

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH



HCMUTE

ARTIFICIAL INTELLIGEN OBJECT DETECTION CNN

GVHD: ThS. Nguyễn Trường Thịnh

Mã môn học: ARIN337629_22_2_09

Sinh viên thực hiện: Nguyễn Đức Quyền

MSSV: 20146148

Ho Chi Minh, Tháng 04 năm 2023

MỤC LỤC

PHẦN 1: GIỚI THIỆU BÀI TOÁN.....	1
1.1 Đặt vấn đề:.....	1
1.2 Mục tiêu.....	4
PHẦN 2: KIẾN THỨC TÌM HIỂU.....	5
2.1 Mạng Nơ-ron	5
2.2 Mạng Nơ-ron tích chập	6
PHẦN 3: THIẾT KẾ VÀ THỰC HIỆN.....	11
3.1 Môi trường lập trình và công cụ hỗ trợ	11
3.2 Mô tả dữ liệu	11
3.3 Tiền xử lý dữ liệu	11
3.4 Xây dựng mô hình.....	15
3.5 Kết quả	17
3.6 Code thêm về các bài tập còn lại (bài2-5)	20
KẾT LUẬN	21
TÀI LIỆU THAM KHẢO	22
PHỤ LỤC	Error! Bookmark not defined.

PHẦN 1: GIỚI THIỆU BÀI TOÁN

1.1 Đặt vấn đề:

Do sự phát triển của khoa học công nghệ và internet, nhu cầu khai phá tri thức từ các nguồn dữ liệu khổng lồ được lưu trữ ngày càng tăng. Hiện nay, trí tuệ nhân tạo (AI) là xu thế phát triển của thế giới, và Machine learning (học máy) là một trong những lĩnh vực quan trọng của AI được sinh ra nhằm khai thác thông tin khổng lồ này.

Machine learning – học máy là chương trình chạy trên một mạng thần kinh nhân tạo, có khả năng huấn luyện máy tính "học" từ một lượng lớn dữ liệu được cung cấp để giải quyết những vấn đề cụ thể. Chẳng hạn, nếu muốn dạy robot cách đi từ nơi này đến nơi kia để nó chỉ đường cho người khác, theo cách truyền thống bạn sẽ đưa cho nó một loạt quy tắc hướng dẫn cách nhìn trái phải hay tránh người đi qua, chọn đường đi thoáng nhất và ngắn nhất...

Khi dùng Machine learning, bạn sẽ cho robot xem hàng chục nghìn video quay cảnh người đi lại và hàng chục nghìn video quay cảnh ai đó và chạm để nó tự học theo. Phần khó nhất là làm sao cho máy tính hiểu được những video này ngay từ đầu.

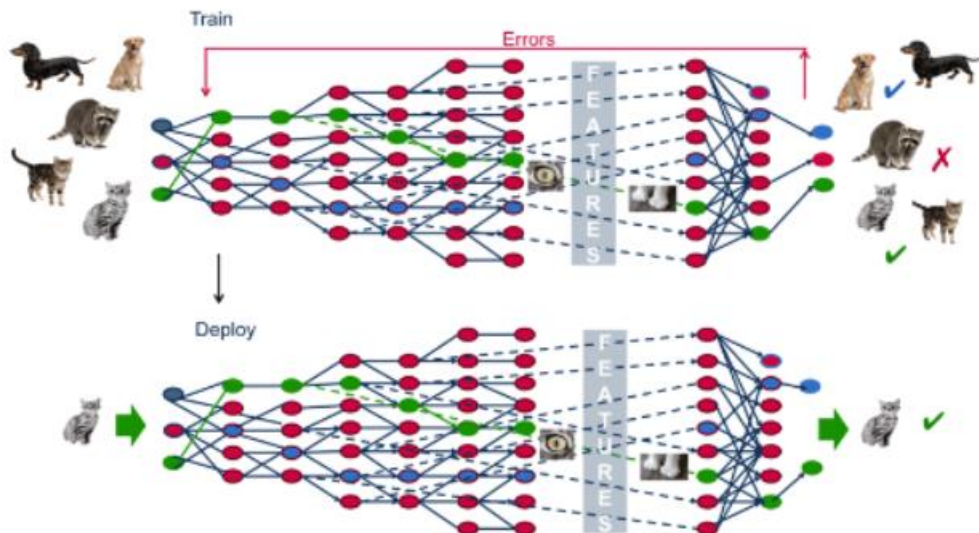
Để học máy hiệu quả, chính xác hơn, Deep Learning (học sâu) đã ra đời. Deep learning sẽ giúp robot học nhanh, kỹ và áp dụng chính xác hơn. Lấy ý tưởng từ bộ não sinh học, các mô hình học sâu xây dựng các thuật toán giúp máy suy nghĩ và xử lý thông tin giống như bộ não con người. Nhờ có nhiều lớp thần kinh nhân tạo hơn Machine learning, Deep learning sẽ có khả năng tự học và nhận diện nhiều vấn đề có cấp độ phức tạp hơn nhiều. Deep learning cho phép máy tính tự động hiểu, xử lý và học từ dữ liệu để thực thi nhiệm vụ được giao, cũng như cách đánh giá giúp tăng tính hiệu quả.

Ví dụ, robot dẫn đường ở sân bay Incheon (Hàn Quốc) có thể thay đổi tốc độ trên đường đi, biết dừng lại tránh người đi tới, biết chọn phần đường ít người đi qua và biết thay mặt cười khi người đi cùng muốn chụp ảnh selfie.



Hình 1.1 Các Robot của LG vừa được đưa vào sử dụng thí điểm tại sân bay quốc tế Incheon

Một ví dụ khác là khi bạn dạy máy tính nhận diện hình ảnh một mỹ nhân – khái niệm có tính trừu tượng. Chúng ta sẽ lập trình ra nhiều lớp trong mạng thần kinh nhân tạo, mỗi lớp có khả năng xác định một đặc điểm cụ thể của mỹ nhân như mắt, miệng, mũi, số đo ba vòng... rồi cho máy xem hàng chục nghìn bức ảnh mỹ nhân (chỉ ra rằng “Đây là mỹ nhân”) cùng hàng trăm nghìn bức ảnh của các đối tượng khác nhau (chỉ ra rằng “Đây không phải mỹ nhân”). Khi mạng thần kinh nhân tạo này xem hết các bức ảnh, các lớp của nó sẽ dần nhận ra đặc điểm và biết lớp nào là quan trọng, lớp nào không để tự định nghĩa về khái niệm mỹ nhân. Nó cũng sẽ nhận ra rằng mỹ nhân là một con người và là phụ nữ, nên khi



cần xác định mỹ nhân, nó sẽ lọc tìm hình ảnh phụ nữ, kèm những đặc điểm khác như tỷ lệ khuôn mặt, các số đo cơ thể chuẩn...

Hình 1.2 Minh họa quá trình học và nhận diện hình ảnh trong Deep learning

Cụ thể, chúng dựa trên các dữ liệu người dùng phát sinh để gợi ý thêm những sản phẩm họ sẽ quan tâm, thích (trên các nền tảng mua sắm), gợi ý các bài quảng cáo/được tài trợ (trên Google, Facebook, Naver...) hay các khóa học người học quan tâm (trên các nền tảng học online như Funix, Coursera..) và rất nhiều lĩnh vực khác nữa.



Hình 1.3 Facebook tự động nhận diện khuôn mặt trên các bức ảnh

Deep learning cũng được ứng dụng nhiều trong lĩnh vực y tế, đặc biệt là lĩnh vực y sinh. Trí tuệ nhân tạo Watson của IBM có thể xác định chứng leukemia hiếm gặp ở loài người trong khoảng mười phút trong khi các bác sĩ không thể tìm ra. Để làm được điều này, Watson đã so sánh bộ gene của bệnh nhân với hơn 20 triệu kết quả nghiên cứu bệnh khác. Công nghệ này cũng được ứng dụng vào khoa học tội phạm và điều tra, hay đưa vào ứng dụng camera công cộng tại nhiều quốc gia để quản lý an ninh công cộng. Chẳng hạn, ở Hàn Quốc hay Trung Quốc, hệ thống camera an ninh tại đây sẽ nhận diện chính xác khuôn mặt của tất cả mọi người dù ở góc độ nào. Bạn có thể tự tin vào an ninh công cộng vì hệ thống an ninh biết chính xác từng khuôn mặt ngoài đường.

Hiện nay, trước tiềm năng của công nghệ này, các tập đoàn công nghệ lớn trên thế giới như Google, Apple, Tencent, Hanwa... luôn ưu tiên các startup về robot và deep learning. Những nghiên cứu quy mô từ các tập đoàn lớn này đang tạo ra những công nghệ ngày càng đột phá, như trợ lý ảo như Siri, Google Assistant, robot quản gia, các hệ thống xe tự lái.. cho đến phát triển các nguyên liệu mới giúp robot thấu hiểu thế giới xung quanh, các nghiên cứu rút ngân quỹ đạo cho tàu không gian... Deep learning đang ngày càng cho thấy một tương lai đầy hứa hẹn. Khả năng phân tích dữ liệu lớn và sử dụng deep learning vào các hệ thống máy tính khiến máy tính có thể tự thích nghi với những gì chúng tiếp nhận mà không cần đến bàn tay lập trình của con người. Sự tiến bộ đó sẽ nhanh chóng mở đường cho nhiều đột phá trong tương lai về trí tuệ nhân tạo.

1.2 Mục tiêu

Việc học Deep Learning trong sở hữu hình ảnh được áp dụng rộng rãi trong nhiều lĩnh vực, bao gồm y tế, an ninh, giao thông, marketing và nhiều lĩnh vực khác. Ví dụ, trong lĩnh vực y tế, Deep Learning có thể được sử dụng để phát hiện các khối u, nhận dạng các bệnh trên hình ảnh siêu âm và CT, và giúp bác sĩ chẩn đoán và điều trị bệnh tật một cách chính xác hơn.

Ngoài ra, việc học Deep Learning trong sở hữu hình ảnh cũng giúp máy tính hiểu được hình ảnh và đưa ra các phản hồi thông minh. Ví dụ, các hệ thống nhận diện hình ảnh có thể được sử dụng để giám sát giao thông, phát hiện tai nạn giao thông, nhận dạng biển số xe và quản lý luồng giao thông. Để hiểu hơn về deep learning trong xử lý ảnh ta có các bài toán như sau:

Bài 1. Nhận diện gương mặt

Bài 2. Tiền giấy VN (5000đ, 10000đ, 20k, 50k, 100k, 500k)

Bài 3. 10 món ăn Việt Nam (bún bò, chè, bánh xèo.....)

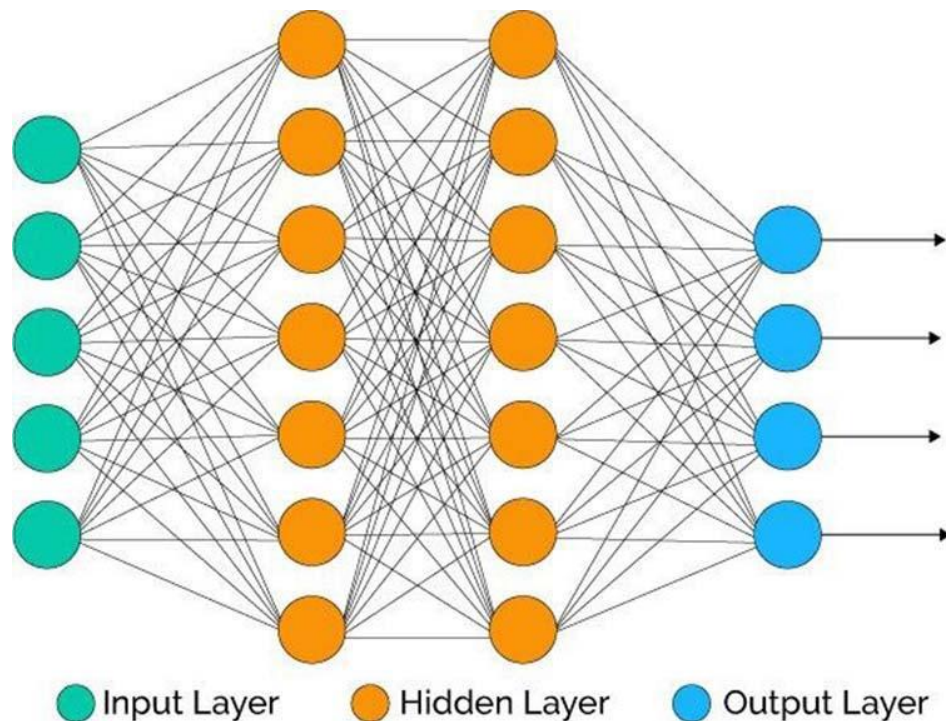
Bài 4. 5 loại hoa (hồng, sen, súng, mai, cúc, hồng)

Bài 5. Nhận dạng tất cả các thành viên trong lớp từ ảnh khuôn mặt (bạn sưu tầm)

PHẦN 2: KIẾN THỨC TÌM HIỂU

2.1 Mạng Nơ-ron

Định nghĩa mạng Nơ-ron Mạng nơ-ron, còn được gọi là mạng nơ-ron nhân tạo (ANN) hoặc mạng nơ-ron mô phỏng (SNN), là một tập hợp con của học máy và là trung tâm của các thuật toán học sâu. Tên và cấu trúc của chúng được lấy cảm hứng từ bộ não con người, bắt chước cách các tế bào thần kinh sinh học truyền tín hiệu cho nhau. Mạng thần kinh nhân tạo (ANN) bao gồm một lớp nút, chứa lớp đầu vào, một hoặc nhiều lớp ẩn và lớp đầu ra. Mỗi nút, hoặc nơ-ron nhân tạo, kết nối với nút khác và có trọng số và ngưỡng liên quan. Nếu đầu ra của bất kỳ nút riêng lẻ nào cao hơn giá trị ngưỡng đã chỉ định, nút đó sẽ được kích hoạt, gửi dữ liệu đến lớp tiếp theo của mạng. Mặt khác, không có dữ liệu nào được chuyển sang lớp tiếp theo của mạng.



Mạng nơ-ron dựa vào dữ liệu đào tạo để học và cải thiện độ chính xác của chúng theo thời gian. Tuy nhiên, một khi các thuật toán học tập này được tinh chỉnh để đạt được độ chính xác, chúng sẽ là những công cụ mạnh mẽ trong khoa học máy tính và trí tuệ nhân tạo, cho phép chúng ta phân loại và phân cụm dữ liệu với tốc độ cao. Các tác vụ trong nhận dạng giọng nói hoặc nhận dạng hình ảnh có thể mất vài phút so với hàng giờ so với nhận

dạng thủ công của các chuyên gia con người. Một trong những mạng thần kinh nổi tiếng nhất là thuật toán tìm kiếm của Google.

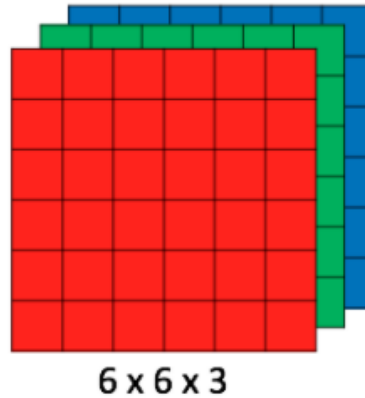
Mạng nơ-ron nhân tạo có thể được phân loại theo phương thức dữ liệu được truyền từ nút đầu vào đến nút đầu ra. Dưới đây là một số ví dụ:

- **Mạng nơ-ron truyền thẳng:** Mạng nơ-ron truyền thẳng xử lý dữ liệu theo một chiều, từ nút đầu vào đến nút đầu ra. Mỗi nút trong một lớp được kết nối với tất cả các nút trong lớp tiếp theo. Mạng truyền thẳng sử dụng một quy trình phản hồi để cải thiện dự đoán theo thời gian.
- **Thuật toán truyền ngược:** Mạng nơ-ron nhân tạo liên tục học hỏi bằng cách sử dụng vòng lặp phản hồi hiệu chỉnh để cải thiện phân tích dự đoán của chúng. Đơn giản mà nói, bạn có thể coi rằng dữ liệu truyền từ nút đầu vào đến nút đầu ra qua nhiều lối đi khác nhau trong mạng nơ-ron. Chỉ có duy nhất một lối đi chính xác, ánh xạ nút đầu vào đến nút đầu ra thích hợp.
- **Mạng nơ-ron tích chập:** Những lớp ẩn trong mạng nơ-ron tích chập thực hiện các chức năng toán học cụ thể, như tóm tắt hoặc sàng lọc, được gọi là tích chập. Chúng rất hữu ích trong việc phân loại hình ảnh vì chúng có thể trích xuất các đặc điểm liên quan từ hình ảnh, điều này có lợi cho việc nhận dạng và phân loại hình ảnh. Biểu mẫu mới dễ xử lý hơn mà không làm mất đi các đặc điểm quan trọng để đưa ra dự đoán chính xác. Mỗi lớp ẩn trích xuất và xử lý các đặc điểm hình ảnh khác nhau, như các cạnh, màu sắc và độ sâu.

2.2 Mạng Nơ-ron tích chập

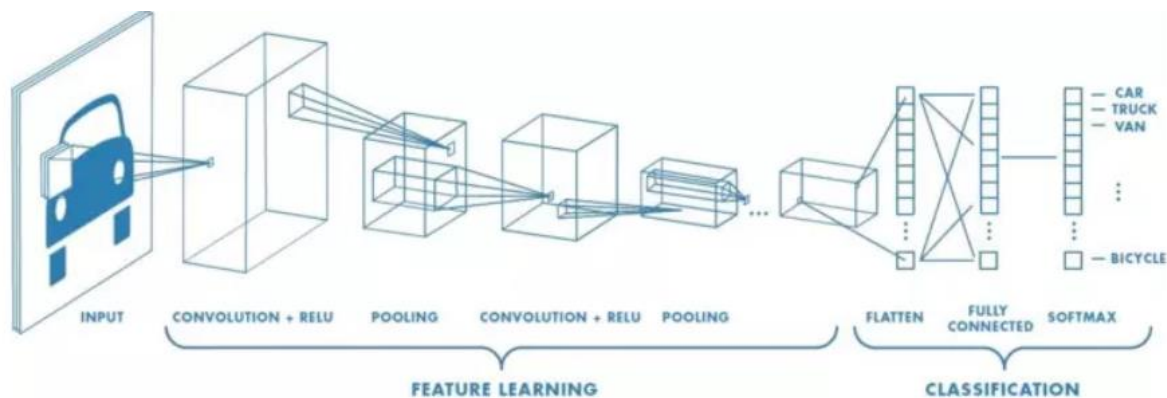
2.2.1 Giới thiệu

Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.



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 $H \times W \times D$ (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB $6 \times 6 \times 3$ (3 ở đây là giá trị RGB).

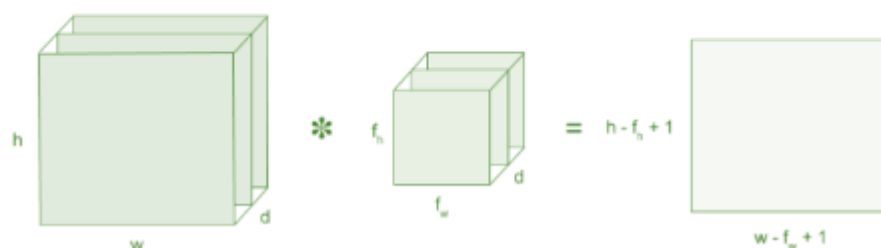
Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernels), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ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



2.2.2 Lớp tích chập (Convolution Layer)

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**



Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 – Image Matrix

*

1	0	1
0	1	0
1	0	1

3 x 3 – Filter Matrix

Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.

1	1	1	0	0
0	1	1	1	0
0	0	1 _{x1}	1 _{x0}	1 _{x1}
0	0	1 _{x0}	1 _{x1}	0 _{x0}
0	1	1 _{x1}	0 _{x0}	0 _{x1}

Image

4	3	4
2	4	3
2	3	4



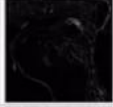
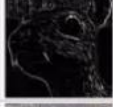



Convolved
Feature

Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.

2.2.3 Đường viền - Padding

Đôi khi kernel không phù hợp với hình ảnh đầu vào. Ta có 2 lựa chọn:

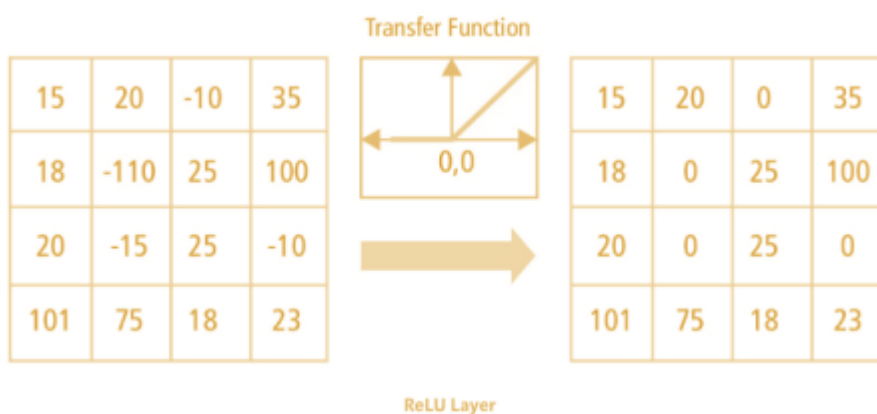
- Chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding).
- Cắt bớt hình ảnh tại những điểm không phù hợp với kernel.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

2.2.4 Hàm phi tuyến - ReLU

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: $f(x) = \max(0, x)$.

Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.



Có 1 số hàm phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

2.2.5 Hàm kích hoạt lớp cuối cùng

Hàm kích hoạt được áp dụng cho lớp kết nối đầy đủ cuối cùng thường khác với các lớp khác. Một hàm kích hoạt thích hợp cần được lựa chọn theo từng task. Các lựa chọn điển hình của lớp cuối cùng chức năng kích hoạt cho các loại tác vụ khác nhau được tóm tắt trong bảng sau

PHẦN 3: THIẾT KẾ VÀ THỰC HIỆN

3.1 Môi trường lập trình và công cụ hỗ trợ

- Môi trường lập trình: Google Codelabs
- Ngôn ngữ sử dụng: Python
- Thư viện hỗ trợ: os,sklearn, keras, numpy, ...

3.2 Mô tả dữ liệu

Cấu trúc tập dữ liệu:

Bai1: với 393 dữ liệu của 5 gương mặt được đưa vào

Bai2: với 3350 hình ảnh chụp các loại tiền từ 1.000vnđ tới 500.000vnđ hình ảnh màu trong số đó mỗi loại tiền từ 200-250 hình

Bài 3: 5500 bức ảnh của 10 món ăn được đưa vào training

Bài 4: 5 loài hoa, mỗi loài từ 60-80 bức ảnh

Bài 5: 578 bức ảnh của 10 bạn trong lớp

3.3 Tiền xử lý dữ liệu

Các bước tiền xử lý dữ liệu bao gồm:

- Kết nối gg colab với drive:

```
✓ [2] from google.colab import drive  
18s drive.mount('/content/drive')
```

Mounted at /content/drive

- Thêm tất cả các thư viện sử dụng trong bài:

```

✓ [21] import numpy as np
0s import matplotlib.pyplot as plt
from os import listdir
from numpy import asarray, save
from sklearn.model_selection import train_test_split
from time import time
from keras.utils import load_img, img_to_array
from keras.utils import to_categorical
from keras.models import Sequential, Model
from keras.layers import *
from keras.layers import LeakyReLU
from keras.optimizers import Adam
from keras.engine.training import optimizers
from keras.backend import categorical_crossentropy

```

Tạo đường dẫn tới hình ảnh, quy định size ảnh 100 pixel và gán giá trị cho các tên ảnh đặt tên giống nhau thành 1 giá trị số và in ra màn hình để kiểm tra các giá trị vừa gán. Tiếp theo có thể lưu các giá trị đó để lần sau chỉ cần gọi ra để sử dụng.

```

✓ 3s folder = '/content/drive/MyDrive/guong_mat/'

img_size = 100
photos, labels = list(), list()
for file in listdir(folder) :
    output = 0.0
    if file.startswith('ibrahimovic '):
        output = 1.0
    if file.startswith('ronandinho'):
        output = 2.0
    if file.startswith('pele'):
        output = 3.0
    if file.startswith('messi'):
        output = 4.0
    if file.startswith('ronaldo'):
        output = 5.0
    photo = load_img(folder + file, target_size = (img_size, img_size))
    photo = img_to_array(photo)
    photos.append(photo)
    labels.append(output)
photos = asarray(photos)
labels = np.array(labels)
print(photos.shape, labels.shape)
# save('/content/drive/MyDrive/anh/hoa/hoa_photos.npy', photos)
# save('/content/drive/MyDrive/anh/hoa/hoa_labels.npy', labels)

(393, 100, 100, 3) (393,)

```

Đổi định dạng sang kiểu float32.

```

✓ [23] photos = photos.astype('float32')/255.0
0s print(photos.shape)

(393, 100, 100, 3)

```

Chia ảnh ra thành 3 thư mục để train, test và validation với test=20% train lúc đầu và validation=20% train lúc sau.


```

✓ 0s ▶ [(trainX, testX, trainY, testY)] = train_test_split(photos, labels, test_size=0.20, stratify=labels, random_state=10)

(trainX, valX, trainY, valY) = train_test_split(trainX, trainY, test_size=0.20, random_state=10)

print(trainX.shape)
print(trainY.shape)
print(testX.shape)
print(testY.shape)
print(valX.shape)
print(valY.shape)

```

```

(251, 100, 100, 3)
(251,)
(79, 100, 100, 3)
(79,)
(63, 100, 100, 3)
(63,)

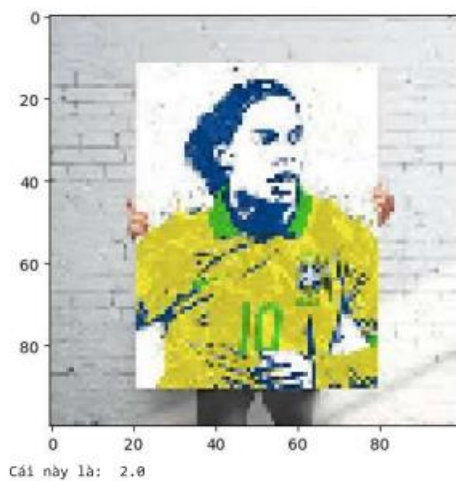
```

Test thử 1 bức ảnh.

```

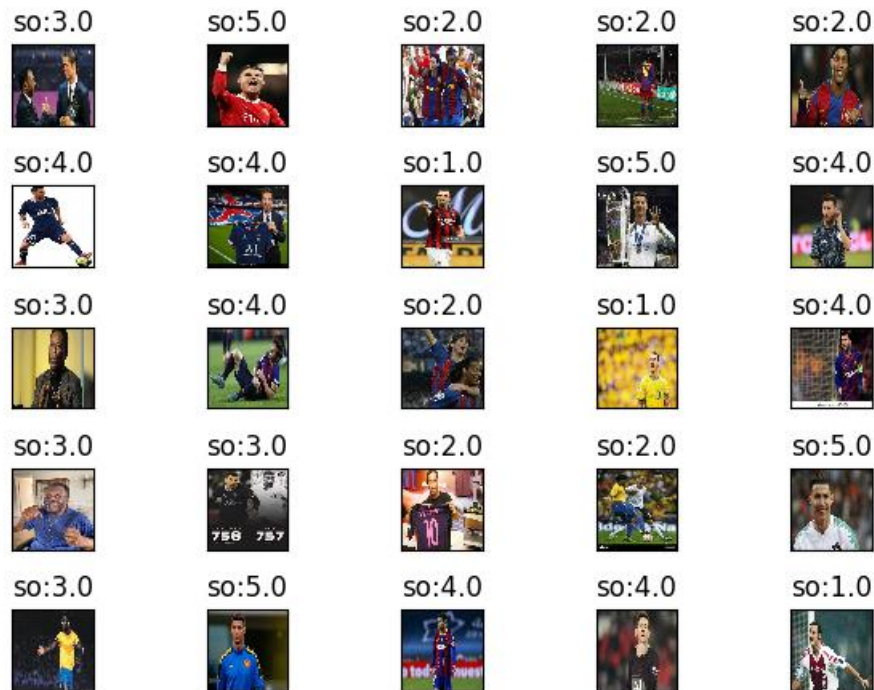
image_test=50
hình = trainX[image_test]
plt.imshow(hình)
plt.show()
print('Cái này là: ', trainY[image_test])

```



Test thử 25 bức ảnh để kiểm lại lỗi.

```
[ ]
for i in range(25): # 30 hình ảnh
    plt.subplot(5,5,i+1) # 6 cột , 5 hàng
    plt.tight_layout()
    plt.imshow(trainX[i], interpolation='none')
    plt.title('so:{}'.format(trainY[i]))
    plt.xticks([])
    plt.yticks([])
```



Chuyển ảnh về one hot encoding

```
[ ] #one hot encoding giatri mau tung diem anh tu thap phan duoi dang 10 bit
trainY = to_categorical (trainY)
testY=to_categorical(testY)
valY=to_categorical(valY)

print(trainY.shape)
print(testY.shape)
print(valY.shape)
```

```
(251, 6)
(79, 6)
(63, 6)
```

3.4 Xây dựng mô hình

Quy định các giá trị cho bước tạo mạng nơ ron và train. Tạo 5 lớp CNN và 1 lớp ANN để chuẩn bị cho việc train dữ liệu.

```
batch_size=128
epochs=50
classes =6

#Tạo 1 mạng có 28x28 input và 10 output
model = Sequential()

# 1 bộ VGG
model.add(Conv2D(32, kernel_size=(3,3), activation = 'linear', input_shape=(img_size,img_size,3),padding = 'same'))
model.add(LeakyReLU(alpha=0.1)) # Hàm xử lý dữ liệu (Ngưỡng)
model.add(MaxPooling2D((2,2), padding='same'))

# Bộ 2 VGG
model.add(Conv2D(64,(3,3) , activation = 'linear' , padding = 'same'))
model.add(LeakyReLU(alpha = 0.1)) # Hàm xử lý dữ liệu (Ngưỡng)
model.add(MaxPooling2D((2,2), padding = 'same'))

# Bộ 3 VGG
model.add(Conv2D(128,(3,3) , activation = 'linear' , padding = 'same'))
model.add(LeakyReLU(alpha = 0.1)) # Hàm xử lý dữ liệu (Ngưỡng)
model.add(MaxPooling2D((2,2), padding = 'same'))

# Bộ 4 VGG
model.add(Conv2D(256,(3,3) , activation = 'linear' , padding = 'same'))
model.add(LeakyReLU(alpha = 0.1)) # Hàm xử lý dữ liệu (Ngưỡng)
model.add(MaxPooling2D((2,2), padding = 'same'))

# Bộ 5 VGG
model.add(Conv2D(512,(3,3) , activation = 'linear' , padding = 'same'))
model.add(LeakyReLU(alpha = 0.1)) # Hàm xử lý dữ liệu (Ngưỡng)
model.add(MaxPooling2D((2,2), padding = 'same'))

# ANN Model
model.add(Flatten()) # Lấy kích thước trước rồi làm phẳng
model.add(Dense(512, activation = 'relu'))
# model.add(Dense(512, activation = 'relu'))

model.add(LeakyReLU(alpha = 0.1))
model.add(Dense(classes, activation = 'softmax'))

model.compile(loss = categorical_crossentropy, optimizer = Adam() , metrics=['accuracy'])
model.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
=====		
conv2d_18 (Conv2D)	(None, 100, 100, 32)	896
leaky_re_lu_22 (LeakyReLU)	(None, 100, 100, 32)	0
max_pooling2d_18 (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_19 (Conv2D)	(None, 50, 50, 64)	18496
leaky_re_lu_23 (LeakyReLU)	(None, 50, 50, 64)	0
max_pooling2d_19 (MaxPooling2D)	(None, 25, 25, 64)	0
conv2d_20 (Conv2D)	(None, 25, 25, 128)	73856
leaky_re_lu_24 (LeakyReLU)	(None, 25, 25, 128)	0
max_pooling2d_20 (MaxPooling2D)	(None, 13, 13, 128)	0
conv2d_21 (Conv2D)	(None, 13, 13, 256)	295168
leaky_re_lu_25 (LeakyReLU)	(None, 13, 13, 256)	0
max_pooling2d_21 (MaxPooling2D)	(None, 7, 7, 256)	0
conv2d_22 (Conv2D)	(None, 7, 7, 512)	1180160
leaky_re_lu_26 (LeakyReLU)	(None, 7, 7, 512)	0
max_pooling2d_22 (MaxPooling2D)	(None, 4, 4, 512)	0
flatten_4 (Flatten)	(None, 8192)	0
dense_8 (Dense)	(None, 512)	4194816
leaky_re_lu_27 (LeakyReLU)	(None, 512)	0
dense_9 (Dense)	(None, 6)	3078
=====		
Total params: 5,766,470		
Trainable params: 5,766,470		
Non-trainable params: 0		

Train dữ liệu.

```
start = time()
train = model.fit(trainX, trainY, batch_size=batch_size, epochs=epochs, verbose = 1, validation_data=(valX, valY))
print(time()-start)
```

```

2/2 [=====] - 0s 104ms/step - loss: 0.1924 - accuracy: 0.9243 - val_loss: 1.4884 - val_accuracy: 0.6667
Epoch 41/50
2/2 [=====] - 0s 98ms/step - loss: 0.1304 - accuracy: 0.9681 - val_loss: 1.6104 - val_accuracy: 0.6032
Epoch 42/50
2/2 [=====] - 0s 100ms/step - loss: 0.1066 - accuracy: 0.9761 - val_loss: 1.4211 - val_accuracy: 0.6825
Epoch 43/50
2/2 [=====] - 0s 104ms/step - loss: 0.1106 - accuracy: 0.9562 - val_loss: 1.7440 - val_accuracy: 0.6508
Epoch 44/50
2/2 [=====] - 0s 98ms/step - loss: 0.0670 - accuracy: 0.9801 - val_loss: 1.8871 - val_accuracy: 0.5556
Epoch 45/50
2/2 [=====] - 0s 103ms/step - loss: 0.0673 - accuracy: 0.9761 - val_loss: 2.0217 - val_accuracy: 0.6190
Epoch 46/50
2/2 [=====] - 0s 98ms/step - loss: 0.0277 - accuracy: 1.0000 - val_loss: 2.1901 - val_accuracy: 0.6825
Epoch 47/50
2/2 [=====] - 0s 98ms/step - loss: 0.0267 - accuracy: 1.0000 - val_loss: 1.8752 - val_accuracy: 0.6825
Epoch 48/50
2/2 [=====] - 0s 100ms/step - loss: 0.0187 - accuracy: 1.0000 - val_loss: 1.9885 - val_accuracy: 0.6667
Epoch 49/50
2/2 [=====] - 0s 118ms/step - loss: 0.0146 - accuracy: 1.0000 - val_loss: 2.2500 - val_accuracy: 0.6190
Epoch 50/50
2/2 [=====] - 0s 118ms/step - loss: 0.0122 - accuracy: 1.0000 - val_loss: 2.4989 - val_accuracy: 0.6508
9.86219334602356

```

3.5 Kết quả

Vẽ đồ thị accuracy, loss cho train và validation.

```

acc=train.history['accuracy']
val_acc=train.history['val_accuracy']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')

plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1.1])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')

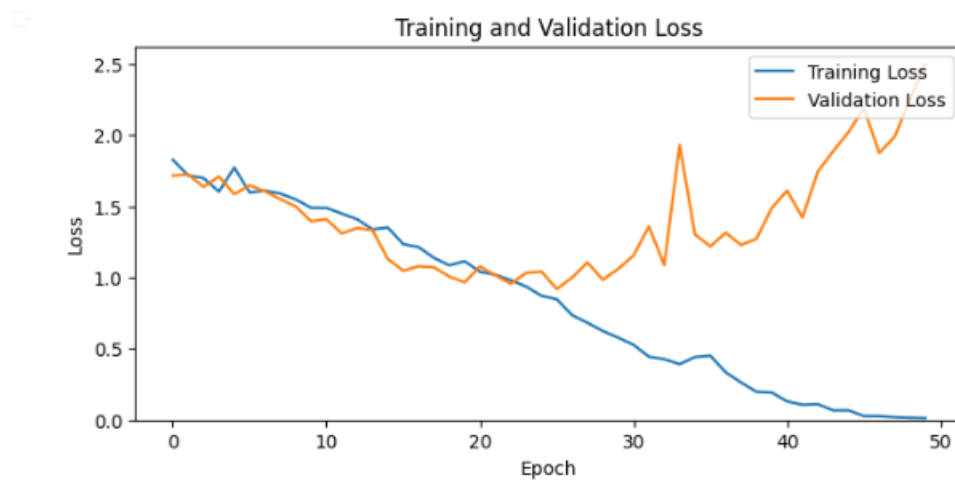
```

Text(0.5, 0, 'Epoch')



```
[ ]
acc=train.history['accuracy']
val_acc=train.history['val_accuracy']
loss=train.history['loss']
val_loss=train.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Loss')
plt.ylim(0)
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.show()
```



Test lại độ chính xác của chương trình vừa train.

```
# Số lượng ảnh cần lấy
num_images = 5

# Tạo ra một mảng các chỉ số ngẫu nhiên
indices = np.random.choice(range(len(testX)), num_images, replace=False)

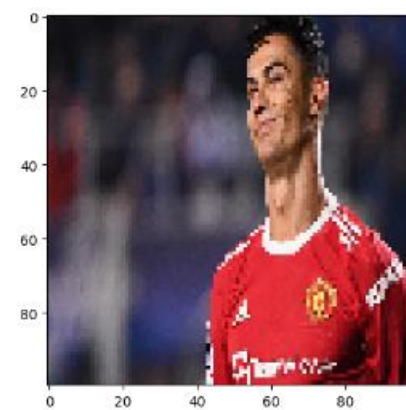
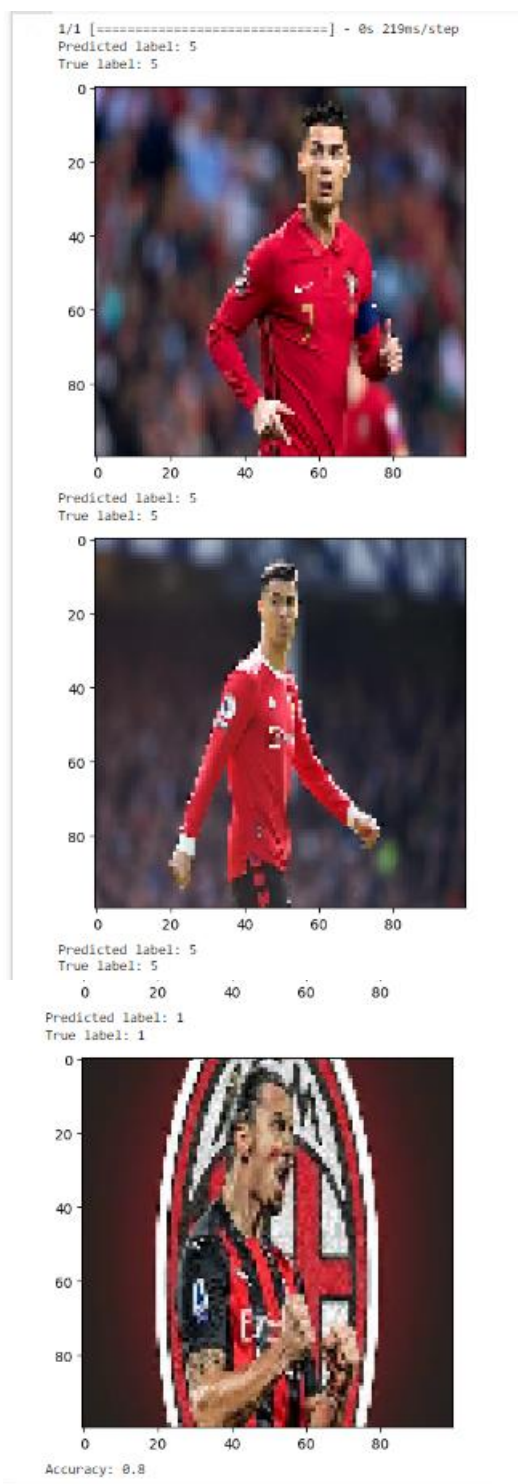
# Lấy các ảnh tương ứng từ tập testX
images = testX[indices]

# Lấy các nhãn tương ứng từ tập testY
true_labels = np.argmax(testY[indices], axis=1)

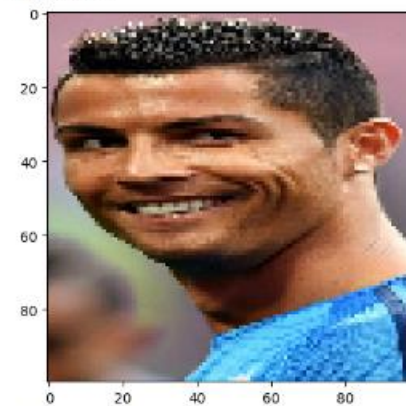
# Dự đoán các nhãn cho các ảnh lấy ra
predicted_labels = np.argmax(model.predict(images), axis=1)

# In ra các nhãn dự đoán và nhãn thực tế tương ứng
for i in range(num_images):
    print("Predicted label:", predicted_labels[i])
    print("True label:", true_labels[i])
    plt.imshow(images[i])
    plt.show()

# Tính toán độ chính xác
accuracy = np.sum(predicted_labels == true_labels) / num_images
print("Accuracy:", accuracy)
```



Predicted label: 3
True label: 5



Predicted label: 1
True label: 1

3.6 Code thêm về các bài tập còn lại (bài2-5)

KẾT LUẬN

Thông qua bài toán trên ta có thể hiểu thêm và cấu trúc mạng CNN. Đề tài đồ án cung cấp cho sinh viên những kiến thức căn bản về Trí tuệ nhân tạo, cung cấp những kiến thức căn bản cũng như những kỹ năng cần thiết cho quá trình tiếp theo khi tiếp tục nghiên cứu Trí tuệ nhân tạo. Bản thân em cũng đã có thể từ hướng dẫn của thầy tạo ra một sản phẩm có tính thực tiễn và độ chính xác khá tốt.

Trong quá trình nghiên cứu, thực hiện đồ án, dù thiếu các kiến thức cơ bản ban đầu. Tuy nhiên, nhờ hướng dẫn của thầy cũng như những anh chị, bạn bè em cũng đã có thể hoàn thành một bài report tuy không quá xuất sắc nhưng quá trình làm việc luôn nghiêm túc, chính chu nhất có thể.

Trong quá trình viết báo cáo, khó tránh khỏi sai sót, rất mong các thầy hướng dẫn góp ý. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp thầy hướng dẫn để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài báo cáo sắp tới.

TÀI LIỆU THAM KHẢO

- Mạng nơ-ron là gì?* (không ngày tháng). Được truy lục từ Amazon Web Services:
<https://aws.amazon.com/vi/what-is/neural-network/>
- Tuấn, T. Q. (2019, tháng 03 01). *Deep learning - công nghệ phía sau trí tuệ nhân tạo*. Được truy lục từ Funx: <https://funix.edu.vn/tin-tuc-funix/deep-learning-cong-nghe-phia-sau-tri-tue-nhan-tao/>