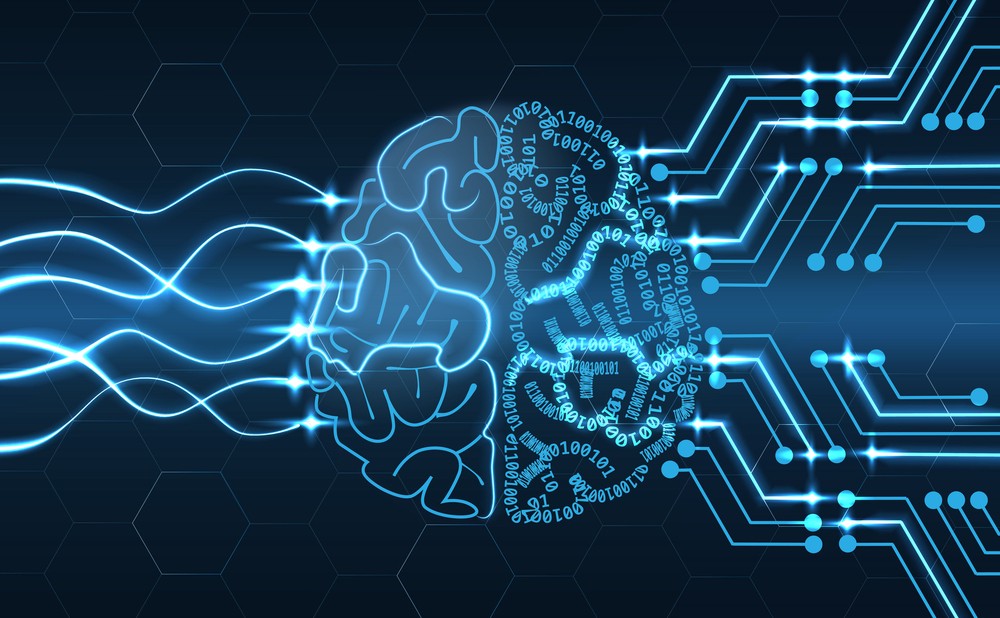
# Thuật toán CNN

## Tổng quan về CNN

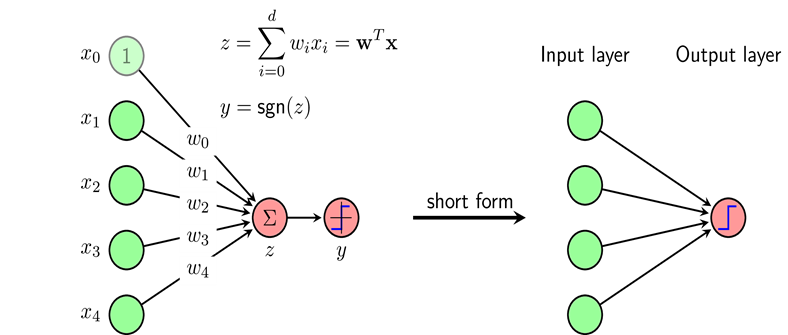
**CNN (Convolutional Neural Network)** là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay như hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Ví dụ như tự động nhận diện khi chúng ta đăng một ảnh lên Facebook, hay khi search một từ bất kì lên google search, ví dụ "Mèo" thì trong tab "Hình ảnh", google sẽ hiển thị rất nhiều ảnh có mèo trong đó. CNN chính là một dạnh Artificial Neural Network, một Multiplayer Perceptron nhưng mang thêm 1 vài cải tiến, đó là **Convolution** và **Pooling**.



Hình 7: Convolutional Neural Network

* 1. Đặc điểm của CNN
  2. Neural Network

Mạng nơron nhân tạo, Artificial Neural Network (ANN) là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ nơron sinh học. Nó được tạo nên từ một số lượng lớn các phần tử (nơron) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng nơron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu,...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất học chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơron.



Hình 8: Mạng Neural Network cơ bản

Các thành phần cơ bản của một nơron nhân tạo bao gồm:

* Tập các đầu vào: Là các tín hiệu vào (input signals) của nơron, các tín hiệu này thường được đưa vào dưới dạng một vector N chiều.
* Tập các liên kết: Mỗi liên kết đƣợc thể hiện bởi một trọng số liên kết – Synaptic weight. Trọng số liên kết giữa tín hiệu vào thứ j với nơron k thường được kí hiệu là w\_kj. Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.
* Bộ tổng (Summing function): Thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.
* Ngưỡng (còn gọi là một độ lệch - bias): Ngưỡng này thường được đưa vào như một thành phần của hàm truyền.
* Hàm truyền (Transfer function): Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi nơron. Nó nhận đầu vào là kết quả của hàm tổng và ngưỡng.
* Đầu ra: Là tín hiệu đầu ra của một nơron, với mỗi nơron sẽ có tối đa là một đầu ra.
  1. Ưu và nhược điểm của CNN

Trong thực tế, mỗi bức ảnh đầu vào của một bài toán có kích thước là D \* R \* C (D là chiều dài, R là chiều rộng, C là số màu) thì số lượng tham số cần phải xử lý là rất lớn. Giả sử một bức hình đơn giản sẽ có 28x28x3=2352 tham số, đây là một con số rất lớn nhưng trong thực tế thì kích thước của một bức ảnh còn lớn hơn rất nhiều (Đối với một bức ảnh X-ray thông thường thì chiều dài và chiều rộng ít nhất là 425 và 512). Và CNN sẽ giải quyết được vấn đề này, cái mà ANN không thể làm được. Vì sao? Vì CNN sẽ giúp số tham số đầu vào của ANN trở nên ít lại nhưng vẫn giữ nguyên các thuộc tính cần thiết cho bức hình.

Nhưng CNN gặp phải các khó khăn sau:

* Cần lượng dữ liệu rất lớn để có thể xây dựng một model chuẩn.
* Tiêu tốn nhiều tài nguyên để thực hiện thuật toán.
* Các bức ảnh đầu vào phải có cùng một kích thước (width, height, dimension,…).
* Khi các bức hình đầu vào có kích thước lớn hơn thì kéo theo số lượng layer tăng rất nhiều.

## Activation function

Activation function giới hạn phạm vi giá trị tới một giá trị hữu hạn cho phép. Một số hàm activation thường dùng : Sigmoid, Rectified linear unit (ReLU), Softmax.

1. Sigmoid function

Sigmoid function là một logistic function có phạm vi (0, 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.

Áp dụng tốt nhất có bài toán classification với chỉ hai class.

(1)

(2)



Hình 9: Đồ thị hàm Sigmoid

1. Softmax

Hàm softmax có output nằm trong khoảng từ 0 đến 1, giống như hàm Sigmoid. Nhưng nó chia mỗi output sao cho tổng số output bằng 1.

(3)

1. Rectified linear unit (ReLU)

ReLU (Rectified Linear Unit) được sử dụng rộng rãi gần đây vì tính đơn giản của nó. Nó có công thức toán học - rất đơn giản. Ưu điểm chính của nó là:

* ReLU được chứng minh giúp cho việc training các Deep Networks nhanh hơn rất nhiều. Sự tăng tốc này được cho là vì ReLU được tính toán gần như tức thời và gradient của nó cũng được tính cực nhanh với gradient bằng 1 nếu đầu vào lớn hơn 0, bằng 0 nếu đầu vào nhỏ hơn 0.
* Mặc dù hàm ReLU không có đạo hàm tại , trong thực nghiệm, người ta vẫn thường định nghĩa ReLU và khẳng định thêm rằng, xác suất để input của một unit bằng 0 là rất nhỏ.



Hình 10: Đồ thị hàm ReLU

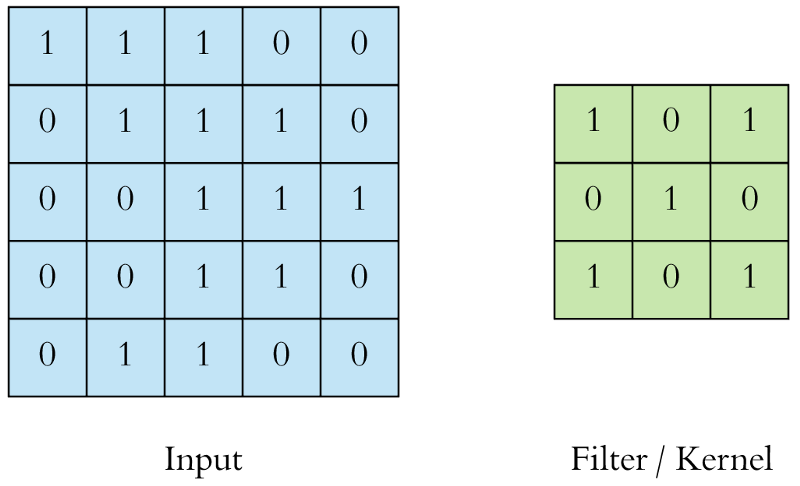
Với bất kỳ input có giá trị âm đưa vào cho hàm ReLU sẽ biến giá trị thành 0, do đó ảnh hưởng đến biểu đồ kết quả khi không ánh xạ các giá trị âm một cách thích hợp.

## Các thành phần cơ bản của mạng CNN

1. Convolution

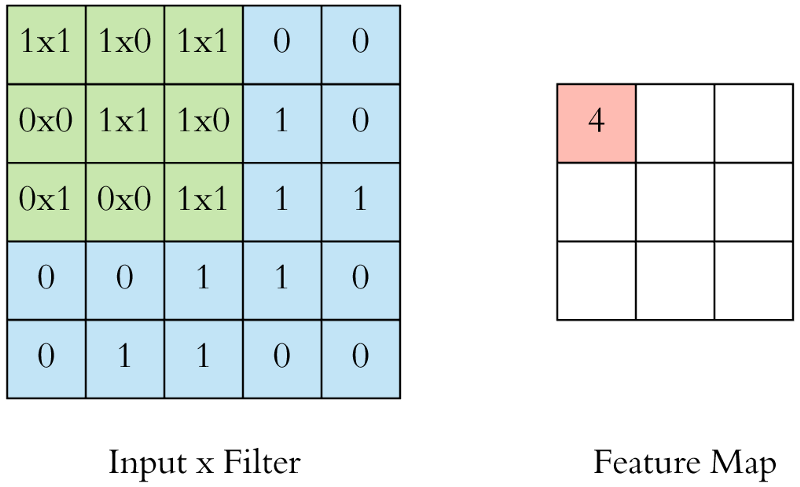
Convolution gồm 2 khái niệm khác là **Convolution Filter** và **Convolutional Layer.** Trong mạng Neural Network thông thường, từ input, ta cho qua các Hidden Layer rồi ra được output. Với CNN, Convolutional Layer cũng chính là Hidden Layer, khác ở chỗ, Convolutional Layer là một tập các feature map và mỗi feature map này là một bản scan của input ban đầu, nhưng được trích xuất ra các feature/đặc tính cụ thể. Scan như thế nào thì lại dựa vào Convolution Filter hay kernel. Đây là một ma trận sẽ quét qua ma trận dữ liệu đầu vào, từ trái qua phải, trên xuống dưới, và nhân tương ứng từng giá trị của ma trận đầu vào mà ma trận kernel rồi cộng tổng lại, đưa qua activation funciton (sigmoid, relu, elu, ... ), kết quả sẽ là một con số cụ thể, tập hợp các con số này lại là 1 ma trận nữa, chính là feature map.

Ví dụ: có một ma trận đầu vào input và một kernel.



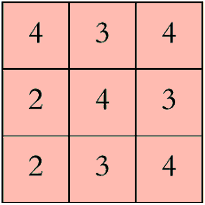
Hình 11: Ma trận đầu vào (trái) và kernel (phải)

Quét kernel qua từng phần tử của input. Và tính toán như trên: nhân tương ứng, rồi cộng tổng kết quả, đưa qua activation function, ta thu được một giá trị tại feature map:



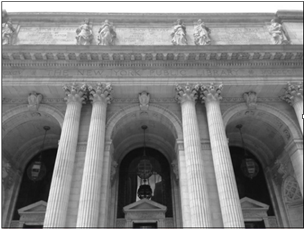
Hình 12: Kernel quét qua từng phần tử của mảng input

Thực hiện lần lượt cho đến hết, ta thu được Feature Map cuối cùng:



Hình 13: Kết quả

Chúng ta không cần phải quan tâm tới việc phải tìm bao nhiêu kernel hay lưu các kernel về để dùng dần. Đó là việc của CNN, nó sẽ tự động tìm các kernel, tự dò ra các feature.



Hình 14: Ví dụ thực tế, trước và sau

* **Chi tiết về Convolution Layer:** Là một layer đầu tiên để tách features từ một bức hình đầu vào. Tích chập bảo toàn mối quan hệ giữa các điểm ảnh bằng cách học features của bức hình sử dụng những hình vuông nhỏ của dữ liệu đầu vào. Nó là một thuật toán mà xem 2 dữ liệu đầu vào là ma trận hình ảnh và một kernel.

Một số khái niệm cơ bản: **Filter**, **Kernel** hay **Feature** **Detector** đều là cách gọi của ma trận lọc (như đã đề cập ở trên). Thông thường, ở các lớp đầu tiên của Conv Layer sẽ có kích thước là [5x5x3]:

- Convolved Feature, Activation Map hay Feature Map là đầu ra của ảnh khi cho bộ lọc chạy hết bức ảnh với phép tích vô hướng.

- Receptive field là vùng ảnh được chọn để tính tích chập, hay bằng đúng cái kích thước của bộ lọc.

- Depth là số lượng bộ lọc.

- Stride được hiểu là khoảng cách dịch chuyển của bộ lọc sau mỗi lần tính. Ví dụ khi stride=2. Tức sau khi tính xong tại 1 vùng ảnh, nó sẽ dịch sang phải 2 pixel. Tương tự với việc dịch xuống dưới.

- Zero-Padding là việc thêm các giá trị 0 ở xung quanh biên ảnh, để đảm bảo phép tích chập được thực hiện đủ trên toàn ảnh.

Cách tính kích thước đầu ra của ảnh mỗi layer:

Giả sử ảnh đầu ra là [ x x ]

Thì:

= (5)

= (6)

=

Trong đó:

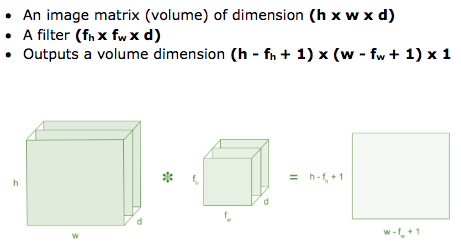
[ x x ]: Kích thước đầu vào

: Kích thước bộ lọc Kernel

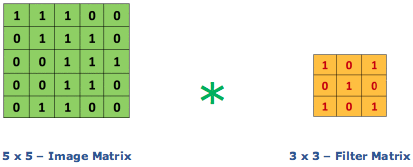
: giá trị Stride

: số lượng zero-padding thêm vào viền ảnh

: Số lượng bộ lọc (Depth)



Hình 15: Mô hình tổng quát của Conv layer

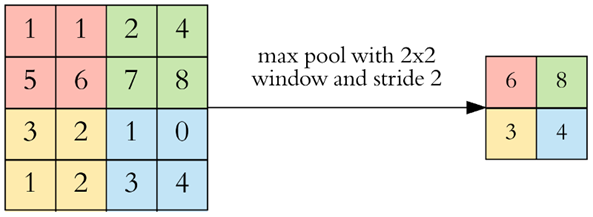


Hình 16: Minh họa toán học

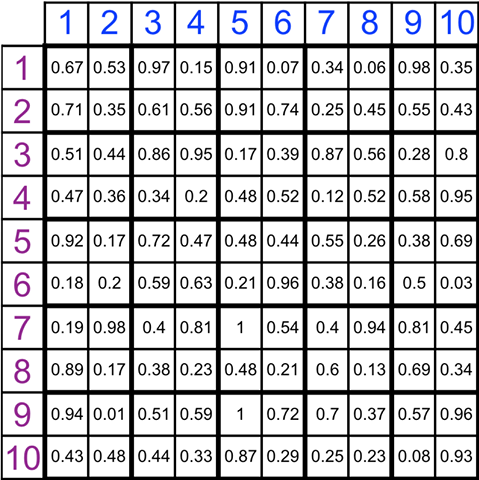
1. Max pooling

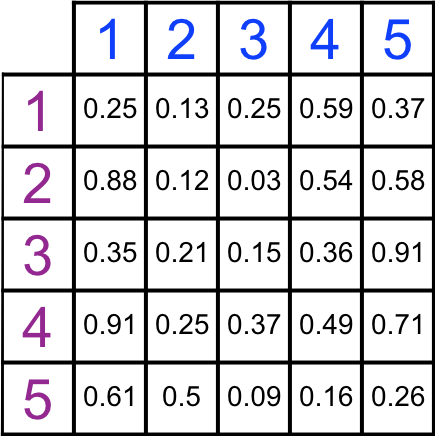
Pooling được sử dụng trong CNN để giảm số hyperparameter cần phải tính toán, từ đó giảm thời gian tính toán, tránh overfitting. Có 2 loại Pooling: Max pooling (dùng nhiều nhất) và Average pooling.

Pooling layer cung cấp tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling).

****Trong bài toán này, chúng ta sẽ sử dụng Max pooling: giữ lại chi tiết quan trọng hay hiểu ở trong bài toán này chính giữ lại pixel có giá trị lớn nhất. Nếu một đặc trưng được phát hiện ở một vùng nào đó bị bao phủ bởi bộ lọc, giá trị cao nhất trong vùng sẽ được giữ lại.

Hình 17: Minh họa Max Pooling (1)

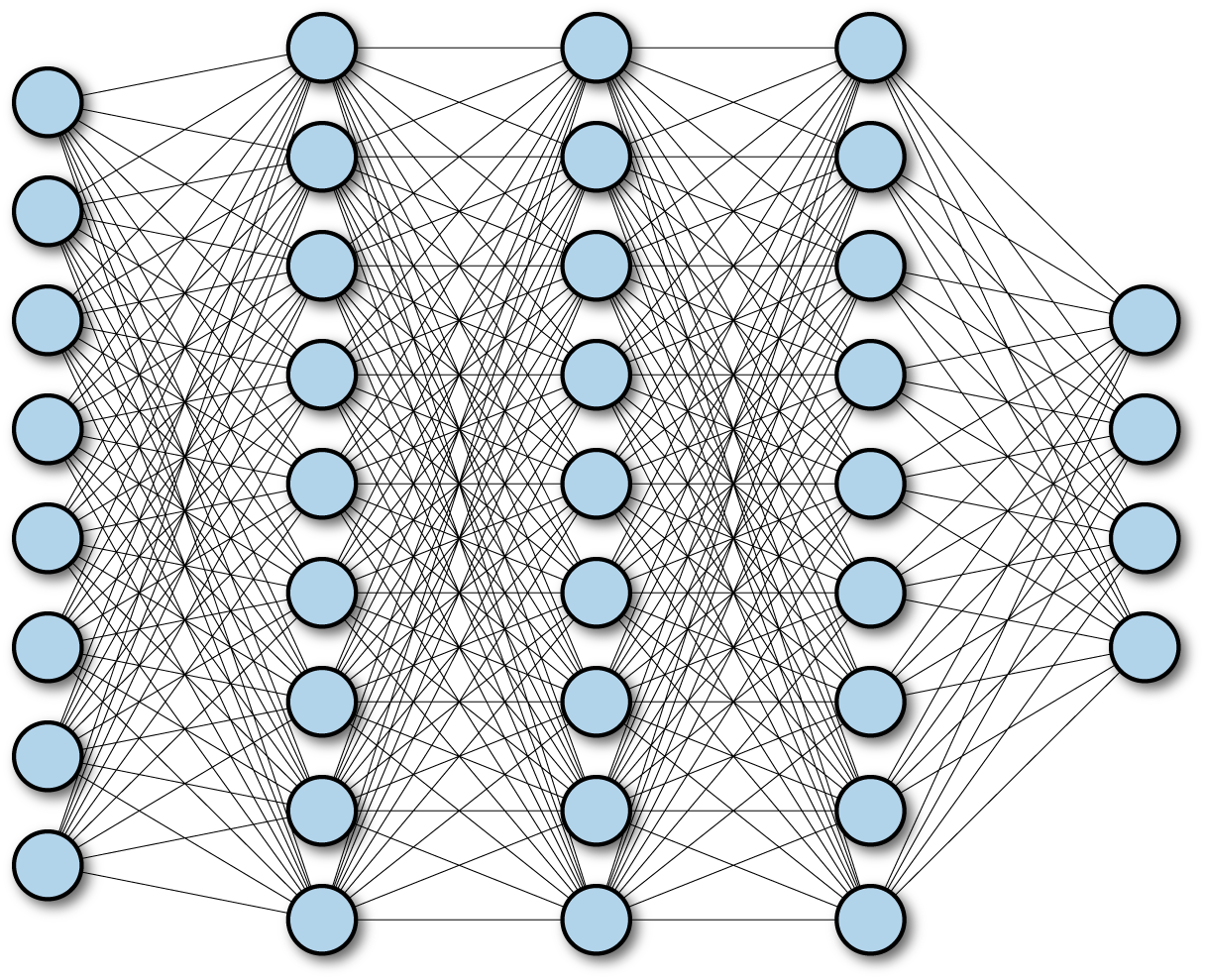




Hình 18: Minh họa Max pooling (2)

1. Fully – connected

Mục đích: kết nối mọi neuron trong một lớp với mọi neuron trong một lớp khác. Về nguyên tắc, nó giống như multi-layer perceptron neural network truyền thống. Thực hiên sau convolutional layers + pooling layers. Vì ouput của 2 layer này có kích thước 3D, còn output của fully connected layer là mảng 1D nên phải Flatten output của pooling layer cuối cùng. Flatten chỉ đơn giản là sắp xếp khối 3D của các số thành một vector 1D ( n hàng, 1 cột).



Hình 19: Sơ đồ Fully - Connected

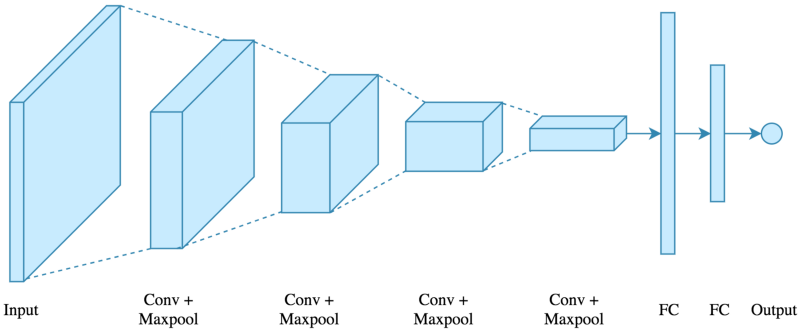
## Cách hoạt động của CNN

CNN có kiến trúc được hình thành từ các thành phần cơ bản bao gồm Convolution (CONV), Pooling (POOL), ReLU, Fully-connected (FC) về mặt xây dựng kiến trúc tổng quát CNN được mô tả như sau (dấu mũi tên thể hiện thứ tự sắp xếp các tầng từ trước đến sau):

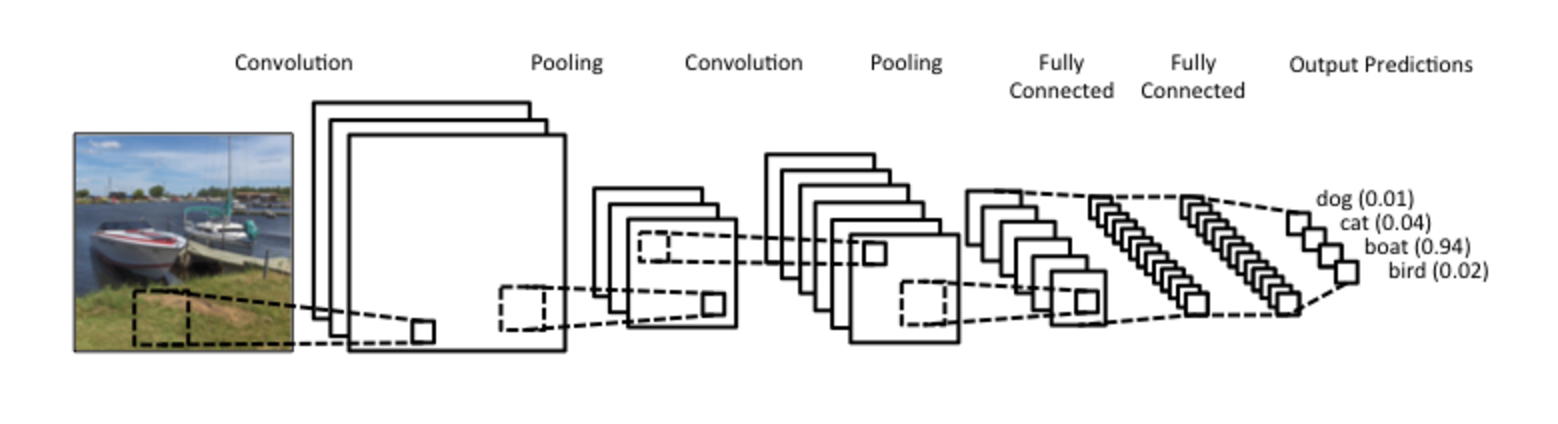
[[CONV 🡪 RELU]\*N 🡪 POOL?]\*M 🡪 [FC 🡪 RELU]\*K 🡪 FC

Trong đó:

* **[CONV -> RELU]\*N** tức là trong kiến trúc này sau tầng CONV là tầng RELU, trong CNN kiến trúc 2 tầng này có thể lặp N lần.
* **POOL?** là tầng Pooling cho người thiết kế quyết định có thể có hoặc không.
* **[[CONV -> RELU]\*N -> POOL?]\*M** trong kiến trúc CNN có thể lặp lại M lần kiểu sau tầng CONV là tầng RELU và kế tới là tầng Pooling.
* **[FC -> RELU]\*K** trong CNN có thể lặp K lần cấu trúc kiểu sau tầng FC là tầng RELU nhưng trước nó phải có tầng [CONV -> RELU].



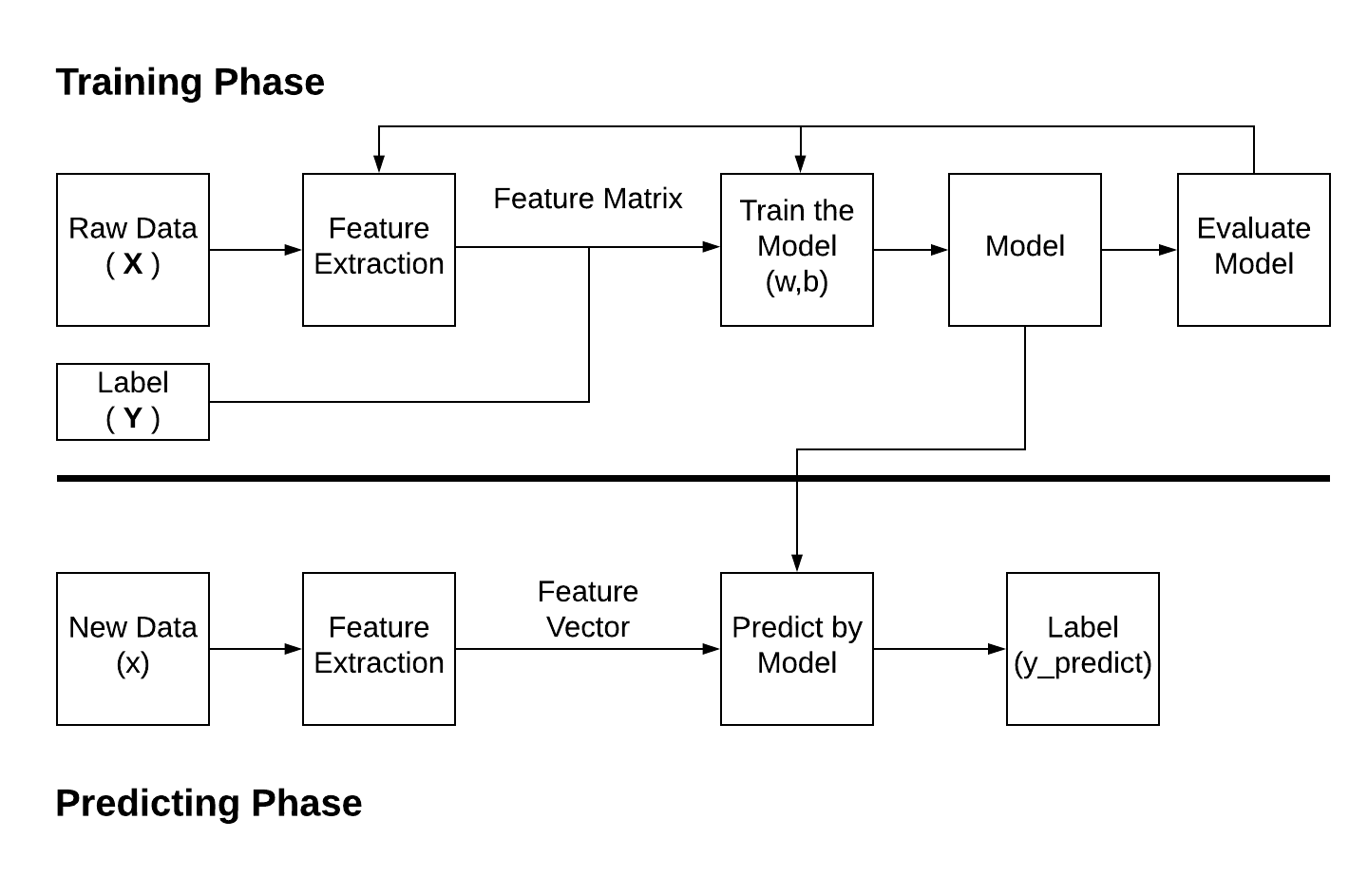
Hình 20: Sơ đồ hoạt động của CNN



Hình 21: Sơ đồ hoạt động của CNN với ảnh thực tế

Trong suốt quá trình huấn luyện, CNNs sẽ tự động học được các thông số.

## Diagram cho bài toán



Hình 22: Diagram

* Training phase:
* Feature Extraction: mục đích là tạo ra một Feature Extractor biến dữ liệu thô ban đầu thành dữ liệu phù hợp với từng mục đích khác nhau.
* Raw data: là tất cả các thông tin ta biết về dữ liệu. Ví dụ: với ảnh thì là giá trị của từng pixel; với văn bản thì là từng từ, từng câu; với file âm thanh thì nó là một đoạn tín hiệu,… Dữ liệu thô này thường không ở dạng vector, không có số chiều như nhau. Thậm chí có thể có số chiều như nhau nhưng số chiều quá lớn, như một bức ảnh màu 1000 pixel x 1000 pixel thì số elements đã là 3×106 (3 vì ảnh màu thường có 3 channels: Red, Green, Blue). Đây là một con số quá lớn, không lợi cho lưu trữ và tính toán.
* Sau quá trình Train model thì chúng ta sẽ thu được một model với dữ liệu (X) được gắn nhãn (Y) tương ứng. Tiếp theo là quá trình đánh giá model, nếu vẫn chưa tốt thì quay lại bước Feature Extraction và lại tiếp tục tiến trình.
* Predicting phase:
* New data: là dữ liệu mới hoàn toàn được đưa vào để thực hiện quá trình dự đoán. Dữ liệu này cũng sẽ trải qua quá trình Feature Extraction ở trên để phân tích.
* Sau khi hoàn tất quá trình Feature Extraction, quá trình dự đoán sẽ được thực hiện với model được tạo ra ở Training phase. Cuối cùng dữ liệu đầu ra sẽ được gán nhãn tương ứng.