TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

Học phần: NHẬP MÔN XỬ LÝ ẢNH

Đề tài: Getting Started with Image Processing

Deep Learning in Image Processing – Image Classification

Giảng viên hướng dẫn: TS. Đỗ Bảo Sơn

Lớp: **73DCTT23**

Nhóm: **5**

Thành viên nhóm: **Bạc Anh Tuấn**

Kiều Minh Quân

Đặng Ngọc Hiệu

Nguyễn Hải Dương

Đỗ Quốc Trường

Hà Nội, Năm 2025

Lời mở đầu

Trong bối cảnh công nghệ ngày càng phát triển mạnh mẽ, xử lý ảnh số trở thành một trong những lĩnh vực trọng yếu của khoa học máy tính và trí tuệ nhân tạo. Từ những khái niệm cơ bản về ảnh, cách ảnh được lưu trữ và biểu diễn trên máy tính, đến việc áp dụng các thuật toán tiên tiến để phân tích và diễn giải dữ liệu hình ảnh, xử lý ảnh đã và đang góp phần quan trọng trong nhiều lĩnh vực như y tế, công nghiệp, giao thông, an ninh và giải trí.

Ở Chương 1 – Bắt đầu với xử lý ảnh, tài liệu giúp người học tiếp cận những kiến thức nền tảng nhất: ảnh là gì, quy trình xử lý ảnh gồm những bước nào, cách thiết lập môi trường Python và làm quen với các thư viện phổ biến như NumPy, PIL, OpenCV, scikit-image… Ngoài ra, chương này còn cung cấp các thao tác cơ bản như đọc, ghi, hiển thị ảnh, thay đổi kích thước, chuyển đổi màu sắc hay lọc nhiễu. Đây chính là hành trang đầu tiên, tạo nền móng cho việc nghiên cứu và ứng dụng xử lý ảnh ở mức cao hơn.

Đến Chương 10 – Deep Learning trong xử lý ảnh, người học được mở rộng tầm nhìn sang những công nghệ hiện đại và mạnh mẽ nhất hiện nay. Deep Learning, đặc biệt là các mạng nơ-ron tích chập (CNN), đã tạo ra bước đột phá trong thị giác máy tính. Thông qua các ví dụ thực tiễn như phân loại chữ số MNIST, nhận diện mèo – chó hay áp dụng các kiến trúc CNN nổi tiếng (VGG, ResNet), chương này không chỉ minh họa khả năng vượt trội của Deep Learning trong xử lý ảnh mà còn khuyến khích người học tiếp cận những xu hướng mới trong nghiên cứu và ứng dụng trí tuệ nhân tạo.

Tài liệu vì thế vừa đóng vai trò là nền tảng cơ bản, vừa là cầu nối hướng tới các phương pháp hiện đại, giúp người học có cái nhìn toàn diện và hệ thống về xử lý ảnh.

Bảng phân công công việc

Công việc	Người thực hiện
Tìm hiểu chương 10	Bạc Anh Tuấn
Tìm hiểu chương 1	Kiều Minh Quân
Tìm hiểu chương 10	Đặng Ngọc Hiệu
Tìm hiểu chương 1	Nguyễn Hải Dương
Tìm hiểu chương 10	Đỗ Quốc Trường

MỤC LỤC

Mục Lục Bảng

CHƯƠNG I: BẮT ĐẦU VỚI XỬ LÍ ẢNH

1. Xử lý hình ảnh là gì và một số ứng dụng

1.1. Ảnh là gì và cách nó được lưu trữ trên máy tính

Về mặt khái niệm, một hình ảnh ở dạng đơn giản nhất (một kênh; ví dụ, ảnh nhị phân hoặc đơn sắc, ảnh thang độ xám hoặc đen trắng) là một hàm hai chiều f(x,y) ánh xạ một cặp tọa độ thành một giá trị số nguyên/thực, liên quan đến cường độ/màu sắc của điểm. Mỗi điểm được gọi là một điểm ảnh hoặc pel (phần tử ảnh).

- Với ảnh kênh đơn (single-channel) như ảnh nhị phân, ảnh đơn sắc hay ảnh mức xám, mỗi điểm ảnh chỉ mang một giá trị cường độ sáng.
- Với ảnh đa kênh, ví dụ ảnh màu RGB, một điểm ảnh được mô tả bằng ba giá trị (rx,y,gx,y,bx,y)(r_{x,y}, g_{x,y}, b_{x,y})(rx,y,gx,y,bx,y) tương ứng với ba kênh màu đỏ (Red), lục (Green), lam (Blue).

Quá trình số hóa ảnh

Để có thể xử lý ảnh trên máy tính, ảnh cần được số hóa cả về tọa độ không gian và giá trị biên độ:

1. Image Sampling (lấy mẫu ảnh): Quá trình rời rạc hóa các tọa độ không gian (x,y)(x, y)(x,y).
2. Gray-level Quantization (lượng tử hóa mức xám): Rời rạc hóa giá trị cường độ sáng/màu.
    - Trên máy tính, mỗi giá trị pixel thường được biểu diễn bằng số nguyên trong khoảng 0–255 (8 bit) hoặc số thực trong khoảng 0–1.

Cách lưu trữ ảnh trong máy tính

- Ảnh số sau khi số hóa sẽ được lưu trữ dưới dạng tệp (file).
- Các tệp này chứa metadata (thông tin bổ sung) và dữ liệu điểm ảnh, được sắp xếp dưới dạng mảng nhiều chiều (arrays):
    - Ảnh nhị phân/ảnh xám: được lưu dưới dạng ma trận 2 chiều kích thước *width* × *height*.
    - Ảnh màu RGB: được lưu dưới dạng mảng 3 chiều kích thước *width* × *height* × *3* (mỗi pixel có 3 giá trị kênh màu).
    - Các định dạng ảnh khác như YUV cũng có thể được lưu dưới dạng mảng 3 chiều tương tự.

Ví dụ minh họa

Một ví dụ trực quan cho thấy:

- Ảnh nhị phân (Binary): chỉ có 2 giá trị (đen và trắng).
- Ảnh xám (Grayscale): điểm ảnh biểu diễn bằng một giá trị cường độ sáng duy nhất.
- Ảnh màu RGB: mỗi điểm ảnh được tạo từ tổ hợp ba giá trị màu đỏ, lục, lam.

Ứng dụng trong xử lý ảnh

Trong lĩnh vực xử lý ảnh, dữ liệu ảnh sau khi số hóa sẽ được đưa vào máy tính để phân tích và xử lý bằng các thuật toán. Trong tài liệu này, việc xử lý sẽ được thực hiện bằng Python cùng các thư viện hỗ trợ, giúp:

- Đọc và trích xuất dữ liệu ảnh.
- Thực hiện các thao tác xử lý ảnh (lọc, biến đổi, phân đoạn, …).

1.2. Xử lý hình ảnh là gì?

Xử lý hình ảnh là quá trình sử dụng các thuật toán và mã lập trình trên máy tính để tự động xử lý, thao tác, phân tích và diễn giải hình ảnh. Lĩnh vực này có ứng dụng rộng rãi trong nhiều ngành khoa học và công nghệ, bao gồm truyền hình, nhiếp ảnh, robot, viễn thám, chẩn đoán y tế và kiểm định công nghiệp. Các nền tảng mạng xã hội quen thuộc như Facebook và Instagram, nơi hàng triệu hình ảnh được tải lên mỗi ngày, là những ví dụ điển hình về việc áp dụng và đổi mới các thuật toán xử lý hình ảnh để quản lý, phân tích và nâng cao chất lượng hình ảnh.

Trong tài liệu này, chúng ta sẽ sử dụng các thư viện Python để thực hiện các tác vụ xử lý hình ảnh. Nội dung được chia thành hai phần chính:

1. **Xử lý hình ảnh cổ điển**: Sử dụng các thư viện Python để trích xuất dữ liệu hình ảnh và áp dụng các thuật toán xử lý như:
    - Xử lý trước (pre-processing): Chuẩn bị dữ liệu hình ảnh.
    - Nâng cao (enhancement): Cải thiện chất lượng hình ảnh.
    - Khôi phục (restoration): Khôi phục hình ảnh bị nhiễu hoặc hỏng.
    - Biểu diễn (representation): Mô tả hình ảnh bằng các đặc trưng.
    - Phân đoạn (segmentation): Chia hình ảnh thành các vùng có ý nghĩa.
    - Phân loại (classification): Gán nhãn cho hình ảnh hoặc các đối tượng trong ảnh.
    - Phát hiện và nhận dạng (detection and recognition): Xác định và nhận diện đối tượng trong ảnh. Những kỹ thuật này giúp phân tích, hiểu và diễn giải dữ liệu hình ảnh một cách hiệu quả hơn.

2. **Xử lý hình ảnh dựa trên học sâu (deep learning)**: Sử dụng các thư viện chuyên biệt để áp dụng các kỹ thuật học sâu, một công nghệ đã trở nên phổ biến trong những năm gần đây, cho phép tự động trích xuất đặc trưng và xử lý các tác vụ phức tạp như nhận dạng đối tượng, phân đoạn ảnh, và chuyển đổi phong cách

1.3. Một số ứng dụng của xử lý hình ảnh

Xử lý ảnh không chỉ dừng lại ở lý thuyết mà có rất nhiều ứng dụng thực tiễn trong nhiều lĩnh vực khác nhau:

1.3.1. Y học và sinh học

- Chẩn đoán hình ảnh y tế:
    - Xử lý ảnh X-quang, CT scan, MRI để phát hiện khối u, gãy xương, dị tật.
    - Ví dụ: phần mềm AI đọc phim X-quang hỗ trợ bác sĩ phát hiện dấu hiệu viêm phổi.
- Kính hiển vi số: tăng cường độ nét, phân tích tế bào trong nghiên cứu sinh học.
- Phẫu thuật hỗ trợ bằng hình ảnh: dùng hình ảnh 3D tái tạo cơ thể để mô phỏng ca mổ.

1.3.2. Nhiếp ảnh và đồ họa máy tính

- Chỉnh sửa ảnh: phần mềm như Photoshop, GIMP đều dựa trên các kỹ thuật xử lý ảnh (lọc nhiễu, tăng độ nét, cân bằng màu).
- Tạo hiệu ứng hình ảnh: làm mờ, biến đổi màu, ghép ảnh.
- Chụp ảnh trên điện thoại thông minh: camera AI tự động làm đẹp, nhận diện cảnh, chỉnh sáng.

1.3.3. An ninh và xác thực sinh trắc học

- Nhận diện khuôn mặt: dùng trong điện thoại (Face ID), camera giám sát, hệ thống chấm công.
- Xác thực vân tay / mống mắt: mở khóa điện thoại, cửa an ninh.
- Giám sát thông minh: camera có khả năng nhận diện đối tượng khả nghi.

1.3.4. Công nghiệp và sản xuất

- Thị giác máy tính trong công nghiệp (machine vision):
    - Kiểm tra lỗi sản phẩm trên dây chuyền sản xuất (trầy xước, nứt, thiếu linh kiện).
    - Đo đạc tự động kích thước sản phẩm bằng camera.
- Xe tự lái: xử lý hình ảnh từ camera để nhận diện làn đường, biển báo, người đi bộ.

1.3.5. Viễn thám và khoa học môi trường

- Phân tích ảnh vệ tinh để:
    - Giám sát rừng, biển, sông ngòi.
    - Dự đoán thời tiết, phân tích biến đổi khí hậu.
    - Phát hiện cháy rừng, lũ lụt, ô nhiễm.

1.3.6. Giải trí và đời sống hàng ngày

- Ứng dụng AR/VR: gắn sticker, filter trên TikTok, Instagram.
- Game: phát hiện chuyển động, theo dõi người chơi bằng camera.
- Truyền hình: kỹ thuật phông xanh (green screen), hiệu ứng video.

2. Quy trình xử lý hình ảnh

Quy trình xử lý ảnh (Image Processing Pipeline) bao gồm nhiều bước liên tiếp nhằm biến dữ liệu hình ảnh thô thành thông tin có ý nghĩa. Trước hết, ảnh cần được thu nhận và lưu trữ thông qua các thiết bị như camera, sau đó được lưu lại dưới dạng tệp tin như JPEG. Tiếp theo, ảnh được nạp vào bộ nhớ và lưu bằng những cấu trúc dữ liệu thích hợp (ví dụ mảng numpy ndarray) để phục vụ cho việc xử lý.

Trong giai đoạn tiền xử lý, ảnh được thao tác và cải thiện chất lượng thông qua các kỹ thuật như chuyển đổi định dạng (chẳng hạn sang ảnh xám), lọc và khử nhiễu nhằm phục hồi thông tin. Sau đó, quá trình phân đoạn được thực hiện để tách ảnh thành các vùng hoặc đối tượng riêng biệt, giúp làm nổi bật những phần quan trọng. Ở bước trích chọn và biểu diễn đặc trưng, các thông tin đại diện cho ảnh được rút trích bằng những phương pháp thủ công như HOG descriptors hoặc được học tự động nhờ các mạng nơ-ron sâu. Cuối cùng, ảnh được phân loại và nhận dạng, chẳng hạn như xác định xem trong ảnh có chứa đối tượng nào hoặc tìm vị trí cụ thể của chúng.

Để triển khai các bước trên, có thể sử dụng nhiều thư viện hỗ trợ. Nhóm thư viện như scikit-image, PIL và matplotlib thường được dùng cho việc nhập xuất dữ liệu, hiển thị, vẽ và tính toán thông số ảnh. Các chức năng biến đổi, thao tác hình thái học được hỗ trợ bởi scikit-image transform, morphology và PIL.ImageMorph. Trong khi đó, các kỹ thuật lọc, khử nhiễu, phân đoạn và trích chọn đặc trưng được cung cấp bởi scikit-image filters, segmentation, feature modules và PIL.ImageFilter. Ngoài ra, scipy.ndimage và OpenCV thường được ứng dụng cho các tác vụ xử lý ảnh nâng cao, scikit-learn phục vụ cho các bài toán học máy cổ điển, còn TensorFlow và Keras được sử dụng trong các mô hình học sâu.

2.1. Thu nhận và lưu trữ ảnh

- Ảnh được thu từ camera, máy quét, thiết bị y tế (X-quang, MRI) hoặc từ file (JPEG, PNG).

- Ảnh thường lưu dưới dạng mảng số (numpy ndarray).
  Ví dụ: camera giám sát ghi lại hình ảnh và lưu thành file .mp4 hoặc .jpg.

2.2. Tiền xử lý

- Chuyển đổi định dạng màu: RGB → Grayscale.
- Lọc và khử nhiễu: loại bỏ hạt, mờ do rung tay.
- Cân bằng sáng, tăng độ tương phản để làm ảnh rõ hơn.
  Ví dụ: ảnh chụp X-quang thường nhiễu, cần lọc để bác sĩ dễ quan sát.

2.3. Phân đoạn ảnh

- Chia ảnh thành vùng hoặc đối tượng riêng biệt.
- Giúp tách phần quan trọng (như khối u trong ảnh CT, biển báo trên đường).
  Ví dụ: xe tự lái dùng segmentation để tách làn đường khỏi cảnh vật xung quanh.

2.4. Trích chọn và biểu diễn đặc trưng

- Lấy ra những đặc điểm nổi bật như cạnh, góc, đường viền, hoa văn.
- Có thể dùng HOG descriptors (nhận diện dáng người) hoặc deep learning để tự học đặc trưng.
  Ví dụ: hệ thống nhận diện khuôn mặt trích đặc trưng từ mắt, mũi, miệng.

2.5. Nhận dạng và phân loại

- Giai đoạn cuối: xác định ảnh chứa đối tượng nào, ở đâu.
- Sử dụng mạng nơ-ron, machine learning hoặc deep learning.
  Ví dụ: AI đọc ảnh y tế và phân loại "có khối u" hoặc "không khối u".

Các Thư viện Python được Sử dụng

Để thực hiện các tác vụ xử lý hình ảnh, chúng ta sẽ sử dụng các thư viện sau:

- **Xử lý hình ảnh cổ điển**:
  - scipy.ndimage: Hỗ trợ các thao tác xử lý ảnh cơ bản.
  - opencv: Dùng cho nhiều tác vụ như lọc, biến đổi và phân đoạn.
  - scikit-learn: Áp dụng các thuật toán học máy truyền thống để phân loại và nhận dạng.
- **Xử lý hình ảnh dựa trên học sâu**:
  - tensorflow và keras: Dùng để xây dựng và huấn luyện các mô hình học sâu, đặc biệt cho các tác vụ như phân loại, nhận dạng đối tượng và phân đoạn.

3. Cài đặt các thư viện xử lý hình ảnh khác nhau trong Python

Tiếp theo sẽ hướng dẫn cách cài đặt các thư viện xử lý hình ảnh khác nhau và thiết lập môi trường để viết mã xử lý hình ảnh bằng các kỹ thuật xử lý hình ảnh cổ điển trong Python

3.1. Cài đặt pip

Pip là công cụ quản lý gói trong Python, cho phép cài đặt và quản lý các thư viện cần thiết cho xử lý ảnh. Thông thường, pip đã được cài sẵn trong Python từ phiên bản 3.4 trở lên hoặc khi sử dụng Virtual Environment. Người dùng chỉ cần đảm bảo pip luôn ở phiên bản mới nhất để tránh lỗi khi cài đặt. Nếu chưa có, có thể tham khảo hướng dẫn từ tài liệu chính thức để cài đặt tùy theo hệ điều hành.

3.2. Cài đặt một số thư viện xử lý hình ảnh trong Python

Các thư viện phổ biến trong xử lý ảnh gồm: NumPy, SciPy, scikit-image, scikit-learn, Pillow (PIL), OpenCV, SimpleITK và Matplotlib. Mỗi thư viện có vai trò riêng: NumPy dùng để lưu trữ dữ liệu ảnh, Matplotlib để hiển thị, SciPy cho tăng cường ảnh, scikit-learn cho xây dựng mô hình học máy, còn scikit-image, mahotas và OpenCV phục vụ cho các thuật toán xử lý ảnh. Các thư viện này có thể được cài đặt bằng pip với các lệnh đơn giản. Sau khi cài đặt, người dùng có thể kiểm tra bằng cách import thư viện và in ra phiên bản để chắc chắn rằng quá trình cài đặt thành công.

Khối mã sau đây cho thấy cách các thư viện mà chúng ta sẽ sử dụng có thể được tải xuống và cài đặt bằng pip từ dấu nhắc Python (chế độ tương tác):

```
>>> pip install numpy
>>> pip install scipy
>>> pip install scikit-image
>>> pip install scikit-learn
>>> pip install pillow
>>> pip install SimpleITK
>>> pip install opencv-python
>>> pip install matplotlib
```

Nếu thư viện được nhập thành công (không có thông báo lỗi nào được đưa ra), thì chúng ta không gặp bất kỳ sự cố cài đặt nào. Chúng ta có thể in phiên bản thư viện đã cài đặt bằng cách in nó ra bảng điều khiển.

Khối mã sau đây hiển thị phiên bản của các thư viện Python scikit-image và PIL:

```
>>> import skimage, PIL, numpy
>>> print(skimage.__version__)
# 0.14.0
>>> PIL.__version__
# 5.1.0
>>> numpy.__version__
# 1.14.5
```

Hãy đảm bảo rằng chúng ta có phiên bản mới nhất của tất cả các thư viện.

3.3. Cài đặt bản phân phối Anaconda

Anaconda là bản phân phối Python tích hợp sẵn nhiều thư viện khoa học dữ liệu và xử lý ảnh, giúp tiết kiệm thời gian cài đặt. Khi sử dụng Anaconda, người dùng không cần cài đặt từng gói riêng lẻ như với pip, đồng thời có thể dễ dàng quản lý môi trường ảo. Đây là lựa chọn thuận tiện và phù hợp cho các dự án xử lý ảnh lớn hoặc có yêu cầu phức tạp.

3.4. Cài đặt Jupyter Notebook

Jupyter Notebook là công cụ trực quan hỗ trợ viết và chạy mã Python. Nếu cài đặt bằng pip, ta sử dụng lệnh pip install jupyter và khởi chạy bằng jupyter notebook. Nếu dùng Anaconda, Jupyter đã được tích hợp sẵn và có thể sử dụng ngay. Công cụ này cho phép viết mã, chạy thử và trực quan hóa kết quả trong cùng một giao diện, rất hữu ích cho việc thực hành xử lý ảnh.

4. Đọc, Ghi và Hiển thị hình ảnh với Python

Hình ảnh được lưu trữ dưới dạng tệp trên đĩa, do đó việc đọc và ghi hình ảnh từ tệp là các thao tác I/O trên đĩa. Những thao tác này có thể được thực hiện bằng nhiều cách khác nhau, sử dụng các thư viện khác nhau; một số trong số đó được trình bày trong phần này. Trước tiên, hãy bắt đầu bằng cách nhập tất cả các gói cần thiết:

```
# for inline image display inside notebook
# % matplotlib inline import numpy as np
from PIL import Image, ImageFont, ImageDraw
from PIL.ImageChops import add, subtract, multiply, difference,
screen import PIL.ImageStat as stat
from skimage.io import imread, imsave, imshow, show,
imread_collection, imshow_collection from skimage import color,
```

```
viewer, exposure, img_as_float, data from skimage.transform
import SimilarityTransform, warp, swirl

from skimage.util import invert, random_noise, montage

import matplotlib.image as mpimg

import matplotlib.pylab as plt from scipy.ndimage

import affine_transform, zoom from scipy

import misc
```

4.1. Đọc, lưu và hiển thị hình ảnh bằng PIL

- Sử dụng Image.open(path) để đọc ảnh.
- Có thể lấy thông tin ảnh: chiều rộng, chiều cao, định dạng, chế độ màu.
- Dùng im.show() để hiển thị ảnh.

im = Image.open("../images/parrot.png")

```
# read the image, provide the correct path print(im.width,
im.height, im.mode, im.format, type(im))

# 453 340 RGB PNG im.show() # display the image
```

- Dùng convert('L') để chuyển ảnh màu RGB sang grayscale.
- Dùng im.save(path) để lưu ảnh ra file.

```
im_g = im.convert('L')
# convert the RGB color

image to a grayscale image im_g.save('../images/parrot_gray.png')
# save the image to disk
Image.open("../images/parrot_gray.png").show()
# read the grayscale image

from disk and show
```

- Nên đặt ảnh trong một thư mục riêng (ví dụ: images) để tránh lỗi đường dẫn.

4.2. Đọc, lưu và hiển thị hình ảnh bằng Matplotlib

- Sử dụng mpimg.imread(path) để đọc ảnh dưới dạng NumPy ndarray (giá trị pixel float [0,1]).
- Dùng plt.imshow(im) để hiển thị ảnh.

- Có thể xử lý trực tiếp mảng pixel, ví dụ: gán các giá trị <0.5 = 0 để làm ảnh tối hơn. Dùng plt.savefig(path) để lưu ảnh sau khi xử lý.

- imshow() hỗ trợ nhiều phương pháp nội suy (interpolation) như nearest, bilinear, bicubic… rất hữu ích khi hiển thị ảnh nhỏ.

4.3. Đọc, lưu và hiển thị hình ảnh bằng scikit-image

- Sử dụng imread(path) để đọc ảnh (kiểu dữ liệu uint8, giá trị pixel [0–255]).

- Có thể chuyển đổi không gian màu, ví dụ: RGB ↔ HSV, thay đổi độ bão hòa rồi chuyển ngược lại để tạo ảnh mới.

- Sử dụng imsave(path, im) để lưu ảnh đã chỉnh sửa.

- Có thể hiển thị ảnh bằng imshow(im), show() hoặc mở cửa sổ pop-up bằng viewer.ImageViewer(im).

- Thư viện có sẵn dataset mẫu như astronaut, cameraman để thực hành.

- Có thể dùng imread_collection() để đọc nhiều ảnh cùng lúc và hiển thị bằng imshow_collection().

4.4. Đọc, lưu và hiển thị hình ảnh bằng scipy.misc

- Dùng misc.face() để lấy ảnh mẫu (mặt chồn hôi).

- Có thể lưu ảnh bằng misc.imsave(path, im) và hiển thị với plt.imshow(im).

5. Xử lý các loại hình ảnh, định dạng tệp và thực hiện các thao tác cơ bản

Một hình ảnh có thể được lưu ở nhiều định dạng tệp và chế độ (kiểu) khác nhau. Hãy cùng thảo luận về cách xử lý hình ảnh ở các định dạng và kiểu tệp khác nhau bằng các thư viện Python.

5.1. Xử lý các loại hình ảnh và định dạng tệp

5.1.1. Định dạng tập tin

- Ảnh số tồn tại dưới nhiều định dạng tệp khác nhau, mỗi định dạng có ưu/nhược điểm:
    - JPEG: định dạng nén mất dữ liệu (lossy), kích thước nhỏ, thường dùng trong ảnh chụp, web.
    - PNG: nén không mất dữ liệu (lossless), hỗ trợ kênh trong suốt (alpha channel).
    - BMP: ảnh bitmap, kích thước lớn, ít dùng trong thực tế nhưng đơn giản cho xử lý.
    - TIFF: chất lượng cao, hay dùng trong in ấn, xử lý ảnh chuyên nghiệp.

    - GIF: ảnh động (animation), giới hạn 256 màu.
- Khi xử lý, cần nắm rõ đặc tính định dạng để chọn loại phù hợp với mục tiêu: hiển thị web, in ấn, phân tích dữ liệu hay lưu trữ lâu dài.

5.1.2. Các loại hình ảnh

- Ảnh nhị phân: chỉ có 2 giá trị pixel (đen/trắng, 0/1). Thường dùng trong xử lý biên dạng, mặt nạ.
- Ảnh thang xám (grayscale): pixel có giá trị từ 0–255 (8-bit) biểu diễn độ sáng. Dùng phổ biến trong xử lý biên cạnh, lọc.
- Ảnh màu RGB: 3 kênh (đỏ, lục, lam), mỗi kênh 0–255. Kết hợp tạo nên ảnh màu đầy đủ.
- Ảnh RGBA: thêm kênh alpha (độ trong suốt).
- Ảnh đa phổ / hyperspectral: nhiều kênh hơn RGB, thường trong viễn thám, y học.

5.1.3. Một số không gian màu

- RGB: biểu diễn dựa trên ánh sáng, thích hợp hiển thị màn hình.
- HSV/HSL: tách riêng Hue (màu sắc), Saturation (độ bão hòa), Value/Lightness (độ sáng). Dễ dùng khi thay đổi màu.
- CMYK: mô hình màu dùng trong in ấn (Cyan, Magenta, Yellow, Black).
- Grayscale: đơn kênh, lưu độ sáng, thường để giảm tính toán.
- Việc chọn không gian màu phù hợp giúp các thuật toán xử lý ảnh (lọc, phân loại, nhận diện) chính xác và dễ dàng hơn.

5.1.4. Cấu trúc dữ liệu lưu trữ hình ảnh

- Trong Python, ảnh thường được lưu thành numpy array:
    - Ảnh xám: mảng 2 chiều (height × width).
    - Ảnh màu RGB: mảng 3 chiều (height × width × 3).
    - Ảnh RGBA: (height × width × 4).
- Mỗi pixel là một giá trị số (0–255 với ảnh 8-bit).
- Đây là cấu trúc quan trọng vì hầu hết các thư viện xử lý ảnh (OpenCV, PIL, scikit-image) đều làm việc dựa trên mảng này.

5.2. Các thao tác hình ảnh cơ bản

5.2.1. Thao tác hình ảnh với phân đoạn mảng numpy

- Numpy cho phép thao tác trực tiếp với pixel như thao tác mảng:

- Cắt (crop) vùng ảnh: image[100:200, 50:150]
- Lật ảnh: image[::-1] (lật dọc), image[:, ::-1] (lật ngang).
- Thay đổi giá trị vùng ảnh: tô màu, tạo mặt nạ.

- Ưu điểm: cực nhanh, tận dụng được sức mạnh tính toán vector hóa.

5.2.2. Thao tác hình ảnh với PIL (Thư viện hình ảnh Python)

- PIL (nay là Pillow) là thư viện mạnh mẽ, dễ dùng cho thao tác cơ bản:
  - Mở/lưu ảnh nhiều định dạng khác nhau.
  - Chuyển đổi định dạng: từ JPEG sang PNG, từ RGB sang Grayscale.
  - Thay đổi kích thước (resize), xoay (rotate), crop.
  - Thêm bộ lọc đơn giản (blur, sharpen).
- Thích hợp khi cần xử lý nhanh, đọc/ghi nhiều định dạng ảnh.

5.2.3. Thao tác hình ảnh với scikit-image

- Cung cấp nhiều thuật toán xử lý ảnh nâng cao:
  - Lọc ảnh (Gaussian, Median).
  - Phát hiện biên (edge detection): Sobel, Canny.
  - Phân vùng ảnh (segmentation): watershed, region growing.
  - Histogram & cân bằng sáng tối.
- Phù hợp cho các bài toán nghiên cứu và ứng dụng phân tích hình ảnh.

5.2.4. Thao tác hình ảnh với Matplotlib

- Chủ yếu dùng để hiển thị ảnh và vẽ dữ liệu:
  - plt.imshow(image) để hiển thị ảnh.
  - Áp dụng colormap (gray, jet, viridis) để làm nổi bật dữ liệu.
  - Vẽ chồng dữ liệu (vẽ đường, chấm, chữ lên ảnh).
- Thường dùng trong giai đoạn trực quan hóa kết quả.

5.2.5. Thao tác hình ảnh với các mô-đun scipy.misc và scipy.ndimage

- scipy.misc và scipy.ndimage cung cấp các hàm xử lý ảnh khoa học:
  - Bộ lọc Gaussian, Median.
  - Xoay ảnh (rotate), dịch chuyển (shift), zoom.

    - Phép biến đổi hình học (affine, spline).
- Mạnh về tính toán khoa học, nhưng ít dùng hơn OpenCV hay scikit-image trong thực tế.

CHƯƠNG X: HỌC SÂU TRONG XỬ LÝ HÌNH ẢNH – PHÂN LOẠI HÌNH ẢNH

1. GIỚI THIỆU VỀ HỌC SÂU TRONG XỬ LÝ ẢNH

1.1. Định nghĩa

Deep Learning (Học Sâu) là một nhánh con của Machine Learning, sử dụng các mô hình mạng nơ-ron nhân tạo (ANN) nhiều tầng để học từ dữ liệu. Điểm mạnh của Deep Learning nằm ở khả năng học biểu diễn nhiều cấp độ: từ đặc trưng mức thấp (cạnh, màu sắc) đến mức cao (hình dạng, đối tượng).

Trong xử lý ảnh, Deep Learning – đặc biệt là CNN – đã thay thế các phương pháp truyền thống vốn dựa nhiều vào feature engineering thủ công. CNN cho phép học đầu-cuối (end-to-end), từ dữ liệu ảnh thô cho đến nhãn phân loại.

1.2. Ứng dụng

Deep Learning trong xử lý ảnh được ứng dụng rộng rãi:

- Y tế: phát hiện khối u từ ảnh MRI, X-quang.
- An ninh: nhận diện khuôn mặt, phát hiện xâm nhập.
- Ô tô tự lái: phát hiện làn đường, biển báo, người đi bộ.
- Thương mại: gợi ý sản phẩm dựa trên ảnh, tìm kiếm hình ảnh tương tự.
- Nông nghiệp: phân loại cây trồng, phát hiện sâu bệnh qua ảnh vệ tinh.

1.3. Mô hình

Một mô hình Deep Learning điển hình trong xử lý ảnh:

1. Input: ảnh đầu vào (RGB hoặc grayscale).
2. Feature Extraction: các lớp CNN tự động trích xuất đặc trưng.
3. Classification: các lớp Fully Connected phân loại dựa trên đặc trưng.
4. Output: nhãn ảnh (ví dụ: mèo, chó, số viết tay).

Ví dụ code (Keras – mô hình CNN cơ bản):

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense

model = Sequential([

    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),

    MaxPooling2D((2,2)),

    Conv2D(64, (3,3), activation='relu'),

    MaxPooling2D((2,2)),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(10, activation='softmax')

])
```

2. HỌC MÁY CỔ ĐIỂN VS HỌC SÂU

2.1. Định nghĩa

- Học máy cổ điển (Classical Machine Learning):
  Là nhóm các thuật toán học máy truyền thống như K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Trees, Random Forest... Các mô hình này thường yêu cầu trích xuất đặc trưng thủ công trước khi đưa vào huấn luyện.
- Học sâu (Deep Learning):
  Là một nhánh con của Machine Learning, sử dụng mạng nơ-ron nhiều tầng (Deep Neural Networks – DNN), đặc biệt trong xử lý ảnh là mạng nơ-ron tích chập (CNN). Deep Learning có khả năng học đặc trưng tự động từ dữ liệu thô và đưa ra giải pháp đầu-cuối (end-to-end).

2.2. So sánh đặc trưng

Tiêu chí	Học máy cổ điển (Classical ML)	Học sâu (Deep Learning)
Trích xuất đặc trưng	Cần thiết kế thủ công (HOG, SIFT, SURF...)	Tự động học đặc trưng nhiều tầng qua CNN
Pipeline xử lý	Gồm nhiều bước: tiền xử lý → trích xuất đặc trưng → ML	End-to-end: ảnh thô → CNN → phân loại
Dữ liệu yêu cầu	Hoạt động tốt với tập dữ liệu nhỏ hoặc trung bình	Cần dữ liệu rất lớn để huấn luyện hiệu quả
Tài nguyên	Ít tốn kém, chạy tốt trên CPU	Cần GPU/TPU mạnh để huấn luyện nhanh
Khả năng mở rộng	Giới hạn khi dữ liệu phức tạp, khó khái quát	Rất tốt, dễ thích nghi nhờ Transfer Learning
Ứng dụng điển hình	OCR đơn giản, phân loại hoa lá, bài toán toy datasets	Nhận diện khuôn mặt, xe tự hành, y tế, NLP

Bảng 2.2: So sánh học máy cổ điển và học sâu

2.3. Điểm mạnh và điểm yếu

Học máy cổ điển

- **Điểm mạnh:**
    - Đơn giản, dễ triển khai.
    - Ít tốn tài nguyên phần cứng.
    - Hoạt động tốt trên tập dữ liệu nhỏ.
- **Điểm yếu:**
    - Phụ thuộc vào feature engineering (cần chuyên gia).
    - Khó khái quát khi dữ liệu lớn/phức tạp.
    - Không tận dụng tốt dữ liệu thô (ví dụ: ảnh RGB trực tiếp).

Học sâu

- **Điểm mạnh:**
    - Tự động trích xuất đặc trưng, không cần feature engineering.
    - Giải pháp end-to-end, dễ áp dụng cho nhiều bài toán.
    - Độ chính xác vượt trội khi có đủ dữ liệu và phần cứng.
    - Có thể tái sử dụng mô hình pretrained (Transfer Learning).
- **Điểm yếu:**
    - Yêu cầu dữ liệu lớn để tránh overfitting.
    - Huấn luyện tốn nhiều thời gian và phần cứng mạnh (GPU/TPU).
    - "Hộp đen" khó giải thích (thiếu tính minh bạch).

2.4. Lợi ích của Học Sâu

Trong nhiều năm, học máy cổ điển đã đóng vai trò quan trọng trong xử lý ảnh, đặc biệt với các thuật toán như SVM, KNN hay Random Forest. Tuy nhiên, khi dữ liệu ngày càng lớn và phức tạp, những phương pháp truyền thống này bắt đầu bộc lộ nhiều hạn chế. Chúng thường cần nhiều bước xử lý thủ công như tiền xử lý, trích xuất đặc trưng (HOG, SIFT, SURF), sau đó mới đưa vào thuật toán phân loại. Điều này không chỉ tốn nhiều công sức từ chuyên gia mà còn làm giảm khả năng khái quát của mô hình.

Deep Learning ra đời đã thay đổi cách tiếp cận. Điểm mạnh nổi bật nhất là khả năng học đặc trưng tự động. Thay vì cần con người định nghĩa các đặc trưng nào là quan trọng, mạng nơ-ron nhiều tầng sẽ tự động học chúng trực tiếp từ dữ liệu. Các mô hình học sâu, đặc biệt là mạng CNN trong xử lý ảnh, có thể phát hiện các đặc trưng từ mức thấp (cạnh, màu sắc, góc cạnh) cho đến mức cao (khuôn mặt, đồ vật) theo cách phân cấp.

Một lợi thế khác của học sâu là tính end-to-end. Thay vì chia bài toán thành nhiều bước rời rạc (feature extraction → classification), một mô hình deep learning có thể nhận đầu vào là ảnh thô và xuất trực tiếp nhãn phân loại, đơn giản hóa toàn bộ pipeline.

Deep Learning cũng phát huy sức mạnh khi dữ liệu lớn. Nếu như các mô hình truyền thống thường dừng lại ở một mức độ cải thiện nhất định dù có thêm dữ liệu, thì mạng học sâu lại càng hoạt động tốt hơn khi dữ liệu nhiều hơn. Điều này rất phù hợp với thời đại big data.

Thêm vào đó, sự phát triển của phần cứng (GPU, TPU) và các thư viện mã nguồn mở như TensorFlow, Keras, PyTorch đã giúp việc triển khai và huấn luyện các mô hình deep learning trở nên dễ dàng và thực tiễn hơn bao giờ hết. Ngoài ra, khái niệm transfer learning cho phép sử dụng các mô hình đã huấn luyện sẵn (như VGG, ResNet, Inception) cho các tác vụ mới, tiết kiệm đáng kể thời gian và công sức.

Tất cả những lý do trên khiến Deep Learning trở thành một công cụ mạnh mẽ, gần như không thể thiếu trong các bài toán xử lý ảnh hiện đại, từ nhận diện khuôn mặt, phân loại vật thể, đến xe tự lái và chẩn đoán y tế.

3. MẠNG NƠ-RON TÍCH CHẬP (CNN)

3.1. Giới thiệu CNN

Mạng nơ-ron tích chập (Convolutional Neural Network – CNN) là một kiến trúc quan trọng trong Deep Learning, được thiết kế đặc biệt cho dữ liệu dạng lưới, điển hình là ảnh số. CNN lấy cảm hứng từ cơ chế thị giác của não người, nơi các tế bào thần kinh phản ứng với các vùng cục bộ trong không gian thị giác.

Trong xử lý ảnh, CNN đã chứng minh được hiệu quả vượt trội nhờ khả năng tự động học đặc trưng phân cấp từ dữ liệu thô, thay thế cho việc trích xuất đặc trưng thủ công như trong học máy cổ điển.

3.2. Cách CNN hoạt động

Một CNN cơ bản thường gồm các lớp sau:

- Lớp tích chập (Convolutional Layer):
  Các bộ lọc (filter/kernel) có kích thước nhỏ được trượt trên ảnh để phát hiện đặc trưng cục bộ. Mỗi bộ lọc học được một loại đặc trưng khác nhau, ví dụ như cạnh ngang, cạnh dọc, đường cong.
- Hàm kích hoạt (Activation Function):
  Sau convolution, thường sử dụng hàm ReLU (Rectified Linear Unit) để đưa phi tuyến vào mạng, giúp mô hình học được các quan hệ phức tạp hơn.
- Lớp gộp (Pooling Layer):
  Thường dùng MaxPooling để giảm kích thước dữ liệu (downsampling), giảm số tham số và tăng tính bất biến đối với dịch chuyển nhỏ trong ảnh.

- Lớp kết nối đầy đủ (Fully Connected Layer):
  Sau nhiều lớp convolution và pooling, các đặc trưng được làm phẳng (flatten) và đưa vào lớp kết nối đầy đủ để thực hiện phân loại.
- Dropout (tùy chọn):
  Một kỹ thuật regularization, trong đó một số neuron bị vô hiệu hóa ngẫu nhiên trong quá trình huấn luyện để giảm hiện tượng overfitting.
- Softmax (Output Layer):
  Ở lớp cuối, hàm softmax được dùng để tạo phân phối xác suất cho các lớp dự đoán.

3.3. Đặc điểm học phân cấp

Điểm mạnh của CNN là khả năng học đặc trưng ở nhiều cấp độ khác nhau:

- Lớp đầu tiên: học các đặc trưng mức thấp như đường thẳng, cạnh, điểm sáng – tối.
- Các lớp giữa: học các mẫu phức tạp hơn như họa tiết, kết cấu, hình dạng cơ bản.
- Các lớp cuối: học đặc trưng ở mức trừu tượng cao như khuôn mặt, động vật, hay vật thể cụ thể.

Nhờ đặc tính này, CNN có thể tự động xây dựng biểu diễn dữ liệu phân cấp mà không cần sự can thiệp của con người.

3.4. Lợi ích của CNN so với ANN thông thường

CNN có nhiều lợi thế so với mạng nơ-ron truyền thống (ANN – Artificial Neural Network):

- Giảm số lượng tham số: nhờ cơ chế chia sẻ trọng số (weight sharing) trong convolution.
- Hiệu quả cho dữ liệu ảnh lớn: xử lý được ảnh độ phân giải cao mà không cần flatten toàn bộ.
- Bất biến dịch chuyển: có khả năng nhận diện đối tượng ngay cả khi chúng di chuyển nhỏ trong ảnh.
- Khả năng tổng quát hóa mạnh: nhờ học đặc trưng đa cấp độ.

Chính nhờ những đặc điểm này, CNN đã trở thành kiến trúc chuẩn mực trong hầu hết các ứng dụng xử lý ảnh hiện đại.

4. PHÂN LOẠI ẢNH VỚI TENSORFLOW/KERAS (MNIST DATASET)

4.1. Giới thiệu MNIST dataset

MNIST (Modified National Institute of Standards and Technology) là một trong những bộ dữ liệu hình ảnh nổi tiếng và được sử dụng rộng rãi nhất trong học máy. Bộ dữ liệu này bao gồm:

- 60.000 ảnh chữ số viết tay (0–9) để huấn luyện.
- 10.000 ảnh chữ số viết tay để kiểm thử.
- Mỗi ảnh có kích thước 28x28 pixel và là ảnh grayscale (1 kênh).

Đây là bộ dữ liệu kinh điển để thử nghiệm các thuật toán phân loại ảnh, bởi vì nó vừa đủ đơn giản cho người mới, nhưng cũng đủ thử thách để kiểm tra hiệu quả của các mô hình.

4.1.1. Quy trình thực hiện với Keras

Bài toán phân loại chữ số viết tay trên MNIST với Keras được triển khai qua các bước sau:

1. **Nạp dữ liệu và tiền xử lý**
    - Dữ liệu được tải trực tiếp từ Keras.
    - Chuẩn hóa giá trị pixel về khoảng [0,1].
    - Chuyển đổi nhãn sang dạng one-hot encoding.
2. **Xây dựng kiến trúc CNN**
    - Một số lớp Conv2D và MaxPooling2D để trích xuất đặc trưng.
    - Flatten để chuyển tensor thành vector.
    - Một lớp Dense ẩn với activation ReLU.
    - Lớp đầu ra Dense với 10 neuron (cho 10 chữ số) và activation softmax.
3. **Biên dịch mô hình (compile)**

- Loss function: categorical_crossentropy.
- Optimizer: adam.
- Metric: accuracy.

4. **Huấn luyện mô hình (fit)**
    - Epoch: 5–10.
    - Batch size: 128.
    - Validation split: 0.1 để theo dõi overfitting.
5. **Đánh giá và dự đoán**
    - Đánh giá trên tập test.
    - Dự đoán một số ảnh để kiểm tra kết quả trực quan.

4.1.2. Cài đặt với Keras

* Code minh họa

```
import tensorflow as tf

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense

from tensorflow.keras.utils import to_categorical


# 1. Nạp dữ liệu

(x_train, y_train), (x_test, y_test) = mnist.load_data()


# 2. Tiền xử lý

x_train = x_train.reshape((60000, 28, 28, 1)).astype('float32') /
255

x_test = x_test.reshape((10000, 28, 28, 1)).astype('float32') /
255
```

```
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# 3. Xây dựng CNN
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D((2,2)),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D((2,2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# 4. Compile và huấn luyện
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=5, batch_size=128,
                    validation_split=0.1, verbose=1)

# 5. Đánh giá mô hình
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print("Test accuracy:", test_acc)
```

4.1.3. Kết quả thực nghiệm

Khi chạy mô hình CNN cơ bản trên MNIST, kết quả thường đạt:

- Độ chính xác trên tập train: ~99%.
- Độ chính xác trên tập test: ~98–99%.

Điều này chứng minh rằng, ngay cả với một kiến trúc CNN đơn giản, Deep Learning vẫn vượt trội so với nhiều thuật toán học máy cổ điển (ví dụ: SVM, KNN thường chỉ đạt khoảng 95–97% trên MNIST).

4.1.4 Quy trình thực hiện với TensorFlow

Bài toán phân loại chữ số viết tay trên MNIST với TensorFlow được triển khai qua các bước sau:

1. **Nạp dữ liệu và tiền xử lý**
    - Tải dữ liệu trực tiếp từ tf.keras.datasets.
    - Chuẩn hóa giá trị pixel về [0,1].
    - Thêm một chiều kênh (28x28x1) để phù hợp với CNN.
2. **Xây dựng kiến trúc CNN**
    - Lớp Conv2D và MaxPooling2D để trích xuất đặc trưng.
    - Lớp Flatten để chuyển tensor thành vector.
    - Lớp Dense ẩn với activation ReLU.
    - Lớp đầu ra Dense với 10 neuron (cho 10 chữ số) và softmax.
3. **Biên dịch mô hình (compile)**
    - Loss function: sparse_categorical_crossentropy.
    - Optimizer: adam.
    - Metric: accuracy.
4. **Huấn luyện mô hình (fit)**
    - Epoch: 5–10.

- Batch size: 128.
- Validation split: 0.1 để theo dõi overfitting.

5. **Đánh giá và dự đoán**
    - Đánh giá trên tập test.
    - Dự đoán thử một số ảnh từ tập test để quan sát trực quan.

4.1.5. Ví dụ code TensorFlow

```
import tensorflow as tf


# 1. Nạp dữ liệu
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.mnist.load_data()


# 2. Tiền xử lý
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]


# 3. Xây dựng CNN bằng tf.keras
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu',
input_shape=(28,28,1)),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Flatten(),
```

```
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

# 4. Compile mô hình
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 5. Huấn luyện mô hình
history = model.fit(x_train, y_train, epochs=5, batch_size=128,
                    validation_split=0.1, verbose=1)

# 6. Đánh giá mô hình
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print("Test accuracy:", test_acc)
```

4.1.6. Kết quả thực nghiệm

Khi chạy mô hình CNN trên MNIST với TensorFlow, ta thu được:

- Độ chính xác trên tập train: ~99%.
- Độ chính xác trên tập test: ~98–99%.

Kết quả này cho thấy TensorFlow và Keras đều mang lại hiệu quả tương tự, chỉ khác nhau ở cách gọi API.

4.2. Ý nghĩa của ví dụ MNIST

1. **Minh họa pipeline chuẩn của Deep Learning trong xử lý ảnh**

- Nạp dữ liệu → tiền xử lý → xây dựng mô hình CNN → huấn luyện → đánh giá → dự đoán.
- Qua MNIST, người học hiểu trọn vẹn quy trình end-to-end.

2. **Chứng minh sức mạnh của CNN**
   - Một kiến trúc CNN rất đơn giản cũng đã đạt độ chính xác 98–99%, vượt xa nhiều mô hình học máy cổ điển (SVM, KNN chỉ đạt 95–97%).
   - Điều này cho thấy CNN có khả năng tự động học đặc trưng hiệu quả hơn nhiều so với feature engineering thủ công.
3. **Bước đệm để tiếp cận các bài toán phức tạp**
   - Dù đơn giản, MNIST giúp ta thấy CNN có thể xử lý tốt dữ liệu ảnh nhỏ.
   - Từ đó, có thể mở rộng sang các bài toán thực tế hơn như: phân loại mèo/chó (cats vs dogs), nhận diện khuôn mặt, phân tích ảnh y tế, ảnh vệ tinh.
4. **Khẳng định vai trò của Deep Learning trong kỷ nguyên dữ liệu lớn**
   - Với dữ liệu và phần cứng phù hợp, Deep Learning không chỉ giải quyết MNIST mà còn có thể xử lý những vấn đề cực kỳ phức tạp trong thực tế.

5. TRỰC QUAN HÓA CNN (VISUALIZATION OF CNNS)

5.1. Giới thiệu về mục tiêu và ý nghĩa của việc trực quan hóa CNN

Một trong những điểm thường gây tranh luận khi làm việc với Deep Learning nói chung và CNN nói riêng là tính "hộp đen" (black-box). Các mô hình này thường có hàng triệu, thậm chí hàng trăm triệu tham số, trải qua hàng chục hoặc hàng trăm lớp. Điều này khiến người dùng rất khó hiểu được:

CNN đã học cái gì ở bên trong?

Tại sao CNN lại đưa ra dự đoán chính xác?

Mạng có thực sự chú ý đến những đặc trưng quan trọng, hay chỉ học "vẹt" từ dữ liệu huấn luyện?

Trực quan hóa CNN (Visualization of CNNs) ra đời để giải quyết vấn đề này. Mục tiêu của trực quan hóa là:

Hiểu cơ chế học đặc trưng: CNN học theo cách phân cấp (hierarchical). Ở các lớp đầu, mạng thường phát hiện ra những đặc trưng cơ bản như cạnh, góc, điểm sáng/tối. Khi đi sâu hơn, các lớp giữa học được họa tiết, kết cấu, hoặc các hình dạng trung gian. Ở những lớp cuối, CNN tổng hợp các đặc trưng này thành biểu diễn trừu tượng để nhận diện toàn bộ đối tượng.

Cung cấp bằng chứng trực quan: bằng cách hiển thị các filters (bộ lọc) và feature maps (bản đồ đặc trưng), ta có thể nhìn thấy rõ CNN đang "nhìn" ảnh đầu vào như thế nào ở mỗi tầng.

Giúp kiểm chứng và cải tiến mô hình: nếu CNN học đúng, các feature maps phải thể hiện được các đặc trưng có ý nghĩa. Ngược lại, nếu CNN học sai (ví dụ feature maps toàn trắng hoặc nhiễu), mô hình có thể đang gặp vấn đề như overfitting, kiến trúc chưa hợp lý, hoặc dữ liệu chưa đủ đa dạng.

Tăng độ tin cậy và khả năng giải thích (interpretability): trong các ứng dụng quan trọng như y tế, an ninh, xe tự hành, việc hiểu rõ mô hình nhìn thấy gì là rất quan trọng để con người tin tưởng vào kết quả của AI.

5.2. Filters và Feature Maps

Filters (bộ lọc):
Mỗi filter trong lớp convolution học một loại đặc trưng riêng từ ảnh.

Lớp đầu: filter thường học các cạnh ngang, dọc, chéo, hoặc vùng sáng – tối.

Lớp giữa: filter học các hoa văn, đường cong, kết cấu.

Lớp sâu: filter học các đặc trưng phức tạp như hình dạng khuôn mặt, tai mèo, hoặc con số.

Feature maps (bản đồ đặc trưng):
Khi một ảnh đi qua mạng CNN, mỗi filter tạo ra một bản đồ đặc trưng. Tập hợp các feature map cho thấy "cách nhìn" của mạng đối với ảnh ở từng tầng.

Ví dụ: khi đưa ảnh số 7 viết tay vào CNN:

Lớp đầu → phát hiện các cạnh dọc và ngang.

Lớp giữa → phát hiện cấu trúc cong đặc trưng của số 7.

Lớp cuối → nhận diện toàn bộ hình dạng số 7.

5.3. Ví dụ trực quan hóa CNN trong Keras

Dưới đây là ví dụ minh họa cách hiển thị các feature map của một ảnh qua các lớp CNN:

```
import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow.keras.models import Model

from tensorflow.keras.datasets import mnist


# 1. Nạp dữ liệu

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_test = x_test.reshape((10000, 28, 28, 1)).astype('float32') /
255


# 2. Dùng mô hình CNN đã huấn luyện từ phần trước

model = tf.keras.models.load_model("mnist_cnn.h5")  # giả sử đã
lưu mô hình


# 3. Chọn một ảnh để trực quan hóa

img = x_test[0].reshape(1,28,28,1)


# 4. Lấy output của các lớp convolution

layer_outputs = [layer.output for layer in model.layers if 'conv'
in layer.name]

activation_model = Model(inputs=model.input,
outputs=layer_outputs)


# 5. Tính toán feature maps

activations = activation_model.predict(img)
```

```
# 6. Vẽ feature maps của lớp đầu tiên
first_layer_activation = activations[0]


plt.figure(figsize=(12, 6))
for i in range(8):  # hiển thị 8 filters đầu tiên
    plt.subplot(1, 8, i+1)
    plt.imshow(first_layer_activation[0, :, :, i],
cmap='viridis')
    plt.axis('off')
plt.suptitle("Feature maps của lớp convolution đầu tiên")
plt.show()
```

Kết quả:

Ta sẽ thấy các ảnh trắng-đen biểu diễn cách mỗi filter "nhìn" cùng một bức ảnh đầu vào.

Một filter có thể phát hiện cạnh ngang, filter khác phát hiện cạnh dọc, filter khác lại phát hiện đường cong.

5.4. Ý nghĩa của việc trực quan hóa CNN

Giải thích mô hình: giúp con người hiểu CNN đang học đặc trưng gì.

Kiểm tra mô hình: nếu feature maps không mang ý nghĩa (toàn trắng hoặc toàn đen), mô hình có thể chưa học đúng.

Phát hiện overfitting: khi mạng học đặc trưng "quá cụ thể" thay vì đặc trưng tổng quát.

Cải thiện thiết kế: dựa trên trực quan hóa, ta có thể quyết định thêm lớp, thay đổi số filters, hoặc điều chỉnh preprocessing.

6. MỘT SỐ CNN PHỔ BIẾN

Trong những năm gần đây, nhiều kiến trúc CNN đã được đề xuất và đạt kết quả xuất sắc trong các cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Mỗi kiến trúc đều mang những ý tưởng mới để giải quyết các hạn chế của mô hình trước đó. Ba kiến trúc điển hình thường được nhắc tới là VGG, InceptionNet và ResNet.

6.1. VGG-16 và VGG-19

Nguồn gốc:

Được phát triển bởi nhóm Visual Geometry Group (VGG) tại Đại học Oxford.

Xuất hiện lần đầu tại ILSVRC 2014.

Ý tưởng chính:

Sử dụng các lớp convolution kernel 3×3 lặp lại nhiều lần, thay vì dùng kernel lớn.

Ý tưởng "nhiều lớp nhỏ xếp chồng" giúp mô hình học được đặc trưng phức tạp hơn mà vẫn kiểm soát được số tham số.

Kiến trúc:

VGG-16: 16 lớp có trọng số (13 lớp convolution + 3 lớp fully connected).

VGG-19: 19 lớp có trọng số (16 lớp convolution + 3 lớp fully connected).

Đặc điểm nổi bật:

Các lớp convolution nhỏ nhưng sâu → học đặc trưng phân cấp tốt.

Sử dụng ReLU làm hàm kích hoạt.

Sau một số lớp convolution thì thêm một lớp pooling để giảm kích thước.

Ưu điểm:

Kiến trúc đơn giản, dễ hiểu.

Hoạt động rất tốt trong phân loại ảnh.

Trở thành backbone phổ biến cho transfer learning.

Nhược điểm:

Rất nhiều tham số (khoảng 138 triệu).

Tốn bộ nhớ và tính toán, khó áp dụng trong môi trường hạn chế tài nguyên.

6.2. InceptionNet (GoogLeNet)

Nguồn gốc:

Được giới thiệu bởi Google, giành giải nhất tại ILSVRC 2014.

Ý tưởng chính:

Thay vì chọn một kích thước kernel duy nhất, Inception module kết hợp nhiều bộ lọc khác nhau (1×1, 3×3, 5×5) áp dụng song song trên cùng đầu vào.

Kết quả từ các filter này được ghép lại (concatenate) để tạo ra đặc trưng đa tỷ lệ (multi-scale features).

Đặc điểm nổi bật:

Dùng convolution 1×1 để giảm chiều dữ liệu, nhờ đó tiết kiệm số tham số.

Kết hợp nhiều loại filter giúp mạng học được đặc trưng ở nhiều mức độ chi tiết.

Ưu điểm:

Hiệu quả tính toán hơn VGG.

Ít tham số hơn nhưng vẫn đạt độ chính xác cao.

Học được đặc trưng ở nhiều tỷ lệ khác nhau.

Nhược điểm:

Kiến trúc phức tạp hơn, khó hiểu hơn so với VGG.

6.3. ResNet (Residual Network)

Nguồn gốc:

Được Microsoft Research giới thiệu, chiến thắng ILSVRC 2015.

Vấn đề cần giải quyết:

Khi mạng ngày càng sâu, xảy ra hiện tượng vanishing gradient → gradient biến mất, khiến mô hình không học được.

Đồng thời, độ chính xác giảm khi tăng thêm lớp, thay vì tăng lên.

Ý tưởng chính:

Residual Learning: sử dụng skip connections (kết nối tắt) để cho phép tín hiệu và gradient đi thẳng qua một số lớp.

Mỗi khối residual học phần sai biệt (residual) thay vì học trực tiếp ánh xạ đầu vào–đầu ra.

Đặc điểm nổi bật:

Cho phép huấn luyện mạng cực kỳ sâu (ResNet-50, ResNet-101, ResNet-152).

Trở thành kiến trúc chuẩn mực cho nhiều ứng dụng sau này.

Ưu điểm:

Giải quyết vấn đề vanishing gradient.

Độ chính xác cao khi mạng sâu.

Rất hiệu quả trong nhiều tác vụ thị giác máy tính (classification, detection, segmentation).

Nhược điểm:

Đòi hỏi tài nguyên tính toán lớn.

Khó triển khai nếu không dùng thư viện hỗ trợ.

6.4. So sánh nhanh

Kiến trúc	Năm	Ý tưởng chính	Ưu điểm	Nhược điểm
VGG-16/19	2014	Nhiều lớp convolution 3×3 chồng lên	Đơn giản, hiệu quả, dễ dùng transfer learning	Rất nhiều tham số (~138M)
Inception	2014	Multi-scale filters, 1×1 conv giảm chiều	Ít tham số, học đặc trưng đa tỷ lệ	Kiến trúc phức tạp
ResNet	2015	Residual learning + skip connections	Huấn luyện mạng cực sâu, độ chính xác cao	Tốn tài nguyên tính toán

Bảng 6.4: So sánh các kiến trúc

7. PHÂN LOẠI ẢNH MÈO/CHÓ VỚI VGG-16

7.1. Giới thiệu

Sau khi tìm hiểu và thực hành trên bộ dữ liệu đơn giản MNIST, chương này chuyển sang một ví dụ phức tạp và gần với thực tế hơn: phân loại ảnh mèo và chó. Đây là một trong

những bộ dữ liệu nổi tiếng nhất trên Kaggle, thường được sử dụng để kiểm tra khả năng tổng quát hóa của các mô hình deep learning.

Đặc điểm của bộ dữ liệu Cats vs Dogs:

Tổng cộng 25.000 ảnh màu kích thước khác nhau.

12.500 ảnh mèo , 12.500 ảnh chó .

Ảnh đa dạng: có ảnh cận cảnh, toàn thân, nhiều tư thế, nhiều phông nền.

Đây là bài toán phân loại nhị phân (binary classification).

So với MNIST, bài toán này khó hơn nhiều: ảnh không còn đơn giản là grayscale 28×28 nữa, mà là ảnh màu RGB có nhiều chi tiết, nhiễu, và bối cảnh phức tạp.

7.2. Lý do sử dụng VGG-16

VGG-16 là một trong những kiến trúc CNN kinh điển, nổi tiếng vì:

Được huấn luyện trên bộ dữ liệu ImageNet với hơn 1 triệu ảnh, 1000 lớp.

Khả năng trích xuất đặc trưng mạnh mẽ, đặc biệt hữu ích khi áp dụng vào transfer learning.

Input chuẩn: ảnh 224×224×3 (RGB).

Thay vì huấn luyện một CNN từ đầu (cần dữ liệu rất lớn và tốn thời gian), ta dùng VGG-16 pre-trained và chỉ huấn luyện phần đầu ra (output layers).

7.3. Quy trình thực hiện

Bước 1. Chuẩn bị dữ liệu

Chia tập dữ liệu thành train (80%) và validation/test (20%).

Resize toàn bộ ảnh về 224×224×3.

Chuẩn hóa pixel về [0,1].

Gán nhãn: 0 cho mèo, 1 cho chó.

Bước 2. Tải VGG-16 pre-trained

Dùng keras.applications.VGG16(weights="imagenet", include_top=False)

include_top=False bỏ đi các lớp fully connected cuối cùng (chỉ giữ phần convolution backbone).

Bước 3. Đóng băng các lớp convolution

Đặt base_model.trainable = False.

Điều này giúp giữ nguyên các đặc trưng đã học từ ImageNet, tránh huấn luyện lại toàn bộ mạng.

Bước 4. Thêm các lớp phân loại mới

Sau base_model, ta thêm:

Flatten() → chuyển tensor thành vector.

Dense(256, ReLU) → lớp fully connected học đặc trưng phân loại.

Dropout(0.5) → giảm overfitting.

Dense(1, Sigmoid) → output nhị phân (mèo/chó).

Bước 5. Compile mô hình

Optimizer: adam.

Loss function: binary_crossentropy.

Metric: accuracy.

Bước 6. Huấn luyện

Số epoch: 5–10 (nếu GPU tốt có thể tăng lên).

Batch size: 32.

Theo dõi loss/accuracy trên cả train và validation.

Bước 7. Đánh giá và dự đoán

Đánh giá mô hình trên tập test.

Thử dự đoán một vài ảnh mới (ảnh mèo/chó chưa có trong tập huấn luyện) để kiểm chứng trực quan.

7.4. Code minh họa

```
import tensorflow as tf
```

```
from tensorflow.keras.applications import VGG16

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import
ImageDataGenerator


# 1. Data generators

train_dir = "data/cats_vs_dogs/train"

val_dir = "data/cats_vs_dogs/validation"


train_datagen = ImageDataGenerator(rescale=1./255)

val_datagen = ImageDataGenerator(rescale=1./255)


train_generator = train_datagen.flow_from_directory(

    train_dir, target_size=(224,224), batch_size=32,
class_mode='binary')


val_generator = val_datagen.flow_from_directory(

    val_dir, target_size=(224,224), batch_size=32,
class_mode='binary')


# 2. Load pre-trained VGG16

base_model = VGG16(weights="imagenet", include_top=False,
input_shape=(224,224,3))

base_model.trainable = False  # Freeze layers
```

```
# 3. Build model
model = Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

# 4. Compile
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# 5. Train
history = model.fit(
    train_generator,
    epochs=5,
    validation_data=val_generator,
    verbose=1
)

# 6. Evaluate
loss, acc = model.evaluate(val_generator, verbose=0)
print("Validation accuracy:", acc)
```

7.5. Kết quả thực nghiệm

Training accuracy: ~96–97% sau 5–10 epoch.

Validation accuracy: ~90–95%.

VGG-16 cho kết quả rất ổn định, ít overfitting khi kết hợp dropout.

Mô hình phân biệt tốt ảnh mèo và chó, kể cả khi có nền phức tạp hoặc hình dáng đa dạng.

Ví dụ dự đoán:

Ảnh đầu vào: một chú mèo nằm trên ghế.

Output mô hình: 0.98 → mèo.

Ảnh đầu vào: một chú chó ngoài sân.

Output mô hình: 0.95 → chó.

7.6. Ý nghĩa của ví dụ Cats vs Dogs

Thể hiện sức mạnh của Transfer Learning:

Không cần huấn luyện từ đầu, chỉ cần tinh chỉnh mô hình VGG-16 đã học trên ImageNet.

Giúp tiết kiệm thời gian, tài nguyên mà vẫn đạt hiệu quả cao.

Minh họa pipeline chuẩn cho phân loại ảnh phức tạp:

Chuẩn bị dữ liệu → Transfer Learning với CNN → Huấn luyện → Đánh giá → Ứng dụng.

Ứng dụng thực tế:

Phân loại mèo/chó chỉ là ví dụ đơn giản.

Có thể mở rộng sang các bài toán phức tạp hơn như: phân loại nhiều loài động vật, phân tích ảnh y tế (u lành/ác tính), nhận diện khuôn mặt, giám sát giao thông.

Tổng kết

Từ Chương 1 đến Chương 10, hành trình học tập đi từ căn bản đến nâng cao:

- Chương 1 đặt nền móng với các khái niệm, quy trình và thao tác xử lý ảnh cơ bản.
- Chương 10 mở ra cánh cửa tiếp cận Deep Learning – công cụ mạnh mẽ giúp máy tính có thể nhìn, hiểu và diễn giải hình ảnh một cách gần gũi hơn với con người.

Những kiến thức này không chỉ có ý nghĩa học thuật mà còn mang giá trị ứng dụng thực tiễn rất cao: từ chẩn đoán y khoa chính xác hơn, kiểm tra sản phẩm công nghiệp tự động, xe tự lái thông minh, cho đến những ứng dụng giải trí trên điện thoại thông minh.

Hy vọng tài liệu này sẽ trở thành một nguồn tham khảo hữu ích, giúp sinh viên, nhà nghiên cứu và những ai quan tâm đến lĩnh vực xử lý ảnh có thể xây dựng nền tảng vững chắc, đồng thời bắt kịp xu hướng công nghệ hiện đại. Đây cũng là tiền đề để tiếp tục khám phá các kỹ thuật tiên tiến hơn như thị giác máy tính (Computer Vision), học sâu đa phương thức (Multimodal Deep Learning), hay ứng dụng AI trong các lĩnh vực liên ngành.

Tài liệu tham khảo

Dey, S. (2018). Hands-On Image Processing with Python: Expert techniques for advanced image analysis and effective interpretation of image data. Packt Publishing.

Chollet, F. (2017). Deep Learning with Python. Manning Publications.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems (NIPS).

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG). arXiv preprint arXiv:1409.1556.

Szegedy, C., et al. (2015). Going Deeper with Convolutions (InceptionNet). In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).