

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO THÍ NGHIỆM/THỰC NGHIỆM**  
**HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**  
**ĐỀ TÀI:**  
**ỨNG DỤNG TENSORFLOW VÀ MẠNG VGG16 ĐỂ**  
**PHÂN LOẠI ẢNH**

Sinh viên thực hiện : 1. Hoàng Ngọc Sơn Hà  
2. Tăng Mạnh Đạt  
3. Phạm Lê Minh Đức  
4. Đỗ Mạnh Dũng

Nhóm: 03

Lớp: 20241IT6094003

Khóa: K17

Giảng viên hướng dẫn : Trần Thanh Huân

**Hà Nội, Tháng 12/2024**

BỘ CÔNG THƯƠNG  
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ  
NỘI

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT  
NAM  
Độc lập – tự do – hạnh phúc  
-----o0o-----

## **PHIẾU GIAO ĐỀ TÀI**

### **BÁO CÁO THỰC NGHIỆM/THÍ NGHIỆM**

Nhóm thực hiện: 1) Hoàng Ngọc Sơn Hà  
2) Tăng Mạnh Đạt  
3) Phạm Lê Minh Đức  
4) Đỗ Mạnh Dũng

Lớp học phần: 20241IT6094003  
thông tin

Khóa: K17

Khoa Công nghệ

Ngành học: Trí Tuệ Nhân Tạo

Tên đề tài: Phân loại ảnh trong thư viện

Mục đích: Mạng VGG16, một mạng nơron được huấn luyện trên tập dữ liệu lớn, có khả năng nhận diện đặc trưng trong ảnh, là cơ sở để nhóm phát triển ứng dụng phân loại ảnh bằng Tensorflow. Ứng dụng này nhằm hỗ trợ người dùng dễ dàng quản lý và tìm kiếm ảnh trên điện thoại.

Yêu cầu:- Hoàn thành các nhiệm vụ của đề tài

-Sử dụng kỹ năng làm việc nhóm trong quá trình thực hiện báo cáo

-Trình bày báo cáo đúng yêu cầu, sử dụng kỹ năng “Viết báo cáo” khi thực hiện đề tài

-Sử dụng kỹ năng viết tài liệu kỹ thuật và phi kỹ thuật trong phần mở đầu của báo cáo

Kết quả thu được: Bản báo cáo đề tài (bản cứng và bản mềm); sản phẩm đề tài

Ngày giao đề tài: 20/11/2024

Ngày hoàn thành: 20/12/2024

Giảng viên hướng dẫn: Trần Thanh Huân

Hà nội, Ngày 06 Tháng 11 Năm 2024

GIẢNG VIÊN

Trần Thanh Huân

## PHIẾU PHÂN CÔNG NHIỆM VỤ

Nhóm 03, gồm 4 thành viên

- 1) Hoàng Ngọc Sơn Hà (nhóm trưởng)
- 2) Tăng Mạnh Đạt
- 3) Phạm Lê Minh Đức
- 4) Đỗ Mạnh Dũng

TT	Người thực hiện	Công việc	Kết quả đạt được	Nhận xét của GV
<b>Tuần 1</b>				
1	Hoàng Ngọc Sơn Hà	1) Định nghĩa cơ bản về Neural Networks 2) Mô hình của một mạng Neuron 3) Mô hình của một Neuron 4) Một số loại hàm kích hoạt	Đã tìm được bản mềm tài liệu, xây dựng khung báo cáo	
2	Tăng Mạnh Đạt	1) Các loại Neural Networks 2) Mạng Perceptron 3) Ứng dụng của Neural Networks	Đã tìm được bản mềm tài liệu, chưa sắp xếp tài liệu theo yc của nhóm trưởng	
3	Phạm Lê Minh Đức	1) Các lớp cơ bản của một Convolutional Neural Network 2) Cách thức hoạt động của Convolutional Neural Network	Đã tìm được bản mềm tài liệu, chưa sắp xếp tài liệu theo yc của nhóm trưởng	
4	Đỗ Mạnh Dũng	1) Tìm hiểu về Tensorflow và Keras	Đã tìm được bản mềm tài liệu, chưa sắp xếp tài liệu theo yc của nhóm trưởng	
<b>Tuần 2</b>				
1	Hoàng Ngọc Sơn Hà	Tìm hiểu về bài toán phân loại ảnh	Tìm được bản mềm về định nghĩa và cách hoạt động của bài toán	

2	Tăng Mạnh Đạt	Tìm hiểu về mô hình phân loại ảnh ResNet	Tìm được bản mềm về khái niệm, ưu điểm và kiến trúc	
3	Phạm Lê Minh Đức	Tìm hiểu về mô hình phân loại ảnh Inception	Tìm được bản mềm về định nghĩa và cách huấn luyện	
4	Đỗ Mạnh Dũng	Tìm hiểu về mô hình phân loại ảnh VGG16	Tìm được bản mềm khái niệm và phân tích	
<b>Tuần 3</b>				
1	Hoàng Ngọc Sơn Hà	Phân tích cách áp dụng Tensorflow, mạng VGG16 và xây dựng chương trình	Phân tích được cách áp dụng Tensorflow, mạng VGG16 và xây dựng được khung chương trình	
2	Tăng Mạnh Đạt	Phân tích bài toán và xây dựng chương trình	Phân tích được cách áp dụng bài toán và xây dựng chương trình	
3	Phạm Lê Minh Đức	Thu thập và xử lý dữ liệu	Thu thập và xử lý hình ảnh theo yêu cầu để huấn luyện	
4	Đỗ Mạnh Dũng	Xây dựng chương trình và kiểm tra, đánh giá hiệu quả của mô hình	Xây dựng chương trình và kiểm tra chất lượng sau khi đạt được	

## LỜI NÓI ĐẦU

Để có kết quả như ngày hôm nay, nhóm em đã nhận được rất nhiều sự chỉ bảo, hướng dẫn của thầy giáo **Trần Thanh Huân** - giảng viên khoa Công nghệ thông tin trường Đại học Công nghiệp Hà Nội. Cũng như sự góp ý, giúp đỡ của các bạn trong lớp học phần Trí tuệ nhân tạo. Nhóm em bày tỏ lòng biết ơn với thầy cũng như với các bạn đã nhiệt tình đóng góp nhiều kiến thức quý báu. Những kiến thức này không chỉ giúp chúng em trong quá trình thực hiện bài tập lớn mà còn là hành trang tiếp bước cho chúng em trong quá trình học tập và lập nghiệp sau này. Trong quá trình làm bài tập lớn, chúng em đã cố gắng hết sức để hoàn thành nhiệm vụ được giao. Nhóm em hy vọng có thể nhận thêm nhiều lời nhận xét, góp ý từ thầy giáo và các bạn!

***Chúng em xin chân thành cảm ơn!***

## MỤC LỤC

MỤC LỤC.....	1
DANH MỤC CÁC KÝ HIỆU, CHỮ CÁI VIẾT TẮT.....	3
DANH MỤC HÌNH ẢNH .....	4
DANH MỤC BẢNG BIỂU.....	5
LỜI CẢM ƠN.....	6
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT .....	9
1.1 Tổng quan về Neural Networks .....	9
1.1.1 Định nghĩa cơ bản.....	9
1.1.2 Mô hình của một mạng Neuron .....	10
1.1.3 Mô hình của một Neuron .....	10
1.1.4 Một số loại hàm kích hoạt .....	11
1.1.5 Các loại Neural Networks .....	13
1.1.6 Mạng Perceptron .....	15
1.1.7 Ứng dụng của Neural Networks .....	17
1.1.7.1 Phân loại mẫu .....	18
1.1.7.2 Dự đoán .....	18
1.1.7.3 Tối ưu hóa .....	18
1.2 Convolutional Neural Network (CNN).....	19
1.2.1 Các lớp cơ bản của một Convolutional Neural Network.....	19
1.2.1.1 Convolutional layer .....	19
1.2.1.2 Lớp Pooling .....	22
1.2.1.3 Lớp Fully Connected .....	23
1.2.2 Cách Convolutional Neural Network hoạt động.....	23
1.2.2.1 Khởi tạo Weight.....	24
1.2.2.2 Quy định Network .....	24
1.3 Giới thiệu về Tensorflow và Keras .....	25
1.3.1 Thư viện Tensorflow.....	25
1.3.2 Framework Keras.....	26
CHƯƠNG 2: MỘT SỐ MÔ HÌNH GIẢI QUYẾT BÀI TOÁN PHÂN LOẠI ẢNH .....	28
2.1 Bài toán phân loại ảnh.....	28
2.2 Một số mô hình phân loại ảnh .....	29
2.2.1 ResNet .....	29

2.2.2 Inception .....	32
2.2.3 VGG16.....	33
<b>CHƯƠNG 3: ỨNG DỤNG TENSORFLOW VÀ MẠNG VGG16 VÀO</b>	
<b>PHÂN LOẠI ẢNH .....</b>	<b>35</b>
3.1 Phân tích cách áp dụng Tensorflow và mạng VGG16.....	35
3.2 Phân tích bài toán.....	36
3.2.1 Phát biểu bài toán .....	36
3.2.2 Dữ liệu đầu vào .....	36
3.2.3 Dữ liệu đầu ra .....	36
3.3 Các bước thực hiện.....	36
3.3.1 Thu thập dữ liệu .....	36
3.3.2 Xây dựng chương trình .....	38
3.3.2.1 Tiền xử lý dữ liệu .....	38
3.3.2.2 Xây dựng mô hình .....	38
3.3.2.3 Kiểm tra, đánh giá hiệu quả của mô hình .....	42
<b>KẾT LUẬN.....</b>	<b>44</b>
<b>KIẾN NGHỊ.....</b>	<b>45</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>46</b>

## DANH MỤC HÌNH ẢNH

Hình 1. Ví dụ về mô hình của một Neuron	10
Hình 2. Hàm Relu	11
Hình 3. Hàm Sigmoid	12
Hình 4. Hàm Softmax	12
Hình 5. Feedforward network với single-layer của neuron	13
Hình 6. Fully connected feedforward multi-layer network với một layer ẩn và một layer đầu ra	14
Hình 7. Rosenblatt's Perceptron	15
Hình 8. Biểu đồ luồng tín hiệu của một perceptron	16
Hình 9. Mô tả phép tính tích chập convolution	18
Hình 10. Ví dụ về các lớp convolution	20
Hình 11. Pooling operation	21
Hình 12. Fully Connected Layer	22
Hình 13. Mô hình kiến trúc cơ bản của mạng RESNET	30
Hình 14. Phần trăm training error và test error của mạng RESNET	30
Hình 15. Minh họa kiến trúc của VGG16	32
Hình 16. Tổng quan về dữ liệu	35
Hình 17. Lấy dữ liệu	36
Hình 18. Chuẩn bị kiến trúc VGG16	37
Hình 19. Tiến hành huấn luyện mô hình	37
Hình 20. Đồ thị loss trong quá trình huấn luyện	38
Hình 21. Đồ thị accuracy trong quá trình huấn luyện	38
Hình 22. Thử lại với 31 ảnh ngẫu nhiên trong tập album	39
Hình 23. Hình thu về kết quả không chính xác	40



## **DANH MỤC BẢNG BIỂU**

Bảng 1. Mô tả ký hiệu của biểu đồ luồng tín hiệu của một perceptron	17
---------------------------------------------------------------------	----

## DANH MỤC CÁC KÝ HIỆU, CHỮ CÁI VIẾT TẮT

STT	Viết tắt	Dịch nghĩa
1	CNN	Convolutional Neural Network
2	ReLU	Rectified Linear Unit
3	ILSVRC	ImageNet Large Scale Visual Recognition Competition
4	AR	Augmented Reality
5	ORL	Olivetti Research Laboratory
6	CPU	Central Processing Unit
7	GPU	Graphic Processing Unit
8	RNNs	Recurrent Neural Networks
9	ResNet	Residual Network
10	DNN	Deep Neural Network

# MỞ ĐẦU

## 1. Tên đề tài

Ứng dụng Tensorflow và mạng VGG16 để phân loại ảnh.

## 2. Lý do chọn đề tài

Ngày nay các kỹ thuật về mạng nơron đã và đang được nghiên cứu, phát triển một cách rộng rãi. Mạng VGG16 là một trong số các mạng được huấn luyện trên tập dữ liệu lớn và có khả năng nhận diện nhiều đặc trưng khác nhau trong ảnh. Bên cạnh đó, hầu hết điện thoại của mọi người hiện nay chứa rất nhiều loại ảnh khác nhau, khó để tìm kiếm khi muốn lấy lại thông tin. Cũng bởi lý do đó, nhóm em chọn đề tài “Ứng dụng Tensorflow và mạng VGG16 để phân loại ảnh” nhằm giúp mọi người có thể phân loại ảnh một cách dễ dàng hơn.

## 3. Mục tiêu của đề tài

- Nhận diện ảnh thuộc thư mục ảnh cụ thể.
- Phân loại ảnh từ album ra các thư mục ảnh mong muốn.

## 4. Phương pháp nghiên cứu

- Sử dụng bộ dữ liệu thu thập từ nhiều nguồn trên Internet.
- Sử dụng kiến thức đã nghiên cứu được để tiến hành viết chương trình, báo cáo.

## 5. Đối tượng nghiên cứu

- Mạng CNN áp dụng cho bài toán phân loại.
- Thư viện Tensorflow và Framework Keras.

## 6. Phạm vi nghiên cứu

- Dữ liệu là các ảnh được đánh nhãn tương ứng với loại ảnh. Ví dụ: ảnh chụp màn hình, ảnh núi, ảnh chân dung
- Thử nghiệm trên thư viện ảnh của thành viên nhóm.

## **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

### **1.1 Tổng quan về Neural Networks**

#### **1.1.1 Định nghĩa cơ bản**

Neural Networks là một hệ thống phức tạp, phi tuyến tính, song song, bao gồm các đơn vị xử lý nhỏ và đơn giản đơn được kết nối với nhau để có thể thực hiện các phép tính. Neural Networks có thể được sửa đổi bằng cách thay đổi độ mạnh yếu của từng kết nối giữa các đơn vị xử lý để đạt được đầu ra mong muốn thông qua một quá trình gọi là ‘Học’.

Neural Networks tự động trích xuất các tính năng từ các tập data training trong quá trình học. Với khả năng tổng quát hóa, Neural Networks được đào tạo để có thể phân loại được các dữ liệu đầu vào mới vào các lớp đầu ra đã được đào tạo. Có một số thuộc tính của Neural Networks: neural Networks có thể tuyến tính (linear) hoặc phi tuyến tính (nonlinearity). Tầm quan trọng của phi tuyến tính (nonlinearity) là khi dự kiến một Network sẽ tính toán đầu ra mong muốn từ những tín hiệu đầu vào phi tuyến tính như tín hiệu giọng nói. Một Neural Networks thường ánh xạ một tập hợp các đầu vào thành một tập hợp các đầu ra, trong đó một tập hợp các ví dụ huấn luyện bao gồm các tín hiệu đầu vào duy nhất được chỉ định và phản hồi ra một đầu ra mong muốn tương ứng. Một cách ngẫu nhiên, các ví dụ đào tạo được chọn từ một tập hợp và được cung cấp cho Mạng. Độ mạnh của các kết nối giữa các nút của mạng sau đó được sửa đổi để giảm thiểu sự khác biệt giữa phản hồi mong muốn và phản hồi thực tế được tạo ra bởi Mạng. Quá trình này được lặp lại cho đến khi chênh lệch giữa phản hồi mong muốn và phản hồi thực tế bằng 0 hoặc nhỏ nhất, do đó đạt được trạng thái ổn định cho mạng.

Neural Networks có tính thích nghi, vì chúng có thể điều chỉnh trọng lượng khớp thần kinh với những thay đổi xung quanh.

### 1.1.2 Mô hình của một mạng Neuron

Một đơn vị xử lý đơn giản được kết nối với các đơn vị khác của Neural Network được gọi là một “Neuron”. Neuron bao gồm ba yếu tố cơ bản.

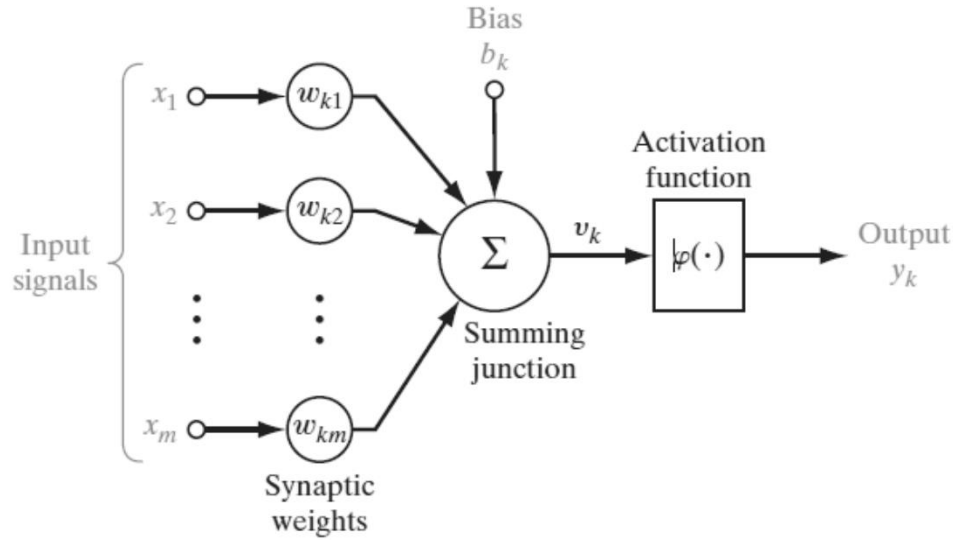
- **Phần tử đầu tiên** là một tập hợp các khớp thần kinh hoặc các liên kết kết nối, mỗi phần tử được đặc trưng bởi một trọng lượng (weight) hoặc sức mạnh (strength) của riêng nó. Đầu vào là một tập các liên kết kết nối được nhân với độ mạnh của kết nối đó. Trong trường hợp của một artificial neuron, phạm vi giá trị có thể bao gồm giá trị âm hoặc dương.

- **Phần tử thứ hai** là adder, một bộ kết hợp tuyến tính tính tổng mỗi tín hiệu đầu vào tương ứng nhân với cường độ của kết nối.

- **Phần tử thứ ba** là hàm kích hoạt (activation function), được sử dụng để giới hạn biên độ đầu ra của nơ-ron trong khoảng  $[0,1]$  hoặc  $[-1,1]$ , tùy thuộc vào hàm kích hoạt.

### 1.1.3 Mô hình của một Neuron

Hình ảnh dưới đây là một ví dụ về neuron.



Hình 1. Ví dụ về mô hình của một Neuron

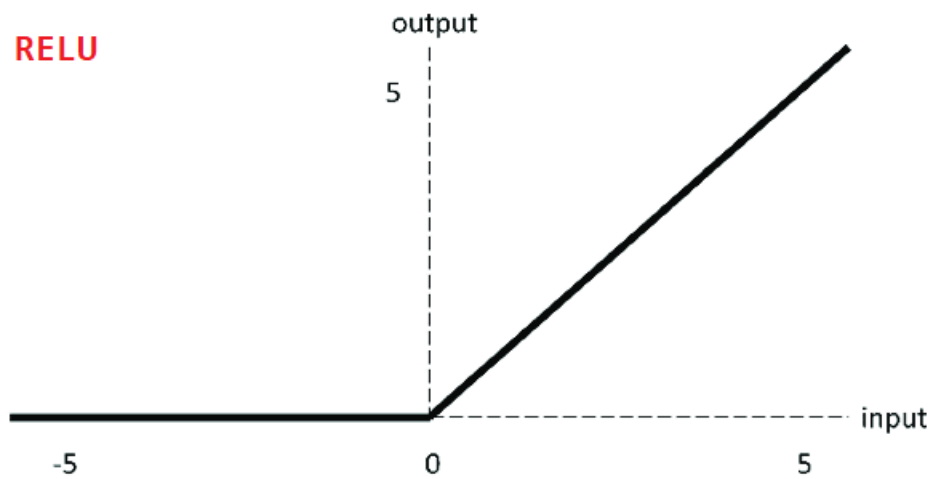
Chúng ta có thể thấy một neuron cơ bản sẽ có 3 lớp cơ bản là Input layer, Hidden layer và Output layer.

#### 1.1.4 Một số loại hàm kích hoạt

Hàm kích hoạt (Activation function) là những hàm phi tuyến tính được áp dụng vào đầu ra của các neuron trong tầng ẩn (Hidden layer) của một mô hình mạng (Network), đầu ra được chuyển đổi này được sử dụng làm đầu vào của neuron ở layer tiếp theo.

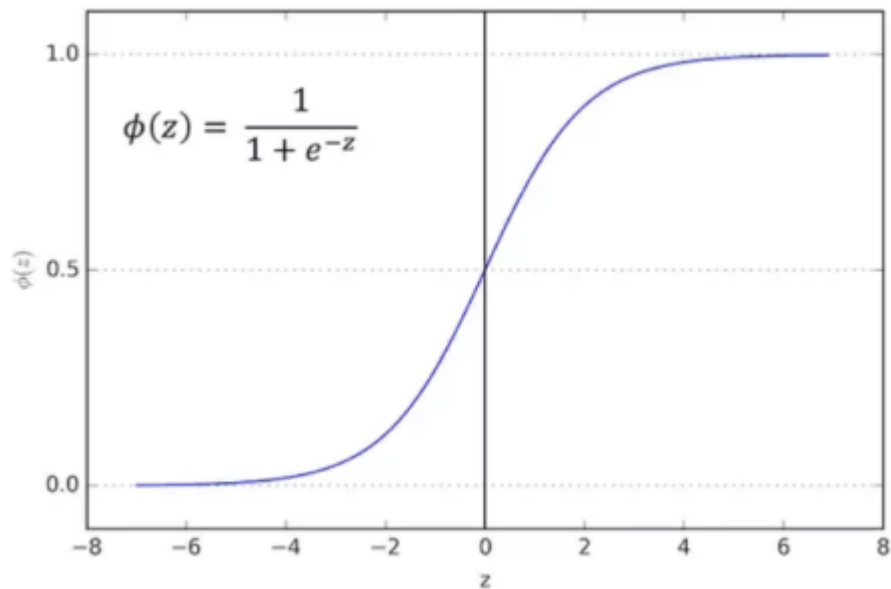
Một số Activation Function thường gặp là:

- ReLU: Hàm ReLU cho đầu ra bằng đầu vào nếu đầu vào lớn hơn 0 và bằng 0 nếu đầu vào nhỏ hơn hoặc bằng 0. Hàm ReLU thường được sử dụng trong các lớp nơ-ron ẩn của mạng nơ-ron và có thể giúp tăng tốc độ huấn luyện.



*Hình 2. Hàm Relu*

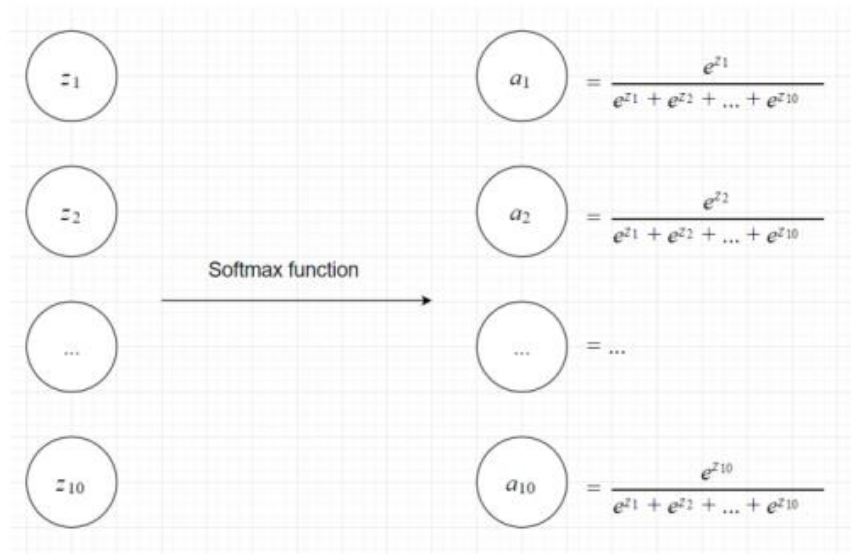
- Sigmoid: Hàm sigmoid chuyển đổi đầu vào thành một giá trị trong khoảng từ 0 đến 1. Hàm sigmoid được sử dụng rộng rãi trong các lớp nơon đầu ra và trong các mô hình phân loại nhị phân.



*Hình 3. Hàm Sigmoid*



- Softmax: Hàm softmax thường được sử dụng trong lớp nơ-ron đầu ra của mạng nơ-ron phân loại. Nó chuyển đổi đầu vào thành một phân phối xác suất trên các lớp đầu ra, giúp tạo ra các dự đoán xác suất cho từng lớp.



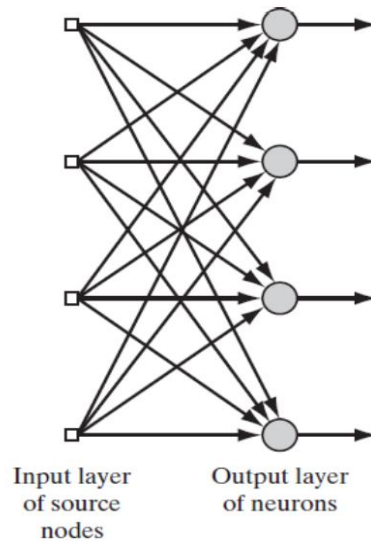
Hình 4. Hàm Softmax

### 1.1.5 Các loại Neural Networks

Có 2 loại Neural Networks là Single-layer feedforward network và Multilayer feedforward network.

#### ❖ Single-layer feedforward network

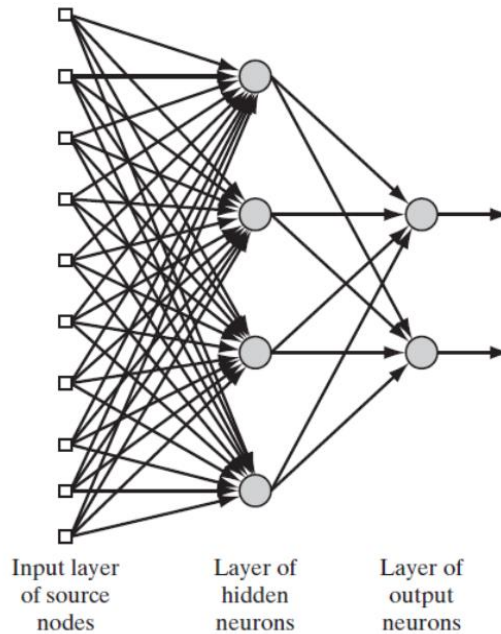
Một Single-layer network là một loại mạng feedforward network (mạng truyền thẳng) chiếu các đầu vào từ lớp của các nút nguồn trên lớp đầu ra của các neuron (computation nodes). “Single layer” là một phép tính được thực hiện bởi một lớp duy nhất, là lớp đầu ra. Không có tính toán nào được thực hiện bởi lớp đầu vào của các nút nguồn.



*Hình 5. Feedforward network với single-layer của neuron*

#### ❖ **Multilayer feedforward network**

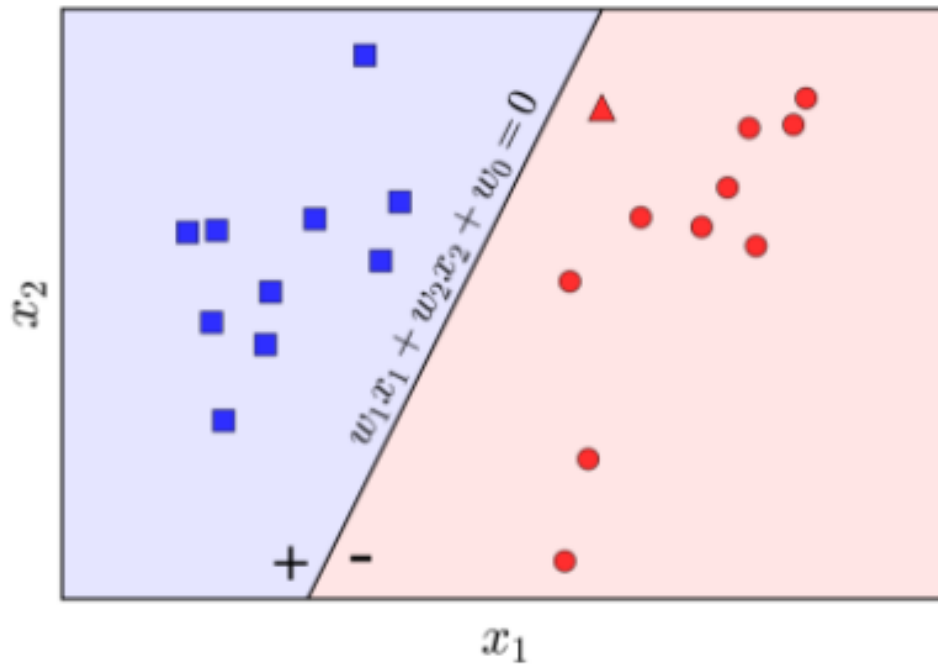
Một Multilayer network là một feedforward network với một layer của các nút nguồn, một hoặc nhiều layer ẩn của lớp neurons và lớp đầu ra của các neuron (Hình 1.3). Cả lớp ẩn và lớp đầu ra đều bao gồm các nút tính toán. Thuật ngữ lớp “Ẩn” đề cập đến thực tế là lớp như vậy không được nhìn thấy trực tiếp từ đầu ra hoặc đầu vào của mạng và chúng nhận đầu vào từ các nút nguồn hoặc các nút của các lớp ẩn trước đó và kết quả đầu ra của mạng.



Hình 6. Fully connected feedforward multi-layer network với một layer ẩn và một layer đầu ra

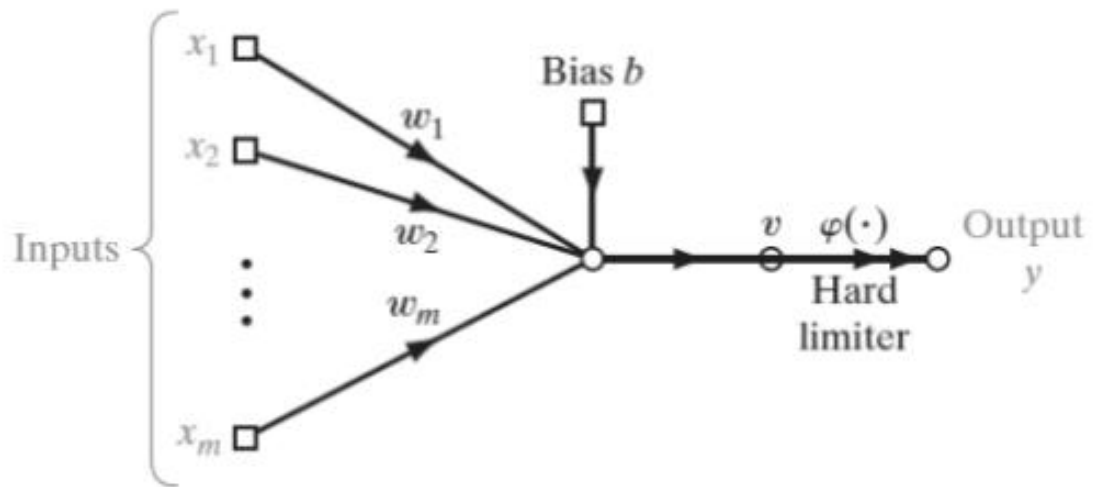
### 1.1.6 Mạng Perceptron

Mạng Perceptron là dạng đơn giản nhất của một Neural Network bao gồm một neuron duy nhất có trọng số liên kết (synaptic weights) và độ lệch (bias) có thể điều chỉnh được. Một neuron riêng lẻ được sử dụng nghiêm ngặt như một bộ phân loại cho hai lớp của các mẫu có thể phân tách tuyến tính, nằm trên các mặt đối diện của một siêu phẳng. Trong Hình 1.7, một mẫu có thể phân tách tuyến tính của hai lớp được phân loại bằng cách sử dụng một Rosenblatt's Perceptron.



*Hình 7. Rosenblatt's Perceptron*

Cách mà một Perceptron được sử dụng cho bài toán phân loại hai lớp là lấy các tín hiệu đầu vào khác nhau  $x_m$  và phân loại chúng thành một trong hai lớp,  $C_1$  hoặc  $C_2$ , tùy thuộc vào giá trị đầu ra  $y$  của nơ-ron, cho dù đó là  $+1$  hay  $-1$  (Hình 3.2). Kết quả phân loại mẫu có thể được vẽ trên một mặt phẳng hai chiều để hiểu rõ hơn về hoạt động của perceptron. Hai lớp trên mặt phẳng hai chiều được ngăn cách bởi một ranh giới quyết định, đó là một siêu không gian; đường phân cách hai lớp với nhau.



Hình 8. Biểu đồ luồng tín hiệu của một perceptron

Ký Hiệu	Mô tả
$x_m$	Tín hiệu đầu vào cho nơ-ron
$w_m$	synaptic weights của nơ-ron
$b$	bias
$v$	Trường cực bộ gây ra hoặc activation potential của nơ-ron
$\varphi(\cdot)$	Activation potential
$y$	Tín hiệu đầu ra của nơ-ron

Bảng 1. Mô tả ký hiệu của biểu đồ luồng tín hiệu của một perceptron

### 1.1.7 Ứng dụng của Neural Networks

Phân loại mạng nơ-ron, hoặc các ứng dụng dự báo đã được triển khai rộng rãi trong các lĩnh vực công nghiệp, kinh doanh và khoa học với nhiều ứng dụng khác nhau do các đặc tính của mạng nơ-ron là học và tổng quát hóa, bất kể thuật toán là gì, cho dù là backpropagation, radial basis functions, genetic hoặc Kohonen's learning vector quantization. Mạng nơ-ron đã đáp ứng nhu cầu đa dạng theo ba loại: phân loại mẫu, dự đoán và tối ưu hóa.

#### **1.1.7.1 Phân loại mẫu**

Mạng nơ-ron nhân tạo có thể xử lý nhiều tín hiệu đầu vào và suy ra các mối quan hệ phi tuyến tính, chúng được sử dụng để xử lý hình ảnh và nhận dạng các ký tự. Các ứng dụng xử lý hình ảnh bao gồm một loạt các ngành công nghiệp, nhận dạng hình ảnh cho điện thoại thông minh hoặc để chống lại tội phạm bằng cách phát hiện tội phạm bị truy nã từ cộng đồng, cũng như xử lý hình ảnh vệ tinh cho mục đích nông nghiệp để phân biệt các thuộc tính khác nhau của các loại cây trồng khác nhau trên cùng một cánh đồng rộng lớn. Ứng dụng nhận dạng ký tự phù hợp hơn với các ngành ngân hàng và bảo mật, nơi chữ viết tay trên séc có thể được nhận dạng và xử lý thành các số được số hóa. Chữ ký viết tay hợp pháp cũng có thể được phân biệt với những chữ ký giả mạo.

#### **1.1.7.2 Dự đoán**

Với khả năng mô hình hóa các mối quan hệ phi tuyến tính mà không thể xác định từ các dữ liệu đầu vào khác nhau, mạng nơ-ron có khả năng dự báo các xu hướng dữ liệu trong tương lai. Có rất nhiều trường hợp sử dụng để dự báo mạng nơ-ron nhân tạo trong các lĩnh vực bán hàng, phân bổ tài chính giữa các sản phẩm, thị trường chứng khoán. Trong phân tích tiếp thị, mạng nơ-ron đã được sử dụng để cắt giảm chi phí của các chiến dịch tiếp thị bằng cách dự báo những khách hàng chưa có khả năng từ những khách hàng tiềm năng dựa trên tập mẫu của tập dữ liệu đã được thu thập từ họ.

#### **1.1.7.3 Tối ưu hóa**

Các phương pháp tối ưu hóa được sử dụng để tối đa hóa hoặc giảm thiểu các chức năng, mục tiêu nhất định của hệ thống phi tuyến tính. Mạng nơ-ron nhân tạo giúp tính gần đúng các hàm mục tiêu, do đó cho phép sử dụng các kỹ thuật phù hợp để phát triển các phương trình đa thức nhằm tính toán lời giải cần thiết để đạt được sự tối ưu hóa của hệ thống.

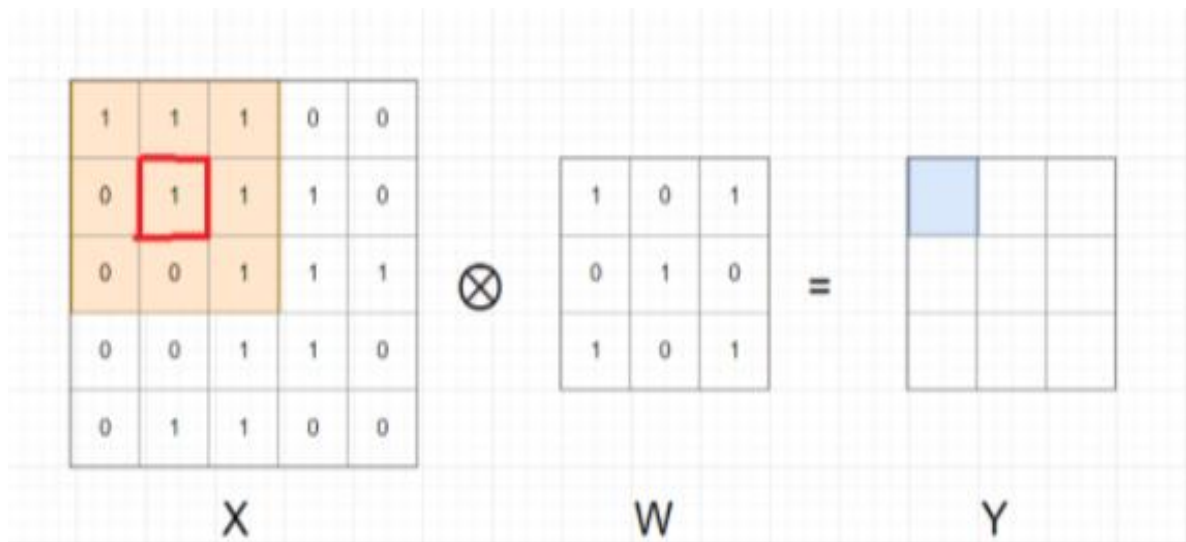
## 1.2 Convolutional Neural Network (CNN)

### 1.2.1 Các lớp cơ bản của một Convolutional Neural Network

#### 1.2.1.1 Convolutional layer

##### ❖ Phép tính convolution.

Ta giả sử ma trận cần tính convolution là ma trận  $X$  có kích thước  $n \times m$ . Và 1 ma trận  $k$  có kích thước là  $x \times x$ . Kí hiệu phép tính convolution ( $\otimes$ ), kí hiệu  $Y = X \otimes W$ . Với mỗi phần tử  $x_{ij}$  trong ma trận  $X$  lấy ra một ma trận có kích thước bằng kích thước của kernel  $W$  có phần tử  $x_{ij}$  làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận  $A$ . Sau đó tính tổng các phần tử của phép tính element-wise của ma trận  $A$  và ma trận  $W$ , rồi viết vào ma trận kết quả  $Y$ .



Hình 9. Mô tả phép tính tích chập convolution

Để thấy với phép tính như trên thì shape của ma trận  $Y$  sẽ nhỏ hơn shape (kích thước) của ma trận  $X$  đầu vào. Với những trường hợp cần ma trận  $Y$  có cùng kích thước với ma trận  $X$  chúng ta thêm 1 hệ số được gọi là padding vào ma trận  $X$  rồi thực hiện phép tính convolution như bình thường.

### ❖ Lớp tích chập (Convolution)

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.

Giả sử input của 1 convolutional layer tổng quát là tensor kích thước  $H \times W \times D$ . Kernel có kích thước  $F \times F \times D$  (kernel luôn có depth bằng depth của input và  $F$  thường là số lẻ vì ô vuông lưới chẵn  $\times$  chẵn thì sẽ không có 1 ô vuông ở tâm đối xứng dẫn tới giảm độ chính xác), stride:  $S$ , padding:  $P$ . Convolutional layer áp dụng  $K$  kernel.

Output sẽ có kích thước:








$$\left(\frac{H + 2 \times P - F}{S} + 1\right) \times \left(\frac{W + 2 \times P - F}{S} + 1\right) \times K$$

Ta cũng có thể chồng nhiều lớp convolution lên nhau để lấy được đặc trưng của ảnh. Trước khi output của lớp convolution trước làm input của lớp sau thì ta đưa qua 1 hàm phi tuyến tính.

### ❖ Ý nghĩa của lớp Convolution

Convolution sẽ giúp làm mờ, làm nét ảnh. Lấy được các đặc trưng của ảnh. Mỗi kernel (ma trận tích chập) khác nhau sẽ đều có những tác dụng khác nhau.



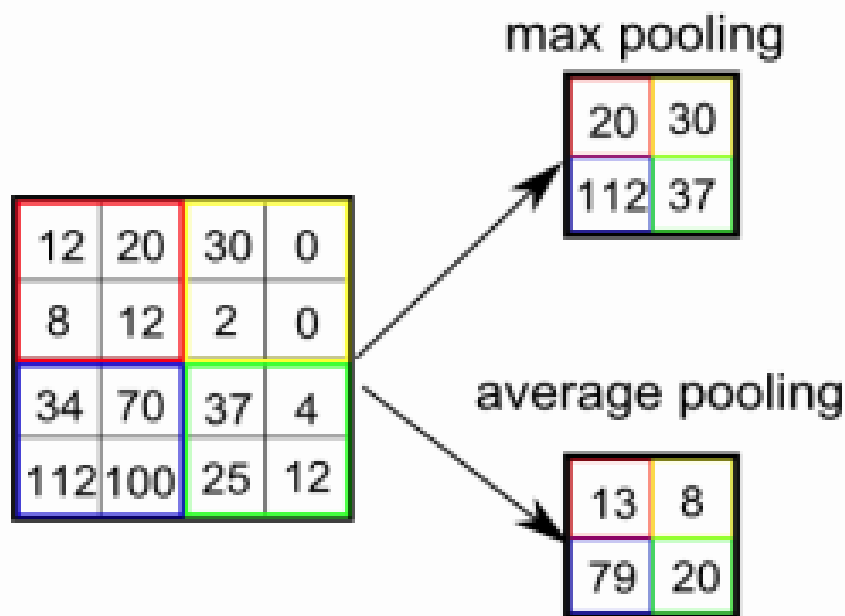
Operation	Filter	Convolved Image
<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>Edge detection</b>	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1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

Hình 10. Ví dụ về các lớp convolution

### 1.2.1.2 Lớp Pooling

Giống như convolution layer, dữ liệu đầu vào được convoluted với một bộ lọc để tạo thành một ma trận convolution.

Ví dụ trong một Pooling layer một cụm  $3 \times 3$  được lấy từ dữ liệu đầu vào  $5 \times 5$ . Điều này sẽ được miêu tả ở Hình 11, có hai hàm pooling, average hoặc max còn tùy thuộc vào hàm. Giá trị trung bình của tất cả các phần tử sau đó được đặt vào  $C_{11}$  hoặc giá trị tối đa của cụm được đặt vào  $C_{11}$ . Quá trình này được lặp đi lặp lại tùy thuộc vào chức năng, từ phần tử  $C_{11}$  cho đến phần tử  $C_{33}$  cho pooling  $3 \times 3$ .



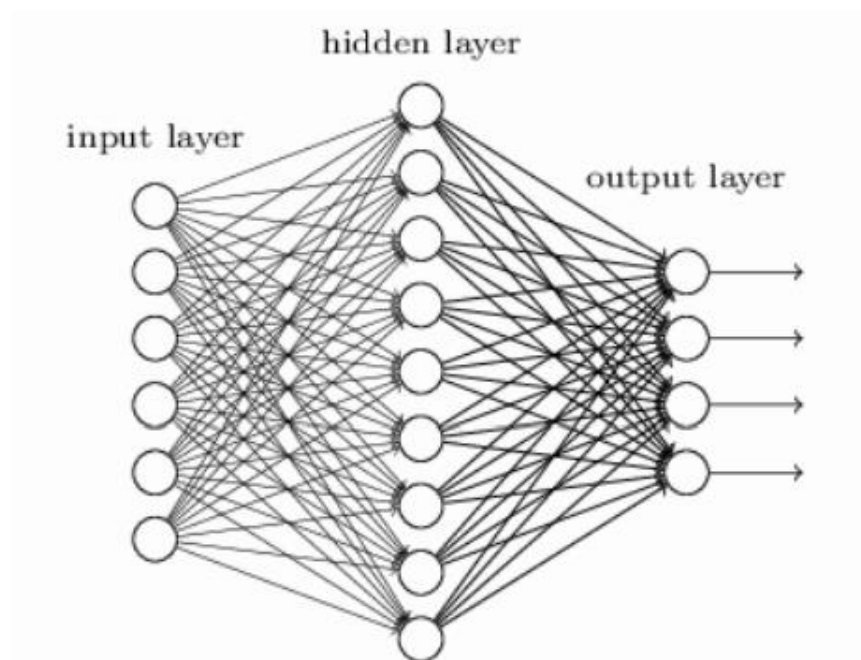
Hình 11. Pooling operation

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Việc giảm kích thước dữ liệu giúp giảm các phép tính toán trong model. Bên cạnh đó, với phép pooling

kích thước ảnh giảm, do đó lớp convolution học được các vùng có kích thước lớn hơn.

### 1.2.1.3 Lớp Fully Connected

Bằng nhiều convolution và pooling layer, đầu ra được nhận đến một fully connected layer dưới dạng đầu vào như trong *Hình 1.12*. Một fully connected layer thường được đặt ở phần cuối trong cấu trúc CNN, hoặc ở đâu đó trung gian. Loại layer này rất giống với đa lớp truyền tiếp perceptron (feedforward multilayer perceptron). Trong layer này, tất cả thông tin được lọc, lấy mẫu được thu thập để CNN bắt đầu học. Hầu hết các trọng số cũng nằm trong phần này của mạng.



*Hình 12. Fully Connected Layer*

### 1.2.2 Cách Convolutional Neural Network hoạt động

Đối với bất kì nhiệm vụ nào được yêu cầu thực hiện bởi CNN, các thông số cần phải được điều chỉnh tương ứng với mỗi nhiệm vụ cần thực thi. Các cơ chế và kỹ thuật khác nhau như khởi tạo trọng số và điều chỉnh mạng được thực hiện để tối ưu hóa mạng Neuron.

### 1.2.2.1 Khởi tạo Weight

Để huấn luyện CNN một cách ổn định, cần khởi tạo một trọng số chính xác, nếu không điều này sẽ dẫn đến những khó khăn khác nhau trong quá trình học, tùy vào phương pháp học được sử dụng. Ví dụ, trong trường hợp sử dụng phương pháp back-propagation (lan truyền ngược), việc khởi tạo trọng số không chính xác sẽ dẫn đến việc biến mất hoặc bùng nổ của vấn đề gradient (Vanishing and Exploding Gradient). Có kỹ thuật này là khởi tạo Gaussian Random và khởi tạo Uniform Random.

#### ❖ Khởi tạo Gaussian Random

Kỹ thuật khởi tạo này được sử dụng cho các convolution và fully connected layers của CNN với ma trận ngẫu nhiên của các phần tử được lấy mẫu từ Gaussian với giá trị trung bình bằng 0 và độ lệch nhỏ.

#### ❖ Khởi tạo Uniform Random

Kỹ thuật khởi tạo này được sử dụng cho các convolution và fully connected layers của CNN với ma trận ngẫu nhiên của các phần tử được lấy mẫu từ Uniform với giá trị trung bình bằng 0 và độ lệch nhỏ.

### 1.2.2.2 Quy định Network

CNN được mong đợi sẽ học và khái quát hóa; tuy nhiên, trong một số trường hợp các mạng không thể học và tổng quát hóa vì chúng có xu hướng phù hợp quá mức với dữ liệu đào tạo trong quá trình học. Mạng được kỳ vọng sẽ là học hỏi thay vì ghi nhớ để có thể khái quát hóa và thích ứng tốt hơn với dữ liệu không nhìn thấy được sử dụng để ngăn chặn over-fitting. Data augmentation, dropout, batch normalization, ensemble model averaging, và early stopping là những ví dụ về kỹ thuật chính quy hóa.

CNN có thể phân loại dữ liệu ngẫu nhiên thành các lớp (class) khác nhau. Dữ liệu được phân loại dựa trên các tính năng, đặc điểm và phẩm chất được chia sẻ

với các bộ dữ liệu khác trong các danh mục tương ứng. Khi so sánh với các thuật toán phân loại khác, CNN dễ huấn luyện hơn vì nó yêu cầu ít kết nối và tham số hơn, như đã được chứng minh bằng kết quả thu được từ ILSVRC. Trong cuộc thi này deep CNN đã đạt được top 1 về hiệu suất, hoạt động tốt hơn đáng kể so với các thuật toán được sử dụng trước đây. Deep CNN đã phân loại 1,2 triệu hình ảnh có độ phân giải cao trong cuộc thi thành 1000 lớp khác nhau. CNN được sử dụng trong các ứng dụng khác nhau, phân loại các phạm vi dữ liệu khác nhau từ hình ảnh đến văn bản, ... Một số bộ phân loại CNN hiện đại sẽ được nêu ra trong phần này.

## **1.3 Giới thiệu về Tensorflow và Keras**

### **1.3.1 Thư viện Tensorflow**

TensorFlow là một thư viện mã nguồn mở và một khung (framework) mạnh mẽ trong lĩnh vực học máy và deep learning. Nó được phát triển bởi Google Brain Team và được giới thiệu lần đầu vào năm 2015. TensorFlow cung cấp một cách tiếp cận linh hoạt và hiệu quả để xây dựng và huấn luyện các mô hình học máy phức tạp.

Tầm quan trọng của TensorFlow trong việc xây dựng mô hình học máy là như sau:

- **Linh hoạt và phong phú:** TensorFlow cung cấp một loạt các công cụ và API để xây dựng và triển khai các mô hình học máy. Nó hỗ trợ cả việc tạo ra các mạng nơ-ron truyền thẳng (feedforward neural networks) đơn giản và mô hình học sâu (deep learning) phức tạp. Bạn có thể tạo ra các kiến trúc mạng phức tạp bằng cách kết hợp các tầng, kết nối, và hoạt động khác nhau trong TensorFlow.

- **Xử lý hiệu quả:** TensorFlow được tối ưu để hoạt động trên nhiều nền tảng tính toán, bao gồm CPU và GPU. Nó cung cấp các biểu đồ tính toán và trình tối

ưu hóa để tận dụng tối đa khả năng tính toán song song của GPU, giúp tăng tốc độ huấn luyện và dự đoán của mô hình học máy.

- Khả năng phân phối và triển khai: TensorFlow hỗ trợ phân phối và triển khai các mô hình học máy trên nhiều máy tính, bao gồm cả các hệ thống phân tán. Điều này cho phép bạn huấn luyện và triển khai mô hình trên dữ liệu lớn và môi trường sản xuất thực tế.

- Cộng đồng lớn và hỗ trợ đa dạng: TensorFlow có một cộng đồng đông đảo của các nhà phát triển và nghiên cứu, cung cấp tài liệu phong phú, ví dụ và hỗ trợ cho người dùng. Bạn có thể tìm thấy nhiều nguồn tài liệu, bài viết, mã mẫu và mô hình sẵn có để khám phá và áp dụng trong các dự án của mình.

- Sự phát triển liên tục: TensorFlow được Google và cộng đồng của nó chăm sóc và phát triển tiếp tục. Các phiên bản mới của TensorFlow được phát hành định kỳ, bổ sung các tính năng mới, cải thiện hiệu suất và sửa các lỗi. Điều này đảm bảo rằng TensorFlow luôn cung cấp các công nghệ mới nhất và đáng tin cậy cho việc xây dựng mô hình học máy.

### **1.3.2 Framework Keras**

Keras là một framework cao cấp (high-level) được xây dựng trên TensorFlow, được phát triển bởi François Chollet. Nó cung cấp một giao diện dễ sử dụng và trực quan để xây dựng và huấn luyện mô hình mạng nơ-ron, đặc biệt là mô hình Convolutional Neural Networks (CNNs).

Điểm mạnh của Keras là sự đơn giản và tập trung vào tính linh hoạt. Nó giúp đơn giản hóa việc xây dựng và huấn luyện mô hình học máy bằng cách cung cấp các lớp và module trừu tượng để xây dựng mạng nơ-ron và các phép toán liên quan. Bằng cách sử dụng Keras, bạn có thể tạo ra các mô hình mạng nơ-ron phức tạp một cách dễ dàng chỉ với một vài dòng mã.

Một số đặc điểm và lợi ích chính của Keras là:

- Dễ sử dụng: Keras cung cấp một API đơn giản và dễ hiểu, cho phép người dùng xây dựng và tùy chỉnh các mô hình mạng nơ-ron một cách dễ dàng. Cú pháp của Keras rõ ràng và cho phép người dùng tập trung vào việc xây dựng mô hình mà không phải lo lắng về các chi tiết cài đặt phức tạp.

- Hỗ trợ đa ngôn ngữ: Keras hỗ trợ nhiều ngôn ngữ lập trình, bao gồm Python, R và Julia. Điều này giúp mở rộng cộng đồng sử dụng và đóng góp vào Keras.

- Tích hợp với TensorFlow: Keras được xây dựng trên TensorFlow, cho phép tận dụng toàn bộ tiềm năng của TensorFlow trong việc xử lý và tính toán. Bạn có thể sử dụng các tính năng và công cụ của TensorFlow một cách dễ dàng trong Keras.

- Hỗ trợ mô hình mạng phổ biến: Keras cung cấp các lớp và mô-đun để xây dựng các mô hình mạng nơ-ron phổ biến như CNNs, Recurrent Neural Networks (RNNs) và mạng nơ-ron tái cấu trúc (autoencoders). Bạn có thể sử dụng các lớp này để xây dựng các mô hình phức tạp và đáp ứng nhu cầu của các bài toán khác nhau.

- Hỗ trợ mạnh mẽ cho huấn luyện mô hình: Keras cung cấp các công cụ hỗ trợ cho việc huấn luyện mô hình, bao gồm các thuật toán tối ưu (optimizer), hàm mất mát (loss function), và các đánh giá (metrics). Các tham số này cũng có thể tùy chỉnh và tinh chỉnh để tối ưu hoá quá trình huấn luyện.

## **CHƯƠNG 2: MỘT SỐ MÔ HÌNH GIẢI QUYẾT BÀI TOÁN PHÂN LOẠI ẢNH**

### **2.1 Bài toán phân loại ảnh**

Bài toán phân loại ảnh là một bài toán trong lĩnh vực thị giác máy tính, trong đó mục tiêu là xác định loại hình ảnh nằm trong một số lớp đã xác định trước. Đây là một bài toán phổ biến và có nhiều ứng dụng thực tế, ví dụ như nhận diện chủ thể trong ảnh, phát hiện bệnh trong hình ảnh y khoa, và nhiều hơn nữa.

Bài toán phân loại ảnh thường được giải quyết thông qua việc sử dụng các thuật toán học máy và mạng nơ-ron sâu (deep neural networks). Các bước chính để giải quyết bài toán phân loại ảnh bao gồm:

1. Chuẩn bị dữ liệu: Bước này liên quan đến việc thu thập và chuẩn bị dữ liệu huấn luyện. Dữ liệu huấn luyện bao gồm ảnh đầu vào và nhãn tương ứng với mỗi ảnh.
2. Tiền xử lý dữ liệu: Trước khi đưa dữ liệu vào mô hình, có thể cần thực hiện các bước tiền xử lý như chuẩn hóa, cắt tỉa (cropping), thay đổi kích thước (resizing) hoặc áp dụng các phép biến đổi khác để chuẩn bị dữ liệu cho việc huấn luyện.
3. Xây dựng mô hình: Mô hình phân loại ảnh thường được xây dựng bằng cách sử dụng mạng nơ-ron sâu như Convolutional Neural Network (CNN). Mạng CNN có khả năng tự động học các đặc trưng từ ảnh và xây dựng các mức độ trừu tượng để phân loại ảnh.
4. Huấn luyện mô hình: Mô hình được huấn luyện bằng cách đưa dữ liệu huấn luyện vào mạng CNN và điều chỉnh các trọng số của mạng để tối thiểu hóa sai số giữa nhãn dự đoán và nhãn thực tế.



5. Đánh giá mô hình: Mô hình được đánh giá bằng cách sử dụng dữ liệu kiểm tra mà mô hình chưa nhìn thấy trước đó. Đánh giá có thể được thực hiện bằng cách tính toán các độ đo như độ chính xác (accuracy), độ phủ (recall), độ chính xác dương tính (precision) và F1-score.

6. Dự đoán và phân loại: Sau khi mô hình được huấn luyện và đánh giá, nó có thể được sử dụng để dự đoán và phân loại ảnh mới bằng cách đưa ảnh mới vào mô hình và lấy nhãn dự đoán có xác suất cao nhất.

Trên thực tế, việc giải quyết bài toán phân loại ảnh có thể phức tạp và yêu cầu nhiều kỹ thuật và công cụ phức tạp hơn so với những gì đã được trình bày ở trên. Các mô hình tiên tiến như ResNet, Inception, hoặc Efficient Net cùng với các kỹ thuật như data augmentation, transfer learning và fine-tuning cũng được sử dụng để cải thiện hiệu suất phân loại ảnh.

## **2.2 Một số mô hình phân loại ảnh**

### **2.2.1 ResNet**

ResNet (Residual Network) được giới thiệu đến công chúng vào năm 2015 và thậm chí đã giành được vị trí thứ 1 trong cuộc thi ILSVRC 2015 với tỉ lệ lỗi top 5 chỉ 3.57%. Không những thế nó còn đứng vị trí đầu tiên trong cuộc thi ILSVRC and COCO 2015 với ImageNet Detection, ImageNet localization, Coco detection và Coco segmentation. Hiện tại thì có rất nhiều biến thể của kiến trúc ResNet với số lớp khác nhau như ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152,... Với tên là ResNet theo sau là một số chỉ kiến trúc ResNet với số lớp nhất định.

#### **❖ Lợi thế của ResNet**

Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm hoặc hàng nghìn lớp chập. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều

lớp chập sẽ xảy ra hiện tượng Vanishing Gradient dẫn tới quá trình học tập không tốt.

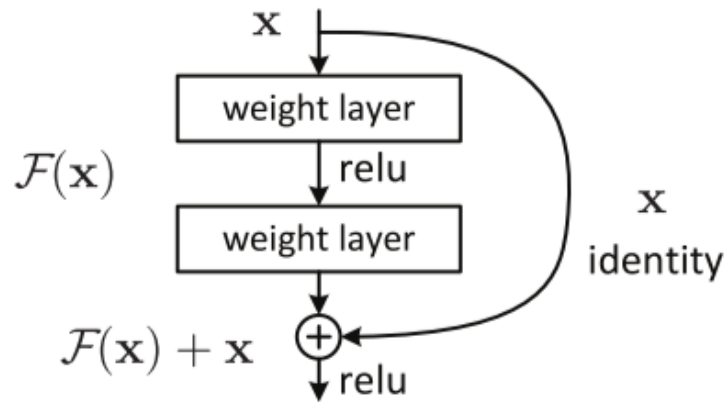
Backpropagation Algorithm (thuật toán lan truyền ngược) là một kỹ thuật thường được sử dụng trong quá trình training DNNs. Ý tưởng chung của thuật toán là sẽ đi từ output layer đến input layer và tính toán gradient của cost function tương ứng cho từng parameter (weight) của network. Gradient Descent, sau đó, sẽ được sử dụng để cập nhật các parameter đó.

Quá trình trên sẽ được lặp lại cho tới khi các parameter của network hội tụ. Thông thường chúng ta sẽ có một hyperparameter định nghĩa cho số lượng vòng lặp để thực hiện quá trình trên. Hyperparameter đó thường được gọi là số Epoch (hay số lần mà training set được duyệt qua một lần và weights được cập nhật). Nếu số lượng vòng lặp quá nhỏ, DNN có thể sẽ không cho ra kết quả tốt, và ngược lại thì thời gian training sẽ quá dài nếu số lượng vòng lặp quá lớn. Ở đây ta có một tradeoff giữa độ chính xác và thời gian training.

Tuy nhiên trên thực tế gradients thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Kết quả là các cập nhật thực hiện bởi Gradient Descent không làm thay đổi nhiều weights của các layer đó, khiến chúng không thể hội tụ và DNN sẽ không thu được kết quả tốt. Hiện tượng này được gọi là Vanishing Gradients => ResNet ra đời để giải quyết vấn đề này.

### ❖ Kiến trúc của mạng Resnet

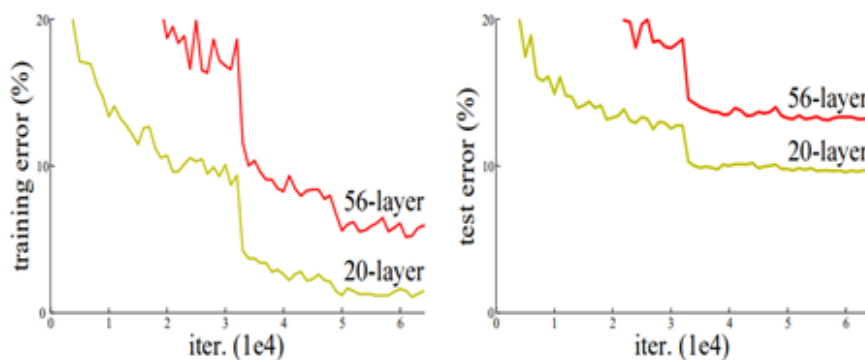
Giải pháp mà ResNet đưa ra là sử dụng kết nối "tắt" đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block, như trong *Hình 1.14*:



Hình 13. Mô hình kiến trúc cơ bản của mạng RESNET

ResNet gần như tương tự với các mạng gồm có convolution, pooling, activation và fully-connected layer. Ảnh bên trên hiển thị khối dư được sử dụng trong mạng. Xuất hiện một mũi tên cong xuất phát từ đầu và kết thúc tại cuối khối dư. Hay nói cách khác là sẽ bổ sung Input X vào đầu ra của layer, hay chính là phép cộng mà ta thấy trong hình minh họa, việc này sẽ chống lại việc đạo hàm bằng 0, do vẫn còn cộng thêm X. Với  $H(x)$  là giá trị dự đoán,  $F(x)$  là giá trị thật (nhãn), chúng ta muốn  $H(x)$  bằng hoặc xấp xỉ  $F(x)$ .

Theo như nghiên cứu [1] Resnet có thể sử dụng được một số lượng layer rất lớn mà không xảy ra hiện tượng mất đạo hàm (Vanishing Gradients).



Hình 14. Phần trăm training error và test error của mạng RESNET

### 2.2.2 Inception

Inception là một kiến trúc mạng neural tích chập (CNN) được sử dụng để phân loại ảnh. Nó được giới thiệu trong bài báo "Going Deeper with Convolutions" của Szegedy et al. vào năm 2014.

Kiến trúc Inception có tên gọi từ module "Inception" trong đó sử dụng các bộ lọc có kích thước khác nhau (1x1, 3x3, 5x5) cùng với pooling để thu thập thông tin từ ảnh đầu vào ở nhiều tỷ lệ khác nhau. Ý tưởng của Inception là kết hợp các bộ lọc với kích thước khác nhau để cùng lúc học được các đặc trưng ở các tỷ lệ và quy mô khác nhau.

Mạng Inception có nhiều phiên bản, ví dụ như Inception-v1, Inception-v2, Inception-v3, và Inception-v4. Các phiên bản này đã được cải tiến theo thời gian với mục tiêu cải thiện hiệu suất và hiệu quả tính toán của mạng.

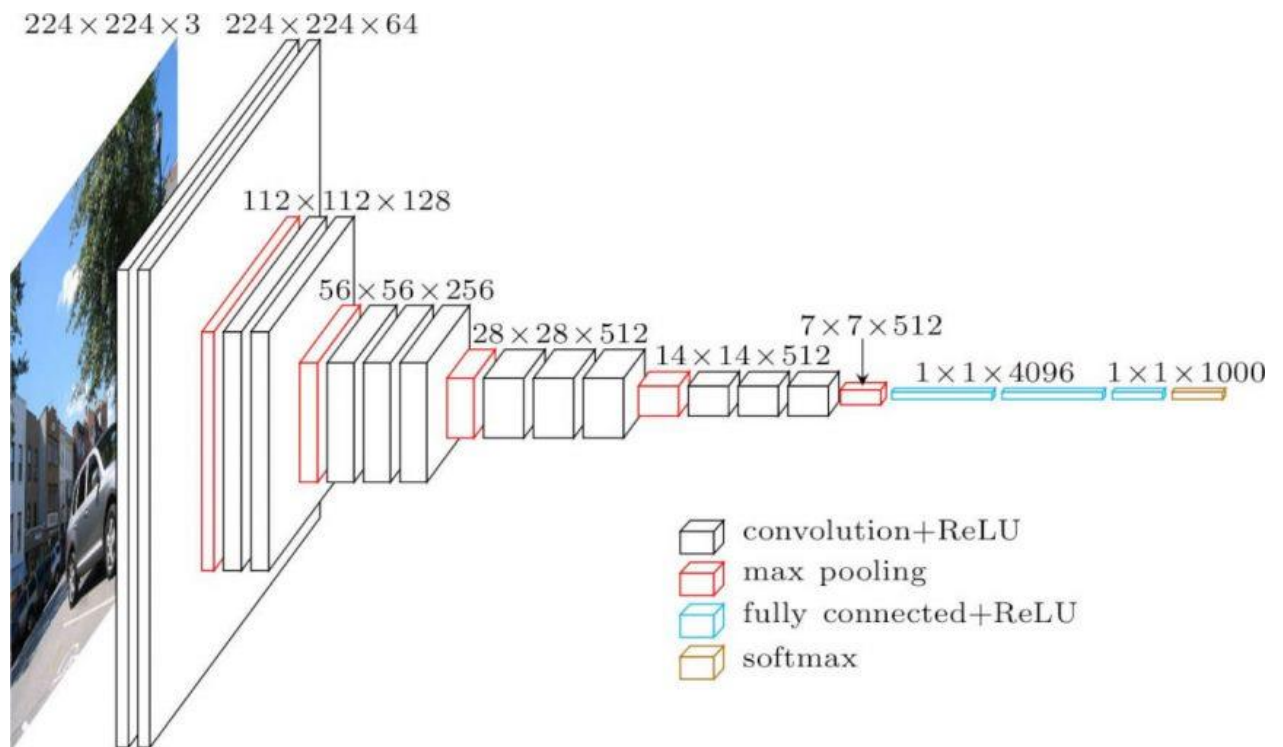
Để phân loại ảnh bằng Inception, ta sẽ sử dụng một mô hình đã được huấn luyện trước trên một tập dữ liệu lớn, chẳng hạn như ImageNet. Mô hình sẽ học từ hàng triệu ảnh và các nhãn tương ứng để hiểu và phân loại các đối tượng trong ảnh.

Sau khi huấn luyện, mô hình Inception sẽ có khả năng nhận diện và phân loại các đối tượng trong ảnh mới dựa trên những kiến thức đã học được từ tập dữ liệu huấn luyện. Việc phân loại ảnh bằng Inception thường được thực hiện bằng cách đưa ảnh vào mô hình và sử dụng đầu ra của mạng để xác định xem ảnh thuộc vào các lớp (nhãn) nào.

### 2.2.3 VGG16

VGG16 là mạng CNN được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford. Mẫu sau khi huấn luyện bởi mạng VGG 16 đạt độ chính xác 92.7% – top 5 test trong dữ liệu ImageNet (<http://www.image-net.org/>) gồm 14

triệu hình ảnh thuộc 1000 lớp khác nhau. Giờ ta áp dụng kiến thức đã nêu trước đó để phân tích mạng VGG16.



Hình 15. Minh họa kiến trúc của VGG16

Phân tích về VGG16:

- Convolutional layer: kích thước  $3 \times 3$ , padding=1, stride=1. Mặc định stride=1 và padding để cho output cùng width và height với input.
- Pool/2 : max pooling layer với size  $2 \times 2$
- $3 \times 3$  conv, 64: thì 64 là số kernel áp dụng trong layer này, hay depth của output của layer này.
- Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
- Sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer.

## CHƯƠNG 3: ỨNG DỤNG TENSORFLOW VÀ MẠNG VGG16 VÀO PHÂN LOẠI ẢNH

### 3.1 Phân tích cách áp dụng Tensorflow và mạng VGG16

1. Chuẩn bị dữ liệu huấn luyện: Cần có một tập dữ liệu huấn luyện có nhãn, trong đó mỗi ảnh được gán nhãn với lớp tương ứng. Tập dữ liệu này được chia thành hai phần: tập huấn luyện (training set) và tập kiểm tra (test set).

2. Tiền xử lý dữ liệu: Trước khi đưa dữ liệu vào mạng VGG16, thường cần tiền xử lý để đảm bảo đầu vào đúng định dạng và phù hợp với mạng. Điều này bao gồm thay đổi kích thước ảnh về đúng kích thước mà VGG16 yêu cầu (thông thường là 224x224 pixel), chuẩn hóa giá trị pixel về khoảng  $[0, 1]$  hoặc  $[-1, 1]$ , và chuẩn bị dữ liệu theo định dạng phù hợp.

3. Tạo mô hình VGG16: Tiếp theo, ta cần tạo mô hình VGG16 bằng cách khởi tạo mạng và tải trọng số được huấn luyện trước (pretrained weights). Trọng số huấn luyện trước có thể được tải từ các nguồn như ImageNet.

4. Đào tạo mô hình: Với mô hình VGG16 đã được khởi tạo và trọng số huấn luyện trước, ta có thể tiến hành đào tạo mô hình trên tập dữ liệu huấn luyện. Quá trình đào tạo này sẽ điều chỉnh các trọng số của mạng để tối thiểu hóa sai số giữa đầu ra dự đoán và nhãn thực tế. Để đạt được điều này, ta cần xác định hàm mất mát (loss function) và thuật toán tối ưu (optimizer) để cập nhật trọng số.

5. Đánh giá mô hình: Sau khi mô hình đã được đào tạo, ta cần đánh giá hiệu suất của nó trên tập kiểm tra để đánh giá khả năng phân loại ảnh. Điều này thường bao gồm tính toán độ chính xác (accuracy) hoặc các độ đo khác như độ chính xác cân bằng (balanced accuracy), độ chính xác theo lớp (class-wise accuracy) và ma trận nhầm lẫn (confusion matrix).

6. Dự đoán lớp ảnh mới: Khi mô hình đã được huấn luyện và đánh giá, ta có thể sử dụng nó để dự đoán lớp của các ảnh mới. Đầu vào ảnh mới sẽ được đưa

qua mạng VGG16, và đầu ra sẽ là xác suất của các lớp phân loại. Lớp với xác suất cao nhất hoặc xác suất vượt ngưỡng nào đó sẽ được chọn là lớp dự đoán của ảnh.

### **3.2 Phân tích bài toán**

#### **3.2.1 Phát biểu bài toán**

Hiện nay, hầu hết điện thoại của mọi người chứa rất nhiều loại ảnh khác nhau, khó để tìm kiếm, sắp xếp. Trong bài toán này, ta dùng mô hình VGG16 để phân loại ảnh vào đúng lớp.

#### **3.2.2 Dữ liệu đầu vào**

Dữ liệu đầu vào cho bài toán phân loại ảnh là một thư mục chứa các hình ảnh thuộc nhiều phân loại ảnh khác nhau, ví dụ: ảnh mèo, ảnh chân dung, ảnh chụp màn hình...

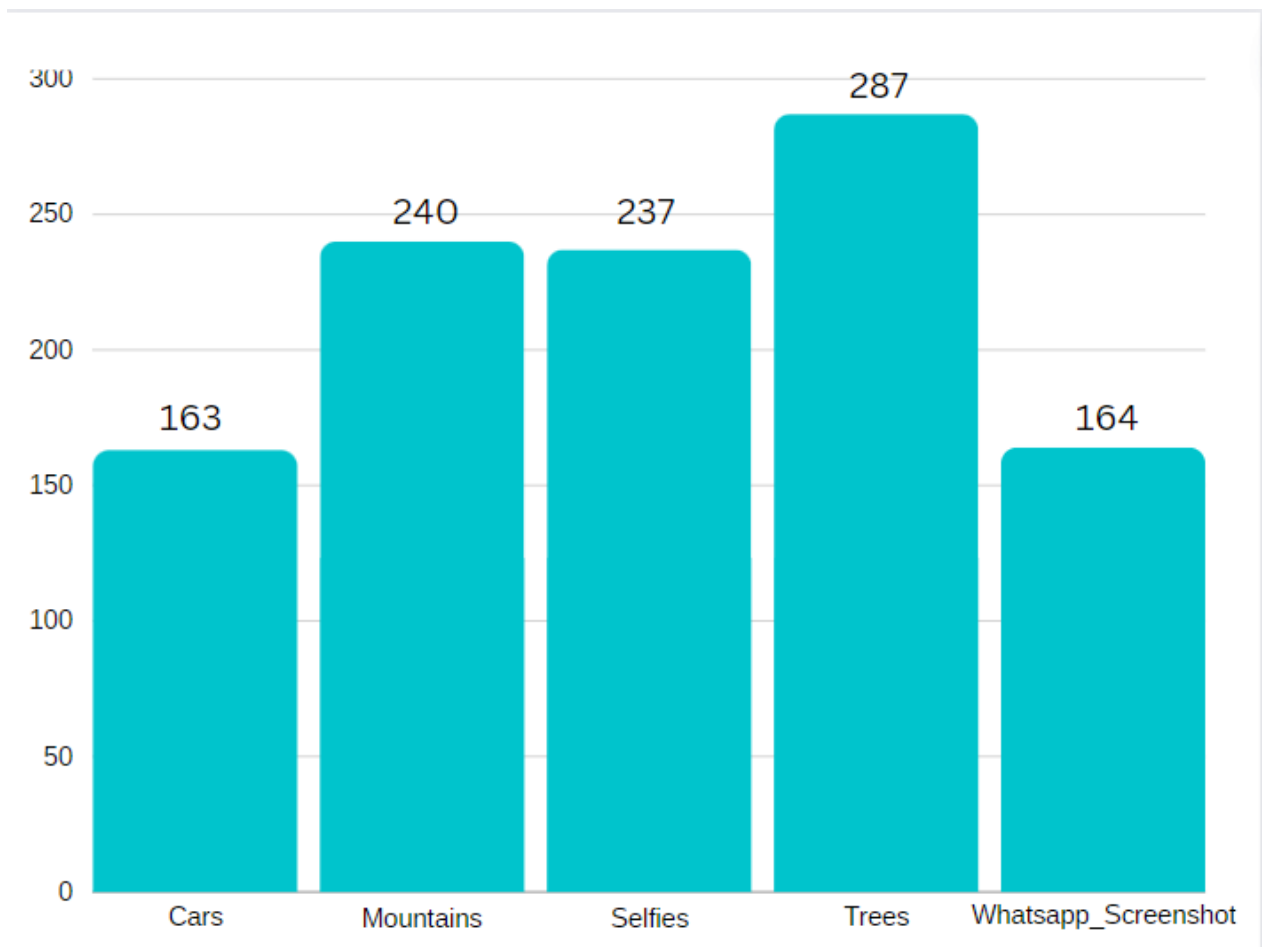
#### **3.2.3 Dữ liệu đầu ra**

Các thư mục chứa các ảnh cùng 1 loại.

### **3.3 Các bước thực hiện**

#### **3.3.1 Thu thập dữ liệu**

Nhóm em đã đi thu thập dữ liệu là các ảnh thuộc các lớp phân loại, sau đó đánh nhãn cho các ảnh.



*Hình 16. Tổng quan về dữ liệu*

Tổng cộng nhóm thu thập 1.200 ảnh, sau khi đánh nhãn và loại bỏ các ảnh không hợp lệ, nhóm còn lại 1.091 ảnh trong đó 163 ảnh ô tô, 240 ảnh núi, 237 ảnh chân dung, 287 ảnh cây cối, 164 ảnh chụp màn hình.

Sau đó nhóm đã chia làm 3 tập:

- Tập train gồm 1.000 ảnh được chia làm 5 thư mục mỗi thư mục là 1 loại ảnh: Có mục đích huấn luyện mẫu và cho mẫu học.
- Tập test: Tập này chiếm gần 10% ảnh (109 ảnh) để giúp mẫu kiểm tra trong lúc huấn luyện.
- Tập album: có mục đích thử mẫu sau khi huấn luyện xong.



### 3.3.2 Xây dựng chương trình

#### 3.3.2.1 Tiền xử lý dữ liệu

Sau khi chia dữ liệu vào các tập đã trình bày như trên. Nhóm em tiến hành xử lý những dữ liệu này.

- Chuyển tất cả dữ liệu về kích thước (224, 224, 3) để huấn luyện mẫu.
- Tiếp đó là chuẩn hóa đầu ra thành dạng vector có kích thước là 5. (Tương ứng với 5 kết quả cần thu về là 5 loại ảnh)

#### 3.3.2.2 Xây dựng mô hình

Tiến hành code model theo kiến trúc VGG16 sử dụng thư viện tensorflow – framework Keras trên python.

```
#Thư viện  
import tensorflow as tf  
from keras.preprocessing.image import ImageDataGenerator  
✓ 49.9s
```

*Hình 17. Lấy dữ liệu*

```

> ~
#Sử dụng VGG16
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model, Sequential
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input

[15]

vgg16 = VGG16(input_shape=INPUT_SIZE + [3], weights='imagenet', include_top=False)

[17]

for layer in vgg16.layers:
    layer.trainable = False

[10]

flatten = Flatten()(vgg16.output)
dense = Dense(units=256, activation='relu')(flatten)
dense_output = Dense(units=len(label_order), activation='softmax')(dense)

[11]

model = Model(inputs=vgg16.input, outputs=dense_output)

[12]

```

Hình 18. Chuẩn bị kiến trúc VGG16

```

model.compile(
    loss='categorical_crossentropy',
    optimizer='rmsprop',
    metrics=['accuracy']
)

[13]

> ~
r = model.fit(
    training_data,
    validation_data=test_data,
    epochs=10,
    steps_per_epoch=len(training_data),
    validation_steps=len(test_data)
)

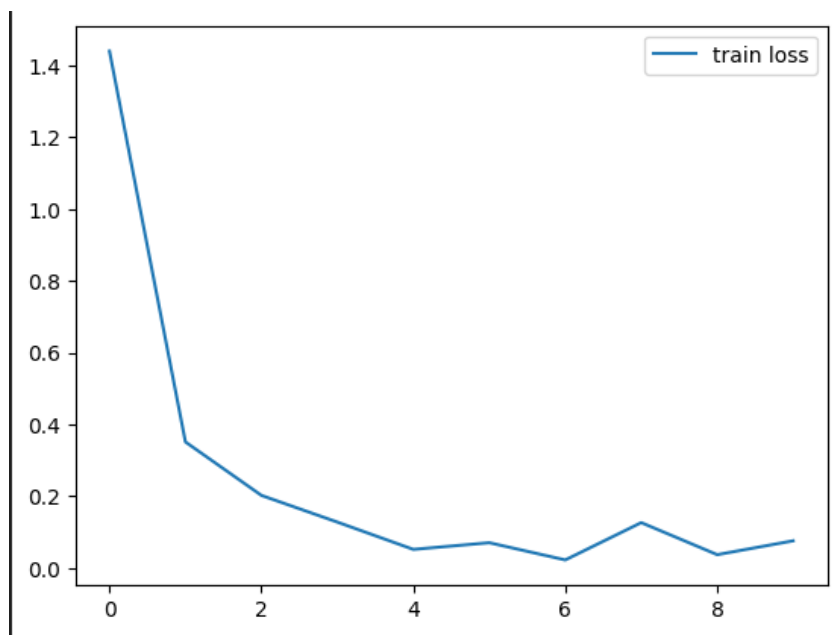
[14]

... Epoch 1/10
61/61 [=====] - 110s 2s/step - loss: 1.4400 - accuracy: 0.8027 - val_loss: 0.2188 - val_accuracy: 0.9375
Epoch 2/10
61/61 [=====] - 105s 2s/step - loss: 0.3514 - accuracy: 0.9148 - val_loss: 0.2082 - val_accuracy: 0.9609
Epoch 3/10
61/61 [=====] - 99s 2s/step - loss: 0.2025 - accuracy: 0.9512 - val_loss: 0.1428 - val_accuracy: 0.9531
Epoch 4/10
61/61 [=====] - 97s 2s/step - loss: 0.1281 - accuracy: 0.9605 - val_loss: 0.3120 - val_accuracy: 0.9453
Epoch 5/10
61/61 [=====] - 96s 2s/step - loss: 0.0521 - accuracy: 0.9865 - val_loss: 0.8361 - val_accuracy: 0.8281
Epoch 6/10
61/61 [=====] - 89s 1s/step - loss: 0.0707 - accuracy: 0.9834 - val_loss: 0.1557 - val_accuracy: 0.9531
Epoch 7/10
61/61 [=====] - 92s 2s/step - loss: 0.0227 - accuracy: 0.9938 - val_loss: 0.7605 - val_accuracy: 0.8516
Epoch 8/10
61/61 [=====] - 94s 2s/step - loss: 0.1266 - accuracy: 0.9730 - val_loss: 0.2725 - val_accuracy: 0.9609
Epoch 9/10
61/61 [=====] - 93s 2s/step - loss: 0.0374 - accuracy: 0.9938 - val_loss: 0.2441 - val_accuracy: 0.9531
Epoch 10/10
61/61 [=====] - 92s 2s/step - loss: 0.0760 - accuracy: 0.9823 - val_loss: 0.1880 - val_accuracy: 0.9609

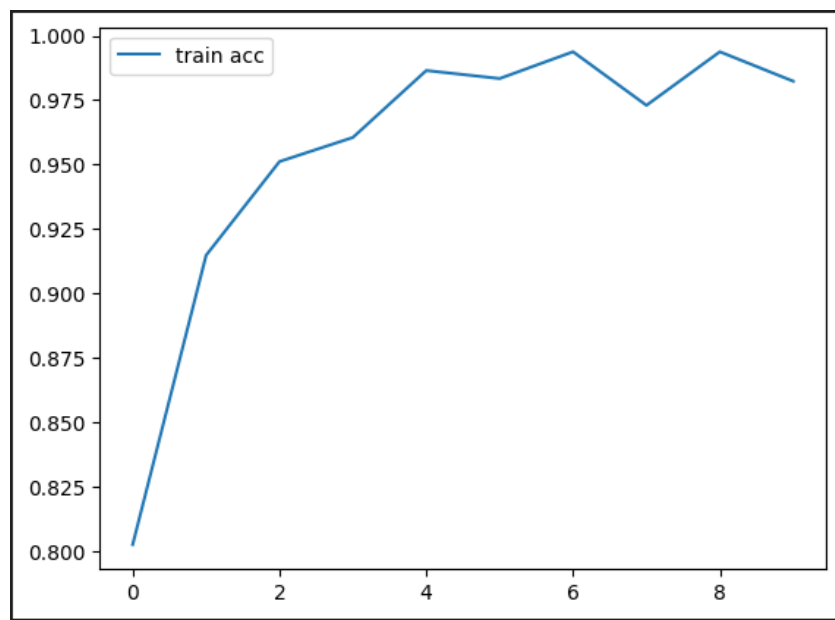
```

Hình 19. Tiến hành huấn luyện mô hình

Sau quá trình huấn luyện với Epoch = 10, nhóm em đã thu về được kết quả là model đã được huấn luyện với độ chính xác khá cao (val\_accuracy = 0.9609)



Hình 20. Đồ thị loss trong quá trình huấn luyện



Hình 21. Đồ thị accuracy trong quá trình huấn luyện

### 3.3.2.3 Kiểm tra, đánh giá hiệu quả của mô hình

Từ các thông số thu được, nhóm em đưa ra một số nhận xét. Model học tốt qua tập dữ liệu, không bị hiện tượng quá khớp (overfitting)..

Thử lại ngẫu nhiên với tập album thu được kết quả khá chính xác.

```
1/1 [=====] - 0s 126ms/step
Cars-1.jpg => Chính xác
1/1 [=====] - 0s 140ms/step
Cars-2.jpg => Chính xác
1/1 [=====] - 0s 130ms/step
Cars-3.jpg => Chính xác
1/1 [=====] - 0s 125ms/step
Cars-4.jpg => Chính xác
1/1 [=====] - 0s 136ms/step
Cars-5.jpg => Chính xác
1/1 [=====] - 0s 142ms/step
Mountains-1.jpg => Chính xác
1/1 [=====] - 0s 135ms/step
Mountains-2.jpg => Chính xác
1/1 [=====] - 0s 156ms/step
Mountains-3.jpg => Chính xác
1/1 [=====] - 0s 119ms/step
Mountains-4.jpg => Chính xác
1/1 [=====] - 0s 131ms/step
Mountains-5.jpg => Chính xác
1/1 [=====] - 0s 131ms/step
Mountains-6.jpg => Chính xác
1/1 [=====] - 0s 122ms/step
Mountains-7.jpg => Chính xác
1/1 [=====] - 0s 115ms/step
...
1/1 [=====] - 0s 118ms/step
Whatsapp_Screenshots-8.jpg => Chính xác
1/1 [=====] - 0s 123ms/step
Whatsapp_Screenshots-9.jpg => Chính xác
```

Hình 22. Thử lại với 31 ảnh ngẫu nhiên trong tập album

Hình cho ra kết quả không chính xác là hình ảnh có chất lượng không tốt, hoặc chứa 2 đối tượng dễ nhầm lẫn



*Hình 23. Hình thu về kết quả không chính xác*

## KẾT LUẬN

Trong đề tài này, nhóm chúng em đã nghiên cứu và thực hiện việc áp dụng TensorFlow, một thư viện học máy phổ biến, cùng với mạng VGG16 để xây dựng một hệ thống phân loại ảnh. Mục tiêu của đề tài là tạo ra một mô hình có khả năng nhận diện và phân loại các loại ảnh khác nhau.

Chúng em đã thực hiện các bước tìm hiểu về mạng VGG16, nghiên cứu kiến trúc của mạng VGG16, bao gồm các lớp convolutional, pooling và fully connected, hiểu cách mạng VGG16 hoạt động và cách nó có thể được sử dụng trong phân loại ảnh. Bước đầu, nhóm em đã thu thập và chuẩn bị dữ liệu ảnh để huấn luyện và kiểm tra mô hình. Tiếp đến, nhóm em tiến hành xây dựng mô hình, sử dụng TensorFlow để xây dựng mô hình phân loại ảnh dựa trên kiến trúc của mạng VGG16, tạo và cấu hình các lớp convolutional, pooling và fully connected tương ứng với kiến trúc VGG16 và thực hiện quá trình huấn luyện để mô hình có khả năng phân loại các loại ảnh. Cuối cùng, chúng em huấn luyện và đánh giá mô hình, sử dụng tập dữ liệu huấn luyện để huấn luyện mô hình và sử dụng tập dữ liệu kiểm tra để đánh giá hiệu suất của mô hình.

Tuy nhiên, hiện tại nhóm chúng em vẫn chưa thực hiện được một số công việc nhất định. Nhóm em vẫn chưa thực hiện được tối ưu hóa mô hình và chưa triển khai ứng dụng thực tế. Hiện tại, mô hình chỉ mới được huấn luyện và đánh giá trên tập dữ liệu nhất định. Tuy nhiên, để ứng dụng trong thực tế, chúng em cần triển khai mô hình vào một hệ thống thực tế, như một ứng dụng web hoặc di động, để có thể phân loại ảnh trong thời gian thực.

Trong tương lai, chúng em sẽ tiếp tục làm việc để hoàn thiện đề tài này. Các công việc tiếp theo mà chúng em dự định thực hiện đầu tiên là tối ưu hóa mô hình bằng cách nghiên cứu và áp dụng các kỹ thuật tối ưu hóa để cải thiện hiệu suất của mô hình, thử nghiệm các kiến trúc mạng khác nhau hoặc sử dụng kỹ

thuật transfer learning để sử dụng mô hình đã được huấn luyện trước đó. Thứ hai, chúng em sẽ mở rộng tập dữ liệu huấn luyện và kiểm tra để đảm bảo mô hình có khả năng phân loại chính xác trên nhiều loại ảnh khác nhau. Bên cạnh đó, chúng em còn dự định sẽ triển khai ứng dụng thực tế để có thể phân loại ảnh trong thời gian thực.

Với những công việc tiếp theo này, chúng em hy vọng rằng đề tài của mình sẽ đóng góp vào việc ứng dụng TensorFlow và mạng VGG16 vào phân loại ảnh một cách hiệu quả và có thể áp dụng trong các lĩnh vực thực tế.

## TÀI LIỆU THAM KHẢO

- [1] Trần Hùng Cường, Nguyễn Phương Nga, Giáo trình Trí tuệ nhân tạo, 2014.
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Microsoft Research, *Deep Residual Learning for Image Recognition*, 2015.
- [3] Karen Simonyan, Andrew Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, v6, 2015.
- [4] Bách khoa toàn thư Việt Nam, *Học sâu*.
- [5] Chi-Feng Wang, *The Vanishing Gradient Problem*, 2019.
- [6] Martin Krasser, *Deep face recognition with Keras, Dlib and OpenCV*, 2018.
- [7] Iván de Paz Centeno, *MTCNN face detection implementation for TensorFlow, as a PIP package*, 3th Edition, 2021.
- [8] Adrian Rosebrock, *Face detection with OpenCV and deep learning*, 2018.
- [9] Zhilu Zhang, Mert R. Sabuncu, *Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels*, v4, 2018