**TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH**

KHOA CÔNG NGHỆ THÔNG TIN

**ĐỒ ÁN MÔN HỌC**

**HỌC SÂU**

**TÊN ĐỀ TÀI**

PHẦN MỀM NHẬN DẠNG 5 BIỂN BÁO GIAO THÔNG

**GIẢNG VIÊN HƯỚNG DẪN:** Tôn Quang Toại

**SINH VIÊN THỰC HIỆN:**

Lương Thiên Bảo – 22DH114453

Ka sa ha bus – 22DH110377

Đặng Trọng Hoàng Anh – 22DH110080

Vũ Trọng Quang – 22DH114700

Thiều Duy Khoa – 22DH111692

**TP. HỒ CHÍ MINH – 04/2025**

MỤC LỤC

[MỤC LỤC I](#_Toc194413129)

[TÓM TẮT ĐỒ ÁN V](#_Toc194413130)

[Chương 1. Phát biểu bài toán 7](#_Toc194413131)

[1.1 Mô tả bài toán 7](#_Toc194413132)

[1.2 Giới hạn bài toán 8](#_Toc194413133)

[1.3 Bố cục đồ án 9](#_Toc194413134)

[Chương 2. Giải pháp đề xuất 10](#_Toc194413135)

[2.1 Phân tích dữ liệu 10](#_Toc194413136)

[2.1.1 Thống kê tập dữ liệu 10](#_Toc194413137)

[2.1.2 Tiền xử lý dữ liệu 11](#_Toc194413138)

[2.2 Thiết kế mô hình 11](#_Toc194413139)

[2.2.1 Mô hình AlexNet 11](#_Toc194413140)

[2.2.2 Mô hình ResNet50 15](#_Toc194413141)

[2.2.3 Mô hình UNet 18](#_Toc194413142)

[Chương 3. Các thực nghiệm, đánh giá và triển khai 24](#_Toc194413143)

[3.1 Các thực nghiệm 24](#_Toc194413144)

[3.1.1 Thực nghiệm AlexNet 24](#_Toc194413145)

[3.1.2 Thực nghiệm ResNet50 28](#_Toc194413146)

[3.1.3 Thực nghiệm UNet 31](#_Toc194413147)

[3.2 Đánh giá 33](#_Toc194413148)

[3.2.1 So sánh 2 mô hình phân loại AlexNet và ResNet50 33](#_Toc194413149)

[3.2.2 Mô hình UNet 34](#_Toc194413150)

[3.3 Triển khai 35](#_Toc194413151)

[KẾT LUẬN 37](#_Toc194413152)

[TÀI LIỆU THAM KHẢO 41](#_Toc194413153)

DANH MỤC HÌNH ẢNH

[Hình 1: Bảng thống kê các tầng mô hình AlexNet 14](#_Toc194254276)

[Hình 2: Minh họa kiến trúc mô hình AlexNet 15](#_Toc194254277)

[Hình 3: Bảng thống kê các tầng mô hình ResNet50 17](#_Toc194254278)

[Hình 4: Minh họa kiến trúc mô hình ResNet50 18](#_Toc194254280)

Hình 5: Trực quan Input đầu vào UNet....................................................................19

[Hình 6: Bảng thống kê các tầng mô hình UNet 21](#_Toc194254281)

[Hình 7: Minh họa kiến trúc mô hình Unet. 22](#_Toc194254282)

[Hình 8: Giao diện trang chủ 34](#_Toc194254283)

[Hình 9: Giao diện dự đoán 35](#_Toc194254284)

[Hình 10: Giao diện kết quả dự đoán 36](#_Toc194254285)

TÓM TẮT ĐỒ ÁN

Bài toán (problem): Đồ án tập trung vào việc xây dựng một hệ thống nhận dạng 5 loại biển báo giao thông bằng Deep Learning. Hệ thống bao gồm hai nhiệm vụ chính:

Phân đoạn (Segmentation): Xác định vùng chứa biển báo trong ảnh.

Phân loại (Classification): Nhận diện loại biển báo từ vùng đã phát hiện.

**Phương pháp đề xuất giải quyết (method):**

Phân đoạn ảnh: Sử dụng mô hình U-Net, một kiến trúc mạnh mẽ cho bài toán phân vùng ảnh, giúp xác định chính xác vị trí biển báo trong ảnh có nhiều đối tượng.

Phân loại biển báo: Sử dụng AlexNet (một mạng CNN sâu) và ResNet50, để nhận diện chính xác từng loại biển báo sau khi đã phân đoạn. Từ đó so sánh hiệu suất giữa 2 thuật toán và lựa chọn ra thuật toán tốt nhất phù hợp với mô hình.

Dữ liệu: Ảnh biển báo được thu thập và tiền xử lý (resize, normalize, augmentation) để đảm bảo tính đa dạng.

Huấn luyện: Áp dụng kỹ thuật tối ưu hóa (Adam optimizer), hàm mất mát phù hợp (Cross-Entropy Loss), và điều chỉnh siêu tham số để đạt hiệu suất tốt nhất.

**Các kết quả đạt được (result):**

Mô hình phân đoạn U-Net đạt độ chính xác cao với IoU trên tập kiểm tra (0.69).

Mô hình phân loại AlexNet đạt độ chính xác cao (Accuracy = 96%) và ResNet50 (96.56%) trong việc nhận diện 5 loại biển báo.

Hệ thống hoạt động tốt trong điều kiện thực tế và có thể triển khai trên ứng dụng hỗ trợ lái xe hoặc camera giám sát giao thông.

1. Phát biểu bài toán
   1. Mô tả bài toán

Phát biểu nội dung bài toán: trong lĩnh vực giao thông, nhận diện biển báo là một nhiệm vụ quan trọng giúp hỗ trợ tài xế và hệ thống xe tự hành đưa ra quyết định chính xác. Đồ án này xây dựng một hệ thống nhận dạng 5 loại biển báo giao thông bằng Deep Learning, gồm hai nhiệm vụ chính:

* Phân đoạn biển báo trong ảnh đầu vào để xác định vị trí biển báo.
* Phân loại biển báo để xác định loại biển báo sau khi đã phát hiện.

Hệ thống giúp tăng cường an toàn giao thông, có thể triển khai trên các phương tiện hỗ trợ lái xe hoặc hệ thống giám sát giao thông thông minh.

**Mô tả Input của bài toán:**

Ảnh màu chứa một hoặc nhiều biển báo giao thông.

Kích thước ảnh có thể thay đổi, góc nhìn và ánh sáng có thể khác nhau.

Ảnh (camera hành trình hoặc camera xe tự lái) có thể có nhiều đối tượng gây nhiễu như phương tiện, người đi đường.

**Mô tả Output của bài toán:**

Bước 1 - Phân đoạn (Segmentation): Ảnh đầu vào sẽ được xử lý để tạo mặt nạ (mask) xác định vùng biển báo.

Bước 2 - Phân loại (Classification): Sau khi xác định vùng chứa biển báo, hệ thống sẽ gán nhãn đúng cho từng biển báo trong ảnh.

Một số ví dụ minh họa: ảnh đầu vào có nhiều đối tượng như người đi bộ, ô tô và một biển báo "Cấm rẽ trái". Hệ thống sẽ phát hiện vùng biển báo trong ảnh và xác định đó là biển "Cấm rẽ trái".

* 1. Giới hạn bài toán

Để đảm bảo tính khả thi và hiệu suất của hệ thống nhận dạng biển báo giao thông, bài toán được giới hạn trong các phạm vi sau:

**Giới hạn hiệu suất và triển khai:**

Hệ thống chỉ nhận diện 5 loại biển báo giao thông phổ biến (biển báo giới hạn tốc độ), thay vì toàn bộ hệ thống biển báo. Do đó hiệu suất có thể bị ảnh hưởng nếu gặp các loại biển báo khác hoặc điều kiện môi trường chưa từng xuất hiện trong dữ liệu huấn luyện.

**Giới hạn về dữ liệu đầu vào:**

Ảnh đầu vào là ảnh chụp từ camera hành trình hoặc camera xe tự lái, không bao gồm dữ liệu từ ảnh vệ tinh hoặc ảnh vẽ đồ họa.

Ảnh có thể chứa nhiều đối tượng gây nhiễu như phương tiện giao thông, người đi đường, cây cối, nhưng hệ thống chỉ tập trung vào nhận diện biển báo.

Kích thước ảnh có thể thay đổi, nhưng phải nằm trong một phạm vi cố định để đảm bảo tính đồng nhất khi đưa vào mô hình.

**Giới hạn về điều kiện môi trường:**

Hệ thống hoạt động hiệu quả trong điều kiện thời tiết tốt, nhưng có thể bị ảnh hưởng trong điều kiện mưa lớn, sương mù dày đặc hoặc ánh sáng yếu.

Các biến dạng của biển báo do góc nhìn nghiêng quá mức, che khuất hoặc hư hỏng vật lý có thể làm giảm độ chính xác của mô hình.

**Giới hạn về mô hình sử dụng:**

Phân đoạn đối tượng trong ảnh được thực hiện OpenCV hoặc bằng U-Net, một kiến trúc mạnh mẽ nhưng yêu cầu tài nguyên tính toán cao.

Phân loại biển báo sử dụng ResNet50 hoặc AlexNet (một mô hình CNN cơ bản) nhưng có thể bị hạn chế khi xử lý các biển báo có sự khác biệt nhỏ về hình dạng.

Hệ thống không sử dụng các phương pháp nhận diện chữ để đọc thông tin văn bản trên biển báo (ví dụ: biển chỉ dẫn có chữ).

* 1. Bố cục đồ án

Đồ án gồm 3 chương chính, trình bày chi tiết quá trình xây dựng hệ thống nhận dạng biển báo giao thông bằng Deep Learning. Cụ thể:

Chương 1: Phát biểu bài toán – Chương này mô tả bài toán cần giải quyết, bao gồm phát biểu bài toán, mô tả dữ liệu đầu vào và đầu ra, các ví dụ minh họa, cũng như các giới hạn và ràng buộc của hệ thống.

Chương 2: Giải pháp đề xuất – Trình bày phương pháp tiếp cận để giải quyết bài toán, bao gồm phân tích dữ liệu, lựa chọn mô hình Deep Learning, thiết kế kiến trúc hệ thống và các kỹ thuật tiền xử lý dữ liệu.

Chương 3: Các thực nghiệm và đánh giá – Trình bày quá trình huấn luyện mô hình, các kết quả thực nghiệm, đánh giá hiệu suất mô hình bằng các chỉ số như độ chính xác (accuracy), độ nhạy (recall), độ đặc hiệu (precision), và thảo luận về kết quả đạt được.

Cuối cùng, đồ án kết luận về những đóng góp chính, hạn chế của hệ thống và hướng phát triển trong tương lai.

1. Giải pháp đề xuất
   1. Phân tích dữ liệu

2.1.1 Thống kê tập dữ liệu

Tập dữ liệu được sử dụng trong đồ án này là German Traffic Sign Recognition Benchmark (GTSRB), được lấy từ Kaggle. Đây là một bộ dữ liệu tiêu chuẩn để nhận diện biển báo giao thông, gồm các hình ảnh chụp biển báo giao thông tại Đức với nhiều điều kiện ánh sáng, góc quay và chất lượng hình ảnh khác nhau.

**Cấu trúc tập dữ liệu bao gồm:**

Tổng số ảnh: Hơn 50.000 ảnh thuộc 43 loại biển báo giao thông.

Tập huấn luyện (Train): Chứa nhiều hình ảnh biển báo với nhãn tương ứng, giúp mô hình học được các đặc trưng cần thiết.

Tập kiểm tra (Test): Được sử dụng để đánh giá độ chính xác của mô hình sau khi huấn luyện.

File csv cho 2 tập train và test để lưu kích thước, tọa độ và đường dẫn của ảnh.

**Phân tích dữ liệu ảnh:**

Ảnh trong tập dữ liệu có độ phân giải không đồng nhất.

Số lượng ảnh trung bình mỗi loại: Khoảng 1.000 – 1.500 ảnh/lớp, nhưng một số lớp có số lượng ảnh ít hơn.

Dữ liệu có tính mất cân bằng, với một số loại biển báo xuất hiện nhiều hơn so với các loại khác. Một số loại biển báo có rất ít ảnh, có thể gây overfitting khi huấn luyện mô hình.

**Thống kê kích thước ảnh:**

Kích thước nhỏ nhất: 15x15 pixels

Kích thước lớn nhất: 250x250 pixels

Kích thước trung bình: 32x32 pixels

Do kích thước ảnh không đồng nhất, cần chuẩn hóa kích thước trước khi đưa vào mô hình để đảm bảo tính đồng nhất. Hầu hết các ảnh có độ phân giải nhỏ, do đó sử dụng mô hình CNN như AlexNet hoặc ResNet50 là phù hợp.

2.1.2 Tiền xử lý dữ liệu

Do bài toán chỉ giới hạn nhận diện 5 loại biển báo giao thông nên nhóm chỉ lọc và lấy các mẫu có nhãn từ 0 – 4. Để tăng hiệu quả huấn luyện mô hình, nhóm đã áp dụng các phương pháp tiền xử lý sau:

**Chuẩn hóa kích thước ảnh:** resize ảnh về 227x227 pixels (AlexNet), 256x256 (ResNet50) và 256x256 (Unet) để đồng bộ dữ liệu.

**Chuyển đổi màu sắc:** Chuyển đổi tất cả ảnh sang định dạng RGB.

**Tăng cường dữ liệu (Data Augmentation):** sử dụng các kĩ thuật như xoay ngẫu nhiên (RandomRotation), lật ngang ảnh ngẫu nhiên (RandomHorizontalFlip), điều chỉnh độ sáng, độ tưởng phảm ảnh (ColorJitter), chuyển đổi ảnh sang dạng tensor (ToTensor).

**Chia tập dữ liệu:**

Training: Dùng để huấn luyện mô hình (90%).

Validation: Dùng để tinh chỉnh tham số mô hình (10%).

Test: Dùng để đánh giá mô hình sau khi huấn luyện.

* 1. Thiết kế mô hình

2.2.1 Mô hình AlexNet

**Input của mô hình:** Ảnh đầu vào có 3 kênh màu (RGB). Kích thước ảnh là 227×227 (chuẩn AlexNet).Dữ liệu đầu vào: mỗi ảnh là một biển báo giao thông từ tập dữ liệu GTSRB và đã được xử lý và chuẩn hóa.

**Output của mô hình:** Kích thước đầu ra: mô hình dự đoán 5 lớp biển báo (0-4), tương ứng với 5 giá trị đầu ra.

**Cấu trúc của mô hình:** Gồm 5 lớp Convolution, 3 lớp Fully Connected, sử dụng Batch Normalization và Dropout để tăng độ ổn định.

**Hàm kích hoạt (activation function):**

* ReLu trong tất cả các lớp ẩn: loại bỏ giá trị âm và giữ giá trị dương.
* Softmax trong lớp đầu ra để tính xác suất (sử dụng CrossEntropyLoss nên không cần thêm Softmax vào trong mô hình).

Bảng mô tả chi tiết các tầng (layer) của kiến trúc:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tầng | Loại | Số lượng filters | Kích thước filter | Stride | Padding | Kích thước đầu ra |
| Input | Ảnh RGB | - | - | - | - | 3x227x227 |
| Conv1 | Convolution | 96 | 11 x 11 | 4 | 0 | 96x55x55 |
| BN1 | BatchNorm2d | - | - | - | - | 96x55x55 |
| ReLu | Activation | - | - | - | - | 96x55x55 |
| MaxPool1 | MaxPooling | - | 3 x 3 | 2 | - | 96x27x27 |
| Conv2 | Convolution | 256 | 5 x 5 | 1 | 2 | 256x27x27 |
| BN2 | BatchNorm2d | - | - | - | - | 256x27x27 |
| ReLu | Activation | - | - | - | - | 256x27x27 |
| MaxPool2 | MaxPooling | - | 3 x 3 | 2 | - | 256x13x13 |
| Conv3 | Convolution | 384 | 3 x 3 | 1 | 1 | 384x13x13 |
| BN3 | BatchNorm2d | - | - | - | - | 384x13x13 |
| ReLu | Activation | - | - | - | - | 384x13x13 |
| Conv4 | Convolution | 384 | 3 x 3 | 1 | 1 | 384x13x13 |
| BN4 | BatchNorm2d | - | - | - | - | 384x13x13 |
| ReLu | Activation | - | - | - | - | 384x13x13 |
| Conv5 | Convolution | 256 | 3 x 3 | 1 | 1 | 256x13x13 |
| BN5 | BatchNorm2d | - | - | - | - | 256x13x13 |
| ReLu | Activation | - | - | - | - | 256x13x13 |
| MaxPool3 | MaxPooling | - | 3 x 3 | 2 | - | 256x6x6 |
| Flatten | Kéo thành vector | - | - | - | - | 9216 |
| FC1 | Fully Connected | 4096 | - | - | - | 4096 |
| ReLu | Actication | - | - | - | - | 4096 |
| Dropout | Dropout (0.5) | - | - | - | - | 4096 |
| FC2 | Fully Connected | 4096 | - | - | - | 4096 |
| ReLu | Actication | - | - | - | - | 4096 |
| Dropout | Dropout (0.5) | - | - | - | - | 4096 |
| FC3 | Fully Connected | 5 | - | - | - | 5 |
| Softmax | Activation | - | - | - | - | 5 |

A screenshot of a computer

AI-generated content may be incorrect.

Hình 1: Bảng thống kê các tầng mô hình AlexNet

A diagram of a computer program

AI-generated content may be incorrect.

Hình 2: Minh họa kiến trúc mô hình AlexNet

2.2.2 Mô hình ResNet50

**Input của mô hình:** Ảnh đầu vào có 3 kênh màu (RGB). Kích thước ảnh là 256x256. Dữ liệu đầu vào: mỗi ảnh là một biển báo giao thông từ tập dữ liệu GTSRB, đã được xử lý và chuẩn hóa trước khi đưa vào mô hình.

**Output của mô hình:** Kích thước đầu ra: mô hình dự đoán 5 lớp biển báo (0-4), tương ứng với 5 giá trị đầu ra.

**Cấu trúc của mô hình:** Gồm 5 giai đoạn chính với tổng cộng 50 lớp.

Các khối Residual giúp mô hình học sâu mà không bị mất thông tin do hiện tượng vanishing gradient.

Sử dụng Batch Normalization để cải thiện độ ổn định trong quá trình huấn luyện.

**Hàm kích hoạt (activation function):**

* ReLU trong tất cả các lớp ẩn: loại bỏ giá trị âm và giữ giá trị dương.
* Softmax trong lớp đầu ra để tính xác suất (sử dụng CrossEntropyLoss nên không cần thêm Softmax vào trong mô hình).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tầng | Loại | Filter | Kích thước filter | Stride | Padding | Kích thước đầu ra |
| Input | Ảnh RGB | - | - | - | - | 3x256x256 |
| Conv1 | Conv2D | 64 | 7x7 | 2 | 3 | 64x128x128 |
| BN1 | BatchNorm2D | 64 | - | - | - | 64x128x128 |
| ReLu | ReLu | - | - | - | - | 64x128x128 |
| Pool1 | MaxPool2D | - | 3x3 | 2 | 1 | 64x64x64 |
| Layer1 | Residual Block x3 | 256 | 1x1,3x3,1x1 | 1 | - | 256x64x64 |
| Layer2 | Residual Block x4 | 512 | 1x1,3x3,1x1 | 2 | - | 512x32x32 |
| Layer3 | Residual Block x6 | 1024 | 1x1,3x3,1x1 | 2 | - | 1024x16x16 |
| Layer4 | Residual Block x3 | 2048 | 1x1,3x3,1x1 | 2 | - | 2048x8x8 |
| Pool2 | AdaptiveAvg  Pool2D | - | - | - | - | 2048x1x1 |
| FC | Fully  Connected | - | - | - | - | 5 |

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

Hình 3: Bảng thống kê các tầng mô hình ResNet50

A diagram of a machine learning process

AI-generated content may be incorrect.

Hình 4: Minh họa kiến trúc mô hình ResNet50

2.2.3 Mô hình UNet

**Input của mô hình:** sử dụng các kĩ thuật thị giác máy tính (OpenCV) để tạo dữ liệu train mô hình từ tập dataset ban đầu.

Ảnh gốc (Original Image): ảnh biển báo gốc, chứa cả biển báo và background.

Mask (Binarry Mask): trong đó: Màu trắng (1): Biển báo, Màu đen (0): Background.

Overlay (Ảnh gốc + Mask chồng lên nhau): Để kiểm tra xem mask có đúng với biển báo không.



Hình 5: Trực quan Input đầu vào UNet

**Output của mô hình:** sử dụng giá trị trong khoảng [0,1] để phân biệt giữa foreground (biển báo) và background.Ảnh mask nhị phân (black & white), trong đó pixel của biển báo có giá trị 1 (trắng), còn background có giá trị 0 (đen).

**Cấu trúc của mô hình:** gồm các khối encoder (downsampling) và decoder (unsampling) với tổng cộng 4 mức pooling.Sử dụng các lớp Conv2D, Batch Normalization, ReLU, MaxPooling, và ConvTranspose2D.

**Hàm kích hoạt (activation function):**

* ReLU trong tất cả các lớp ẩn: giúp mô hình học được các đặc trưng phi tuyến tốt hơn.
* Sigmoid trong lớp đầu ra để đưa giá trị đầu ra về khoảng [0,1] phù hợp với phân đoạn nhị phân.

**Hàm mất mát:** Sử dụng Binary Cross Entropy Loss (BCE Loss) do bài toán là phân đoạn nhị phân.

Bảng mô tả chi tiết các tầng (layer) của kiến trúc:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tầng | Loại | Số lượng filters | Kích thước filter | Stride | Padding | Kích thước đầu ra |
| Input | Ảnh RGB | - | - | - | - | 3x256x256 |
| Encoder1 | Convolutional | 64 | 3x3 | 1 | 1 | 64x256x256 |
| BN1 | BatchNorm2d | - | - | - | - | 64x256x256 |
| ReLu | Activation | - | - | - | - | 64x256x256 |
| MaxPool1 | MaxPooling | - | 2x2 | 2 | 0 | 64x128x128 |
| Encoder2 | Convolutional | 128 | 3x3 | 1 | 1 | 128x128x128 |
| BN2 | BatchNorm2d | - | - | - | - | 128x128x128 |
| ReLu | Activation | - | - | - | - | 128x128x128 |
| MaxPool2 | MaxPooling | - | 2x2 | 2 | 0 | 128x64x64 |
| Encoder3 | Convolutional | 256 | 3x3 | 1 | 1 | 256x64x64 |
| BN3 | BatchNorm2d | - | - | - | - | 256x64x64 |
| ReLu | Activation | - | - | - | - | 256x64x64 |
| MaxPool3 | MaxPooling | - | 2x2 | 2 | 0 | 256x32x32 |
| Encoder4 | Convolutional | 512 | 3x3 | 1 | 1 | 512x32x32 |
| BN4 | BatchNorm2d | - | - | - | - | 512x32x32 |
| ReLu | Activation | - | - | - | - | 512x32x32 |
| MaxPool4 | MaxPooling | - | 2x2 | 2 | 0 | 512x16x16 |
| Bottleneck | Convolutional | 1024 | 3x3 | 1 | 1 | 1024x16x16 |
| BN5 | BatchNorm2d | - | - | - | - | 1024x16x16 |
| ReLu | Activation | - | - | - | - | 1024x16x16 |
| Upconv4 | ConvTranspose2d | 512 | 2x2 | 2 | 0 | 512x32x32 |
| Decoder4 | Convolutional | 512 | 3x3 | 1 | 1 | 512x32x32 |
| Upconv3 | ConvTranspose2d | 256 | 2x2 | 2 | 0 | 256x64x64 |
| Decoder3 | Convolutional | 256 | 3x3 | 1 | 1 | 256x64x64 |
| Upconv2 | ConvTranspose2d | 128 | 2x2 | 2 | 0 | 128x128x128 |
| Decoder2 | Convolutional | 128 | 3x3 | 1 | 1 | 128x128x128 |
| Upconv1 | ConvTranspose2d | 64 | 2x2 | 2 | 0 | 64x256x256 |
| Decoder1 | Convolutional | 64 | 3x3 | 1 | 1 | 64x256x256 |
| Output | Convolutional | 1 | 1x1 | 1 | 0 | 1x256x256 |
| Sigmoid | Activation | - | - | - | - | 1x256x256 |

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.

Hình 6: Bảng thống kê các tầng mô hình UNet

A diagram of a diagram

AI-generated content may be incorrect.Hình 7: Minh họa kiến trúc mô hình Unet.

(a)Kiến trúc mạng. (b)Chi tiết khối tích chập còn lại

1. Các thực nghiệm, đánh giá và triển khai
   1. Các thực nghiệm

Trong phần này, chúng tôi trình bày các thực nghiệm đánh giá hiệu suất của các mô hình, bao gồm quá trình huấn luyện, đánh giá. Đối với yêu cầu phân loại biển báo, chúng tôi sẽ thực hiện so sánh hiệu suất cũng như về độ chính xác của 2 mô hình AlexNet và ResNet50. Đối với yêu cầu phân đoạn, chúng tôi sẽ thực hiện so sánh giữa mô hình UNet và các phương pháp kĩ thuật thị giác máy tính (OpenCV) để chọn ra mô hình tốt nhất và xây dựng giao diện.

**Môi trường thực nghiệm:**

* Phần cứng: GPU T4x2, GPU P100.
* Phần mềm: Python 3.9, Pytorch 2.0, Google Colab/Kaggle Notebook.

3.1.1 Thực nghiệm AlexNet

**Hàm loss:** trong bài toán phân loại biển báo giao thông với 5 lớp, chúng tôi sử dụng hàm mất mát **CrossEntropyLoss**. Hàm này được lựa chọn vì:

* Nó tích hợp cả hàm softmax (vì vậy không cần thêm softmax vào lớp đầu ra).
* Nó giúp mô hình học cách phân biệt các lớp thông qua việc tối thiểu hóa sự chênh lệch giữa phân phối dự đoán và phân phối thực tế.

**Hàm cost:** giá trị hàm loss sau mỗi batch (và trung bình qua các epoch) được sử dụng làm hàm cost để đánh giá sự hội tụ của mô hình trong quá trình huấn luyện.

**Mô tả các siêu tham số:**

Phương pháp học (Optimizer): Sử dụng thuật toán Adam, giúp tối ưu hàm loss hiệu quả và ổn định.

Learning rate: Giá trị này được chọn qua quá trình thử nghiệm để đảm bảo hội tụ ổn định mà không gây ra sự dao động quá mức của hàm loss.

Phương pháp khởi tạo trọng số: Sử dụng phương pháp khởi tạo mặc định của PyTorch (thường là Kaiming Uniform cho các lớp Conv2d).

Batch size: Số lượng mẫu được đưa vào mô hình cùng lúc, giúp tối ưu hiệu suất tính toán trên GPU.

Số Epoch: 20 (Số vòng lặp qua toàn bộ tập dữ liệu huấn luyện)

Số layer: Mô hình AlexNet của chúng tôi có 5 tầng Convolution và 3 tầng Fully Connected (FC).

Số neuron trong layer: Các tầng Conv có số filter lần lượt là 96, 256, 384, 384, 256. Các tầng FC có số neuron là 4096, 4096, và cuối cùng là 5 (tương ứng với 5 lớp).

Hàm activation: ReLU được sử dụng sau mỗi lớp Convolution và Fully Connected, giúp loại bỏ giá trị âm và đưa ra tính phi tuyến cần thiết.

Dropout: Áp dụng dropout với tỉ lệ 0.5 trong các tầng Fully Connected để giảm nguy cơ overfitting.

**Thực nghiệm 1:** Thiết lâp: Learning rate 0.0001. Batch size: 64. Thay đổi tham số đầu vào cho bước Data Argumentation điều chỉnh độ sáng, độ tương phản (transform ColorJitter(brigthness = 0.15, contrast = 0.15, saturation = 0.15, hue = 0.15)

**Biểu đồ huấn luyện:**

* Training Loss: Biểu đồ Loss cho thấy giá trị mất mát giảm dần qua các epoch.

A graph of a graph of a graph

AI-generated content may be incorrect.

* Đánh giá mô hình thông qua Accuracy,Accuracy trên từng lớp, Confusion Matrix và Classcification Report:

A screenshot of a graph

AI-generated content may be incorrect.A graph of a graph of a confusion matrix

AI-generated content may be incorrect. 

A number of vehicles and numbers

AI-generated content may be incorrect.

**Thực nghiệm 2:** Thiết lâp: Learning rate 0.0001. Batch size: 64. Thay đổi tham số đầu vào cho bước Data Argumentation điều chỉnh độ sáng, độ tương phản (transform ColorJitter(brigthness = 0.2, contrast = 0.2, saturation = 0.2, hue = 0.2) thay vì 0.15 đối với thực nghiệm trên.

**Biểu đồ huấn luyện:**

* Training Loss: Biểu đồ Loss cho thấy giá trị mất mát giảm dần qua các epoch.

A graph of a graph of a graph

AI-generated content may be incorrect.

* Đánh giá mô hình thông qua Accuracy, Accuracy trên từng lớp, Confusion Matrix và Classcification Report:

A graph of confusion matrix

AI-generated content may be incorrect.A number of vehicles with numbers and symbols

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.

**Nhận xét về các thực nghiệm:** ta thấy thực nghiệm 1 có các chỉ số cao hơn thực nghiệm 2:

**Độ chính xác tổng thể của mô hình:**

Mô hình đạt 96.00% độ chính xác trên tập validation, cho thấy nó hoạt động tốt trong việc nhận diện biển báo tốc độ.

F1-score trung bình đạt 95.9%, phản ánh sự cân bằng giữa Precision và Recall.

**Hiệu suất trên từng lớp:**

Lớp 20km/h có Recall 98%, tức là mô hình nhận diện rất tốt các mẫu thực sự thuộc lớp này (chỉ bỏ sót 1 mẫu).

Lớp 30km/h có Precision 99.1% và Recall 95.13%, cho thấy hầu hết dự đoán là chính xác, nhưng vẫn có một số nhầm lẫn.

Lớp 50km/h (96.35%), 60km/h (94.12%) và 70km/h (97.16%) có độ chính xác cao nhưng vẫn có một số nhầm lẫn nhỏ.

**Phân tích nhầm lẫn (Confusion Matrix):**

Một số mẫu của lớp 50km/h bị nhầm lẫn thành 60km/h (2 mẫu) và 70km/h (3 mẫu), điều này có thể do biển báo có hình dạng hoặc màu sắc tương tự.

Lớp 60km/h có độ chính xác thấp nhất (94.12%) với 3 mẫu bị nhầm thành 70km/h và 2 mẫu bị nhầm với 50km/h.

Tuy nhiên, không có lỗi nghiêm trọng, chỉ có một số ít trường hợp bị nhầm lẫn.

3.1.2 Thực nghiệm ResNet50

**Mô tả các siêu tham số:**

Batch\_size: dữ liệu được đưa vào theo từng Batch 16 ảnh / lần.

Hệ số học: sử dụng thuật toán tối ưu Adam với tốc độ học Leaning Rate thay đổi theo mỗi thực nghiệm.

Epochs: mô hình được huấn luyện tối đa trong 20 epochs (có thể dừng sớm nếu không cải thiện).

**Thực nghiệm 1:** Thiết lập: Learning Rate: 0.0001, Batch\_size: 16. Thay đổi tham số đầu vào cho bước Data Argumentation điều chỉnh độ sáng, độ tương phản (transform ColorJitter(brigthness = 0.15, contrast = 0.15, saturation = 0.15, hue = 0.15).

**Biểu đồ huấn luyện:**

* Training Loss: Biểu đồ Loss cho thấy giá trị mất mát giảm dần qua các epoch.

A graph of a graph of a line

AI-generated content may be incorrect.

* Đánh giá mô hình thông qua Accuracy, Accuracy trên từng lớp, Confusion Matrix và Classcification Report

A graph of a graph of confusion matrix

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**Thực nghiệm 2:** Thiết lập: Learning Rate: 0.0001, Batch\_size: 16. Thay đổi tham số đầu vào cho bước Data Argumentation điều chỉnh độ sáng, độ tương phản (transform ColorJitter(brigthness = 0.2, contrast = 0.2, saturation = 0.2, hue = 0.2) thay vì 0.15 như thực nghiệm 1.

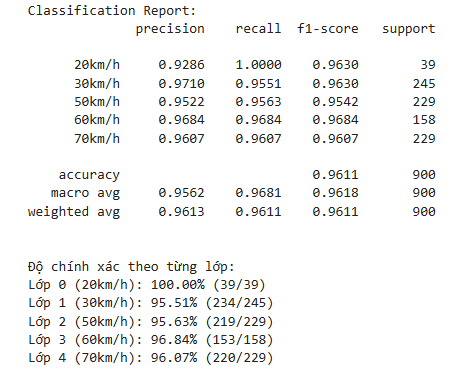
**Biểu đồ huấn luyện:**

* Training Loss: Biểu đồ Loss cho thấy giá trị mất mát giảm dần qua các epoch.

A graph of a graph

AI-generated content may be incorrect.

* Đánh giá mô hình thông qua Accuracy, Accuracy trên từng lớp, Confusion Matrix và Classcification Report

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.

**Nhận xét về các thực nghiệm:** ta thấy thực nghiệm 1 có các chỉ số đánh giá cao hơn thực nghiệm 2:

**Độ chính xác tổng thể của mô hình:**

Mô hình đạt 96.56% độ chính xác trên tập validation, cho thấy hiệu suất rất tốt trong nhận diện biển báo tốc độ.

F1-score trung bình đạt 97.20% (macro avg) và 96.56% (weighted avg), phản ánh sự cân bằng tốt giữa Precision và Recall.

Early stopping được kích hoạt sau epoch 14 với loss là 0.0593, cho thấy mô hình đã hội tụ tốt.

**Hiệu suất trên từng lớp:**

Lớp 20km/h đạt precision, recall và F1-score đều là 100%, thể hiện nhận diện hoàn hảo (31/31 mẫu).

Lớp 30km/h có precision 94.94% và recall 95.74%, cho thấy hiệu suất tốt nhưng có một số nhầm lẫn.

Lớp 50km/h có precision 98.76% và recall 95.20%, với 238/250 mẫu được phân loại đúng.

Lớp 60km/h có precision 98.81% và recall 96.51%, với 166/172 mẫu được phân loại đúng.

Lớp 70km/h có precision 93.72% và recall 98.58%, với 209/212 mẫu được phân loại đúng.

**Phân tích nhầm lẫn (Confusion Matrix):**

Lớp 30km/h có 7 mẫu bị nhầm thành 70km/h và 3 mẫu bị nhầm thành 50km/h.

Lớp 50km/h có 5 mẫu bị nhầm thành 30km/h, 5 mẫu bị nhầm thành 70km/h và 2 mẫu bị nhầm thành 60km/h.

Lớp 60km/h có 4 mẫu bị nhầm thành 30km/h và 2 mẫu bị nhầm thành 70km/h.

Lớp 70km/h có 3 mẫu bị nhầm thành 30km/h.

Lớp 20km/h không có trường hợp nhầm lẫn nào.

3.1.3 Thực nghiệm UNet

Hàm loss và hàm coss: Trong quá trình huấn luyện mô hình UNet, chúng tôi sử dụng hàm mất mát kết hợp (Combined Loss) bao gồm: Binary Cross Entropy (BCE) Loss giúp mô hình học phân biệt giữa các điểm nền và đối tượng trong ảnh.

Mô tả các tham số:

Phương pháp học: Adam Optimizer

Learning rate: Giá trị này được chọn qua quá trình thử nghiệm để đảm bảo hội tụ ổn định mà không gây ra sự dao động quá mức của hàm loss.

Epochs: mô hình được huấn luyện tối đa trong 20 epochs (có thể dừng sớm nếu không cải thiện).

Số Layer: 4 mức downsampling và 4 mức upsampling.

Số Neuron trong mỗi Layer: Tăng dần từ 64 đến 1024.

Hàm Activation: ReLU cho các lớp ẩn, Sigmoid cho đầu ra

Dropout: 0.3.

Thực nghiệm 1: Thiết lập: Learning rate 0.0001. Batch size: 64.

**Biểu đồ loss và IoU theo từng epoch:**

A graph of loss and loss

AI-generated content may be incorrect.

**Phân tích quá trình huấn luyện:**

Mô hình đã chạy trong 20 epoch, kết thúc với loss: 0.2281 và IoU: 0.6927 ở epoch cuối cùng.

Training loss giảm đều từ khoảng 0.33 xuống 0.23, cho thấy mô hình đang học tốt trên tập huấn luyện.

Không có dấu hiệu rõ ràng của overfitting vì loss tiếp tục giảm đến epoch cuối, chưa có đáy cong hay đi ngang.

**Đánh giá IoU (Intersection over Union):**

Metric IoU trên tập validation đạt 0.6927 ở epoch cuối, đây là giá trị khá tốt cho bài toán phân đoạn.

Đồ thị IoU tăng không đều, có nhiều dao động lớn giữa các epoch, cho thấy sự không ổn định trong quá trình học.

Có một số giai đoạn IoU giảm nhẹ (khoảng epoch 9-11) trước khi tiếp tục tăng, điều này có thể liên quan đến sự phức tạp của dữ liệu validation.

**Đánh giá tổng quan:**

Mô hình đang cải thiện cả về loss và IoU theo thời gian, cho thấy quá trình học đang hiệu quả.

Độ dao động của IoU trên tập validation cao hơn nhiều so với độ dao động của loss trên tập training, điều này có thể do một số mẫu trong tập validation khó phân đoạn hơn.

Giá trị IoU cuối cùng là 0.6927 có thể được coi là khá tốt, nhưng còn không gian để cải thiện

## **3.2 Đánh giá**

### **3.2.1 So sánh 2 mô hình phân loại AlexNet và ResNet50**

Dựa vào kết quả đánh giá của hai mô hình AlexNetvà ResNet50, nhóm sẽ so sánh và chọn ra mô hình tốt hơn để kết hợp với UNet và triển khai web.

So sánh chỉ số chính:

|  |  |  |  |
| --- | --- | --- | --- |
| Chỉ số | AlexNet | ResNet50 | Mô hình tốt hơn |
| Độ chính xác tổng thể | 96% | 96.56% | ResNet50 |
| Macro avg F1-score | 95.1% | 97.20% | ResNet50 |
| Weighted avg F1-score | 95.9% | 96.56% | ResNet50 |
| Early stopping | Epoch 18 | Epoch 14 | ResNet50 (hội tụ nhanh hơn) |

So sánh hiệu suất trên từng lớp:

|  |  |  |  |
| --- | --- | --- | --- |
| Lớp | AlexNet (F1-score) | ResNet50 (F1-score) | Mô hình tốt hơn |
| 20km/h | 91.6% | 100% | ResNet50 |
| 30km/h | 96.2% | 95.34% | AlexNet |
| 50km/h | 97.7% | 96.95% | AlexNet |
| 60km/h | 94.4% | 97.65% | ResNet50 |
| 70km/h | 95.8% | 96.09% | ResNet50 |

Kết luận: Mặc dù ResNet50 có các chỉ số cao hơn so với mô hình AlexNet. Nhưng việc xử lý dữ liệu mất cân bằng thì AlexNet làm tốt hơn. Vì ResNet có số lượng tầng và tham số nhiều hơn nên thời gian xử lý cũng như yêu cầu tài nguyên nhiều hơn (thích hợp cho những bài toán phức tạp). Điều này, đối với bài toán đơn giản như phân loại biển báo là điều không cần thiết. Vì vậy nhóm sử dụng mô hình AlexNet.

3.2.2 Mô hình UNet

Sau khi thực hiện xây dựng, đánh giá và kết hợp UNet với mô hình ResNet50 cho ra một mô hình dự đoán tổng thể bao gồm phân đoạn và phân loại. Nhóm có một vài nhận xét và thay đổi sau:

Mô hình (UNet + ResNet50) khi phân đoạn và phân loại cho ra kết quả rất tốt đối với một số trường hợp như:

* Ảnh chỉ chứa biển báo mà không có các vật thể khác (người, phương tiện giao thông).
* Ảnh chứa biển báo không quá xa và không bị mờ hoặc các tác nhân gây nhiễu khác.

**Giải pháp:** thay vì sử dụng UNet để phân đoạn, nhóm đã sử dụng một vài kĩ thuật trong thị giác máy tính (opencv). Vì các biển báo đa phần có hình dạng và màu sắc giống nhau nên nhóm đã dựa vào ngưỡng màu cũng như là hình dạng để phân đoạn biển báo với background.

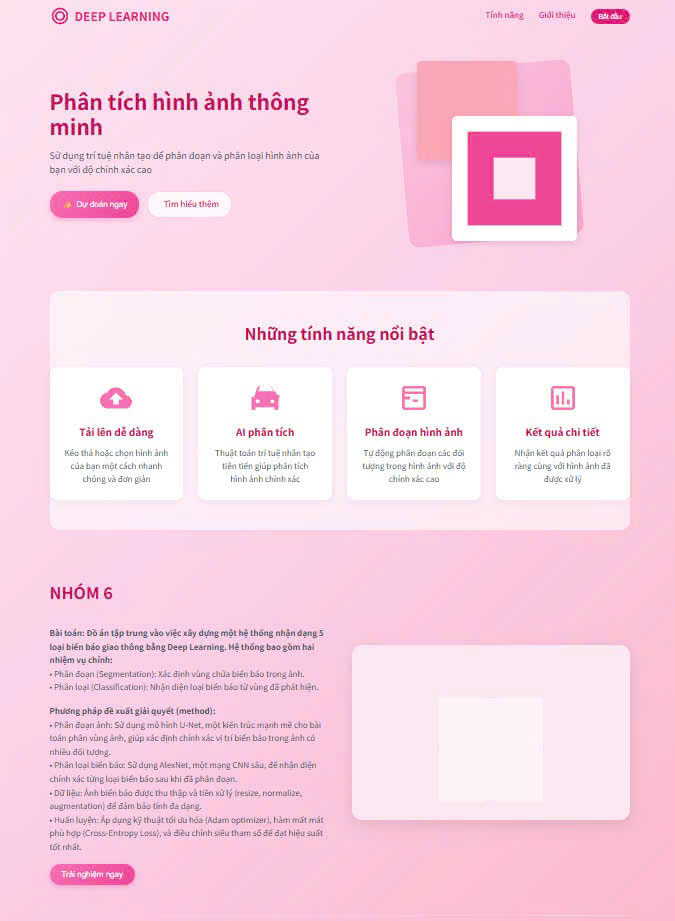
**Kết quả:** khi phân đoạn với các kĩ thuật thị giác máy tính đã cho ra kết quả rất tốt với mọi trường hợp.

3.3 Triển khai

Nhóm thực hiện tạo API với Flask API sau đó triển khai web với Reacjs:

**Giao diện trang chủ:** chứa các thông tin về trang web và các tính năng nổi bật

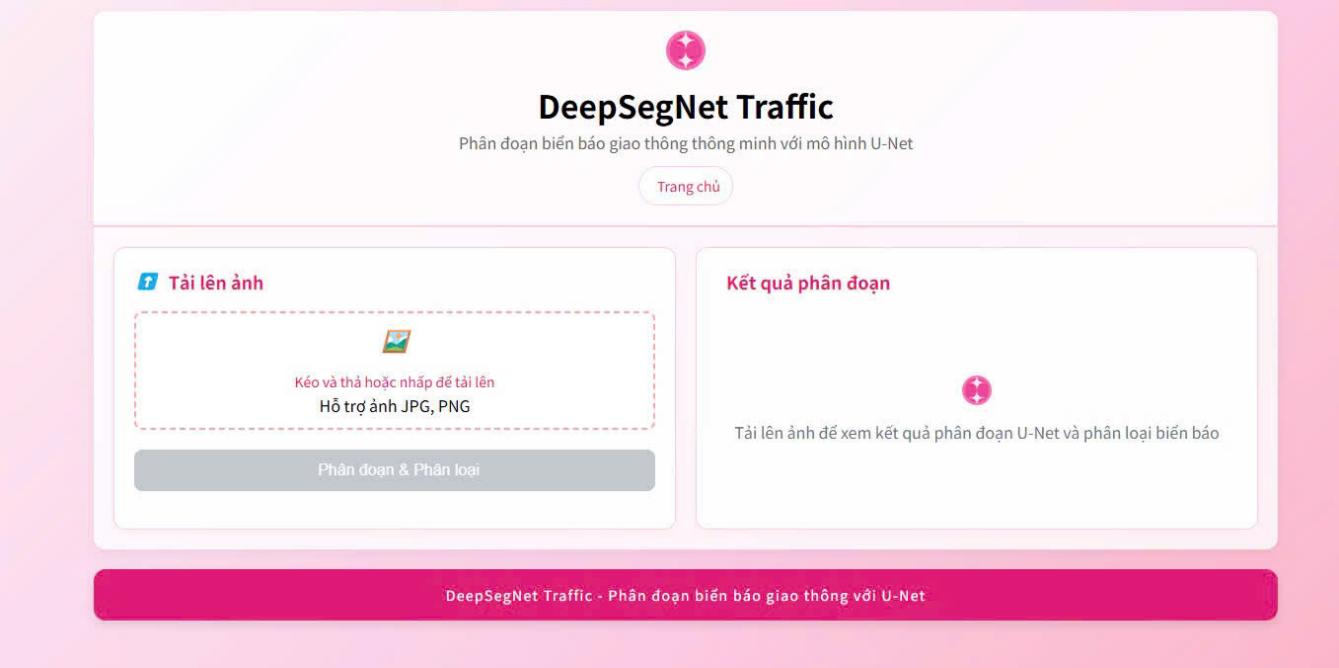
**Cách sử dụng chức năng:** khi người dùng nhấn vào nút dự đoán ngay thì hệ thống sẽ chuyển sang giao diện dự đoán biển báo



Hình 8: Giao diện trang chủ

**Giao diện dự đoán:** chứa các thông tin về trang web và các tính năng nổi bật

**Cách sử dụng các chức năng:** người dùng tải ảnh từ thiết bị lên hệ thống. Sau đó nhấn nút “Phân đoạn và Phân loại”.



Hình 9: Giao diện dự đoán

A screenshot of a computer

AI-generated content may be incorrect.**Giao diện kết quả sau dự đoán:** trả về kết quả dự đoán biển báo

Hình 10: Giao diện kết quả dự đoán

KẾT LUẬN

**Những điều đã làm được**

Trong quá trình huấn luyện mô hình, chúng tôi đã sử dụng WeightedRandomSampler từ PyTorch để giải quyết vấn đề mất cân bằng dữ liệu giữa các lớp. Đây là một cải tiến quan trọng giúp mô hình học được đặc trưng từ tất cả các lớp một cách công bằng, thay vì bị lệch về các lớp có số lượng mẫu lớn hơn.

Cụ thể, WeightedRandomSampler cho phép gán trọng số khác nhau cho từng mẫu dữ liệu, đảm bảo rằng những lớp có số lượng mẫu ít hơn sẽ được lấy mẫu thường xuyên hơn trong mỗi batch huấn luyện. Điều này giúp cải thiện khả năng tổng quát của mô hình, giảm thiểu tình trạng dự đoán sai cho các lớp hiếm, đồng thời tránh được việc phải sử dụng các phương pháp khác như oversampling hoặc undersampling, vốn có thể làm mất đi tính đa dạng của dữ liệu.

Triển khai thành công mô hình AlexNet để giải quyết bài toán phân loại. Quá trình thực hiện bao gồm việc chuẩn bị dữ liệu, tiền xử lý và huấn luyện mô hình với các siêu tham số đã được tối ưu hóa. Chúng tôi đã lựa chọn phương pháp học có giám sát, sử dụng learning rate = 0.0001, batch size = 64, và huấn luyện trong 20 epoch để đảm bảo mô hình có thể học tốt từ dữ liệu mà không bị quá khớp.

Chúng tôi cũng áp dụng hàm kích hoạt ReLU để tăng khả năng học phi tuyến tính của mô hình, cùng với dropout = 0.5 nhằm giảm overfitting. Ngoài ra, quá trình huấn luyện còn sử dụng 5 layers với 8197 neurons mỗi layer, giúp mô hình có đủ khả năng học được các đặc trưng phức tạp trong dữ liệu. Kết quả huấn luyện được đánh giá thông qua biểu đồ loss và accuracy, từ đó phân tích và nhận xét về hiệu suất của mô hình.

Cuối cùng, sau quá trình thực nghiệm, chúng tôi đã rút ra một số đánh giá quan trọng về mô hình, bao gồm việc cải thiện độ chính xác bằng cách tinh chỉnh learning rate hoặc dropout, cũng như xem xét khả năng mở rộng mô hình để cải thiện hiệu suất trên tập dữ liệu lớn hơn.

**Những điều chưa làm được**

Mặc dù đã triển khai thành công mô hình AlexNet và thu được kết quả khả quan, nhưng vẫn còn một số hạn chế chưa được khắc phục trong đồ án này. Trước hết, việc tối ưu siêu tham số chưa được thực hiện một cách toàn diện. Dù đã thử nghiệm với một số giá trị learning rate, batch size, và dropout, nhưng vẫn chưa có quá trình điều chỉnh tự động hoặc sử dụng phương pháp grid search hoặc random search để tìm ra tổ hợp siêu tham số tối ưu nhất.

Ngoài ra, mô hình hiện tại chưa được so sánh với các kiến trúc mạng khác. AlexNet là một mô hình mạnh mẽ, nhưng có thể chưa phải là lựa chọn tốt nhất cho bài toán này. Việc thử nghiệm với các mô hình tiên tiến hơn như VGG, ResNet hoặc EfficientNet sẽ giúp đánh giá hiệu suất một cách khách quan hơn và có thể cải thiện độ chính xác của mô hình.

Một điểm hạn chế nữa là chưa có bước tối ưu hóa hiệu suất tính toán. Mô hình hiện tại chủ yếu chạy trên GPU nhưng chưa tận dụng hết các tài nguyên phần cứng. Chưa có thử nghiệm với các phương pháp như data augmentation nâng cao, sử dụng tensor cores hay các kỹ thuật pruning và quantization để giảm thời gian huấn luyện và tối ưu bộ nhớ.

Cuối cùng, đồ án chưa kiểm tra khả năng tổng quát hóa của mô hình trên dữ liệu thực tế. Mặc dù mô hình đã hoạt động tốt trên tập kiểm tra, nhưng chưa có thử nghiệm trên tập dữ liệu ngoài hoặc dữ liệu đến từ nguồn khác, điều này có thể ảnh hưởng đến tính ứng dụng thực tế của mô hình trong môi trường thật.

**Hướng phát triển**

Trước tiên, việc tối ưu hóa siêu tham số sẽ được cải thiện bằng cách áp dụng các phương pháp như Grid Search, Random Search, hoặc thậm chí là Bayesian Optimization để tìm ra bộ siêu tham số phù hợp nhất, giúp nâng cao hiệu suất của mô hình. Việc thử nghiệm với nhiều learning rate schedulers như ReduceLROnPlateau hoặc Cosine Annealing cũng có thể giúp tối ưu tốc độ hội tụ.

Ngoài ra, mô hình AlexNet có thể được so sánh với các kiến trúc mạng tiên tiến hơn như VGG, ResNet, EfficientNet, hoặc thử nghiệm các mô hình transformer-based như ViT (Vision Transformer). Điều này sẽ giúp đánh giá hiệu suất của mô hình hiện tại so với các kiến trúc hiện đại hơn, từ đó có lựa chọn phù hợp hơn cho bài toán.

Một hướng phát triển quan trọng khác là kiểm tra tính tổng quát hóa của mô hình. Việc đánh giá mô hình trên tập dữ liệu thực tế hoặc các bộ dữ liệu đến từ nhiều nguồn khác nhau sẽ giúp kiểm tra khả năng áp dụng trong thực tế. Nếu phát hiện overfitting, có thể áp dụng các phương pháp như data augmentation nâng cao, regularization mạnh hơn, hoặc transfer learning từ các mô hình đã được huấn luyện trước để cải thiện tính linh hoạt của mô hình.

Triển khai hệ thống phân đoạn và phân loại biển báo xử lí thời gian thực. Áp dụng các mô hình Deep Learning nâng cao như YOLO, FasterRCNN để cho ra kết quả với độ chính xác tốt hơn.

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Wellytambunan, "Youtube," Machine Learning Indonesia, 01 07 2022. [Online]. Available: https://www.youtube.com/watch?v=pg\_r2nblp10. |
| [2] | P. D. Tùng, "Tìm Hiểu Về Mạng Neural Network AlexNet," 11 7 2018. [Online]. Available: https://www.phamduytung.com/blog/2018-06-15-understanding-alexnet/. |
| [3] | Y. Mahdid, "Youtube," Deep Learning with Yacine, 23 10 2020. [Online]. Available: https://www.youtube.com/watch?v=ZUc0Mib5DeI. |
| [4] | A. Krizhevsky, "ImageNet Classification with Deep Convolutional," 25 2 2007. [Online]. Available: https://proceedings.neurips.cc/paper\_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. |
| [5] | P. Đ. Khánh, "ImageSegmentation," 10 07 2020. [Online]. Available: https://phamdinhkhanh.github.io/2020/06/10/ImageSegmention.html. |
| [6] | T. V. Huy, "GitHub," 08 02 2021. [Online]. Available: https://huytranvan2010.github.io/UNet/. |