TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC

NHẬP MÔN PHÂN TÍCH DỮ LIỆU VÀ HỌC SÂU

Đề tài:

IMAGE CLASSIFICATION

USING CNN ON DATASET CIFAR 10

GVHD: Nguyễn Thái Anh

LHP: 233_71ITDS30203_0103

Nhóm 5:

- **Nguyễn Hữu Trường(Leader)**
- **Lê Gia Hào**
- **Thái Quốc Bảo**
- **Đào Duy Luân**

TP. Hồ Chí Minh – năm 2024

LỜI CẢM ƠN

Nhóm 5 chúng em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Thái Anh, người đã tận tâm hướng dẫn, chia sẻ kiến thức và đưa ra những góp ý chân thành để đồ án của chúng em trở nên hoàn thiện hơn. Sự nhấn mạnh vào chất lượng và sự cam kết của thầy đã là động lực lớn giúp nhóm chúng em vượt qua những thách thức trong quá trình nghiên cứu và thực hiện.

Cuối cùng, nhóm 5 chúng em xin tự bày tỏ lòng biết ơn đến tất cả những thành viên trong nhóm đã đóng góp vào công trình này, dù lớn hay nhỏ. Sự hỗ trợ và đóng góp của mọi người đã làm cho đồ án này thành công.

TP.Hồ Chí Minh, ngày 22 tháng 7 năm 2024

Nhóm 5

BẢNG PHÂN CÔNG NHIỆM VỤ

STT	HỌ VÀ TÊN	MSSV	VAI TRÒ	NHIỆM VỤ	THANG ĐIỂM 10
1	Nguyễn Hữu Trường	207ct65867	Nhóm trưởng	Thu thập tài liệu, chỉnh sửa báo cáo hoàn thiện, sửa powerpoint ,viếtcode.	10
2	Đào Duy Luân	2174802010599	Thành viên	Thu thập tài liệu word, làm slide powerpoint , sửa code.	9
3	Thái Quốc Bảo	207ct09955	Thành viên	Thu thập tài liệu, chỉnh sửa powerpoint ,sửa báo cáo.	8.5
4	Lê Gia Hào	207ct65580	Thành viên	Thu thập tài liệu, chỉnh sửa powerpoint ,sửa báo cáo.	8.5

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU

1.1 Lý do chọn chủ đề nghiên cứu:

Trong thời đại số hóa ngày nay, việc xử lý và phân loại hình ảnh đã trở thành một thách thức quan trọng. Hình ảnh là một nguồn dữ liệu lớn và quan trọng, xuất phát từ nhiều lĩnh vực như y tế, công nghiệp, giáo dục và nhiều lĩnh vực khác. Tuy nhiên, việc phân loại hình ảnh một cách chính xác và hiệu quả vẫn là một thách thức đối với nhiều ứng dụng.

Dự án "Image Classification Using CNN" tập trung vào việc áp dụng mạng nơ-ron tích chập (CNN - Convolutional Neural Network) để giải quyết vấn đề phân loại hình ảnh. Mục tiêu chính là xây dựng một hệ thống có khả năng tự động học và nhận diện đối tượng trong hình ảnh với độ chính xác cao.

1.2 Công nghệ sử dụng:

Dự án sẽ sử dụng mô hình CNN, một loại mạng nơ-ron thích hợp cho việc xử lý hình ảnh, để học và trích xuất đặc trưng từ dữ liệu. Quá trình huấn luyện mô hình sẽ được thực hiện trên một bộ dữ liệu đa dạng và đầy đủ (CIFAR10), với mục tiêu tối ưu hóa hiệu suất phân loại.

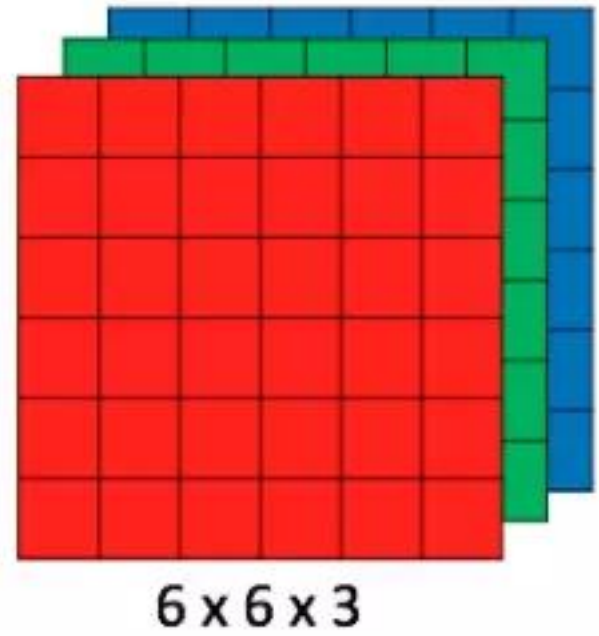
Hình 1. Ảnh minh họa về trực quan dữ liệu

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu

CNN là viết tắt của Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).



Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

2.2 Mạng nơron

Bộ não con người chính là nguồn cảm hứng cho kiến trúc mạng nơ-ron. Các tế bào não của con người, còn được gọi là nơ-ron, tạo thành một mạng lưới phức tạp, có tính liên kết cao và gửi các tín hiệu điện đến nhau để giúp con người xử lý thông tin. Tương tự, một mạng nơ-ron nhân tạo được tạo ra từ các tế bào nơ-ron nhân tạo, cùng nhau phối hợp để giải quyết một vấn đề. Nơ-ron nhân tạo là các mô đun phần mềm, được gọi là nút và mạng nơ-ron nhân tạo là các chương trình phần mềm hoặc thuật toán mà về cơ bản, sử dụng hệ thống máy tính để giải quyết các phép toán.

Một mạng nơ-ron cơ bản bao gồm các nơ-ron nhân tạo liên kết theo 3 lớp:

- Lớp đầu vào: Thông tin từ thế giới bên ngoài đi vào mạng nơ-ron nhân tạo qua lớp đầu vào. Các nút đầu vào xử lý dữ liệu, phân tích hoặc phân loại và sau đó chuyển dữ liệu sang lớp tiếp theo.

- Lớp ẩn: Dữ liệu đi vào lớp ẩn đến từ lớp đầu vào hoặc các lớp ẩn khác. Mạng nơ-ron nhân tạo có thể có một số lượng lớn lớp ẩn. Mỗi lớp ẩn phân tích dữ liệu đầu ra từ lớp trước, xử lý dữ liệu đó sâu hơn và rồi chuyển dữ liệu sang lớp tiếp theo.

- Lớp đầu ra: Lớp đầu ra cho ra kết quả cuối cùng của tất cả dữ liệu được xử lý bởi mạng nơ-ron nhân tạo. Lớp này có thể có một hoặc nhiều nút. Ví dụ: giả sử chúng ta gặp phải một vấn đề phân loại nhị phân (có/không), lớp đầu ra sẽ có một nút đầu ra, nút này sẽ cho kết quả 1 hoặc 0. Tuy nhiên, nếu chúng ta gặp phải vấn đề phân loại nhiều lớp, lớp đầu ra sẽ có thể bao gồm nhiều hơn một nút đầu ra.

2.3 Mô hình CNN

Mô hình CNN (Convolutional Neural Network) thường bao gồm một số lớp cơ bản, mỗi lớp đóng vai trò quan trọng trong việc trích xuất đặc trưng từ hình ảnh đầu vào. Dưới đây là một mô tả tổng quan về các lớp chính của một mô hình CNN và cách chúng hoạt động:

2.3.1 Lớp Convolutional (Convolutional Layer)

- Mô tả: Đây là lớp đầu tiên được sử dụng để trích xuất các đặc trưng khác nhau từ hình ảnh đầu vào. Lớp này thực hiện phép toán convolution giữa hình ảnh đầu vào và một bộ lọc có kích thước cụ thể MxM. Bằng cách trượt bộ lọc qua toàn bộ hình ảnh, tích dot được thực hiện giữa bộ lọc và các phần của hình ảnh đầu vào liên quan đến kích thước của bộ lọc (MxM).

- Hoạt động: Kết quả được gọi là feature map, cung cấp thông tin về hình ảnh như các góc và cạnh. Feature map sau đó được đưa vào các lớp khác để học nhiều đặc trưng khác nhau của hình ảnh đầu vào. Lớp convolutional trong CNN chuyển kết quả đến lớp tiếp theo sau khi áp dụng phép toán convolution.

- Lợi ích: Bảo toàn mối quan hệ không gian giữa các pixel trong hình ảnh.

2.3.2 Lớp Activation (Activation Layer):

- Mô tả: Trong hầu hết các trường hợp, một Lớp Convolutional sẽ được tiếp theo bởi một Lớp Pooling. Mục tiêu chính của lớp này là giảm kích thước của feature map sau phép convolution để giảm chi phí tính toán.

- Hoạt động: Lớp này độc lập hoạt động trên từng feature map và giảm kích thước của chúng. Có nhiều loại phép Pooling như Max Pooling, Average Pooling và Sum Pooling. Các phép Pooling này tóm tắt các đặc trưng được tạo ra bởi lớp convolution.

- Lợi ích: Giảm kích thước tính toán và giữ lại thông tin quan trọng.

2.3.3 Lớp Pooling (Pooling Layer):

- Mô tả: Lớp FC bao gồm trọng số, nơ-ron và độ lệch, được sử dụng để kết nối các nơ-ron giữa hai lớp khác nhau. Thường đặt trước lớp đầu ra và là một số lớp cuối cùng của kiến trúc CNN.

- Hoạt động: Hình ảnh đầu vào từ các lớp trước được làm phẳng và đưa vào lớp FC. Vector được làm phẳng sau đó trải qua một số lớp FC khác nơi các phép toán toán học thường diễn ra. Quá trình phân loại bắt đầu ở đây.

- Lợi ích: Kết nối các đặc trưng đã được trích xuất bởi lớp convolutional và giúp mạng nhận ra các đặc trưng độc lập. Giảm sự giám sát của con người.

2.3.4 Lớp Flatten (Flatten Layer):

- Mô tả: Thường, khi tất cả các đặc trưng được kết nối đến lớp FC, có thể gây ra hiện tượng quá mức khi huấn luyện trên tập dữ liệu. Để giảm thiểu tình trạng này, sử dụng lớp dropout, nơi một số nơ-ron được loại bỏ ngẫu nhiên từ mạng nơ-ron trong quá trình huấn luyện.

- Hoạt động: Trong quá trình dropout, một số nơ-ron được loại bỏ ngẫu nhiên từ mạng, giảm kích thước của mô hình. Ví dụ, khi áp dụng dropout 0.3, 30% số nơ-ron sẽ bị loại bỏ ngẫu nhiên từ mạng.

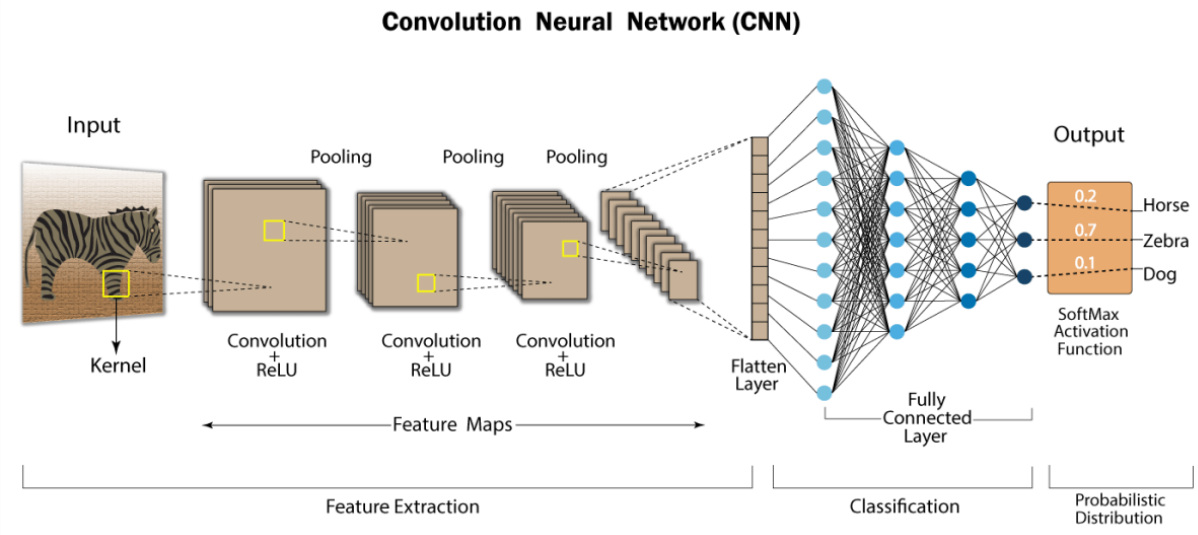
- Lợi ích: Ngăn chặn hiện tượng quá mức bằng cách làm cho mạng trở nên đơn giản hóa.

2.3.5 Lớp Fully Connected (Fully Connected Layer):

- Mô tả: Một trong những tham số quan trọng nhất của mô hình CNN là hàm kích hoạt. Được sử dụng để học và xấp xỉ bất kỳ mối quan hệ nào liên tục và phức tạp giữa các biến của mạng.

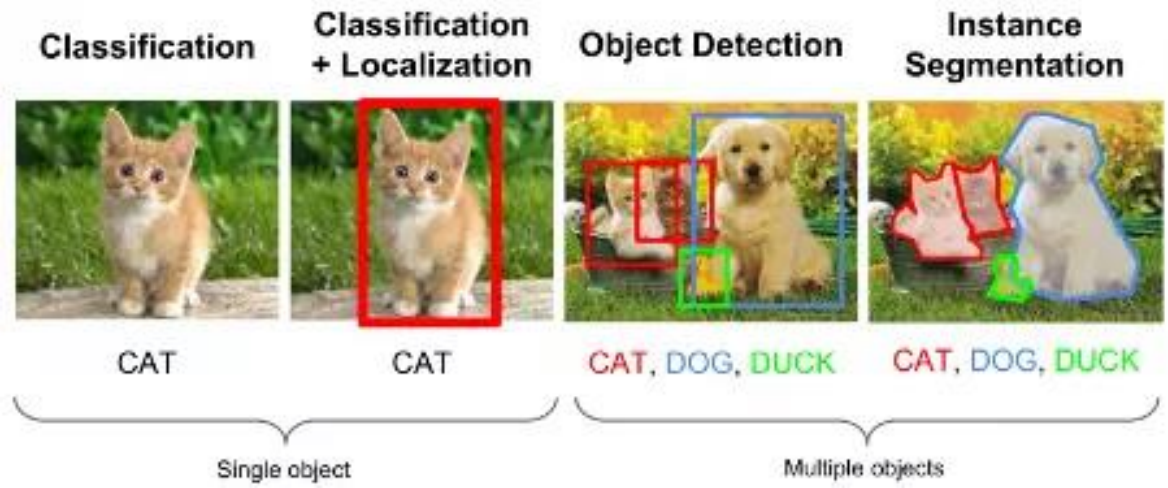
- Hoạt động: Hàm kích hoạt thêm tính phi tuyến tính vào mạng. Có nhiều hàm kích hoạt phổ biến như ReLU, Softmax, tanH và Sigmoid, mỗi hàm có một ứng dụng cụ thể.

- Lợi ích: Quyết định xem một nơ-ron nào nên được kích hoạt hay không. Điều này quyết định liệu đầu vào của mạng có quan trọng hay không trong việc dự đoán, sử dụng các phép toán toán học.

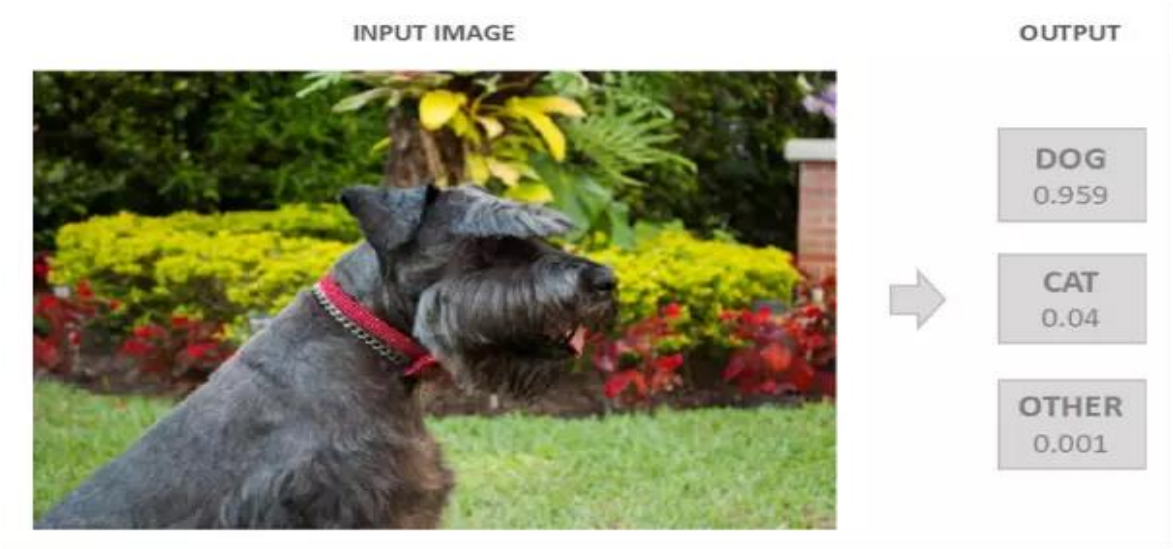


2.3.6 Ứng dụng của mô hình CNN

Mặc dù CNN chủ yếu được sử dụng cho các vấn đề về computer vision, nhưng điều quan trọng là đề cập đến khả năng giải quyết các vấn đề học tập khác của họ, chủ yếu liên quan đến tích chuỗi dữ liệu. Ví dụ: CNN đã được biết là hoạt động tốt trên chuỗi văn bản, âm thanh và video, đôi khi kết hợp với các mạng khác quả cầu kiến trúc hoặc bằng cách chuyển đổi các chuỗi thành hình ảnh có thể được xử lý của CNN. Một số vấn đề dữ liệu cụ thể có thể được giải quyết bằng cách sử dụng CNN với chuỗi dữ liệu là các bản dịch văn bản bằng máy, xử lý ngôn ngữ tự nhiên và gắn thẻ khung video, trong số nhiều người khác.

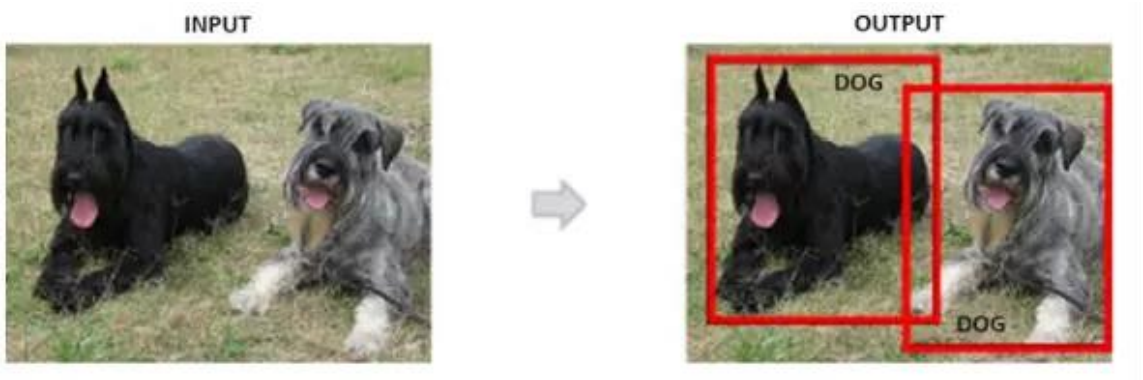


Classification: Đây là nhiệm vụ được biết đến nhiều nhất trong computer vision. Ý tưởng chính là phân loại nội dung chung của hình ảnh thành một tập hợp các danh mục, được gọi là nhãn. Ví dụ: phân loại có thể xác định xem một hình ảnh có phải là của một con chó, một con mèo hay bất kỳ động vật khác. Việc phân loại này được thực hiện bằng cách xuất ra xác suất của hình ảnh thuộc từng lớp, như được thấy trong hình ảnh sau:



Localization: Mục đích chính của localization là tạo ra một hộp giới hạn mô tả vị trí của đối tượng trong hình ảnh. Đầu ra bao gồm một nhãn lớp và một hộp giới hạn. Tác vụ này có thể được sử dụng trong cảm biến để xác định xem một đối tượng ở bên trái hay bên phải của màn hình:

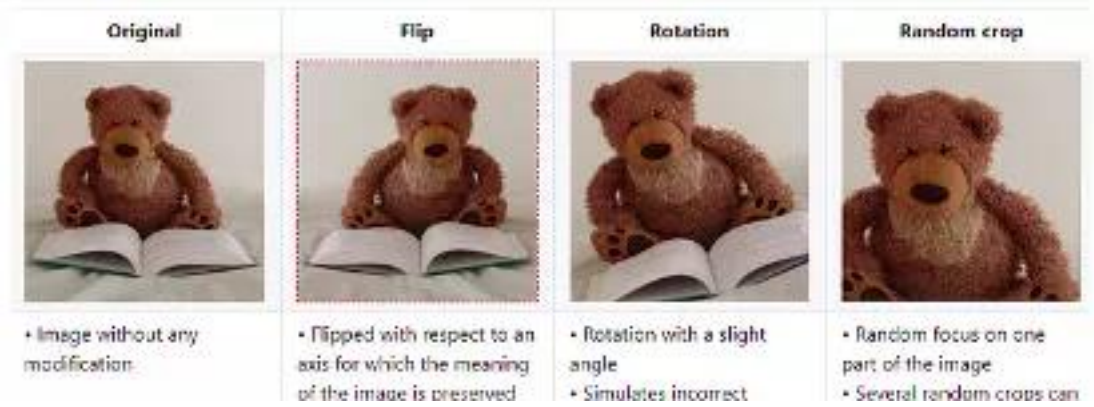
Detection: Nhiệm vụ này bao gồm thực hiện localization trên tất cả các đối tượng trong ảnh. Các đầu ra bao gồm nhiều hộp giới hạn, cũng như nhiều nhãn lớp (một cho mỗi hộp). Nhiệm vụ này được sử dụng trong việc chế tạo ô tô tự lái, với mục tiêu là có thể xác định vị trí các biển báo giao thông, đường, ô tô khác, người đi bộ và bất kỳ đối tượng nào khác có thể phù hợp để đảm bảo trải nghiệm lái xe an toàn:



Segmentation: Nhiệm vụ ở đây là xuất ra cả nhãn lớp và đường viền của mỗi đối tượng hiện diện trong hình ảnh. Điều này chủ yếu được sử dụng để đánh dấu các đối tượng quan trọng của hình ảnh cho phân tích sâu hơn. Ví dụ: tác vụ này có thể được sử dụng để phân định rõ ràng khu vực tương ứng với khối u trong hình ảnh phổi của bệnh nhân. Hình sau mô tả cách vật thể quan tâm được phác thảo và gán nhãn:

2.4 Tăng cường dữ liệu (Data Augmentation)

Hiện nay trong deep learning thì vấn đề dữ liệu có vai trò rất quan trọng. Chính vì vậy có những lĩnh vực có ít dữ liệu để cho việc train model thì rất khó để tạo ra được kết quả tốt trong việc dự đoán. Do đó người ta cần đến một kỹ thuật gọi là tăng cường dữ liệu (data augmentation) để phục vụ cho việc nếu có ít dữ liệu. Phương thức data aumentation cơ bản:



Flip (Lật): lật theo chiều dọc, ngang miễn sao ý nghĩa của ảnh (label) được giữ nguyên hoặc suy ra được. Ví dụ nhận dạng quả bóng tròn, thì lật kiểu gì cũng ra quả bóng. Còn với nhận dạng chữ viết tay, lật số 8 vẫn là 8, nhưng 6 sẽ thành 9 (theo chiều ngang) và không ra số gì theo chiều dọc.

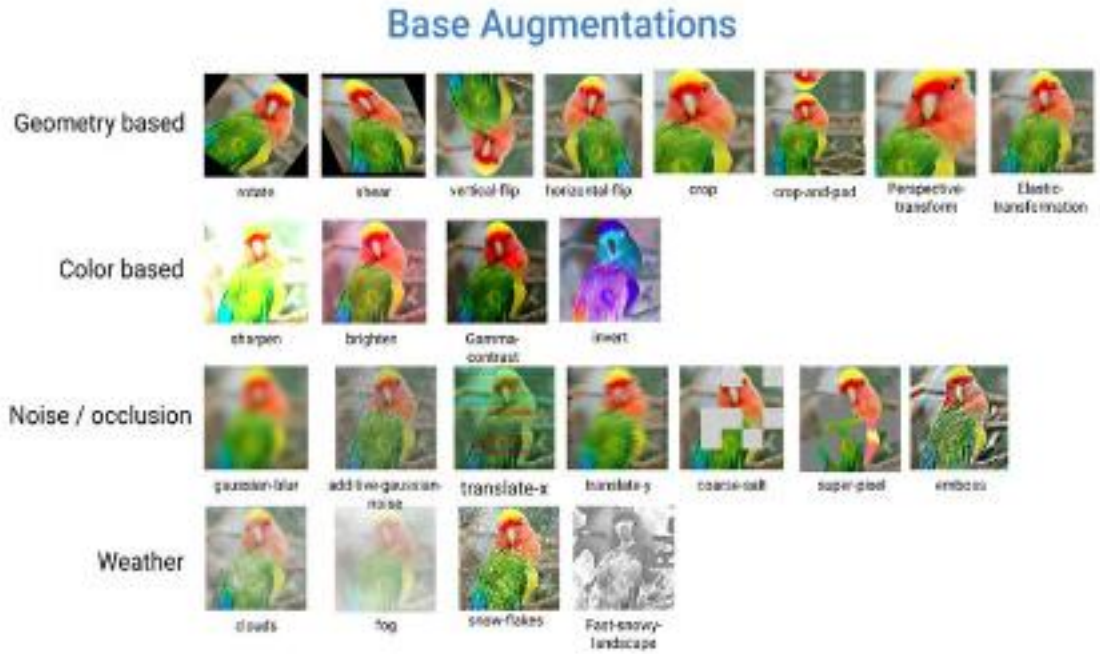
Random crop (Cắt ngẫu nhiên): cắt ngẫu nhiên một phần của bức ảnh. Lưu ý là khi cắt phải giữ thành phần chính của bức ảnh mà ta quan tâm. Như ở nhận diện vật thể, nếu ảnh được cắt không có vật thể, vậy giá trị nhãn là không chính xác.

Color shift (Chuyển đổi màu): Chuyển đổi màu của bức ảnh bằng cách thêm giá trị vào 3 kênh màu RGB. Việc này liên quan tới ảnh chụp đôi khi bị nhiễu --> màu bị ảnh hưởng.

Noise addition (Thêm nhiễu): Thêm nhiễu vào bức ảnh. Nhiễu thì có nhiều loại như nhiễu ngẫu nhiên, nhiễu có mẫu, nhiễu cộng, nhiễu nhân, nhiễu do nén ảnh, nhiễu mờ do chụp không lấy nét, nhiễu mờ do chuyển động… có thể kể hết cả ngày.

I**nformation loss (Mất thông tin):** Một phần của bức hình bị mất. Có thể minh họa trường hợp bị che khuất.

Constrast change (Thay đổi độ tương phản): thay độ tương phản của bức hình, độ bão hòa.



2.5 Ưu điểm, nhược điểm

2.5.1 Ưu điểm:

- Phù Hợp Cho Dữ Liệu Không Gian: CNN được thiết kế đặc biệt để xử lý dữ liệu không gian như hình ảnh. Các lớp convolution giúp mô hình tự động học các đặc trưng cấp thấp đến cấp cao từ dữ liệu.
- Chia Sẻ Trọng Số và Tính Tích Chập: Tính chia sẻ trọng số giữa các vùng không gian giúp giảm lượng tham số cần đào tạo, giảm độ phức tạp của mô hình và tăng cường khả năng tổng quát hóa.
- Tích Chập Cho Đối Tượng Nhỏ: Tích chập giúp mô hình có khả năng nhận biết các đặc trưng ở mọi vị trí trong hình ảnh, bất kể kích thước của chúng.
- Độ Tương Thích Cao với Dữ Liệu 2D: CNN làm việc tốt với dữ liệu 2D như hình ảnh. Các mô hình CNN có thể tự động học các đặc trưng không gian và cấu trúc không gian của dữ liệu.
- Hiệu Suất Cao Trong Phân Loại Hình Ảnh: CNNs đã chứng minh độ hiệu quả và độ chính xác cao trong nhiều nhiệm vụ phân loại hình ảnh, từ nhận diện khuôn mặt đến nhận diện đối tượng trong hình ảnh phức tạp.

2.5.2 Nhược điểm:

- Dễ Bị Overfitting: CNNs có thể dễ dàng bị overfitting đặc biệt là khi dữ liệu đào tạo có lượng nhỏ. Việc sử dụng kỹ thuật như dropout và regularization là quan trọng để kiểm soát overfitting.
- Yêu Cầu Nguồn Lực Lớn: Quá trình đào tạo mô hình CNN có thể đòi hỏi nguồn lực tính toán lớn, đặc biệt là khi làm việc với các mô hình lớn và tập dữ liệu lớn.
- Khó Diễn Giải: Các mô hình CNN thường có số lượng lớn các tham số và lớp, điều này làm cho chúng trở nên khó diễn giải và hiểu lý do mô hình đưa ra các quyết định cụ thể.
- Không Linh Hoạt: Mặc dù CNNs làm việc tốt với hình ảnh, nhưng chúng có thể không linh hoạt đối với dữ liệu không gian khác như văn bản hoặc âm thanh.
- Chưa Hoạt Động Tốt Cho Dữ Liệu Nhất Quán Kém: Nếu dữ liệu đầu vào không nhất quán hoặc bị nhiễu, CNNs có thể không hiệu quả như mong đợi và cần sự chuẩn bị dữ liệu kỹ lưỡng.

CHƯƠNG 3: QUY TRÌNH HOẠT ĐỘNG

3.1 Nền tảng và Thư viện

- Ngôn ngữ lập trình: Python là một trong những ngôn ngữ lập trình phổ biến và mạnh mẽ trong lĩnh vực machine learning và deep learning, và có nhiều lý do mà nó được ưa chuộng khi xây dựng mô hình CNN.

- Thư viện : Keras là một thư viện mã nguồn mở phổ biến trong lĩnh vực machine learning và deep learning. Nó được phát triển bởi Google và cung cấp một cơ sở hạ tầng mạnh mẽ để xây dựng và huấn luyện mô hình machine learning.

- Dữ liệu training: CIFAR10

- Link github đồ án của nhóm 5 :

https://github.com/Truong1103/ImageClassification-Team5

3.2 Triển khai CNN bằng Python trên Google COLAB

Cài thư viện Keras

```
!sudo pip3 install keras
```

1. Import các thư viện cần thiết

```
import numpy as np
# hàm này để hiển thị các tấm ảnh
import matplotlib.pyplot as plt
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
# để chọn ngẫu nhiên các tấm ảnh
import random
import os
# để load dữ liệu trong tập mẫu
from keras.datasets import cifar10
from keras.models import load_model
from matplotlib import image
# các hàm sau để xây dựng mô hình mạng neural
# thêm 1 lớp, thêm hàm kích hoạt activation
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D,Dropout
from keras.layers import Dense, Activation, Flatten
import tensorflow as tf
from tensorflow import keras
from keras import layers, models, optimizers, losses, metrics
```

2. Tải tập dữ liệu Cifar-10

```
#Tải tập dữ liệu
(X_train, y_train), (X_test,y_test) = cifar10.load_data()
```

3. Định nghĩa danh sách các phần tử

```
#Danh sách  gồm các phần tử
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

4. Chuẩn hóa dữ liệu về 0-1

```
from tensorflow.keras import models, layers, optimizers, callbacks, regularizers
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Chuẩn hóa hình ảnh
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

4. Thêm Data Aumentation

```
# Khởi tạo ImageDataGenerator với các phép biến đổi dữ liệu
datagen = ImageDataGenerator(
    rotation_range=20,              # Góc xoay ảnh trong khoảng ±20 độ
    width_shift_range=0.2,          # Dịch chuyển ngang ảnh theo chiều rộng, tỷ lệ 20%
    height_shift_range=0.2,         # Dịch chuyển ngang ảnh theo chiều cao, tỷ lệ 20%
    horizontal_flip=True            # Lật ảnh ngang
)

# Áp dụng phương pháp fit để tính toán thống kê bằng cách sử dụng dữ liệu
datagen.fit(X_train)
```

5. Xây dựng mô hình CNN gồm 6 lớp

```
# Xây dựng mô hình CNN
cnn = models.Sequential([
    # Lớp tích chập đầu tiên với 32 bộ lọc, kernel size 3x3, hàm kích hoạt ReLU, padding 'same'
    layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)),
    layers.BatchNormalization(),  # Lớp chuẩn hóa batch
    layers.MaxPooling2D((2, 2)),  # Lớp pooling 2x2
    layers.Dropout(0.25),         # Lớp dropout với tỉ lệ 0.25

    # Lớp tích chập thứ hai với 64 bộ lọc, kernel size 3x3, hàm kích hoạt ReLU, padding 'same'
    layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    layers.BatchNormalization(),  # Lớp chuẩn hóa batch
    layers.MaxPooling2D((2, 2)),  # Lớp pooling 2x2
    layers.Dropout(0.25),         # Lớp dropout với tỉ lệ 0.25

    # Lớp tích chập thứ ba với 128 bộ lọc, kernel size 3x3, hàm kích hoạt ReLU, padding 'same'
    layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    layers.BatchNormalization(),  # Lớp chuẩn hóa batch
    layers.MaxPooling2D((2, 2)),  # Lớp pooling 2x2
    layers.Dropout(0.25),         # Lớp dropout với tỉ lệ 0.25

    # Lớp tích chập thứ tư với 256 bộ lọc, kernel size 3x3, hàm kích hoạt ReLU, padding 'same'
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.BatchNormalization(),  # Lớp chuẩn hóa batch
    layers.MaxPooling2D((2, 2)),  # Lớp pooling 2x2
    layers.Dropout(0.25),         # Lớp dropout với tỉ lệ 0.25

    # Lớp tích chập thứ năm với 512 bộ lọc, kernel size 3x3, hàm kích hoạt ReLU, padding 'same'
    layers.Conv2D(512, (3, 3), activation='relu', padding='same'),
    layers.BatchNormalization(),  # Lớp chuẩn hóa batch
    layers.MaxPooling2D((2, 2)),  # Lớp pooling 2x2
    layers.Dropout(0.25),         # Lớp dropout với tỉ lệ 0.25

    layers.Flatten(),  # Chuyển từ tensor 3D sang tensor 1D
    layers.Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.001)),  # Lớp fully connected với 512 units, hàm kích hoạt ReLU và regularization L2
    layers.BatchNormalization(),  # Lớp chuẩn hóa batch
    layers.Dropout(0.5),          # Lớp dropout với tỉ lệ 0.5
    layers.Dense(10, activation='softmax')  # Lớp fully connected với 10 units, hàm kích hoạt softmax (cho 10 lớp đầu ra)
])
```

6. Thêm Learning Rate Scheduler , EarlyStopping , Optimizer AdamW

```python
# Sử dụng Learning Rate Scheduler
# ReduceLROnPlateau: Giảm learning rate khi val_loss không cải thiện sau một số epochs
lr_scheduler = callbacks.ReduceLROnPlateau(
    monitor='val_loss',  # Theo dõi độ đo val_loss
    factor=0.5,          # Giảm learning rate đi một nửa
    patience=5,          # Số epochs không cải thiện trước khi giảm learning rate
    min_lr=0.00001       # Learning rate tối thiểu
)

# Sử dụng EarlyStopping
# EarlyStopping: Dừng huấn luyện sớm nếu val_loss không cải thiện
early_stopping = callbacks.EarlyStopping(
    monitor='val_loss',            # Theo dõi độ đo val_loss
    patience=10,                   # Số epochs không cải thiện trước khi dừng huấn luyện
    restore_best_weights=True      # Khôi phục trọng số tốt nhất trước khi dừng
)

# Sử dụng optimizer AdamW với gradient clipping
# AdamW: Optimizer với weight decay và gradient clipping
optimizer = optimizers.AdamW(
    learning_rate=0.001,   # Tốc độ học
    weight_decay=1e-5,     # Lượng giảm trọng số để tránh overfitting
    clipvalue=1.0          # Giá trị tối đa cho gradient để tránh gradient explosion
)
```

7. Huấn luyện mô hình

```python
# Huấn luyện mô hình CNN
cnn.compile(
    optimizer=optimizer,  # Sử dụng optimizer đã được định nghĩa trước đó (ví dụ: AdamW)
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),  # Hàm mất mát
    metrics=['accuracy']  # Đánh giá mô hình theo độ chính xác
)

# Huấn luyện mô hình với dữ liệu được tạo ra bằng data augmentation
history = cnn.fit(
    datagen.flow(X_train, y_train, batch_size=64),  # Generator của dữ liệu huấn luyện với batch_size là 64
    epochs=40,  # Số lượng epochs
    validation_data=(X_test, y_test),  # Dữ liệu validation
    callbacks=[early_stopping, lr_scheduler]  # Các callbacks để điều chỉnh learning rate và early stopping
)
```

-Kết quả:

```
Epoch 20/40
782/782 [==============================] - 33s 42ms/step - loss: 0.8356 - accuracy: 0.7260 - val_loss: 0.8612 - val_accuracy: 0.7280 - lr: 5.0000e-04
Epoch 21/40
782/782 [==============================] - 33s 42ms/step - loss: 0.8248 - accuracy: 0.7301 - val_loss: 0.8435 - val_accuracy: 0.7334 - lr: 5.0000e-04
Epoch 22/40
782/782 [==============================] - 32s 41ms/step - loss: 0.8329 - accuracy: 0.7283 - val_loss: 0.6801 - val_accuracy: 0.7883 - lr: 5.0000e-04
Epoch 23/40
782/782 [==============================] - 33s 42ms/step - loss: 0.8194 - accuracy: 0.7327 - val_loss: 0.6108 - val_accuracy: 0.8043 - lr: 5.0000e-04
Epoch 24/40
782/782 [==============================] - 32s 41ms/step - loss: 0.8132 - accuracy: 0.7337 - val_loss: 0.6616 - val_accuracy: 0.7939 - lr: 5.0000e-04
Epoch 25/40
782/782 [==============================] - 33s 43ms/step - loss: 0.8068 - accuracy: 0.7366 - val_loss: 0.7039 - val_accuracy: 0.7762 - lr: 5.0000e-04
Epoch 26/40
782/782 [==============================] - 32s 41ms/step - loss: 0.7991 - accuracy: 0.7399 - val_loss: 0.6421 - val_accuracy: 0.7968 - lr: 5.0000e-04
Epoch 27/40
782/782 [==============================] - 34s 43ms/step - loss: 0.7947 - accuracy: 0.7412 - val_loss: 0.6767 - val_accuracy: 0.7829 - lr: 5.0000e-04
Epoch 28/40
782/782 [==============================] - 33s 42ms/step - loss: 0.7892 - accuracy: 0.7440 - val_loss: 0.7630 - val_accuracy: 0.7549 - lr: 5.0000e-04
Epoch 29/40
782/782 [==============================] - 33s 42ms/step - loss: 0.7568 - accuracy: 0.7536 - val_loss: 0.6800 - val_accuracy: 0.7850 - lr: 2.5000e-04
Epoch 30/40
782/782 [==============================] - 34s 43ms/step - loss: 0.7440 - accuracy: 0.7552 - val_loss: 0.7718 - val_accuracy: 0.7501 - lr: 2.5000e-04
Epoch 31/40
782/782 [==============================] - 33s 42ms/step - loss: 0.7357 - accuracy: 0.7575 - val_loss: 0.6140 - val_accuracy: 0.8055 - lr: 2.5000e-04
Epoch 32/40
782/782 [==============================] - 34s 43ms/step - loss: 0.7342 - accuracy: 0.7583 - val_loss: 0.5634 - val_accuracy: 0.8181 - lr: 2.5000e-04
Epoch 33/40
782/782 [==============================] - 34s 43ms/step - loss: 0.7281 - accuracy: 0.7576 - val_loss: 0.5749 - val_accuracy: 0.8137 - lr: 2.5000e-04
Epoch 34/40
782/782 [==============================] - 33s 42ms/step - loss: 0.7198 - accuracy: 0.7619 - val_loss: 0.5756 - val_accuracy: 0.8161 - lr: 2.5000e-04
Epoch 35/40
782/782 [==============================] - 33s 43ms/step - loss: 0.7125 - accuracy: 0.7636 - val_loss: 0.6739 - val_accuracy: 0.7866 - lr: 2.5000e-04
Epoch 36/40
782/782 [==============================] - 33s 42ms/step - loss: 0.7159 - accuracy: 0.7641 - val_loss: 0.6496 - val_accuracy: 0.7916 - lr: 2.5000e-04
Epoch 37/40
782/782 [==============================] - 34s 43ms/step - loss: 0.7126 - accuracy: 0.7638 - val_loss: 0.6449 - val_accuracy: 0.7911 - lr: 2.5000e-04
Epoch 38/40
782/782 [==============================] - 33s 42ms/step - loss: 0.6914 - accuracy: 0.7696 - val_loss: 0.5612 - val_accuracy: 0.8219 - lr: 1.2500e-04
Epoch 39/40
782/782 [==============================] - 34s 43ms/step - loss: 0.6891 - accuracy: 0.7702 - val_loss: 0.6335 - val_accuracy: 0.7949 - lr: 1.2500e-04
Epoch 40/40
782/782 [==============================] - 34s 43ms/step - loss: 0.6855 - accuracy: 0.7697 - val_loss: 0.5818 - val_accuracy: 0.8110 - lr: 1.2500e-04
```

-Độ chính xác của tập huấn luyện(Train) sau 40 epochs này là 0.7697 (hay 76.97%).

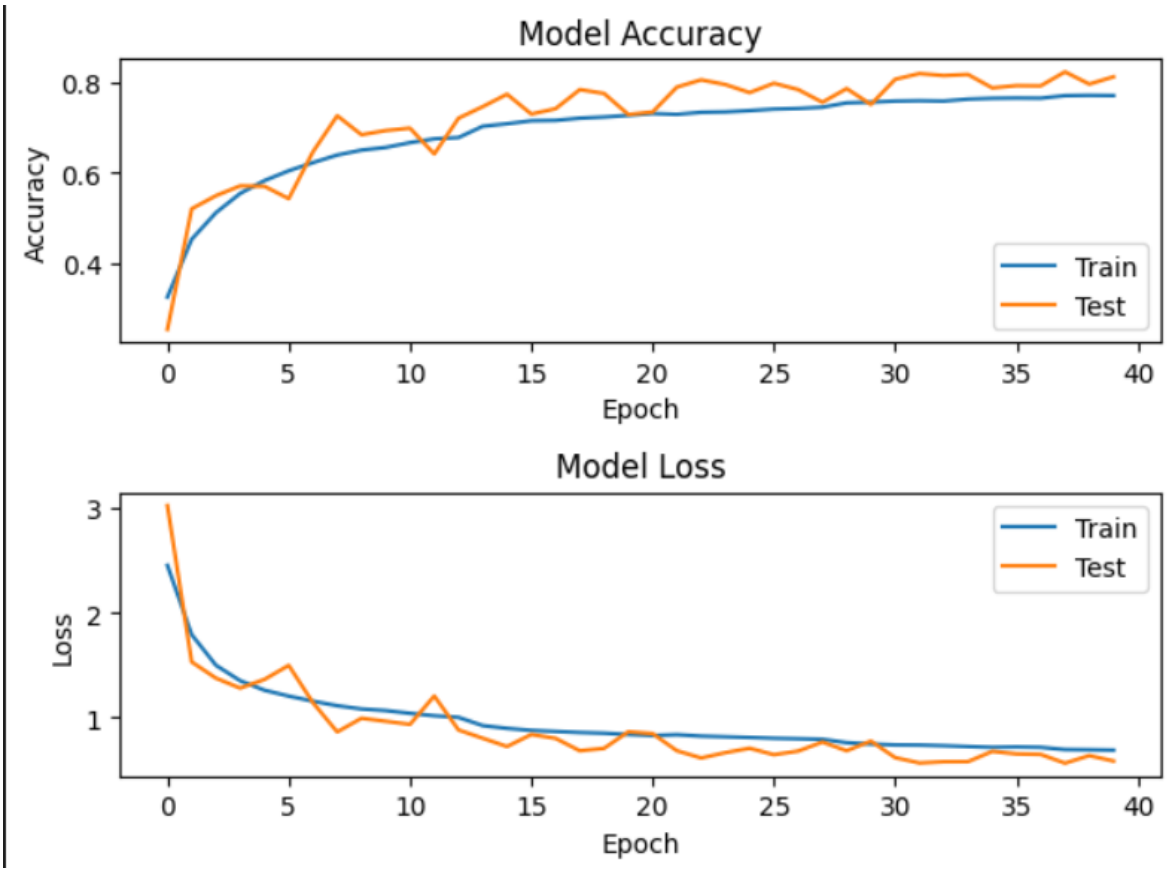
-Mất mát (loss) của tập huấn luyện(Train) sau epoch này là 0.6855.

-Độ chính xác của tập kiểm tra(Test) sau 40 epochs này là 0.8110 (hay 81.10%).

-Mất mát (loss) của tập kiểm tra(Test) sau epoch này là 0.5818.

-Tốc độ học (learning rate) là 0.000125.

8. Biểu đồ đánh giá kết quả

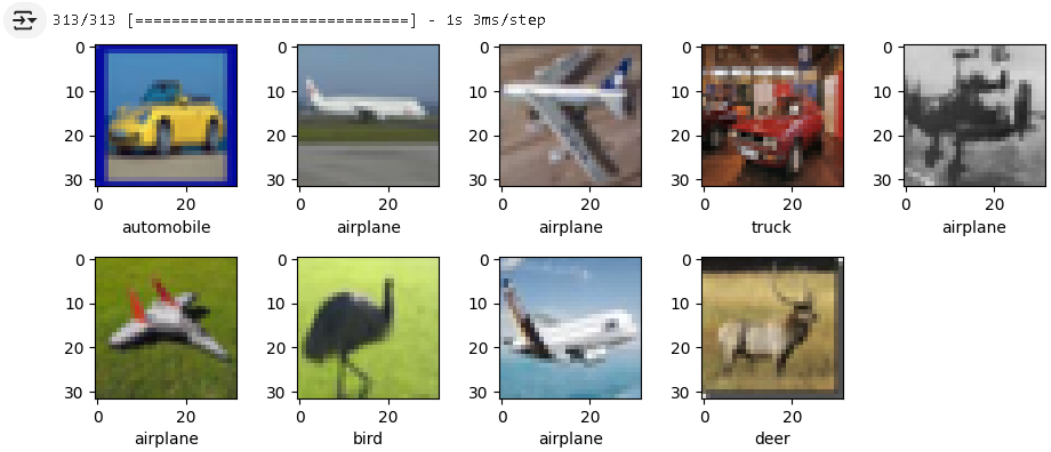


Save model:

```
cnn.save('model.h5')
```

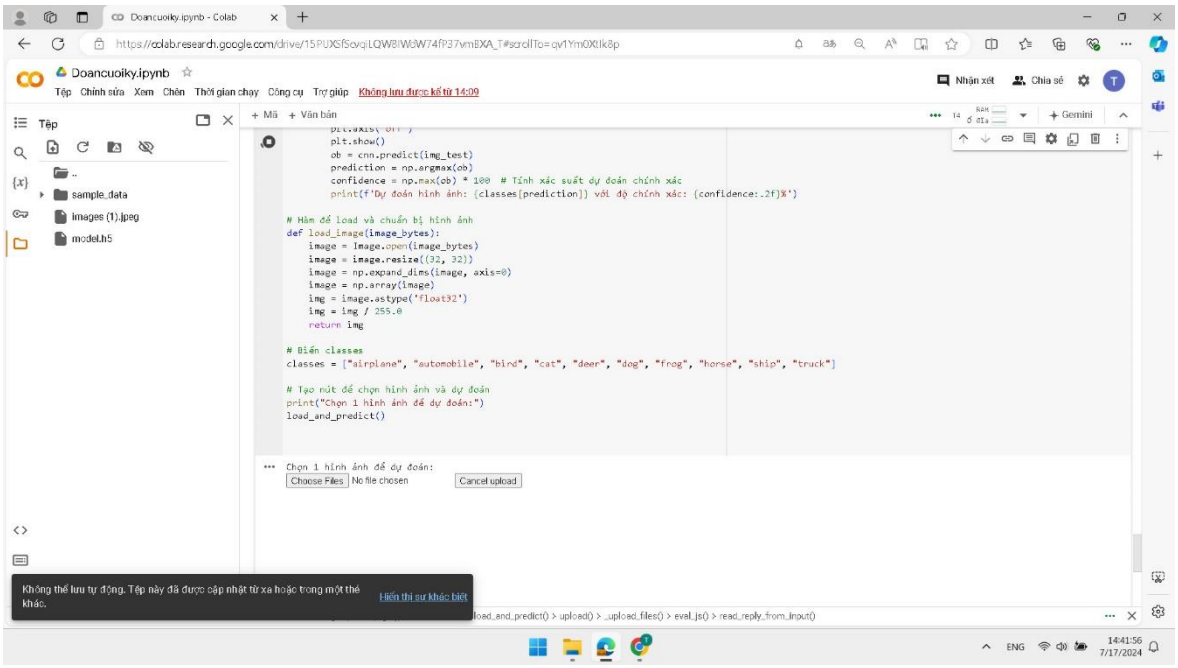
9.Dự đoán hiển thị ngẫu nhiên 9 tấm hình sau khi huấn luyện

```
# Hiển thị ngẫu nhiên 9 tấm hình trong tập test và dự đoán
#Hàm này để chọn ngẫu nhiên các tấm ảnh trong tập dữ liệu
import random
import os
predicted_classes=cnn.predict(X_test)
plt.rcParams['figure.figsize']=(9,9)
for i in range(9):
  plt.subplot(5,5,i+1)
  num=random.randint(0,len(X_test))
  plt.imshow(X_test[num])
  y_classes = [np.argmax(element) for element in predicted_classes]
  plt.xlabel(classes[y_classes[num]])
plt.tight_layout()
```

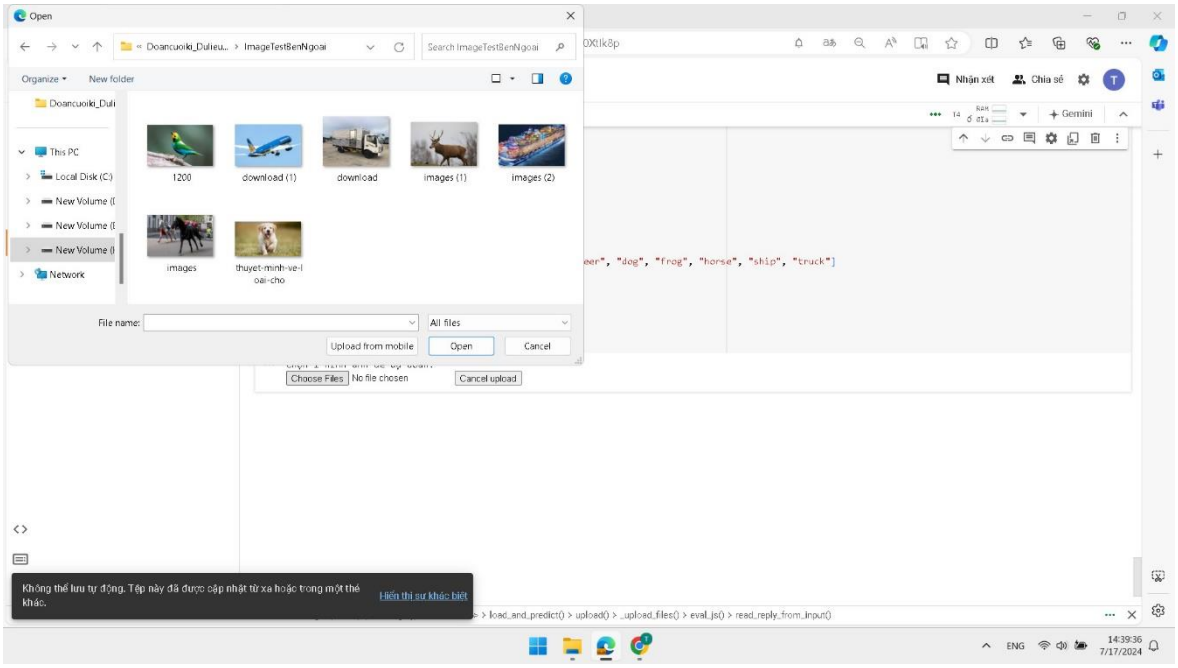


10. Tính năng tải bên ngoài hình ảnh bên ngoài để dự đoán

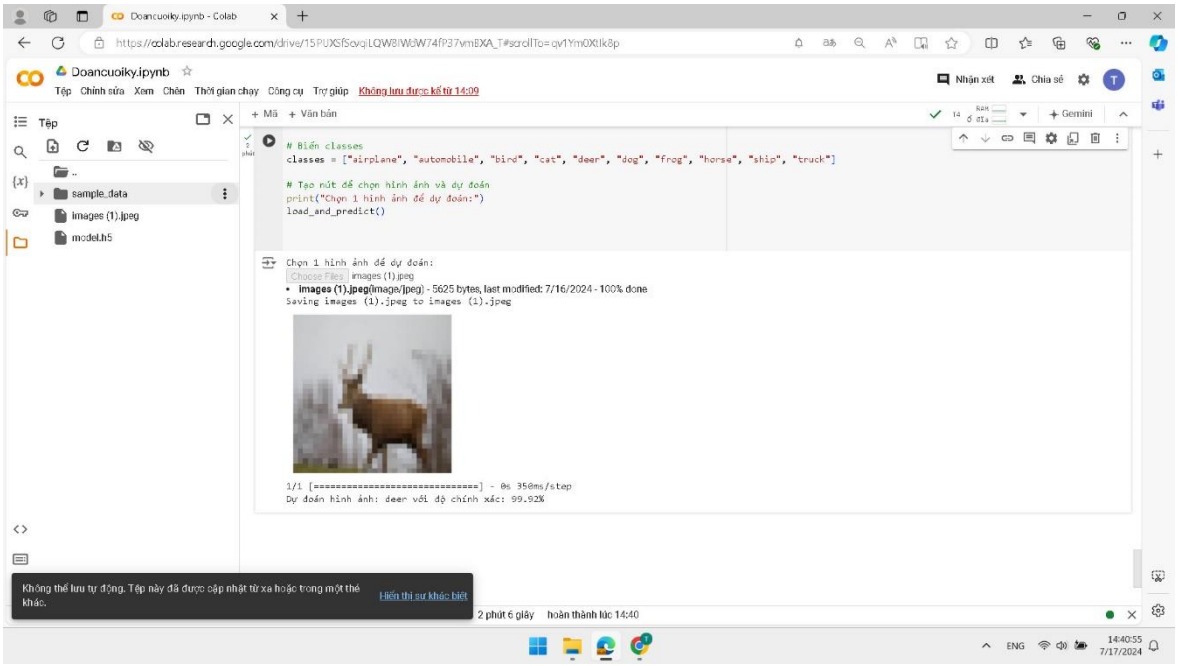
- Ctrl+Enter để chạy , giao diện hiện ra Choose File và Cancel Upload

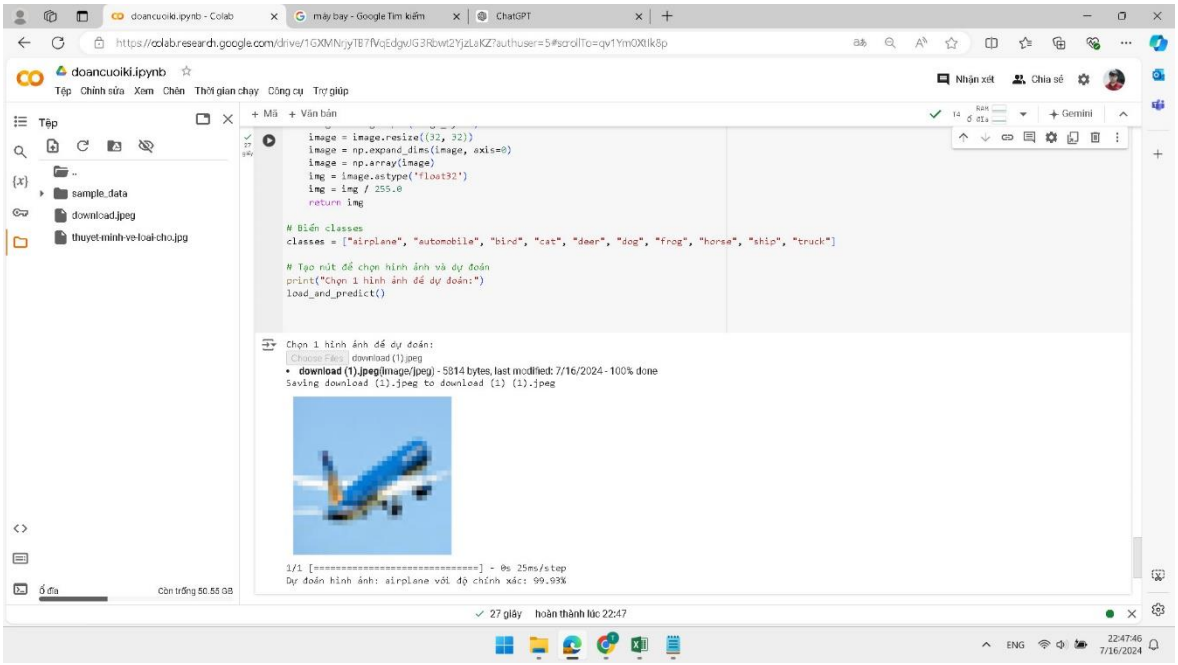


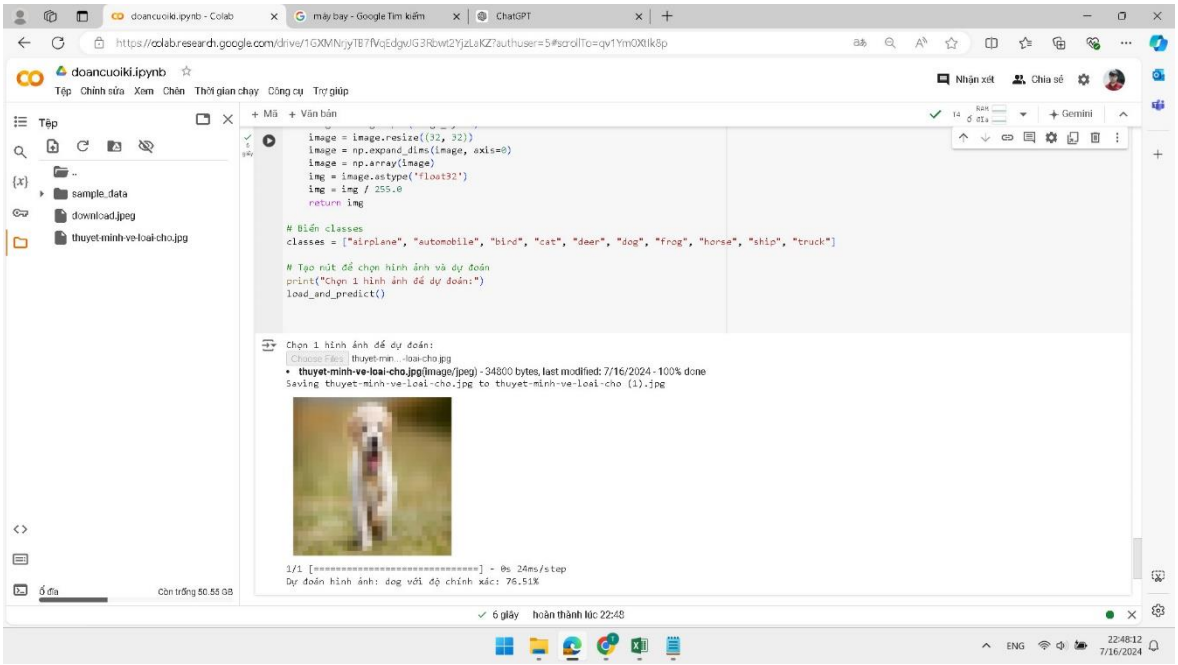
- Tải lên hình ảnh: Nhấn nút "**Choose files**" để chọn một hình ảnh từ máy tính của bạn. Hình ảnh đã tải lên sẽ hiển thị trực tiếp trên giao diện Google Colab.



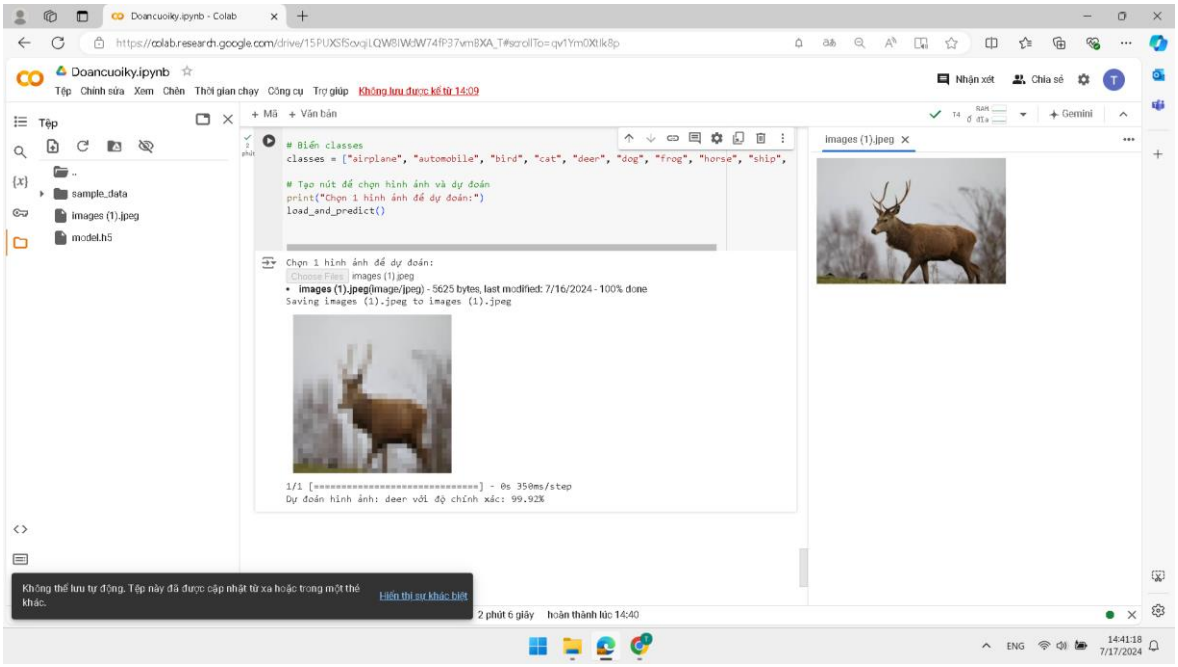
- Dự đoán hình ảnh với độ chính xác dự đoán




doancuoiki.ipynb - Colab
máy bay - Google Tìm kiếm
ChatGPT
doancuoiki.ipynb
Tệp Chỉnh sửa Xem Chèn Thời gian chạy Công cụ Trợ giúp
Nhận xét
Chia sẻ
Gemini
+ Mã + Văn bản
Tệp
sample_data
download.jpeg
thuyet-minh-ve-loai-cho.jpg
Ổ đĩa
Còn trống 50.55 GB
image = image.resize((32, 32))
image = np.expand_dims(image, axis=0)
image = np.array(image)
img = image.astype('float32')
img = img / 255.0
return img
# Biến classes
classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
# Tạo nút để chọn hình ảnh và dự đoán
print("Chọn 1 hình ảnh để dự đoán:")
load_and_predict()
Chọn 1 hình ảnh để dự đoán:
Choose Files download (1).jpeg
download (1).jpeg(image/jpeg) - 5814 bytes, last modified: 7/16/2024 - 100% done
Saving download (1).jpeg to download (1) (1).jpeg
1/1 [==============================] - 0s 25ms/step
Dự đoán hình ảnh: airplane với độ chính xác: 99.93%
27 giây hoàn thành lúc 22:47
ENG
22:47:46
7/16/2024


doancuoiki.ipynb - Colab
máy bay - Google Tìm kiếm
ChatGPT
doancuoiki.ipynb
Tệp Chỉnh sửa Xem Chèn Thời gian chạy Công cụ Trợ giúp
Nhận xét
Chia sẻ
Gemini
+ Mã + Văn bản
Tệp
sample_data
download.jpeg
thuyet-minh-ve-loai-cho.jpg
Ổ đĩa
Còn trống 50.55 GB
image = image.resize((32, 32))
image = np.expand_dims(image, axis=0)
image = np.array(image)
img = image.astype('float32')
img = img / 255.0
return img
# Biến classes
classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
# Tạo nút để chọn hình ảnh và dự đoán
print("Chọn 1 hình ảnh để dự đoán:")
load_and_predict()
Chọn 1 hình ảnh để dự đoán:
Choose Files thuyet-min ...-loai-cho.jpg
thuyet-minh-ve-loai-cho.jpg(image/jpeg) - 34800 bytes, last modified: 7/16/2024 - 100% done
Saving thuyet-minh-ve-loai-cho.jpg to thuyet-minh-ve-loai-cho (1).jpg
1/1 [==============================] - 0s 24ms/step
Dự đoán hình ảnh: dog với độ chính xác: 76.51%
6 giây hoàn thành lúc 22:48
ENG
22:48:12
7/16/2024

- Tự động save images sau khi dự đoán



11.Phân tích kết quả

-Độ chính xác của tập huấn luyện(Train) sau 40 epochs này là 0.7697 (hay 76.97%).

-Mất mát (loss) của tập huấn luyện(Train) sau epoch này là 0.6855.

-Độ chính xác của tập kiểm tra(Test) sau 40 epochs này là 0.8110 (hay 81.10%).

-Mất mát (loss) của tập kiểm tra(Test) sau epoch này là 0.5818.

-Tốc độ học (learning rate) là 0.000125.

Phân tích kết quả tính năng dự đoán hình ảnh từ bên ngoài :

- Tỷ lệ chính xác sẽ **Rất cao** khi hình ảnh đầu vào trực quan rõ vật , dễ nhận diện .

- Deer và Horse : sẽ có thể nó nhận định sai Horse là Deer, vì nếu hình ảnh đầu vào Horse (nếu con Horse gầy ốm).

- Nếu hình ảnh con vật nếu ở vị trí cao trong ảnh, nó sẽ nhận diện là Bird.

- Còn dường như Automobile, Ship , Truck , Airplane tỷ lệ chính xác rất cao và tỷ lệ đoán sai rất ít(hoặc dường như không có).

-Tỷ lệ đoán sai : Phần lớn là do hình ảnh đầu vào , ngoài ra tỷ lệ chính xác Mô hình của tập Train (76.97%), Test(81.10%) vẫn còn hạn chế.

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1.Kết luận

Với khả năng tự động học và trích xuất đặc trưng từ dữ liệu hình ảnh, Mạng Nơ-ron Tích chập (CNN) đã mang lại một cuộc cách mạng trong lĩnh vực xử lý và phân tích hình ảnh.

Thay vì phải dựa vào các phương pháp trích xuất đặc trưng thủ công, vốn tốn kém thời gian và thường không chính xác, CNN cho phép máy tính tự động học các đặc trưng quan trọng trực tiếp từ dữ liệu. Điều này không chỉ nâng cao độ chính xác của các hệ thống nhận dạng hình ảnh mà còn mở ra nhiều ứng dụng mới trong các lĩnh vực như y tế, giao thông, và an ninh.

CNN đã chứng minh rằng việc tự động hóa quá trình học tập từ dữ liệu thô là một bước tiến quan trọng, giúp giảm bớt sự phụ thuộc vào kiến thức chuyên môn của con người và tạo ra các mô hình phân loại vượt trội.

4.2. Hướng phát triển

Để tiếp tục nâng cao hiệu quả và ứng dụng của CNN, có một số hướng phát triển quan trọng cần được xem xét:

1- Cải tiến kiến trúc mô hình:

Nghiên cứu và phát triển các kiến trúc CNN mới để tăng cường khả năng học tập và xử lý dữ liệu. Các cải tiến này có thể bao gồm việc tối ưu hóa số lớp, kích thước lớp tích chập, và cách kết nối giữa các lớp. Ví dụ, các mô hình như ResNet, DenseNet đã cho thấy rằng việc thêm các đường dẫn tắt (skip connections) có thể giúp mô hình học tốt hơn và tránh hiện tượng gradient biến mất.

2- Sử dụng dữ liệu không gán nhãn:

Khai thác các phương pháp học không giám sát (unsupervised learning) hoặc học bán giám sát (semi-supervised learning) để sử dụng hiệu quả dữ liệu không gán nhãn. Các phương pháp này giúp mở rộng tập dữ liệu huấn luyện mà không cần tốn nhiều công sức và chi phí để gán nhãn cho dữ liệu.

3- Tăng cường dữ liệu:

Sử dụng các kỹ thuật tăng cường dữ liệu (data augmentation) như xoay, lật, thay đổi độ sáng, và cắt xén hình ảnh để tạo ra nhiều biến thể của dữ liệu huấn luyện. Điều này giúp mô hình học được nhiều đặc trưng phong phú và giảm thiểu hiện tượng quá khớp (overfitting).

4- Diễn giải mô hình:

Phát triển các phương pháp để giải thích và hiểu rõ hơn cách thức hoạt động của CNN. Điều này không chỉ giúp tăng độ tin cậy của mô hình mà còn giúp phát hiện và sửa lỗi trong quá trình học. Các công cụ như Grad-CAM, LIME có thể được sử dụng để trực quan hóa các vùng ảnh quan trọng mà mô hình chú ý tới.

5- Tối ưu hóa tính toán:

Tối ưu hóa thuật toán và phần cứng để giảm thời gian và chi phí tính toán. Sử dụng các kỹ thuật như lượng tử hóa (quantization), cắt tỉa mô hình (model pruning), và triển khai trên các nền tảng phần cứng chuyên dụng như GPU, TPU để tăng tốc quá trình huấn luyện và suy luận.

6- Tối ưu chất lượng hình ảnh xuất ra:

Nghiên cứu các kỹ thuật cải thiện chất lượng hình ảnh xuất ra từ các mô hình CNN, đặc biệt là trong các ứng dụng như phục hồi hình ảnh, tăng độ phân giải (super-resolution), và lọc nhiễu. Những kỹ thuật này không chỉ giúp nâng cao trải nghiệm người dùng mà còn tăng độ chính xác của các hệ thống phân tích hình ảnh.

TÀI LIỆU THAM KHẢO

[1]Image classification using CNN – 88%

https://www.kaggle.com/code/faressayah/cifar-10-images-classification-using-cnns-88

[2] Image Classification on CIFAR-10 - Papers With Code

https://paperswithcode.com/sota/image-classification-on-cifar-10

[3] Tìm Hiểu về Convolutional CNN

https://viblo.asia/p/tim-hieu-ve-convolutional-neural-networks-cnn-naQZRkr0lvx

[4] Chat GPT

https://chatgpt.com/