torch and a globe

Description automatically generated

**HỌC SÂU(DEEP LEARNING)**

**Mã Học Phần: CT282**

**Đề tài**

**NHẬN DIỆN BIỂN BÁO GIAO THÔNG**

**Giảng Viên: Phạm Nguyên Khang**

|  |  |
| --- | --- |
| **THÀNH VIÊN NHÓM** | |
| **Họ Và Tên** | **MSSV** |
| **Nguyễn Trần Phúc Thịnh** | **B2017006** |
| **Đào Việt Anh** | **B2113303** |

**Học Kỳ I 2024 – 2025**

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

# **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc179149715)

[**PHẦN Ⅰ : MÔ TẢ VẤN ĐỀ** 3](#_Toc179149716)

[1.Giới Thiệu 3](#_Toc179149717)

[2.Dữ Liệu 3](#_Toc179149718)

[**PHẦN Ⅱ : HUẤN LUYỆN VÀ KIỂM TRA HIỆU QUẢ** 5](#_Toc179149719)

[**A/ Tiền Xử Lý Dữ Liệu** 5](#_Toc179149720)

[**B/ Huấn Luyện** 6](#_Toc179149721)

[1.Mô Hình CNN : Kiến trúc mạng cơ bản 6](#_Toc179149722)

[a/ Mô hình 6](#_Toc179149723)

[b/ Quá trình huấn luyện và kiểm tra 7](#_Toc179149724)

[2. Mô Hình CNN : Kiến trúc mạng VGG 9](#_Toc179149725)

[a/ Mô hình 9](#_Toc179149726)

[b/ Quá trình huấn luyện và kiểm tra 11](#_Toc179149727)

[3. Mô Hình CNN : Kiến trúc mạng AlexNet 13](#_Toc179149728)

[a/ Mô hình 13](#_Toc179149729)

[b/ Quá trình huấn luyện và kiểm tra 14](#_Toc179149730)

[**PHẦN Ⅲ : KẾT LUẬN** 17](#_Toc179149731)

[1.Tổng Quát Các Kiến Trúc Mạng 17](#_Toc179149732)

[2.So Sánh Hiệu Quả Và Kết Luận 19](#_Toc179149733)

# **PHẦN Ⅰ : MÔ TẢ VẤN ĐỀ**

## 1.Giới Thiệu

Nhận diện biển báo giao thông là một trong những ứng dụng quan trọng của công nghệ trí tuệ nhân tạo và thị giác máy tính trong lĩnh vực giao thông vận tải. Việc tự động phát hiện và nhận diện các biển báo giao thông giúp tăng cường sự an toàn cho người lái xe, hỗ trợ các hệ thống lái xe tự động và giảm thiểu nguy cơ tai nạn do lỗi con người.

Các biển báo giao thông cung cấp thông tin quan trọng về quy tắc đường bộ, hạn chế tốc độ, hướng đi, và các cảnh báo nguy hiểm. Vì vậy, việc nhận diện chính xác các biển báo này đòi hỏi các thuật toán xử lý ảnh hiệu quả, có khả năng phân loại và trích xuất thông tin từ những hình ảnh phức tạp.

Trong những năm gần đây, các kỹ thuật học sâu (deep learning) đã mang lại nhiều bước tiến đáng kể trong việc giải quyết bài toán này. Cụ thể, các mô hình mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) đã chứng minh được khả năng vượt trội trong việc nhận diện và phân loại chính xác các biển báo giao thông từ các điều kiện môi trường khác nhau.

Bài toán nhận diện biển báo giao thông không chỉ góp phần vào việc phát triển hệ thống giao thông thông minh mà còn mở ra nhiều cơ hội ứng dụng thực tiễn trong việc tối ưu hóa các hệ thống điều khiển xe tự động, cảnh báo va chạm và hỗ trợ người lái xe.

## 2.Dữ Liệu

Dữ liệu ảnh sẽ có 3 loại biển:

* Cấm
* Cảnh Báo
* Chỉ Dẫn

A screenshot of a computer

Description automatically generated

* Dữ liệu đầu vào gồm 1500 ảnh biển báo giao thông:500 ảnh biển báo cấm,500 ảnh biển báo chỉ dẫn và 500 ảnh biển báo cảnh báo với kích thước bằng nhau(150,150)
* Chia tập dữ liệu thành : train (72,5%) , test (12,5%), validation (15%) và 1 file chứa ảnh để kiểm tra

A screenshot of a computer

Description automatically generated

# **PHẦN Ⅱ : HUẤN LUYỆN VÀ KIỂM TRA HIỆU QUẢ**

Các bước được thực hiện trên công cụ google colab hoặc kaggle

## **A/ Tiền Xử Lý Dữ Liệu**

**Khai báo các thư viện cần thiết**

|  |
| --- |
| **import tensorflow as tf**  **from tensorflow.keras.preprocessing.image import ImageDataGenerator**  **from tensorflow.keras.models import Sequential**  **from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense**  **import numpy as np**  **import matplotlib.pyplot as plt**  **import os** |

**Chia dataset và tăng cường dữ liệu:**

|  |
| --- |
| **from tensorflow.keras.preprocessing.image import ImageDataGenerator**  **data\_dir = '/kaggle/input/aaaaaaaa/dataset/dataset'**  **# Tạo ImageDataGenerator cho tập huấn luyện và validation**  **train\_val\_datagen = ImageDataGenerator(**  **rescale=1./255,**  **validation\_split=0.1667,  # 12.5%/75% = 0.1667 để chia validation và train**  **rotation\_range=40,**  **width\_shift\_range=0.2,**  **height\_shift\_range=0.2,**  **shear\_range=0.2,**  **zoom\_range=0.2,**  **horizontal\_flip=True,**  **fill\_mode='nearest'**  **)** |

**Tăng cường và xử lý dữ liệu cho tập huấn luyện (train) và tập xác thực (validation)**

|  |
| --- |
| **# Generator cho tập huấn luyện (75%)**  **train\_generator = train\_val\_datagen.flow\_from\_directory(**  **data\_dir,**  **target\_size=(150, 150),**  **batch\_size=32,**  **class\_mode='categorical',**  **subset='training'  # 75% dữ liệu cho train**  **)**  **# Generator cho tập validation (12.5%)**  **validation\_generator = train\_val\_datagen.flow\_from\_directory(**  **data\_dir,**  **target\_size=(150, 150),**  **batch\_size=32,**  **class\_mode='categorical',**  **subset='validation'  # 12.5% dữ liệu cho validation**  **)** |

**Làm tương tự với tập test (tập test được chia sẵn)**

|  |
| --- |
| **# Tạo ImageDataGenerator riêng cho tập test (rescale không cần augmentation)**  **test\_datagen = ImageDataGenerator(rescale=1./255)**  **# Tạo generator cho tập test từ thư mục 'dataset/test'**  **test\_generator = test\_datagen.flow\_from\_directory(**  **'/kaggle/input/aaaaaaaa/test/test',  # Đường dẫn đến thư mục test**  **target\_size=(150, 150),**  **batch\_size=32,**  **class\_mode='categorical'**  **)** |

**Kết quả trả về số lượng của tập train, validation và test lần lượt :**

|  |
| --- |
| **Found 1155 images belonging to 3 classes.**  **Found 229 images belonging to 3 classes.**  **Found 181 images belonging to 3 classes.** |

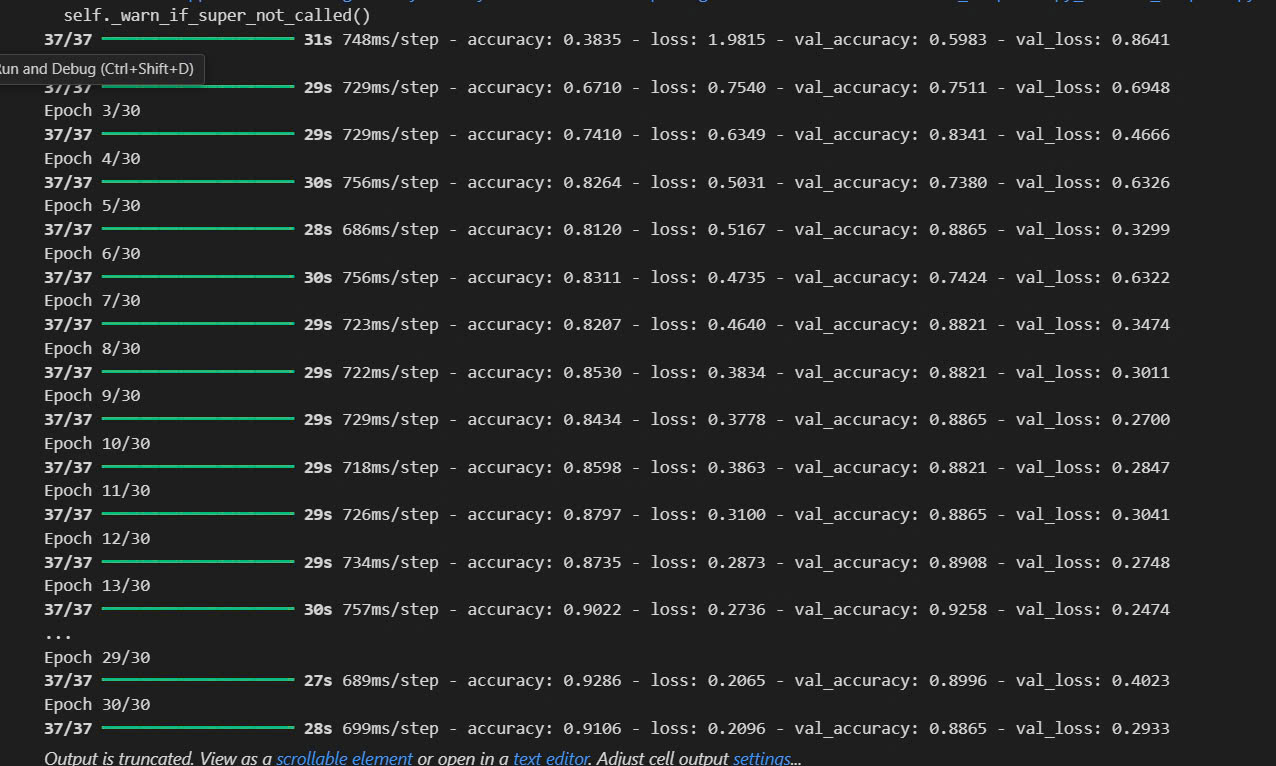
## **B/ Huấn Luyện**

## 1.Mô Hình CNN : Kiến trúc mạng cơ bản

### a/ Mô hình

|  |
| --- |
| model1 = Sequential([      Conv2D(32, (3, 3), activation='relu', input\_shape=(150, 150, 3)),      MaxPooling2D(2, 2),      Conv2D(64, (3, 3), activation='relu'),      MaxPooling2D(2, 2),      Conv2D(128, (3, 3), activation='relu'),      MaxPooling2D(2, 2),      Flatten(),      Dense(512, activation='relu'),      Dense(3, activation='softmax')  # 3 lớp đầu ra tương ứng với 3 loại biển báo  ])  model1.compile(loss='categorical\_crossentropy',                optimizer='adam',                metrics=['accuracy']) |
| # Huấn luyện mô hình  history = model1.fit(      train\_generator,  # Dữ liệu huấn luyện      epochs=30,  # Số lượng epochs, bạn có thể điều chỉnh      validation\_data=validation\_generator,  # Dữ liệu validation  ) |

### b/ Quá trình huấn luyện và kiểm tra



#### Kết quả đánh giá mô hình trên tập test

|  |
| --- |
| # Đánh giá mô hình trên tập test  test\_loss, test\_accuracy = model1.evaluate(      test\_generator,  # Dữ liệu test  )  print(f"Test loss: {test\_loss}")  print(f"Test accuracy: {test\_accuracy}") |



#### **Biểu đồ accuracy và loss của quá trình huấn luyện:**

A graph showing the performance of a performance

Description automatically generated

**Biểu đồ accuracy**

A graph showing loss and validation

Description automatically generated

**Biểu đồ loss**

#### Đưa vào 1 ảnh cho mô hình nhận diện

|  |
| --- |
| from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array  # Đọc ảnh  img\_path = '/kaggle/input/aaaaaaaa/ktra/ktra/img.jpg'  img = load\_img(img\_path, target\_size=(150, 150))  # Tiền xử lý  x = img\_to\_array(img)  x = x / 255.0  x = np.expand\_dims(x, axis=0)  # Dự đoán  predictions = model1.predict(x)  predicted\_class\_index = np.argmax(predictions)  # Mapping nhãn  labels = ['cam', 'canhbao', 'chidan']  predicted\_label = labels[predicted\_class\_index]  # In kết quả  print("Nhãn dự đoán của ảnh là:", predicted\_label) |

A close-up of a sign

Description automatically generatedA screen shot of a computer

Description automatically generated

## 2. Mô Hình CNN : Kiến trúc mạng VGG

### a/ Mô hình

|  |
| --- |
| # VGG16 ailab  modelVGG = Sequential([      # Block 1      Conv2D(64, (3, 3), activation='relu',input\_shape=(150, 150, 3)),      Conv2D(64, (3, 3), activation='relu'),      MaxPooling2D(2, 2),        # Block 2      Conv2D(128, (3, 3), activation='relu'),      Conv2D(128, (3, 3), activation='relu'),      MaxPooling2D(2,2),         # Block 3      Conv2D(256, (3, 3), activation='relu'),      Conv2D(256, (3, 3), activation='relu'),      Conv2D(256, (3, 3), activation='relu'),      MaxPooling2D(2,2),        # Flatten và Fully Connected      Flatten(),      Dense(512, activation='relu'),      Dense(512, activation='relu'),      Dense(3, activation='softmax')  ])  # Biên dịch mô hình  modelVGG.compile(loss='categorical\_crossentropy',                optimizer='adam',                metrics=['accuracy']) |

### b/ Quá trình huấn luyện và kiểm tra

A screenshot of a computer screen

Description automatically generated

#### Kết quả đánh giá mô hình trên tập test

|  |
| --- |
| # Đánh giá mô hình trên tập test  test\_loss, test\_accuracy = modelVGG.evaluate(      test\_generator,  # Dữ liệu test  )  print(f"Test loss: {test\_loss}")  print(f"Test accuracy: {test\_accuracy}") |



#### Biểu đồ accuracy và loss của mô hình

A graph showing the performance of a train and validation accuracy

Description automatically generated

**Biểu đồ accuracy**

A graph of loss and validation

Description automatically generated

**Biểu đồ loss**

#### Đưa vào 1 ảnh cho mô hình nhận diện

|  |
| --- |
| predictionsVGG = modelVGG.predict(x)  predicted\_class\_indexVGG = np.argmax(predictionsVGG)  # Mapping nhãn  labels = ['cam', 'canhbao', 'chidan']  predicted\_labelVGG = labels[predicted\_class\_indexVGG]  # In kết quả  print("Nhãn dự đoán của ảnh là:", predicted\_labelVGG) |

A screen shot of a computer

Description automatically generated

## 3. Mô Hình CNN : Kiến trúc mạng AlexNet

### a/ Mô hình

|  |
| --- |
| modelAlexNet = Sequential([      Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input\_shape=(150, 150, 3)),      MaxPooling2D(pool\_size=(3, 3), strides=(2, 2)),      Conv2D(256, (5, 5), padding='same', activation='relu'),      MaxPooling2D(pool\_size=(3, 3), strides=(2, 2)),        Conv2D(384, (3, 3), padding='same', activation='relu'),      Conv2D(384, (3, 3), padding='same', activation='relu'),      Conv2D(256, (3, 3), padding='same', activation='relu'),      MaxPooling2D(pool\_size=(3, 3), strides=(2, 2)),      # Flatten và Fully Connected      Flatten(),      Dense(512, activation='relu'),      Dense(512, activation='relu'),      Dense(3, activation='softmax')  ])  # Biên dịch mô hình  modelAlexNet.compile(loss='categorical\_crossentropy',                optimizer='sgd',                metrics=['accuracy']) |

### b/ Quá trình huấn luyện và kiểm tra

A screen shot of a computer

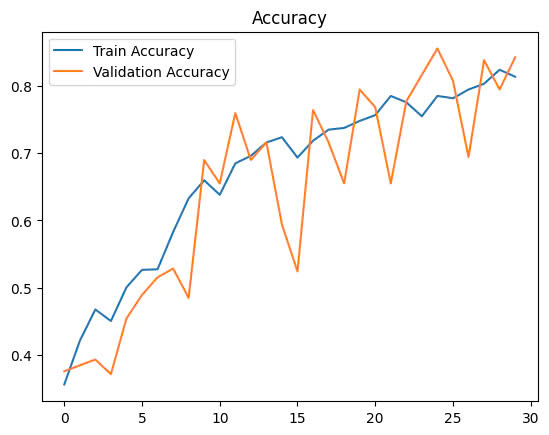
Description automatically generated

#### Kết quả đánh giá mô hình trên tập test

|  |
| --- |
| # Đánh giá mô hình trên tập test  test\_loss, test\_accuracy = modelAlexNet.evaluate(      test\_generator,  # Dữ liệu test  )  print(f"Test loss: {test\_loss}")  print(f"Test accuracy: {test\_accuracy}") |



#### Biểu đồ accuracy và loss của mô hình



**Biểu đồ accuracy**

A graph showing loss and validation

Description automatically generated

**Biểu đồ loss**

#### Đưa vào 1 ảnh cho mô hình nhận diện

|  |
| --- |
| from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array  # Đọc ảnh  img\_path = '/kaggle/input/aaaaaaaa/ktra/ktra/img.jpg'  img = load\_img(img\_path, target\_size=(150, 150))  # Tiền xử lý  x = img\_to\_array(img)  x = x / 255.0  x = np.expand\_dims(x, axis=0)  predictionsAlexNet = modelAlexNet.predict(x)  predicted\_class\_indexAlexNet = np.argmax(predictionsAlexNet)  # Mapping nhãn  labels = ['cam', 'canhbao', 'chidan']  predicted\_labelAlexNet = labels[predicted\_class\_indexAlexNet]  # In kết quả  print("Nhãn dự đoán của ảnh là:", predicted\_labelAlexNet) |



# **PHẦN Ⅲ : KẾT LUẬN**

## 1.Tổng Quát Các Kiến Trúc Mạng

* Kiến trúc mạng cơ bản

A diagram of a software

Description automatically generated

* Kiến trúc mạng VGG

A diagram of a computer

Description automatically generated

* Kiến trúc mạng AlexNet

A diagram of a number of steps

Description automatically generated with medium confidence

## 2.So Sánh Hiệu Quả Và Kết Luận

* Về mặt lý thuyết:

Mô hình CNN với kiến trúc mạng VGG có kiến trúc đơn giản hơn và hiệu suất cao hơn so với AlexNet và kiến trúc mạng cơ bản.

* Về phần dự án:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Cơ bản** | **AlexNet** | **VGG** |
| **Accuracy** | 0.9944751262664795 | 0.9668508172035217 | 0.9613259434700012 |
| **Loss** | 0.021981053054332733 | 0.07372523844242096 | 0.07601171731948853 |

* Kết quả trả về gần như là bằng nhau điều này do:

-Số lượng ảnh trong dataset không đủ nhiều

-Chất lượng ảnh không tốt (ảnh chứa nhiều background)

* Trên thực tế:

-Với số lượng ảnh lớn và nhiều class thì VGG vẫn là kiến trúc mạng tốt nhất