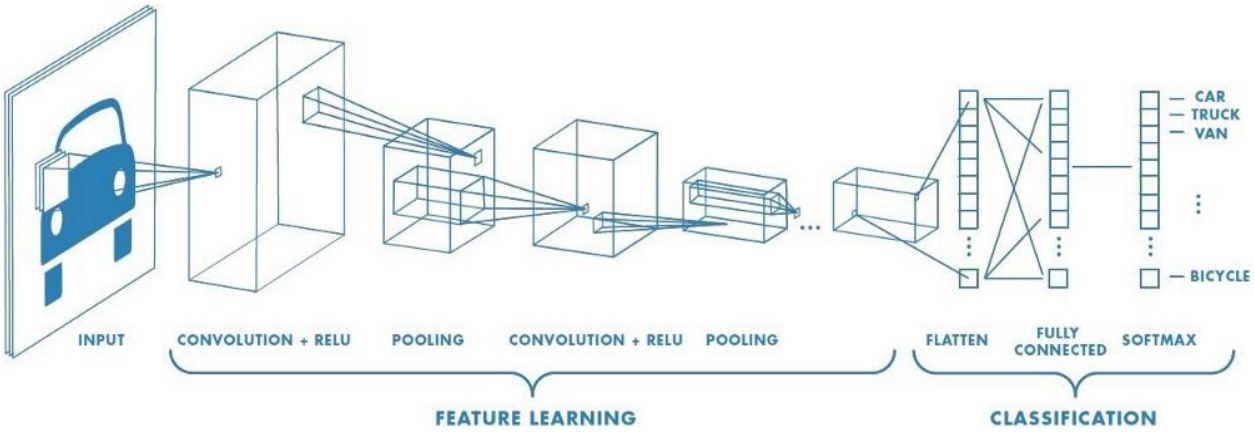
Nhóm 9:

Nguyễn Ngọc Phước	19127519
Hồ Văn Duy	19127373
Nguyễn Trần Thiện Toàn	19127294
Lê Văn Đông	19127363
Đặng Nguyễn Minh Quân	19127523

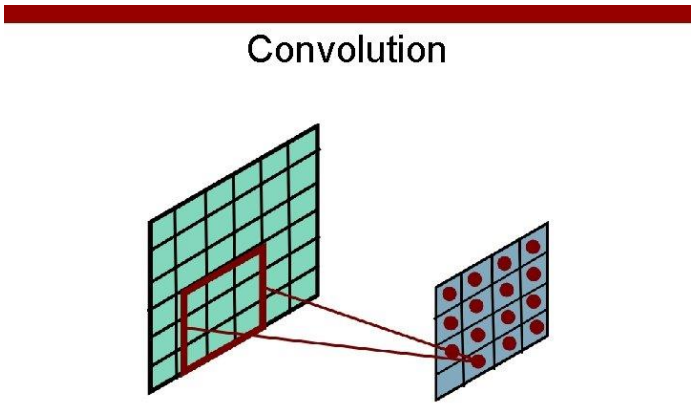
CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Network (CNN), còn được gọi là mô hình mạng neural tích chập, là một trong những mô hình để nhận dạng và phân loại hình ảnh như xác định đối tượng, nhận diện khuôn mặt là những ứng dụng rất phổ biến.



Convolutional Neural network

1. *Convolutional Layer*



Đây là layer chính trong CNN, layer này đảm đương nhiệm vụ trượt kernel qua dữ liệu đầu vào, ở đây cụ thể là dữ liệu pixel của 1 ảnh nào trong mnist dataset.
Convolutional Layers tiến hành tính tổng của tích của từng pixel đầu vào với trọng số tương ứng trên filter, cụ thể:

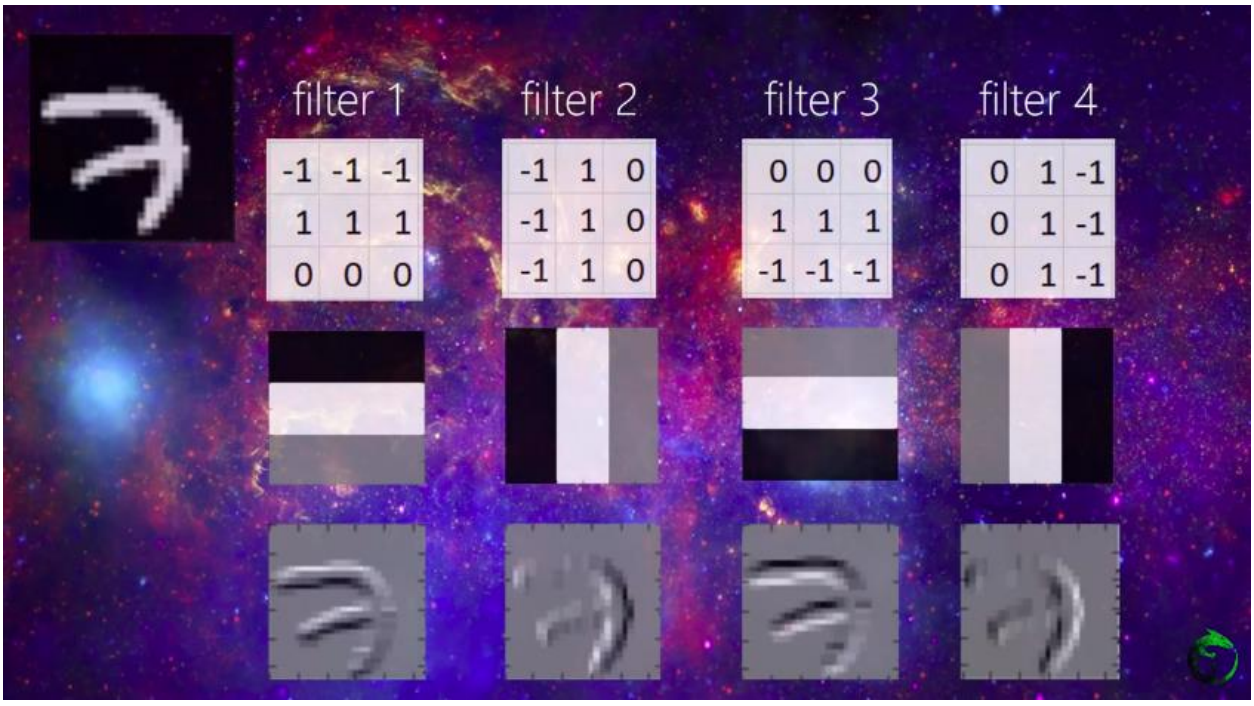
Ta có thể thấy qua cách tính toán như trong hình, dữ liệu pixel đầu vào sẽ qua biến đổi bởi filter và xuất ra dữ liệu đầu ra.

Mục đích của quá trình này là để phát hiện pattern trong ảnh, pattern này có thể từ đi từ các hình đơn giản (cạnh, đường thẳng, đường tròn) cho đến các hình ảnh phức tạp (mắt, mũi, vật thể).

Quá trình trượt chịu ảnh hưởng bởi tham số stride và padding, đây là các kĩ thuật nhằm cải tiến độ chính xác của CNN trong việc phát hiện các pattern.

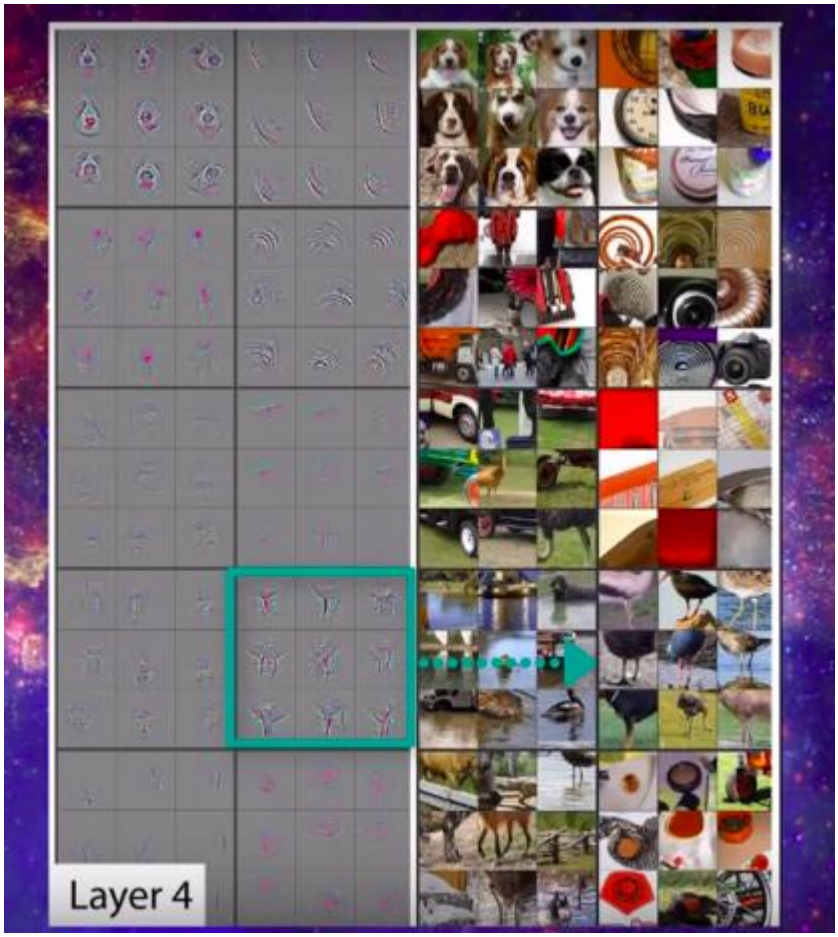
Tham số stride quyết định một lần "trượt" của filter nên dịch qua bao nhiêu pixel. Ở bài toán phân loại ảnh MNIST, số stride thường là 1

Trong quá trình trượt filter, ta có thể để ý các pixel ở góc ảnh không được "trượt" qua nhiều như pixel ở giữa ảnh, như vậy càng nhiều lần "trượt" (tỉ lệ filter size nhỏ hơn so với ảnh nhiều) thì ta càng mất đi dữ liệu càng nhanh ở những pixel ấy, để tránh tình trạng này, padding ra đời, padding là kĩ thuật thêm vào những pixel rác ở ngoài rìa ảnh, khiến cho kích cỡ ảnh lớn hơn, nhưng các pixel trọng tâm của ảnh lại được chú trọng nhiều hơn.



Trong những layer đầu tiên, các filter phát hiện các pattern đơn giản thường tập trung nhiều. Ta có thể thấy như trong hình có 4 filter mang nhiệm vụ phát hiện các cạnh của chữ số viết tay đầu vào.

- Filter 1: Phát hiện những cạnh ở trên
- Filter 2: Phát hiện những cạnh ở bên trái
- Filter 3: Phát hiện những cạnh ở bên dưới
- Filter 4: Phát hiện những cạnh ở bên phải



Càng đi sâu vào từng layer, khả năng phát hiện pattern của CNN nói chung càng phức tạp. Như trên hình ta có thể thấy thậm chí những pattern như chân chim cũng có thể được phát hiện.

Khả năng phát hiện các pattern này được quyết định bởi trọng số của filter.
Tuy nhiên ta cần lưu ý rằng, trọng số của các filter không được định nghĩa từ trước mà được tạo một cách ngẫu nhiên, qua quá trình training, các trọng số này sẽ thay đổi cho phù hợp với mục đích.

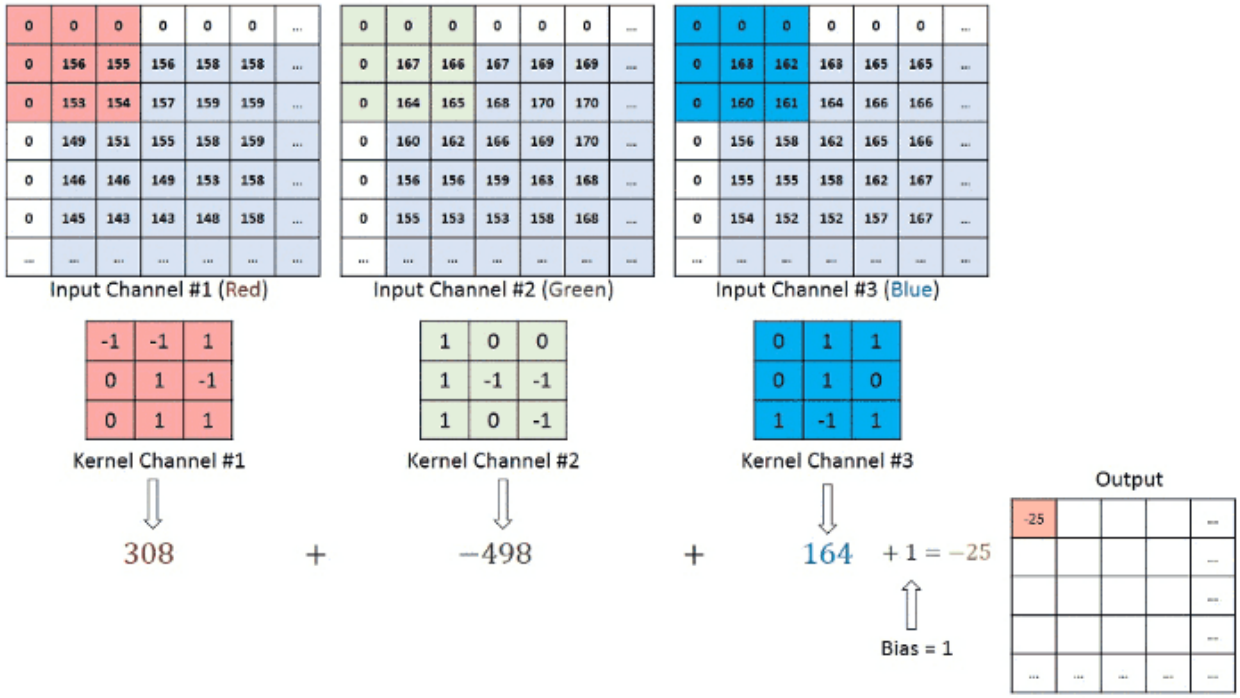
Tất nhiên, để có thể phát hiện nhiều pattern khác nhau trong ảnh, Convolutional Layer không chỉ có 1 filter. Trên thực tế, cả kích cỡ và số lượng của filter cũng như số Convolution Layer còn phải tùy thuộc vào độ phức tạp của ảnh và thực nghiệm

Trong mạng CNN thì các thuật ngữ filter, kernel và feature extractor thường được sử dụng thay thế cho nhau. Chúng đều chỉ một khối mang khả năng trượt trên input volume để trích xuất ra các pattern.

Từng Filter sẽ có số channel bằng với số channel của input volume, về chiều dài và rộng của filter thì còn tùy thuộc vào đặc trưng của từng bài toán khác nhau và độ phức tạp của ảnh. Tuy nhiên kích cỡ thường thấy của filter là 3x3 hoặc 5x5

Mỗi convolutional layer có thể có nhiều filter. Số lượng filter ở mỗi convolutional layer khác nhau trong mạng cnn không được cố định từ trước mà thường phải qua thử nghiệm để xác định số filter tối ưu.

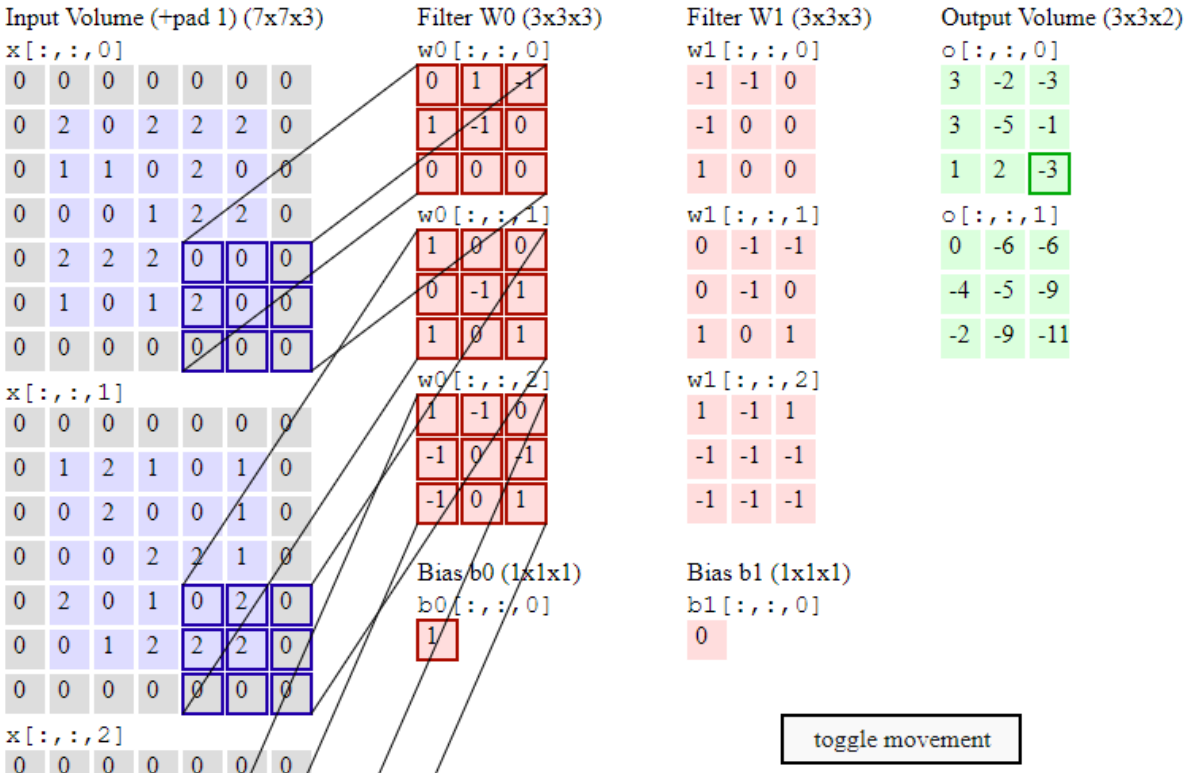
Từng filter này sẽ cùng tiến hành trượt trên input volume, và cứ mỗi filter thì sẽ sinh ra 1 ma trận 2 chiều gọi là feature map. Tính toán thế nào thì ta có thể thấy sau đây.



Filter này có 3 channel, từng channel sẽ tiến hành trượt song song trên channel tương ứng của input volume. Tổng của tích vô hướng của từng channel filter với channel ảnh sẽ là giá trị pixel mới, như ta thấy kết quả của quá trình này là 1 cái ma trận 2 chiều.

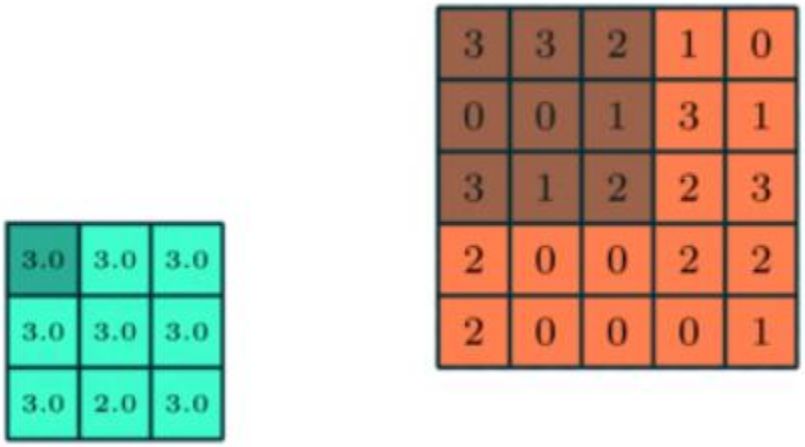
Số filter hiện có trong convolutional layer hiện tại sẽ quyết định số feature map tạo ra, khi ta chồng các feature map 2 chiều này lại với nhau, ta được một khối 3 chiều, khối 3 chiều này sẽ trở thành input volume cho layer tiếp theo.

Đây cũng là ví dụ về convolutional layer có nhiều filter, như ta thấy thì ở đây có 1 volume input có 3 channel và layer này có 2 filter, số channel của từng filter sẽ tương ứng với số channel của volume input, và tương ứng quá trình này sinh ra 2 feature map.

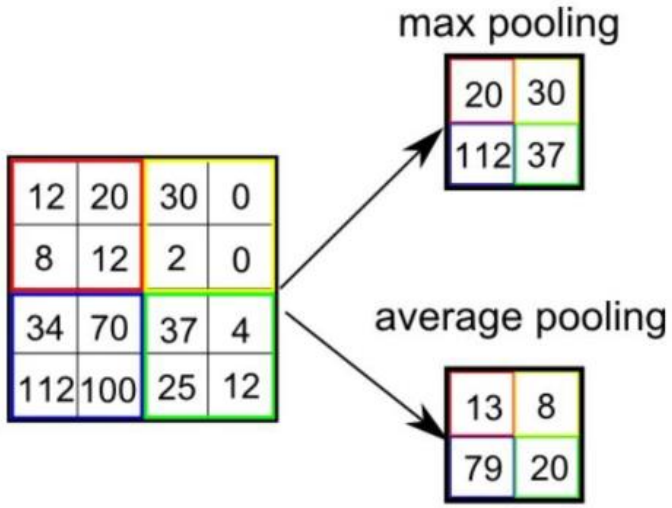


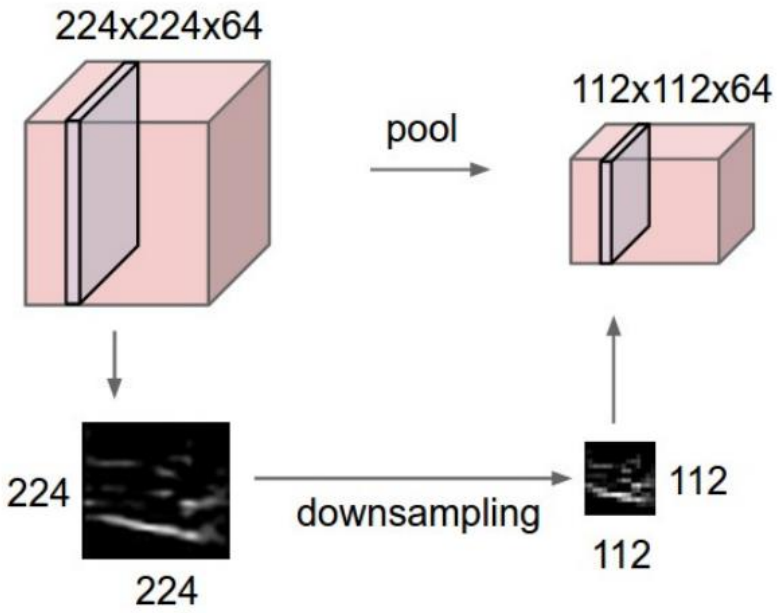
2. *Pooling Layer*

Pooling layer thường được dùng sau convolutional layer để giảm kích thước dữ liệu nhưng vẫn giữ được các đặc trưng của dữ liệu. Khi giảm kích thước dữ liệu sẽ tăng được tốc độ tính toán.



Pooling layer có kích thước là KxK với K thường là K = 2 hoặc K = 4 và hầu hết các pooling layer có kích thước là 2x2 với padding = 0, stride = 2. Và với mỗi ma trận dữ liệu, khi ta áp Pooling layer vào, ta sẽ lấy trung bình pooling hoặc max pooling. Vậy sau khi qua lớp Pooling layer, kích thước dữ liệu sẽ giảm đi 1/2 hoặc 1/4.



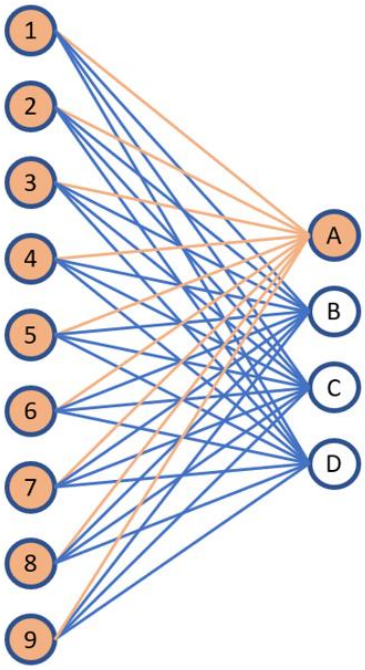


3.

Trong một số trường hợp, trong bước convolutional layer người ta có thể dùng stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.

4. Fully Connected Layer

Mỗi lớp ẩn của mạng noron được gọi là fully connected layer, theo đúng nghĩa đen, có nghĩa là mỗi neuron trong lớp ẩn đó phải liên kết với tất cả các neuron ở lớp trước. Đầu vào của lớp này thường là kết quả của Pooling layer hay Convolutional layer nên thường dùng để kết hợp các đặc điểm của ảnh để ra được output của model.



Ê

Đầu ra của Pooling layer hay Convolutional là ma trận 3 chiều, nên ta thường chuyển dạng nó thành 1 vector. Việc chuyển dạng ma trận sang 1 vector được thực hiện ở **Flatten Layer**.Vector này sau đó được liên kết với một vài fully connected layer và thực hiện các phép toán, giống như mạng thần kinh nhân tạo.

Phép toán được biểu diễn ở mỗi lớp của mạng thần kinh nhân tạo:

$$g(Wx + b)$$

Với: x là vector đầu vào (kết quả ma trận 3 chiều của Pooling layer hay Convolutional layer được chuyển dạng), chiều là [p,1] với a là số neuron của lớp trước

W: ma trận trọng số có chiều là [p,n], với n là số neuron của lớp hiện tại

b: vector hệ số bias, chiều là [p,1]

g: hàm kích hoạt, thường là ReLU

Mỗi phép toán này được lặp lại ở các lớp.

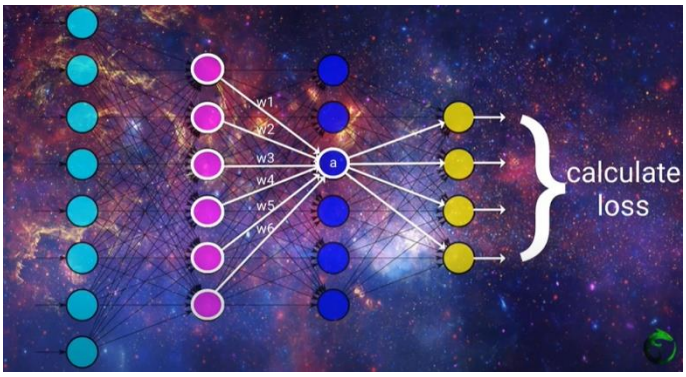
Sau khi được đưa vào fully connected layer, các lớp sẽ sử dụng hàm kích hoạt softmax (thay vì ReLU), để lấy xác suất của một lớp cụ thể ở biến đầu vào. Cuối cùng ta có được xác suất của các đối tượng thuộc các lớp khác nhau của ảnh.

5. *Training filer*

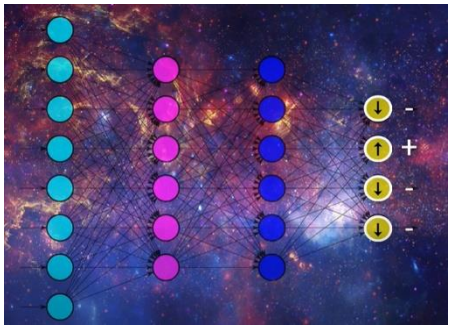
Sau khi đọc và nắm được cơ bản về cơ chế của mạng CNN, ta có thể nhận thấy vẫn còn một điểm chưa được nói đến: Đó là các giá trị weight của filter được tạo ra như thế nào?
Ý tưởng để khởi tạo filter thật ra khá đơn giản: ta chỉ việc chọn ngẫu nhiên giá trị cho filter, tiến hành tính toán đầu ra, dựa vào độ lỗi để cập nhật lại các giá trị filter.

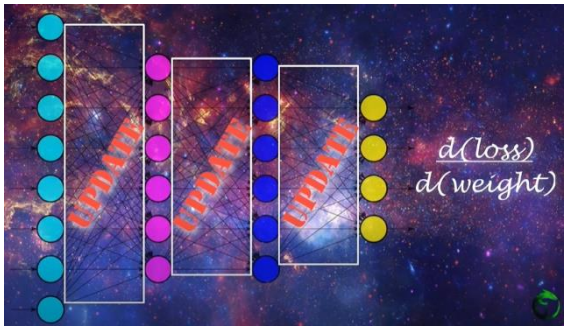
Thức tế, quá trình training filter có thể xem là một quá trình lan truyền ngược (backpropagation) với các giai đoạn như sau:

- Forward Pass: đây chính là epoc đầu hay lần train đầu tiện, lúc này filter sẽ được tạo với giá trị ngẫu nhiên. Cũng chính vì ngẫu nhiên như thế, kết quả đầu ra thường có sai số lớn và hầu như không dự đoán được gì.



- The Loss Function: sau khi có được output, ta tiến hành tính toán độ lỗi. Hàm lỗi thường sử dùng là MSE (Mean Squared Error)





$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

- Backward Pass and Update Weight: Sau khi có được độ lỗi, thuật toán tiến hành đánh giá xem đâu là yếu tố làm tăng độ lỗi, đâu là yếu tố làm giảm. Từ đó sẽ tiến hành các yếu tố (hay weight). Mục đích cuối cùng là tối thiểu độ lỗi đầu ra. Sau khi chọn được các yếu tố để điều chỉnh, độ lỗi sẽ được lan truyền ngược và các giá trị weight sẽ dần được cập nhật theo công thức

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

Reference:

https://stats.stackexchange.com/questions/154798/difference-between-kernel-and-filter-in-cnn/188216
https://stackoverflow.com/questions/9147883/only-png-supports-transparency-is-that-true
https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac
https://towardsdatascience.com/convolution-neural-network-for-image-processing-using-keras-dc3429056306
https://datahacker.rs/convolution-rgb-image/
https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480
https://www.youtube.com/watch?v=YRhxdVk_sIs
https://cs231n.github.io/convolutional-networks/#conv
https://towardsdatascience.com/convolution-neural-network-for-image-processing-using-keras-dc3429056306
https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist