**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**--------\*--------**



**TIỂU LUẬN CHUYÊN NGÀNH**

**Đề tài: Tìm hiểu Neural Network: ANN, RNN, CNN và một số ứng dụng tiêu biểu dùng Neural Network**

**GVHD: PhD.Trần Lê Minh Sang**

**SVTH :**

**Nguyễn Xuân Hùng 15110059**

**Tôn Nữ Minh Châu 15110019**

**TP Hồ Chí Minh, ngày 12 tháng 12 năm 2018**

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

GIẢNG VIÊN HƯỚNG DẪN

Ký tên

# NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

GIẢNG VIÊN PHẢN BIỆN

Ký tên

# LỜI CẢM ƠN

Trong quá trình nghiên cứu, nhóm chúng em đã nhận được sự giúp đỡ rất nhiệt tình của Cô Trần Lê Minh Sang. Để hoàn thành tốt việc nghiên cứu tiểu luận chuyên ngành Cô đã giúp nhóm chúng em rất nhiều bằng việc cung cấp tài liệu, giải quyết những khúc mác và tận tình động viên nhóm chúng em trong suốt thời gian nghiên cứu tiểu luận chuyên ngành.

Đồng thời, nhóm chúng em cũng cảm ơn Thầy Nguyễn Đăng Quang đã sắp xếp để sinh viên chúng em có cơ hội tiếp xúc thực hiện đề tài mới.

Nhóm em xin chân thành cảm ơn !

TP.Hồ Chí Minh, ngày 12 tháng 12 năm 2018

# LỜI NÓI ĐẦU

Công nghệ càng lúc càng phát triển là nhờ có sự ứng dụng AI vào các sản phẩm. Các công ty thời nay không chỉ ứng dụng AI vào lĩnh vực CNTT, mà còn nhiều lĩnh vực khác trong đời sống như: y tế, công nghiệp, nông nghiệp, giáo dục và giao thông. Có thể thấy AI rất quan trọng đối với mọi lĩnh vực, vì vậy việc sinh viên ngành CNTT ngoài các kiến thức về phần mềm thì việc có kiến thức về AI cũng rất cần thiết.

Lĩnh vực AI thì rất rộng và không thể học hết được trong thời gian ngắn. Vì vậy nhóm em đã dành thời gian để tìm lĩnh vực để tìm hiểu, và cuối cùng nhóm tìm ra đề tài để tìm hiểu là "Neural Network".

# DANH MỤC HÌNH

[Hình 2‑1: Quan hệ giữa AI - Machine Learning - Deep Learning 9](#_Toc532389033)

[Hình 2‑2: Đồ thị của Sigmoid Function 12](#_Toc532389034)

[Hình 2‑3: Đồ thị của ReLU Function 13](#_Toc532389035)

[Hình 2‑4: Trực quan hóa việc chọn Learning Rate quá cao hoặc quá thấp 16](#_Toc532389036)

[Hình 2‑5: Mô hình phi tuyến tính của một neuron, có nhãn k 17](#_Toc532389037)

[Hình 2‑6: Kiến trúc của mạng ANN 18](#_Toc532389038)

[Hình 2‑7: Mô hình thể hiện quá trình học của ANN 19](#_Toc532389039)

[Hình 2‑8: Kiến trúc của mạng CNN 21](#_Toc532389040)

[Hình 2‑9: Input và Kernel để thực hiện phép toán Convolution 22](#_Toc532389041)

[Hình 2‑10: Phép nhân ma trận giữa Input và Filter (1) 23](#_Toc532389042)

[Hình 2‑11: Phép nhân ma trận giữa Input và Filter (2) 23](#_Toc532389043)

[Hình 2‑12: Ví dụ về Input 3D 24](#_Toc532389044)

[Hình 2‑13:Biễu diễn Stride=1 (1) 26](#_Toc532389045)

[Hình 2‑14: Biễu diễn Stride=1 (2) 26](#_Toc532389046)

[Hình 2‑15: Biễu diễn Stride=1 (3) 27](#_Toc532389047)

[Hình 2‑16 :Biểu diễn Stride = 2 (1) 27](#_Toc532389048)

[Hình 2‑17: Biểu diễn Stride = 2 (2) 28](#_Toc532389049)

[Hình 2‑18: Biểu diễn Stride = 2 (3) 28](#_Toc532389050)

[Hình 2‑19: Biểu diễn Stride = 2 (4) 28](#_Toc532389051)

[Hình 2‑20: Áp dụng Padding cho Input 29](#_Toc532389052)

[Hình 2‑21: Mạng RNN cơ bản 31](#_Toc532389053)

[Hình 2‑22: Một đoạn phổ quang âm thanh 31](#_Toc532389054)

[Hình 2‑23: Kiến trúc của mạng RNN 33](#_Toc532389055)

[Hình 2‑24: Mạng RNN cơ bản sau khi được "Unfolded" 33](#_Toc532389056)

[Hình 2‑25: Hình minh họa cho BPTT 35](#_Toc532389057)

# MỤC LỤC

[NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN 2](#_Toc532389106)

[NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN 3](#_Toc532389107)

[LỜI CẢM ƠN 4](#_Toc532389108)

[LỜI NÓI ĐẦU 5](#_Toc532389109)

[DANH MỤC HÌNH 6](#_Toc532389110)

[MỤC LỤC 7](#_Toc532389111)

[1. GIỚI THIỆU 9](#_Toc532389112)

[1.1. Mục tiêu 9](#_Toc532389113)

[1.2. Lý do 9](#_Toc532389114)

[1.3. Phạm vi 9](#_Toc532389115)

[2. NỘI DUNG NGHIÊN CỨU 10](#_Toc532389116)

[2.1. Giới thiệu Machine Leaning - Deep Learning 10](#_Toc532389117)

[2.2. Khái quát về Neural Network 11](#_Toc532389118)

[2.2.1. Khái niệm 11](#_Toc532389119)

[2.2.2. Lịch sử phát triển 11](#_Toc532389120)

[2.3. Lý thuyết liên quan 12](#_Toc532389121)

[2.3.1. Activation Funciton 12](#_Toc532389122)

[2.3.2. Cost Funtion 15](#_Toc532389123)

[2.3.3. Optimization (Tối ưu hóa) 15](#_Toc532389124)

[2.4. Artificial neural network (ANN) 17](#_Toc532389125)

[2.4.1. Định nghĩa 17](#_Toc532389126)

[2.4.2. Kiến trúc 19](#_Toc532389127)

[2.4.3. Cách hoạt động 20](#_Toc532389128)

[2.4.4. Ứng dụng 21](#_Toc532389129)

[2.5. Convolutional neural network (CNN) 22](#_Toc532389130)

[2.5.1. Định nghĩa 22](#_Toc532389131)

[2.5.2. Kiến trúc 22](#_Toc532389132)

[2.5.3. Cách hoạt động 23](#_Toc532389133)

[2.5.4. Ứng dụng 31](#_Toc532389134)

[2.6. Recurrent neural network (RNN) 31](#_Toc532389135)

[2.6.1. Định nghĩa 31](#_Toc532389136)

[2.6.2. Kiến trúc 34](#_Toc532389137)

[2.6.3. Cách hoạt động 34](#_Toc532389138)

[2.6.4. Ứng dụng 37](#_Toc532389139)

[2.7. So sánh 38](#_Toc532389140)

[3. KẾT LUẬN 39](#_Toc532389141)

[TÀI LIỆU THAM KHẢO 40](#_Toc532389142)

# GIỚI THIỆU

## Mục tiêu

Có cái nhìn tổng quan về AI, Machine Learning và đặc biệt là Deep Learning.

## Lý do

Neural Network cung cấp nhiều giải pháp tốt nhất cho các bài toán như Nhận diện hình ảnh (Image Recognition), nhận diện lời nói (Speech Recognition) và Xử lý ngôn ngữ tự nhiên (Natural Language Processing).

## Phạm vi

Tìm hiểu về định nghĩa và cách hoạt động của

* Artificial Neural Network.
* Convolutional Neural Network.
* Recurrent Neural Network.

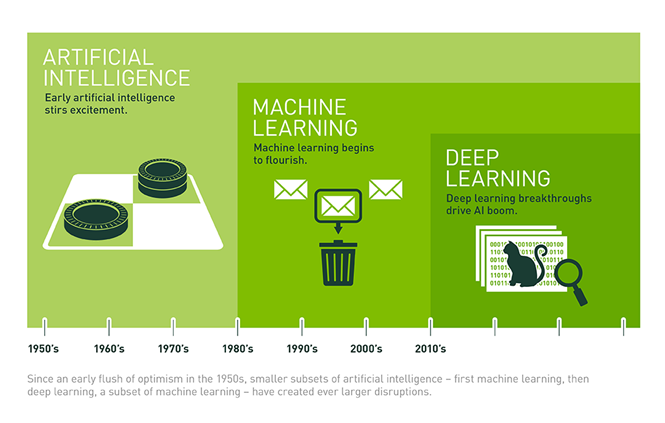
# NỘI DUNG NGHIÊN CỨU

## Giới thiệu Machine Leaning - Deep Learning

Machine Learning là một tập con của AI (Artificial Intelligence - Trí tuệ nhân tạo). Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.

Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, Machine Learning đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi lại Deep Learning.

Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hoặc âm nhạc.



Hình 2‑1: Quan hệ giữa AI - Machine Learning - Deep Learning

## Khái quát về Neural Network

### Khái niệm

Mạng Neural là mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thống thần kinh của sinh vật, bao gồm số lượng lớn các Neuron được gắn kết với nhau để xử lý thông tin.

Mạng Neural giống như bộ não con người, được học bởi kinh nghiệm (thông qua việc huấn luyện), có khả năng lưu trữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết

Neutral Network được phát triển dựa trên nền tảng Perceptron Learning Algorithm (PLA) hay còn gọi là Perceptron.

### Lịch sử phát triển

Vào năm 1943, nhà thần kinh học Warren McCulloch và nhà toán học Walter Pitts đã viết tài liệu về cách thức hoạt động của các neuron. Để diễn tả các hoạt động, làm việc của các neuron trong bộ não, Warren và Walter đã xây dựng mô hình Neural Network đơn giản bằng cách sử dụng mạch điện.

Vào năm 1949, Donald Hebb đã viết ra cuốn sách “The Organization of Behavior”. Cuốn sách này đã chỉ ra sự thật rằng mỗi khi các neuron được sử dụng thì các đường kết nối các neuron với nhau sẽ ngày “dày” hơn, quá trình làm “dày” hơn này xảy ra nhờ việc con người mỗi ngày đều học hỏi nhiều thứ.

Vào năm 1950, việc máy tính trở nên ngày càng phổ biến đã là điều kiện tốt cho việc xây dựng một mạng Neural Network mô phỏng. Nathanial Rochester đã là người đầu tiên sử dụng máy tính để xây dựng mạng Neural Network nhưng thất bại.

Vào năm 1959, Bernard Widrow và Marcian Hoff của đại học Stanford đã phát triển hai mô hình được gọi là “ADALINE” và “MADALINE”.

🡪 “ADALINE” được phát triển để đọc các bit từ đường truyền của điện thoại và dự đoán được bit tiếp theo xuất hiện.

🡪 “MADALINE” là mạng Neural Network đầu tiên mà được áp dụng vào các vấn đề thực tiễn, sử dụng bộ lọc thích hợp để loại bỏ các tiếng vọng trong đường truyền của điện thoại.

Vào năm 1975, mạng Neural Network nhiều Layer (Multilayer Neural Network) đầu tiên được phát triển.

## Lý thuyết liên quan

### Activation Funciton

Trong neural networks, activation function của một neuron xác định output của neuron đó. Output này sau đó được sử dụng làm input cho neuron tiếp theo và cứ thế cho đến khi tìm thấy giải pháp mong muốn cho vấn đề ban đầu.

Nó ánh xạ các giá trị kết quả vào phạm vi mong muốn, chẳng hạn như từ 0 đến 1 hoặc -1 đến 1, …, tùy thuộc vào sự lựa chọn của activation function.

Dưới đây là một số activation function được sử dụng phổ biến trong neural network.

#### Sigmoid

Lý do chính tại sao chúng ta sử dụng hàm sigmoid là vì nó có phạm vi giá trị liên tục từ 0 đến 1. Do đó, nó đặc biệt được sử dụng cho các mô hình mà phải dự đoán xác suất. Vì vậy xác suất của bất cứ điều gì chỉ tồn tại giữa phạm vi 0 và 1. Hàm Sigmoid có dạng như sau:



Hình 2‑2: Đồ thị của Sigmoid Function

#### ReLU (Rectified Linear Unit)

ReLU là activation function được sử dụng nhiều nhất trên thế giới hiện nay, nó được sử dụng trong Convolutional Neural Network (CNN) hoặc Deep Learning. Hàm ReLU có dạng như sau:



Hình 2‑3: Đồ thị của ReLU Function

#### Softmax

Hàm Sigmoid chỉ có thể xử lý hai lớp, vì vậy hàm Sigmoid không thể áp dụng cho các bài toán mà có nhiều hơn hai lớp. Để có thể thao tác với nhiều hơn hai lớp thì hàm Softmax là một lựa chọn tốt vì nó tính xác suất của các class, và các xuất đó có tổng bằng 1. Hàm Softmax có dạng như sau:

với i = 0, 1, 2, …, k.

### Cost Funtion

Trong tất Machine Learning nói chung, Neural Network nói riêng, cost function là hàm mất mát, là độ sai lệch. Mục tiêu của cost function là để tính độ sai lệch của thuật toán bằng cách lấy trung bình sai số giữa các kết quả dự đoán được và kết quả thực tế.

Công thức của cost function có dạng như sau:

Chú thích:

* : Đầu ra dự đoán sử dụng thuật toán.
* : Đầu ra thực tế
* m : Số lượng điểm dữ liệu

### Optimization (Tối ưu hóa)

#### Giới thiệu

Việc tối ưu hàm mất mát là đi tìm giá trị của tham số sao cho hàm đạt giá trị nhỏ nhất, nói cách khác là đi tìm Global minimum. Nhưng việc giải phương trình đạo hàm bằng 0 để tìm nghiệm là bất khả thi. Do đó cần một phương pháp tổng quát để tính được xấp xỉ giá trị của mà tại đó đạt giá trị nhỏ nhất.

Gradient Descent và các biến thể của nó là một trong những phương pháp được dùng nhiều trong Machine Learning và Deep Learning.

#### Gradient Descent

##### Giới thiệu

Gradient được sử dụng trong quá trình train mô hình. Là một thuật toán tối ưu hóa 🡪 Minimize một function tới điểm local minimum của nó.

🡺 Hiểu đơn giản là thuật toán tìm các giá trị của trọng số (Weights) sao cho hàm mất mát càng nhỏ càng tốt.

##### Định nghĩa

Một gradient ước lượng output của một hàm thay đổi như thế nào khi thay đổi các input. Gradient đơn giản là ước lượng sự thay đổi của gia trị sai số khi tất cả trọng số thay đổi.

##### Cách hoạt động của Gradient

Gradient Descent có thể được xem là một người leo núi đi xuống đáy của thung lũng vì Gradient Descent là một thuật toán giảm thiểu

Công thức của Gradient Descent như sau (Lấy ví dụ người leo núi):

b = - η

Chú thích:

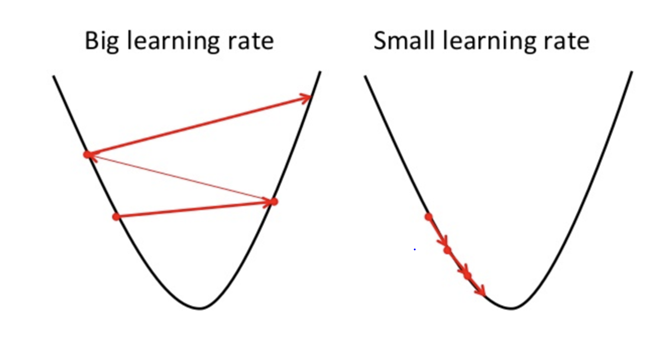
* + b: Vị trí tiếp theo của người leo núi.
  + : Vị trí hiện tại của người leo núi.
  + η: Learning rate (Tốc độ di chuyển của người leo núi khi xuống đáy thung lũng)
  + : Đạo hàm của hàm .

🡺 Công thức trên đơn giản là biểu diễn vị trí tiếp theo mà người leo núi cần phải đi xuống theo độ dốc.

##### Mức độ quan trọng của Learning Rate

Learning Rate được dùng để xác định mức độ nhanh hay chậm để di chuyển tới trọng số tối ưu.

Việc chọn giá trị cho Learning Rate cũng rất quan trọng, nếu chọn Learning Rate quá lớn thì sẽ không hội tụ được tại điểm local minimum. Nhưng nếu chọn Learning Rate quá thấp thì thời gian hội tụ sẽ rất lâu.



Hình 2‑4: Trực quan hóa việc chọn Learning Rate quá cao hoặc quá thấp

## Artificial neural network (ANN)

### Định nghĩa

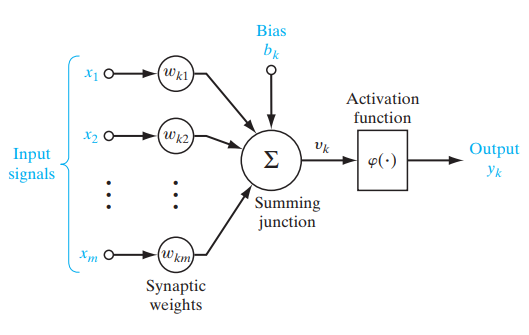
#### Artificial Neutral Network

Artificial Neural Networks là các mô phỏng lấy cảm hứng từ sinh học được thực hiện trên máy tính để thực hiện các nhiệm vụ cụ thể.

#### Neurons

Neuron là một đơn vị xử lý thông tin cơ bản cho hoạt động của neural network.

Một neuron trong bất kỳ layer nào trong network đều được kết nối với tất cả các neuron (node) trong layer trước đó – fully connected.



Hình 2‑5: Mô hình phi tuyến tính của một neuron, có nhãn k

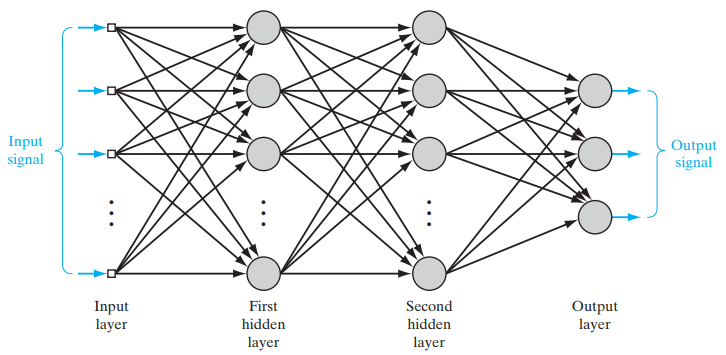
Phương trình viết lại từ mô hình

và

Với

* : input
* : synaptic weight tương ứng của neuron k
* : bias
* : activation function
* : output

### Kiến trúc

Neural network có cấu trúc Multi-layer Perceptrons. Perceptron có thể hiểu tương đương single-layer neural network 

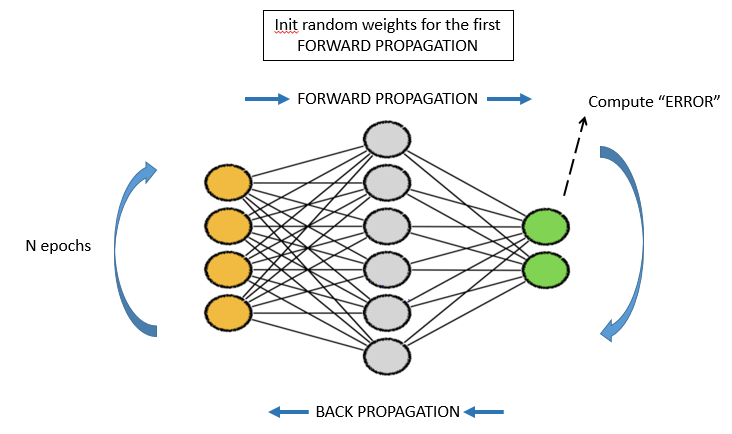
Hình 2‑6: Kiến trúc của mạng ANN

Kiến trúc của mạng ANN bao gồm:

* 1 Input Layer.
* Nhiều Hidden Layer.
* 1 Output Layer.

Mỗi node trong một layer được kết nối với từng node một ở một layer khác. Ta làm mạng “sâu hơn” (deeper) nhờ vào việc tăng số lượng hidden layer.

### Cách hoạt động



Hình 2‑7: Mô hình thể hiện quá trình học của ANN

Quá trình học của nerural network trải qua 2 giai đoạn lớn:

* Forward-Propagation: có nhiệm vụ lấy các đặc tính từ training data, tính toán để tạo model thực hiện nhiệm vụ cụ thể.
* Backward-Propagation: sau khi so sánh sai số với giá trị mong đợi, giai đoạn này thực hiện nhiệm vụ hiệu chỉnh trọng số để được model hiệu suất cao

#### Forward-Propagation

Bước 1: Khởi tạo mạng Neural Network và random các trọng số (Weights).

Bước 2: Đưa toàn bộ dữ liệu training vào Neural Network. Input Layer của Neural Network sẽ nhận các giá trị đầu vào từ bộ dữ liệu training.

Bước 3: Luồng hoạt động của Forward-Propagation là từ trái sang phải (Từ Input Layer cho tới Output Layer). Sau khi đưa dữ liệu đầu vào vào, Neural Network sẽ thực hiện các phép toán TUYẾN TÍNH / PHI TUYẾN TÍNH để cho ra output của output layer.

Bước 4: Sau khi có output của output layer rồi, Neural Network sẽ tính sai số giữa giá trị dự đoán được với giá trị thực tế.

#### Backward-Propagation

Sau khi có sai số rồi, Neural Network sẽ dùng sai số này để điều chỉnh các trọng số bằng việc đạo hàm và sử dụng phương pháp Gradient Descent

Bước 1: Tính đạo hàm của sai số theo từng trọng số (vì trọng số trong Neural Network thay đổi thì sai số cũng thay đổi).

Bước 2: Sau khi tính được đạo hàm của sai số theo từng trọng số rồi thì Neural Network sẽ điều chỉnh từng trọng số bằng công thức Gradient Descent. (Lưu ý: Neural Network chưa áp dụng các trọng số được điều chỉnh cho tới khi quá trình Backward-Propagation kết thúc).

#### Tổng kết

Neural Network thực hiện quá trình Forward-Propagation và Backward-Propagation cho tới khi sai số giữa đầu ra dự đoán và giá trị thực tế là nhỏ nhất. Nhưng thực tế ta không thể xác định được khi nào sai số đạt nhỏ nhất nên quá trình training dừng khi thực hiện hết số epoch quy định trước đó.

### Ứng dụng

ANN được sử dụng cho các loại dữ liệu sau:

* Tabular datasets (.csv)
* Classification prediction problems
* Regression prediction problems

Các ứng dụng của ANN:

* Dự đoán giá chứng khoán.
* Dự báo thời tiết

## Convolutional neural network (CNN)

### Định nghĩa

CNN là một dạng cụ thể của ANN, dạng này dùng các thuật toán Machine Learning và được sử dụng trong Supervised Learning để phân tích dữ liệu.

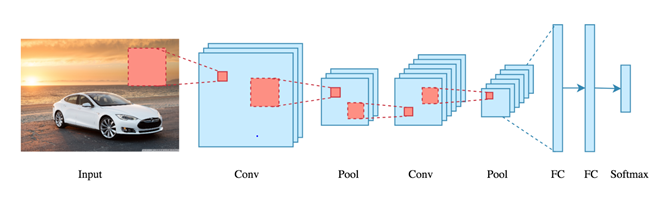
CNN là mạng được sử dụng rất nhiều trong các bài toán liên quan tới hình ảnh. Lợi ích chính của CNN là mạng này có thể tự động dò ra các đặc trưng quan trọng mà không cần sự giám sát của con người.

Ví dụ: Cho một vài tấm hình của chó và mèo. CNN sẽ dò ra các đặc trưng đặc biệt cho từng lớp.

### Kiến trúc

CNN có kiến trúc bao gồm các phần sau:

* Convolutional Layer
* Pooling Layer
* Fully-Connected Layer (FC)



Hình 2‑8: Kiến trúc của mạng CNN

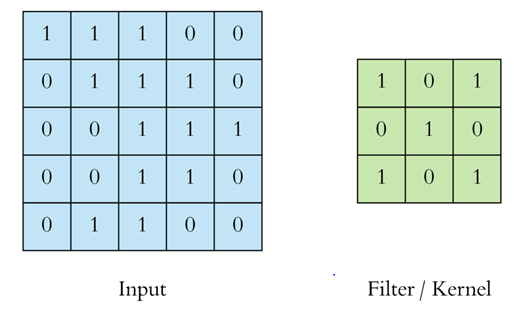
### Cách hoạt động

#### Convolution

Phần chính của CNN là Convolutional Layer. Convolution là phép toán toán học dùng để “hợp nhất” (Merge) hai ma trận dữ liệu.

🡪 Trong trường hợp này, Convolution được áp dụng cho dữ liệu đầu vào bằng cách sử dụng một Convolutional Filter / Kernel để tạo ra một Feature Map.

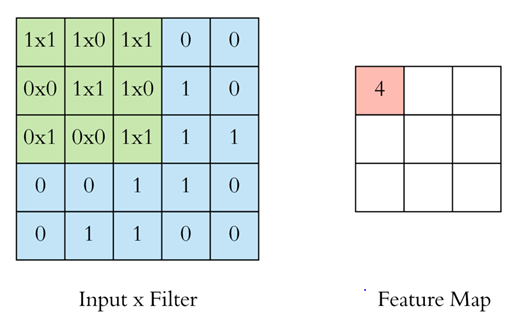
Ví dụ:



Hình 2‑9: Input và Kernel để thực hiện phép toán Convolution

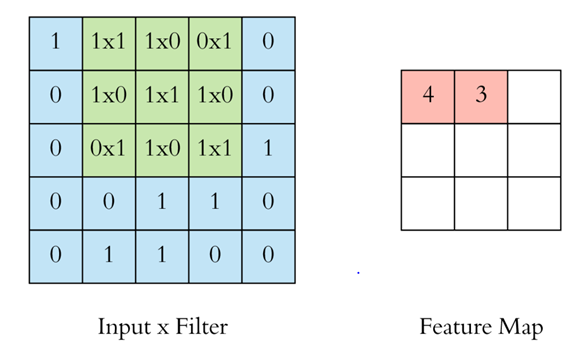
GIẢI THÍCH:

* Ở bên trái là ma trận nhị phân, ma trận này là đầu vào (Input) của Convolution Layer (VD: đầu vào là tấm hình). Ở bên phải là Convolution Filter, còn được gọi là Kernel. Đây được gọi là 3x3 Convolution vì dựa vào kích cỡ của Convolution Filter.
* Biểu diễn phép toán Convolution bằng cách “trượt” (Sliding) Convolution Filter trên Input. Tại mỗi vị trí, ta thực hiện phép nhân ma trận cho từng phần tử và cộng tổng các tích đó lại. Kết quả sau đó sẽ được đem qua Feature Map. Minh họa như hình dưới:



Hình 2‑10: Phép nhân ma trận giữa Input và Filter (1)

Ta tiếp tục phép toán Convolution bằng cách trượt Convolution Filter qua các cột tiếp theo của hàng đầu tiên của Input, và kết quả của lần “trượt” này cũng sẽ được thêm vào Feature Map.



Hình 2‑11: Phép nhân ma trận giữa Input và Filter (2)

* Ta tiếp tục quá trình Convolution cho tới khi Convolution Filter đã trượt hết ma trận Input và trả ra các kết quả và các kết quả này sẽ được bỏ vào Feature Map.

Ở trên là ví dụ về phép toán Convolution 2D sử dụng 3x3 Convolution Filter. Trong thực tế, phép toán Convolution được biểu diễn trên 3D. Trong thực tế, một tấm hình đại diện cho một ma trận 3D với:

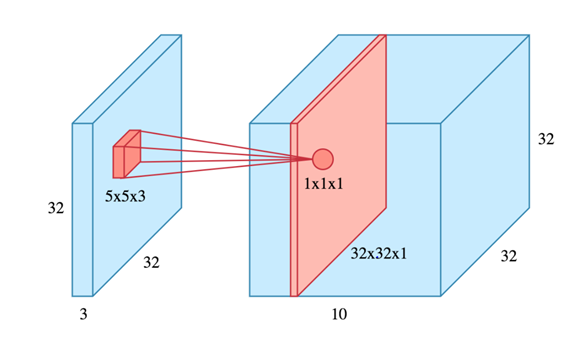
+ Chiều cao. (Height)

+ Chiều rộng. (Width)

+ Chiều sâu. (Depth – Đại diện cho màu sắc)

Một Convolution Filter có chiều cao và chiều rộng cụ thể, như 3x3 hoặc 5x5, và bằng cách thiết kế thì nó bao phủ toàn bộ chiều sâu của dữ liệu đầu vào của nó nên số chiều của Convolution Filter cũng phải là 3D.

Một ví dụ về Convolution Filter 3D dưới đây, biểu diễn nhiều Convolutions trên một Input bằng cách sử dụng các Convolution Filter khác nhau và cho ra các Feature Map khác nhau  Sau đó xếp tất cả các Feature Map tìm được cùng nhau vào “khối”, và “khối” đó là đầu ra cuối cùng của Convolutional Layer.



Hình 2‑12: Ví dụ về Input 3D

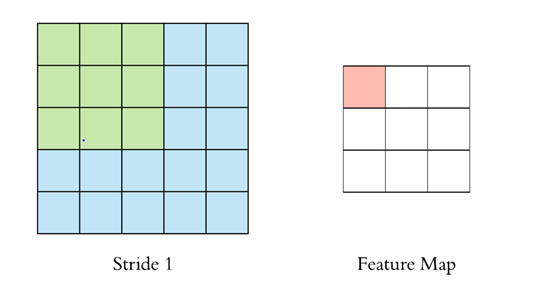
Giả sử có một tấm hình 32x32x3 và Convolution Filter có kích thước 5x5x3 (Chiều sâu của Convolution Filter phải trùng với chiều sâu của Input).

* Khi Convolution Filter ở một vị trí cụ thể nào đó, nó “chứa” một phần nào đó thông tin của Input. Sau đó phép toán Convolution sẽ được thực hiện và kết quả sẽ được cho vào Feature Map tương ứng với Convolution Filter đó 🡺 Lúc này Feature Map có kích thước 32x32x1 (Hình màu đỏ bên phải của hình 10).
* Nếu có 10 Convolution Filter khác nhau thì sau nhiều phép toán Convolution thì sẽ có 10 Feauture Map khác nhau có cùng kích thước là 32x32x1. Và các Feature Map này sẽ được xếp dọc theo chiều sâu, lúc đó các Feature Map sẽ tạo thành một “khối” và “khối” này là output cuối của Convolutional Layer. Kết quả có kích thước 32x32x10 (Hình hộp màu xanh bên phải của hình 10).

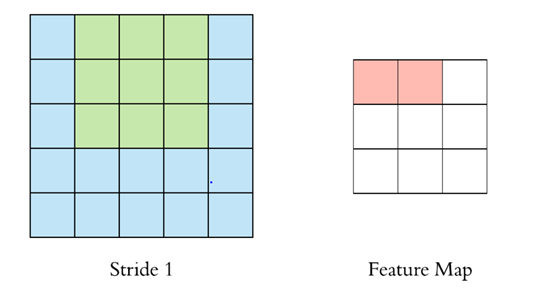
(\*) Để ý thấy chiều cao và chiều rộng của các Feature Map so với Input là không đổi, và việc không đổi này là do Padding.

##### Stride

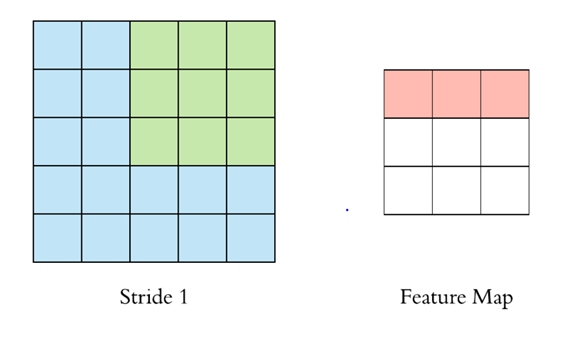
Stride được dùng để chi tiết hóa các Convolution Filter khi trượt trên Input thì sẽ trượt qua bao nhiêu cột tại mỗi bước. Mặc định giá trị Stride sẽ là 1 như các hình dưới.



Hình 2‑13:Biễu diễn Stride=1 (1)

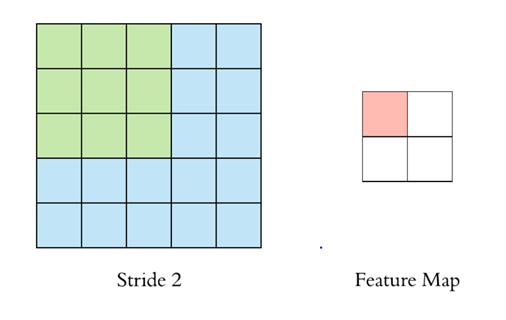


Hình 2‑14: Biễu diễn Stride=1 (2)

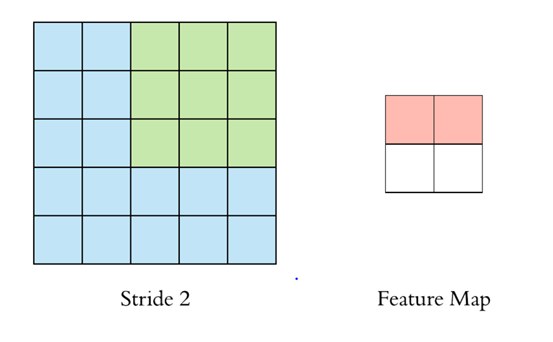


Hình 2‑15: Biễu diễn Stride=1 (3)

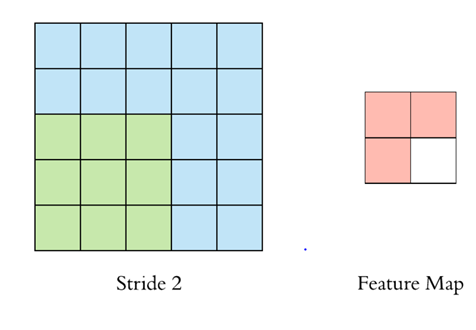
Tùy vào bài toán mà Stride có thể lớn hơn 1 nếu không muốn có sự trùng lặp thông tin của Input. Nếu Stride càng lớn thì kích cỡ của Feature Map sẽ càng lúc càng nhỏ vì Stride càng lớn thì nó sẽ bỏ qua các thông tin đã xuất hiện trước đó. Hình dưới đây minh họa cho Stride = 2 (Lưu ý rằng kích cỡ của Feature Map nhỏ hơn).

S

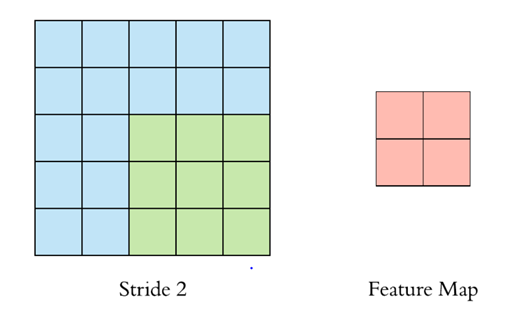
Hình 2‑16 :Biểu diễn Stride = 2 (1)



Hình 2‑17: Biểu diễn Stride = 2 (2)



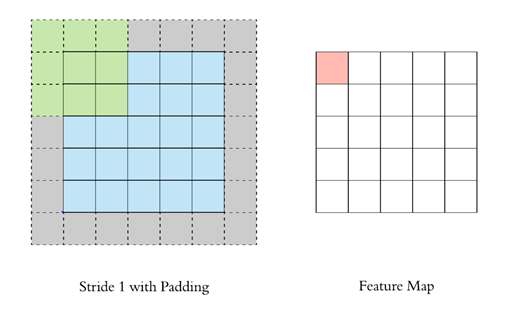
Hình 2‑18: Biểu diễn Stride = 2 (3)



Hình 2‑19: Biểu diễn Stride = 2 (4)

##### Padding

Để ý thấy rằng kích cỡ của Feature Map khi áp dụng Stride = 2 cho input thì nhỏ hơn kích cỡ của Input bởi vì Convolution Filter cần phải được chứa trọn trong Input. Nếu bài toán yêu cầu duy trì kích cỡ của Feature Map thì việc sử dụng Padding là rất cần thiết. Padding “bao bọc” Input bằng các giá trị 0. Hình dưới đây minh họa cho việc sử dụng Padding cho Input.



Hình 2‑20: Áp dụng Padding cho Input

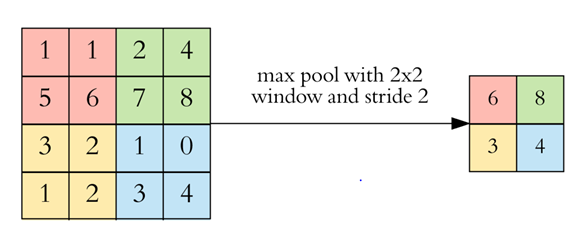
Khu vực màu xám xung quanh Input là Padding. Và kích cỡ của Feature Map lúc này đã trùng khớp với kích cỡ của Input. Padding được sử dụng phổ biến trong CNN để bảo toàn kích cỡ của Feature Map, không thì các Feature Map sau mỗi Layer sẽ có kích cỡ bị giảm thấp, là điều không mong muốn trong các bài toán.

#### Pooling

Sau khi một phép toán Convolution được thực hiện xong thì tiếp theo sẽ là phép toán Pooling. Phép toán Pooling này được dùng để giảm kích thước của dữ liệu. Và việc giảm kích thước dữ liệu này đồng thời cũng giúp giảm đi số lượng các tham số, điều này giúp giảm ngắn thời gian training và giảm được tỉ lệ gặp overfitting. Các pooling layer giảm kích thước của các feature map một cách độc lập, giảm đi chiều cao và chiều rộng nhưng giữ nguyên chiều sâu.

Dạng Pooling phổ biến nhất là Max Pooling, dạng này chỉ lấy giá trị lớn nhất trong cửa sổ mà Pooling trượt. Trái ngược với phép toán Convolution, phép toán Pooling không có tham số. Nó trượt một cửa sổ trên Input của nó, và điều mà nó làm là lấy giá trị lớn nhất trong từng cửa sổ mà nó trượt.

Tương tự phép toán Convolution, kích thước của Pooling cũng phải được cụ thể. Ví dụ về phép toán Max Pooling bằng cách sử dụng cửa sổ Pooling có kích thước 2x2 và Stride 2. Mỗi màu đại diện cho các cửa sổ khác nhau. Vì cả kích thước Pooling và Stride đều là 2 nên thông tin không bị trùng lặp. Hình dưới là hình minh họa cho ví dụ này:



Hình 2.5.14: Biểu diễn kết quả của MaxPool 2x2 và Stride = 2

(\*) LƯU Ý: Sau khi thực hiện phép toán Pooling, output của Pooling layer là một ma trận có kích thước bằng nửa kích thước của Input của Pooling Layer. Đây là tác dụng chính của phép toán Pooling, giảm đi kích thước của Feature Map trong khi vẫn giữ những thông tin quan trọng.

#### Fully-Connected Layer

Sau các phép toán Convolution và Pooling là các Fully Connected Layers. Cấu trúc của Fully Connected Layer này có cấu trúc giống với cấu trúc của mạng ANN.

Output của cả hai phép toán Convolution Layer và Pooling Layer đều là ma trận 3D, nhưng một Fully Connected Layer chỉ nhận đầu vào là vector 1D. Nên trước đó các ma trận 3D phải trải qua bước “Flatten” để đưa về dạng vector 1D.

#### Tổng kết

Một mô hình CNN được xem là sự kết hợp giữa hai thành phần:

* Phần trích đặc trưng: gồm 2 loại layer: convolution layer và pooling layer.
* Phần phân lớp.

### Ứng dụng

CNN được sử dụng cho các loại dữ liệu sau:

* Image Data.
* Classification prediction problems.
* Regression prediction problems.

Ứng dụng cụ thể của CNN:

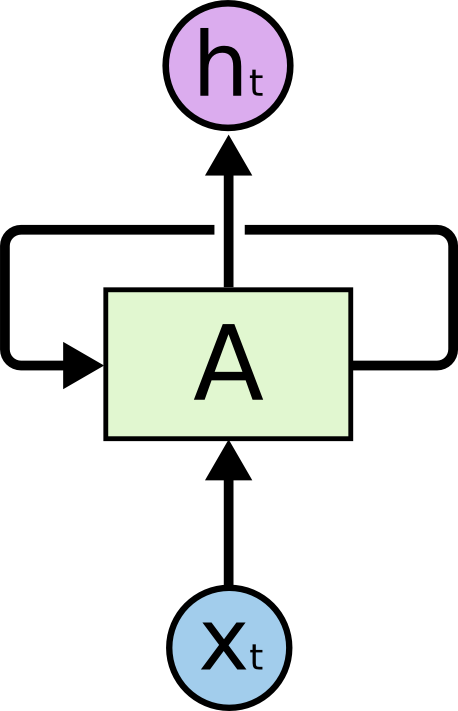
* Nhận diện vật thể trong hình ảnh.
* Nhận diện màu sắc của vật thể.
* Nhận diện khuôn mặt.
* Dự đoán ung thư.

## Recurrent neural network (RNN)

### Định nghĩa

#### Recurrent Neural Network (RNN)

Recurrent Neural Network là một biến thể khác của Artificial Neural Network, trong đó kết nối giữa các nút tạo thành một đồ thị có hướng dọc theo một trình tự. Điều này cho phép nó thể hiện hành vi, động thái, thời gian trong một chuỗi thời gian ( time sequence).



Hình 2‑21: Mạng RNN cơ bản

#### Sequence data

Sequence data có rất nhiều dạng. Ví dụ trong bài toán xử lý tín hiệu đèn giao thông, tín hiệu ở đây là một dạng của dữ liệu trình tự vì các tín hiệu nối tiếp nhau theo thứ tự và từng phần tín hiệu nhỏ góp phần tạo nên ý nghĩa cho cả đoạn tín hiệu lớn.

Âm thanh là một sequence data. Bạn có thể cắt một quang phổ âm thanh thành các phần và nạp vào RNN.



Hình 2‑22: Một đoạn phổ quang âm thanh

Văn bản cũng là một dạng sequence data. Bạn có thể ngắt văn bản thành chuỗi ký tự hoặc chuỗi các từ.

#### Time Step

Trong mạng hồi quy, Time Step là số lần lặp của mạng. Khi mạng hồi quy nhận vào một input thì mạng đã bước sang một Time Step.

Ví dụ: đưa vào mạng hồi quy một câu “I have a good day” và câu này được dùng để train mô hình. Ban đầu, mạng hồi quy sẽ nhận đầu vào là từ “I”, khi mạng nhận vào từ “I” thì đó được xem là Time Step đầu tiên. Tiếp theo mạng sẽ nhận đầu vào là từ “Have” và đó được xem là Time Step thứ hai, nhưng ở Time Step thứ hai này, input của mạng hồi quy lúc này không chỉ có thông tin của từ “Have” mà còn có luôn cả thông tin của từ “I” mà mạng nhận vào ở Time Step đầu tiên. Cứ như vậy cho tới khi mạng nhận vào hết câu mà được dùng để train mô hình

🡺 Output của mỗi Time Step sẽ ảnh hưởng tới Output của mỗi Time Step sau nó.

#### One-hot vector

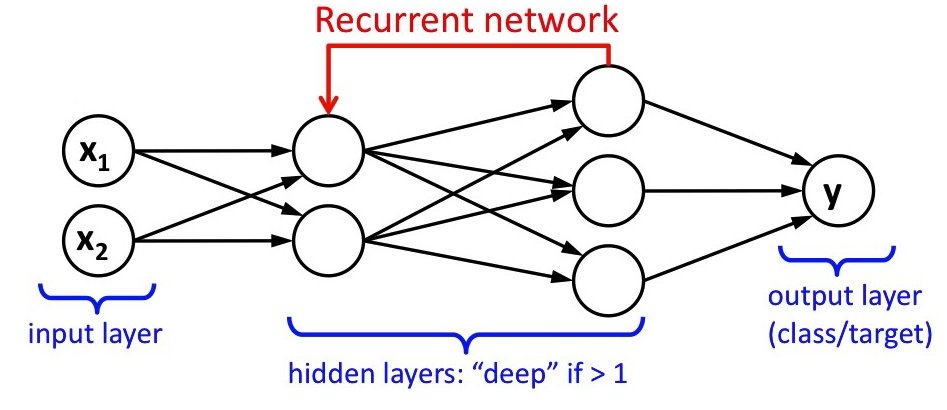
Trong mạng hồi quy, one-hot vector là một ma trận 1xN (hay còn được gọi là một vector). Vector này sẽ bao gồm toàn giá trị 0 ngoại trừ một giá trị 1 duy nhất tại vị trí tương ứng của từ / chữ cái đó trong danh sách từ vựng.

Ví dụ, có câu sau: “Have a good day”. Dựa vào câu này, danh sách từ vựng sẽ là {Have, a, good, day}. Vậy dựa vào vị trí của các từ trong danh sách từ vựng, suy ra được vector one-hot của các từ như sau:

* + Have: [1 0 0 0]
  + a: [0 1 0 0]
  + good: [0 0 1 0]
  + day: [0 0 0 1]

### Kiến trúc

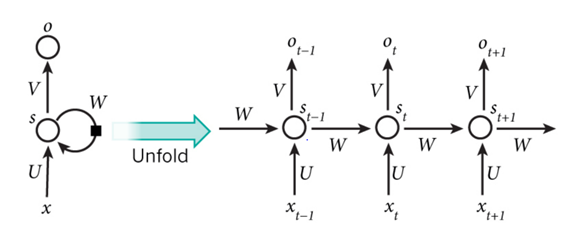
RNN có kiến trúc tương tự như mạng ANN, nhưng khác ở chỗ là trong mạng RNN có vòng lặp ở bên trong, còn mạng ANN thì không. RNN có rất nhiều biến thể.



Hình 2‑23: Kiến trúc của mạng RNN

### Cách hoạt động

Để hiểu rõ hơn cách hoạt động của mạng hồi quy, trước tiên phải “unfold” mạng ra như hình dưới



Hình 2‑24: Mạng RNN cơ bản sau khi được "Unfolded"

Chú thích:

* là input tại Time Step t. Ví dụ, có thể là một one-hot vector đại diện cho từ thứ hai của một câu.
* là Hidden State tại Time Step t. Nó là “bộ nhớ” (Memory) của mạng. được tính toán dựa trên các Hidden State trước đó và input tại Time Step t: = . Hàm thường là hàm phi tuyến tính như hàm tanh hoặc reLU.
* là Output của Time Step t. Ví dụ, nếu muốn dự đoán từ tiếp theo sẽ xuất hiện trong một câu thì sẽ là một vector chứa các xác suất của mỗi từ. = softmax(V).
* W, V, U: ma trận trọng số.

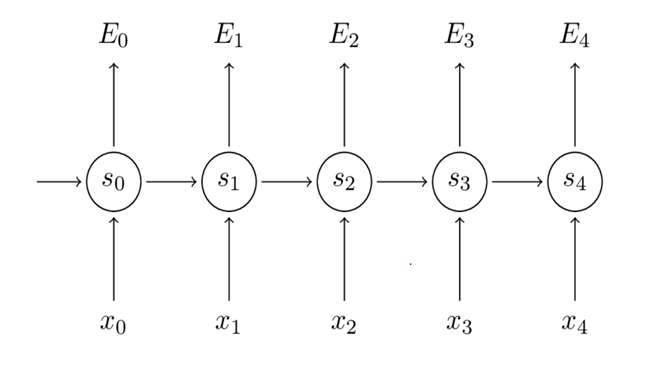
LƯU Ý:

* Có thể xem là bộ nhớ của mạng hồi quy. nắm bắt được thông tin về những gì đã xảy ra ở các Time Step trước.
* Output được tính chỉ dựa trên của Time Step t.
* Trong lý thuyết, mạng hồi quy có thể nhớ toàn bộ các Time Step (tức là mạng có bộ nhớ không giới hạn). Nhưng trong thực thành, mạng chỉ nhớ được một vài Time Step trước đó thôi

Back-propagation through time (BPTT)

Thuật toán back-propagation-through-time (BPTT) để đào tạo một recurrent network là một mở rộng của thuật toán back-propagation.

Tương tự Backpropagation, nhiệm vụ của BPTT là điều chỉnh các trọng số sao cho hàm mất mát là nhỏ nhất, nhưng Backpropagation through time tính sai số ở từng Time Step.



Hình 2‑25: Hình minh họa cho BPTT

Các bước thực hiện chi tiết của thuật toán BPTT:

* Bước 1: Tính sai số ở từng Time Step.
* Bước 2: Sử dụng “Quy tắc chuỗi” (Chain rule) để tính đạo hàm của hàm mất mát theo các trọng số U, V, W. Ở ví dụ này, chỉ sử dụng sai số

1. = . . ()
2. = .

(Vì ) 🡺 phụ thuộc vào và phụ thuộc vào W và , và cứ như vậy)

1. = . . . .

* Bước 3: Cộng tất cả các đạo hàm với nhau vì một câu (Sentence) dùng để training mô hình được xem là một điểm dữ liệu huấn luyện. Ví dụ như sau:

1. =
2. =
3. =

* Bước 4: Dùng các đạo hàm (4), (5), (6) để điều chỉnh ma trận trọng số W, V, U bằng Gradient Descent.

### Ứng dụng

RNN được sử dụng cho các loại dữ liệu sau:

* Text Data.
* Speech Data.
* Classification prediction problems.
* Regression prediction problems.
* Generative models.

Các ứng dụng của RNN là:

* Dự đoán từ tiếp theo của một câu / một văn bản.
* Tạo chú thích cho hình ảnh.
* Nhận diện giọng nói.
* Phân tích cảm xúc của một câu.

## So sánh

|  |  |  |
| --- | --- | --- |
| ANN | CNN | RNN |
| Không có Hidden State | Không có Hidden State | Có Hidden State |
| Mỗi lần train chỉ đưa vào từng input một | Mỗi lần train chỉ đưa vào từng input một | Mỗi lần train sẽ đưa vào input mới và thông tin của các input trước đó |
| Không nhớ được thông tin của những input trước đó | Không nhớ được thông tin của những input trước đó | Lưu thông tin của những input trước đó trong Hidden State |
| Mỗi output của mỗi input không làm ảnh hưởng tới việc tính toán của nhau | Mỗi output của mỗi input không làm ảnh hưởng tới việc tính toán của nhau | Output của Time Step trước sẽ làm ảnh hưởng tới tính toán của Output của Time Step sau nó |
| Được sử dụng cho các bài toán phân lớp, regression | Được sử dụng cho các bài toán phân lớp (đặc biệt là các bài toán nhận diện vật thể trong hình ảnh) | Được sử dụng cho các bài toán có dữ liệu trình tự như: dự đoán từ tiếp theo của một câu, phân tích cảm xúc. |

# KẾT LUẬN

Sau thời gian nghiên cứu và tìm hiểu các kiến thức liên quan tới Neural Network, nhóm đã có tầm nhìn tổng quan về các loại cũng như các cách hoạt động khác nhau của từng loại Neural Network. Đồng thời, nhóm đã học được các phương pháp để tối ưu mô hình sử dụng các kiến thức đại số, giải tích, đại số tuyến tính.

Vì thời gian có hạn và lý thuyết về NeuraL Network là quá nhiều nên nhóm chỉ tìm hiểu được 3 loại Neural Network là: Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN). Và nếu có cơ hội thì nhóm sẽ tiếp tục tìm hiểu các lý thuyết sâu hơn và sau này có thể phát triển ứng dụng.

# TÀI LIỆU THAM KHẢO

1. Ahmed Gad, “Beginners Ask “How Many Hidden Layers/Neurons to Use in Artificial Neural Networks?”, October 2018

<<https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e> >

1. Arden Dertat, “Applied Deep Learning - Part 4: Convolutional Neural Networks”, October 2018

<<https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2> >

1. Arden Dertat, “Applied Deep Learning - Part 1: Artificial Neural Networks”, October 2018

<<https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6> >

1. Debarko De, “RNN or Recurrent Neural Network for Noobs”, November 2018

<<https://hackernoon.com/rnn-or-recurrent-neural-network-for-noobs-a9afbb00e860> >

1. Denny Britz, “Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs”, November 2018

<<http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/> >

1. Denny Britz, “Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients”, November 2018

<<http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/> >

1. Dhruvil Karani, “Introduction to Word Embedding and Word2Vec”, December 2018

<<https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa> >

1. Jason Brownlee, “What is Deep Learning ?”, September 2018

< <https://machinelearningmastery.com/what-is-deep-learning/> >

1. Jason Brownlee, “When to Use MLP, CNN, and RNN Neural Networks”, December 2018

< <https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/> >

1. Karpathy, “ Convolutional Neural Networks (CNNs / ConvNets)”, November 2018

< <http://cs231n.github.io/convolutional-networks/> >

1. Kishan Maladkar, “Types Of Activation Functions In Neural Networks And Rationale Behind It”, September 2018

< <https://www.analyticsindiamag.com/most-common-activation-functions-in-neural-networks-and-rationale-behind-it/> >

1. Kulbear, “ReLU and Softmax Activation Functions”, September 2018

< <https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions> >

1. Matt Mazur, “A Step by Step Backpropagation Example”, October 2018

< <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/> >

1. Michael Nguyen, “Illustrated Guide to Recurrent Neural Networks”, December 2018

<<https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9> >

1. Niklas Donges, “Gradient Descent in a Nutshell”, October 2018

< <https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0> >

1. Niklas Donges, “Recurrent Neural Networks and LSTM”, December 2018

< <https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5> >

1. Simon Haykin, 2009, *Neural Networks and Learning Machines*, Third Edition, Pearson Education, McMaster University Hamilton, Ontario, Canada.
2. Vũ Hữu Tiệp, “Multilayer Perceptron và Backpropagation”, September 2018

< <https://machinelearningcoban.com/2017/02/24/mlp/> >

1. Wikipedia, “Activation Function”, September 2-18

< <https://en.wikipedia.org/wiki/Activation_function> >

1. Sagar Sharma, “Activation Functions: Neural Networks”, September 2018

< <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> >