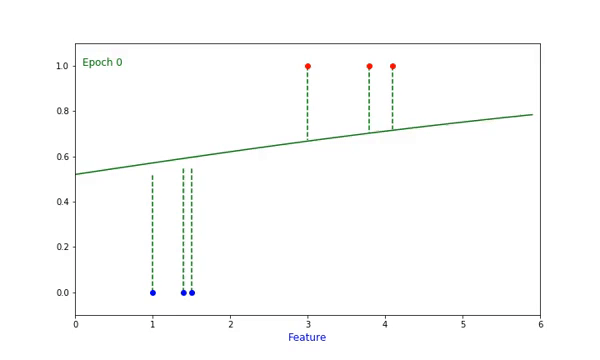
**Báo cáo tìm hiểu tuần 2**

# Logistic Regression

Thường được sử dụng cho các bài toán classification phân loại nhị phân.

**Phương pháp**:

- Sử dụng hàm sigmoid để ánh xạ giá trị x trong miền bị chặn 2 đầu [-R, R] sang miền giá trị [0, 1].



- So sánh giá trị output với một giá trị T (thường là 0.5) để quyết định lớp cho input.

**Hàm loss**: Binary cross entropy

**Nhược điểm:**

- Chỉ phù hợp cho dữ liệu liệu 2 class gần với linear separable, không thể làm được với dữ liệu phi tuyến.

- Yêu cầu dữ liệu độc lập với nhau

# Softmax Regression

Cùng với SVM là 2 thuật toán được dùng nhiều nhất cho bài toán phân loại nhiều lớp.

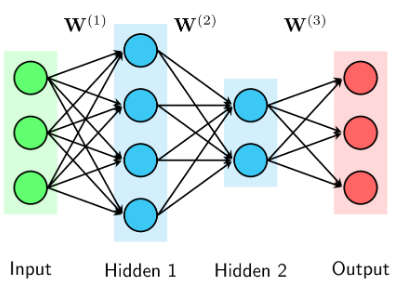
**Phương pháp:**

Dùng hàm softmax: đầu vào là một vector số thực, đầu ra là các giá trị thuộc khoảng (0,1) mà tổng của các giá trị này bằng 1.

Phân lớp của input được xác định theo giá trị output lớn nhất.

**Hàm loss:** Cross Entropy Loss

# Multi-layer Perceptron



MLP sử dụng thêm các lớp (layer) còn được gọi là hidden layer giúp mô hình phân loại tăng khả năng học hỏi từ các bộ dữ liệu phức tạp.

Các node (hay còn gọi là unit) trong hidden layer sẽ tính toán theo 2 bước:

- Tính inner product

- Dùng hàm activation

# Một số khái niệm trong mạng nơ-ron

**Layer:** Một lớp trong mạng. Số layer L = số hidden layer + 1

**Unit:** Một node tròn trong một layer.

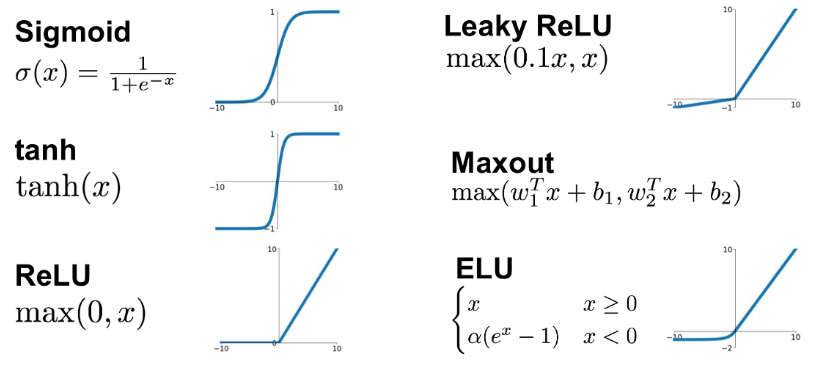
-> Số layer và unit trong mỗi layer là tùy ý.

**Weight:** tham số được truyền từ node của lớp này sang lớp khác.

**Bias:** sự khác nhau giữa giá trị trung bình các dự đoán với giá trị thực tế.

**Activation function:** hàm tính đầu ra từ đầu vào của mỗi input.

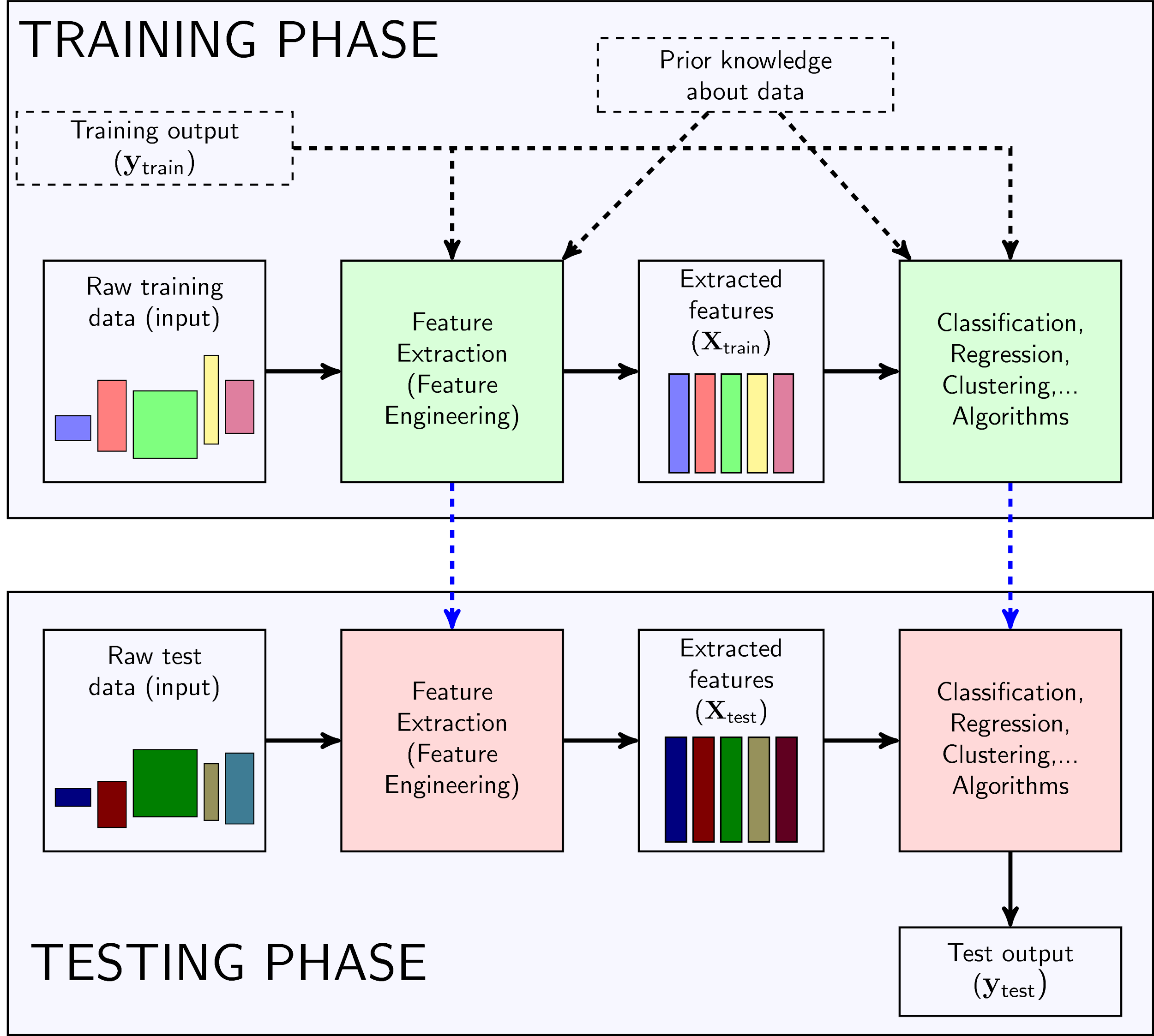
Một số activation function:



# Backpropagation

Là thuật toán sử dụng đạo hàm hàm hợp để tính toán giá trị gradient của từng layers kể từ layer cuối cùng đến layer đầu tiên.

# Feature Engineering



Là công việc chuẩn bị các **extracted features** cho việc dùng để huấn luyện các thuật toán Classification, Regression, Clustering phía sau.

Một số phương pháp:

- Trực tiếp lấy data

- Feature selection: chọn một lượng nhỏ các feature phù hợp trong một lượng lớn feature.

- Dimensionality reduction: (giảm số chiều của dữ liệu) -> giảm bộ nhớ và khối lượng tính toán.

Sử dụng **random projection**: nhân với một ma trận chiếu ngẫu nhiên để giảm số chiều và **Principle Component Analysis** để tránh mất mát thông tin.

- Bag of Words: Tạo ra một từ điển gồm các từ trong văn bản, đến xem mỗi từ xuất hiện bao nhiêu lần. -> Tạo ra một vector thông tin là chuỗi các số lượng các từ với mỗi văn bản riêng.

Nếu vector này có quá nhiều 0 có thể lưu mình vị trí các phần tử khác 0 và giá trị tương ứng.

Có thể lưu chung các từ hiếm gặp vào một phần tử <unknown> nhưng đôi khi có thể sử dụng TF-IDF để tận dụng thông tin từ các từ hiếm.

Có thể sử dụng trong Computer Vision nhưng với kiểu định nghĩa từ điểm khác.

- Feature Scaling và Normalization: chuẩn hóa dữ liệu về cùng đơn vị, độ chênh lệch...

# Tìm hiểu về keras

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense, Activation

**from** keras **import** optimizers

*# 1. Build model*

model **=** Sequential()

model**.**add(Dense(1, input\_shape**=**(2,)))

model**.**add(Activation('linear'))

*# 2. gradient descent optimizer and loss function*

sgd **=** optimizers**.**SGD(lr**=**0.1)

model**.**compile(loss**=**'mse', optimizer**=**sgd)

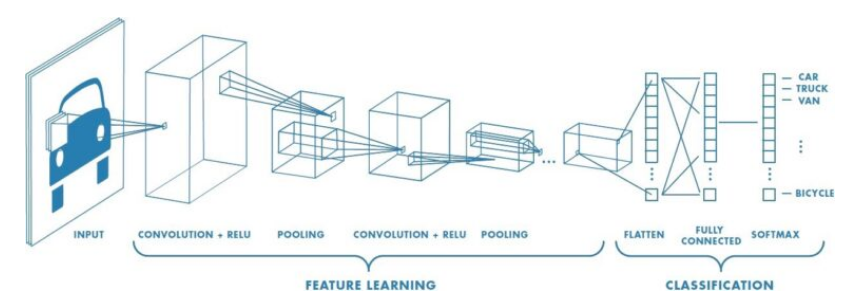
*# 3. Train the model*

model**.**fit(X, y, epochs**=**100, batch\_size**=**2)

Đây là cách xây dựng một mô hình linear regression cơ bản. Ta có thể điều chỉnh các thông số lr (learning rate), epochs, batch\_size phù hợp để thu được kết quả train tốt hơn.

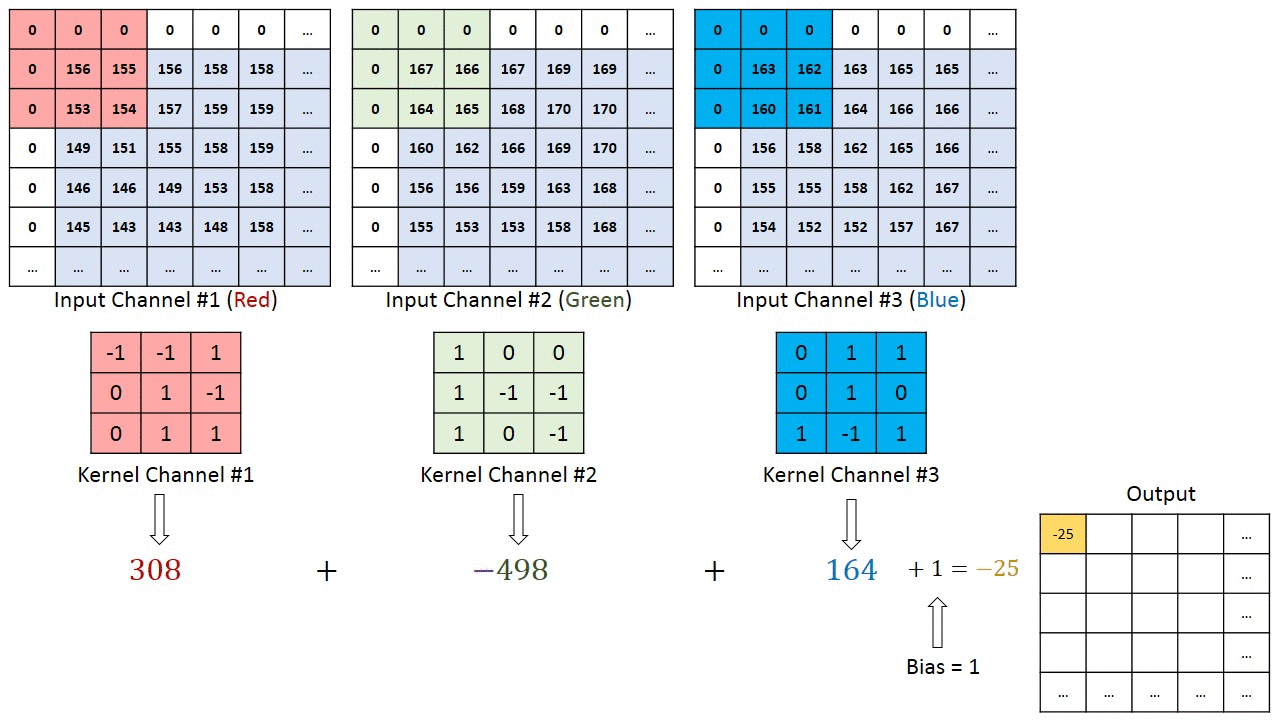
# CNN (Convolutional Neural Network)

Cấu trúc cơ bản của CNN bao gồm: convolution layer, activation layer, pooling layer và fully connected layer

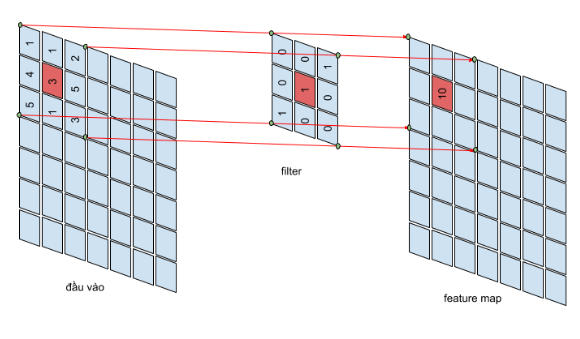


## Convolution Layer:

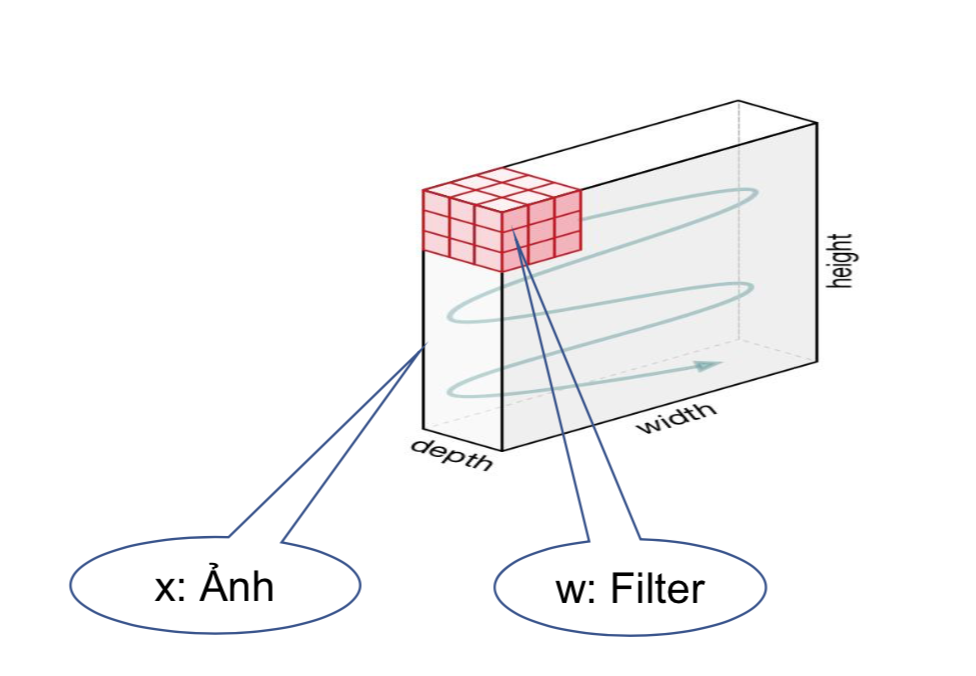
Phát hiện các đặc trưng không gian một cách hiệu quả:



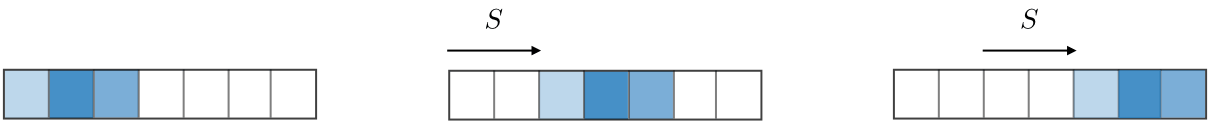
**Filter:** nhỏ, là số lẻ (1x1) (3x3) (5x5)



**2D Convolution:** Loại layer thường được sử dụng. Filter sẽ di chuyển theo 2 chiều.

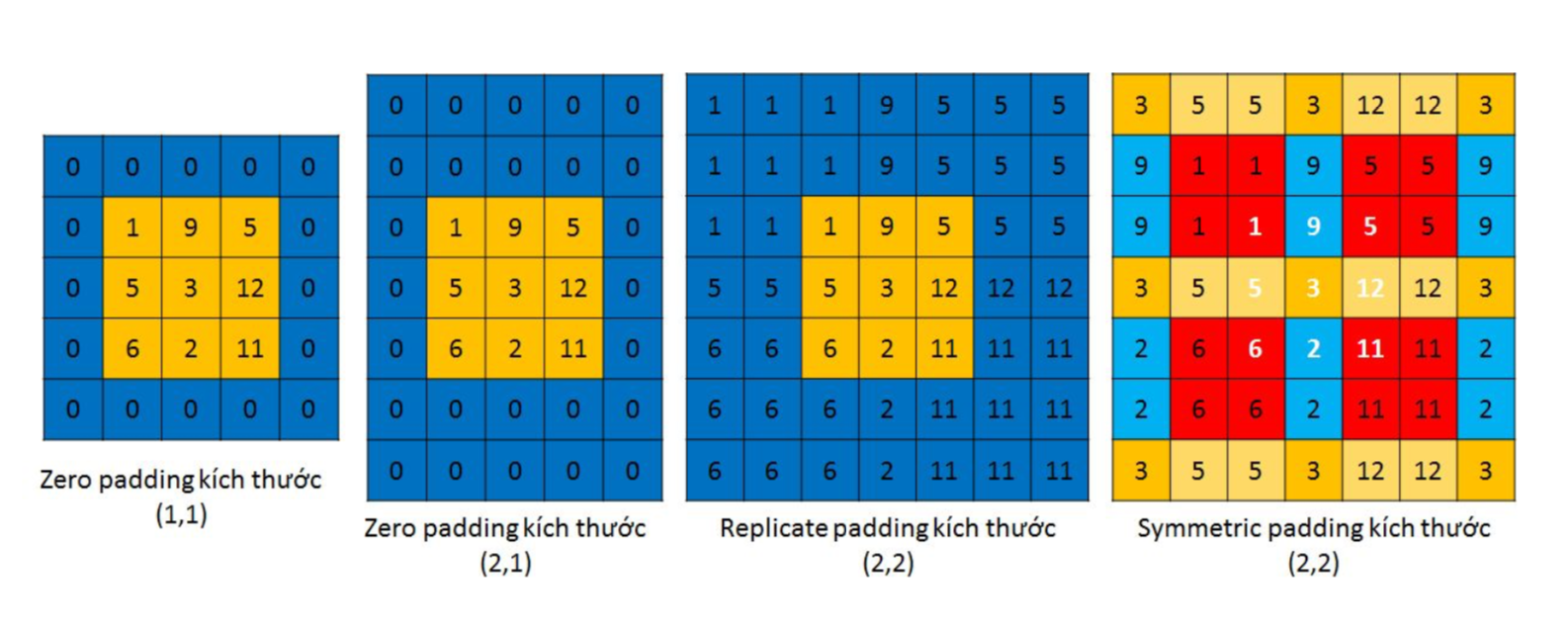


**Stride:** bước trượt của filter.



VD: S = 2

**Padding:** thêm padding cho ảnh để tránh mất thông tin ở viền và giảm tốc độ giảm của đầu ra.

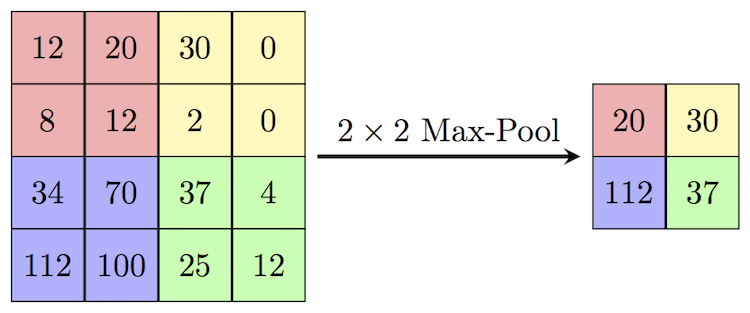


## Activation Layer

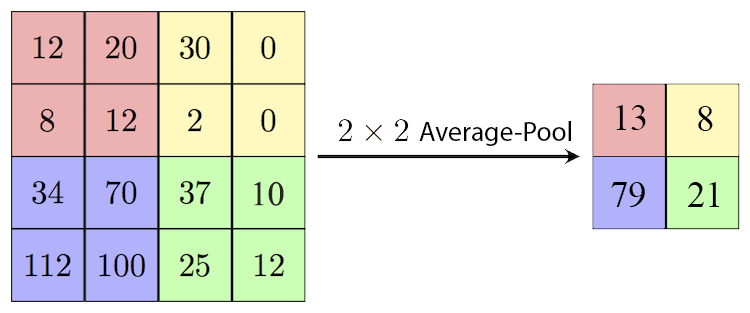
Đi theo sau tầng convolution, phổ biến là làm ReLU.

## Pooling layer

Có 2 loại:

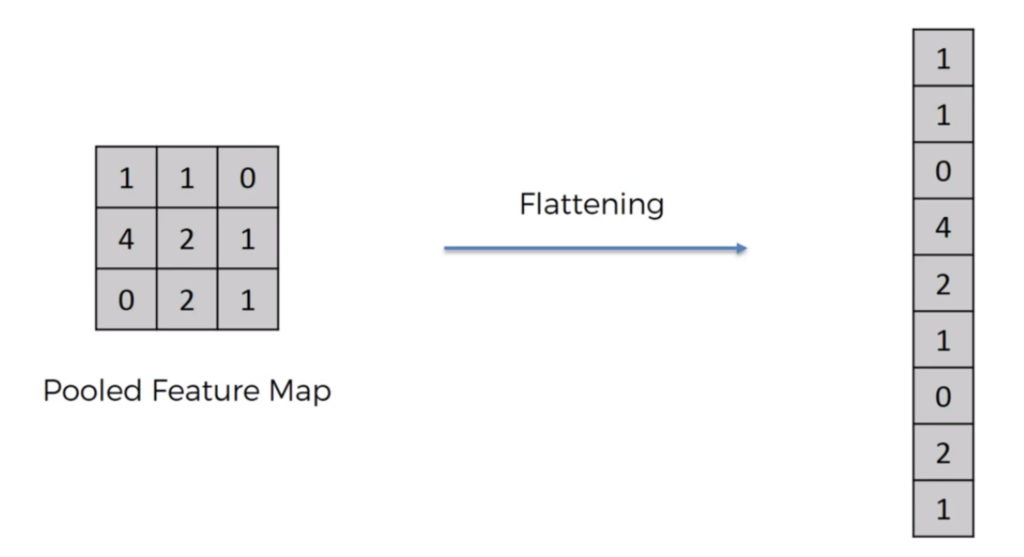
Max Pooling: 

Average Pooling:

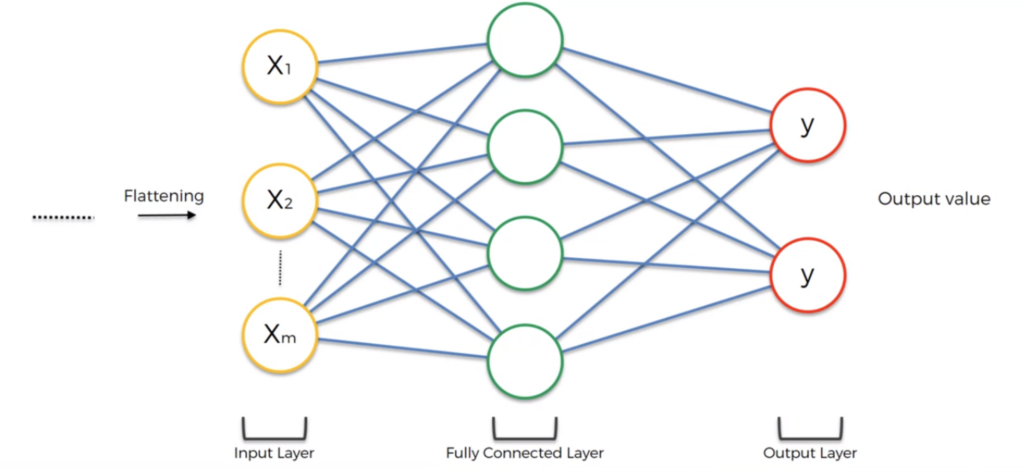


## Fully Connected

Trước khi dữ liệu được đưa vào FC layer cần được làm phẳng trước.



Bản chất FC layer cũng chính là một multi-layer perceptron.



# Batch Normalization

Chuẩn hóa các feature đầu ra mỗi layer sau khi đi qua các activation về trạng thái zero-mean (dữ liệu phân bố quanh điểm 0) với norm 1.

=> x không rơi vào khoảng bão hòa sau khi đi qua các activation.

# 

# L1, L2 Regularization

Cộng thêm tham số vào hàm mất mát. Tham số lamda được điều chỉnh để giảm overfitting. Tuy nhiên lamda lớn có thể gây ra underfitting.

|  |  |
| --- | --- |
| L1 (Lasso Regression) | L2 (Ridge Regression) |
|  |  |
| Lựa chọn đặc trưng tốt hơn.  (L1 đưa các trọng số của các feature ít liên quan về 0) | Hiệu quả hơn trong các bài toán non-sparse (nhiều phần tử khác 0) |

# 