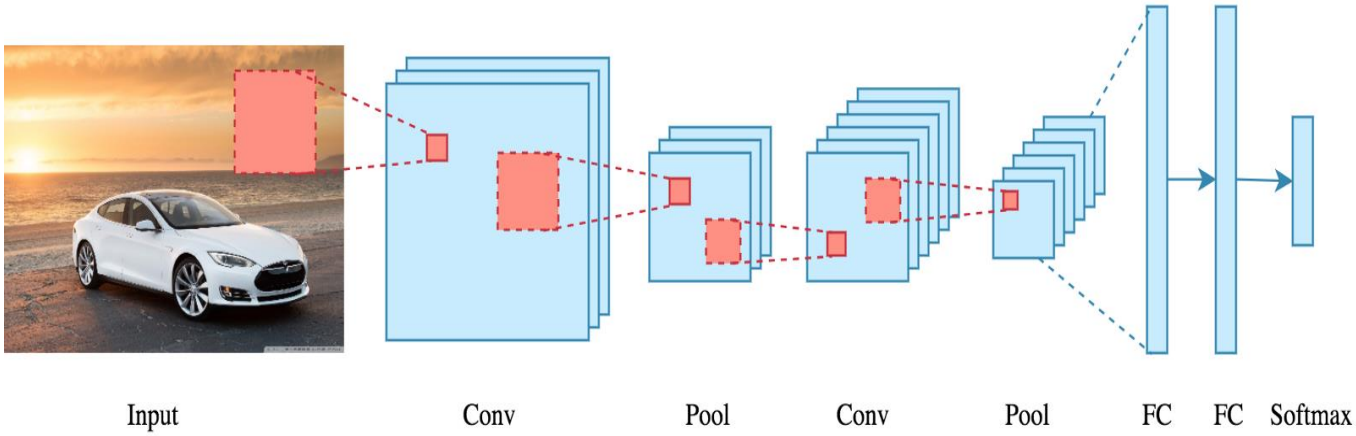
CNN (Convolutional Neural Network)

1. **Giới thiệu:**
    - CNN là mô hình được sử dụng trong các vấn đề liên quan tới hình ảnh.
    - Lợi ích chính của CNN là mô hình này có thể tự động dò ra các đặc trưng mà không cần sự giám sát của con người.
    - VD: Cho một vài tấm hình của chó và mèo. CNN sẽ dò ra các đặc trưng đặc biệt cho từng class.
    - CNN sử dụng 2 toán tử đặc biệt là:
        - Convolution.
        - Pooling.
    - **CNN có thể tự động trích đặc trưng.**
2. **Kiến trúc:**
    - Tất cả mô hình CNN đều có cùng kiến trúc như sau:

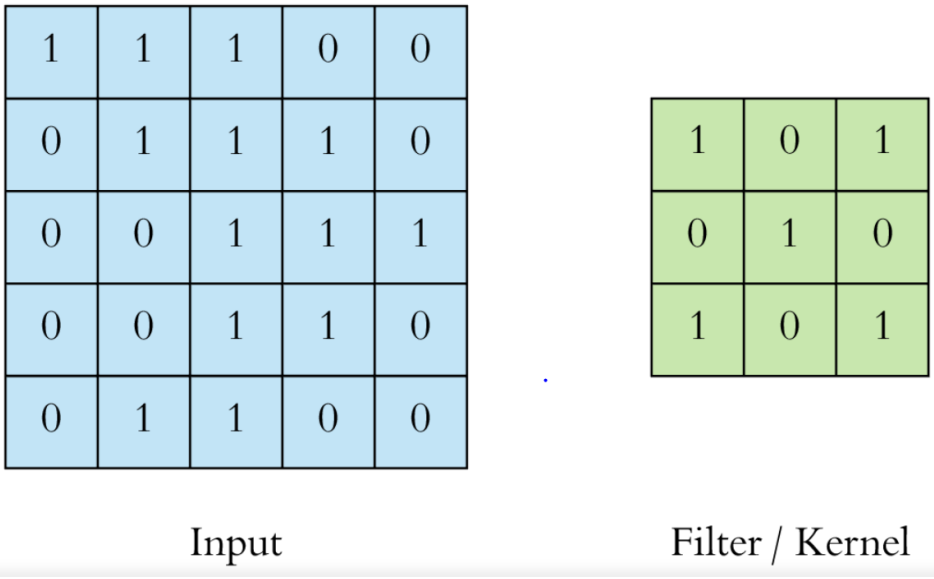


Hình 1: Kiến trúc của mô hình CNN

- Ta có input là một tấm hình. Ta biểu diễn chuỗi các toán tử Conv + Pool, theo sau đó là các Fully Connected Layers.
    - Nếu ta đang phân nhiều lớp thì output layer sẽ dùng hàm Softmax.

- **Convolution:**
    - Phần chính của CNN là **Convolutional Layer.**

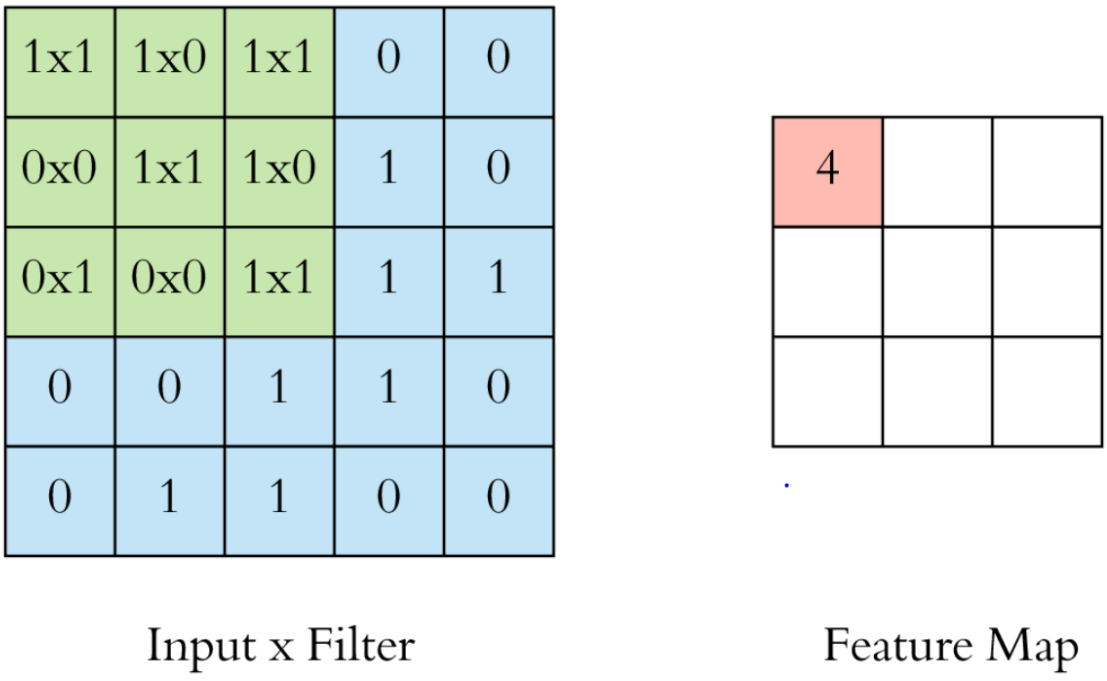
- Convolution là một toán tử toán học dùng để "hợp nhất" 2 tập dữ liệu.
  - Trong toán tử Convolution, một **Feature Map** được tạo ra nhờ áp dụng một **Convolution Filter/ Kernel** lên dữ liệu đầu vào.



Hình 2: Input và Filter/ Kernel

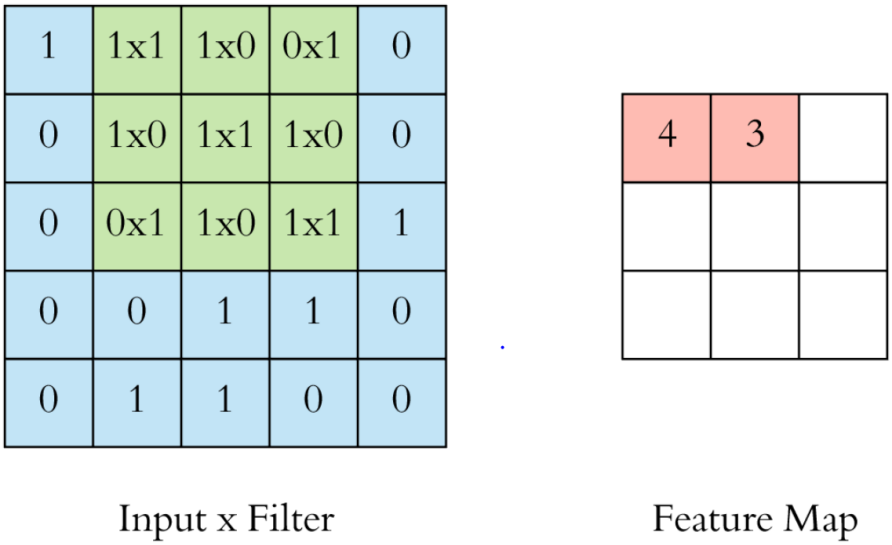
  - Ở hình 2, ta thấy Input là dữ liệu đầu vào của **Convolution Layer**. Đây được gọi là **3x3 Convolution** vì kích thước của ma trận **Filter** là 3x3.

- Cách hoạt động của toán tử Convolution:



Hình 3: Cách hoạt động của toán tử Convolution

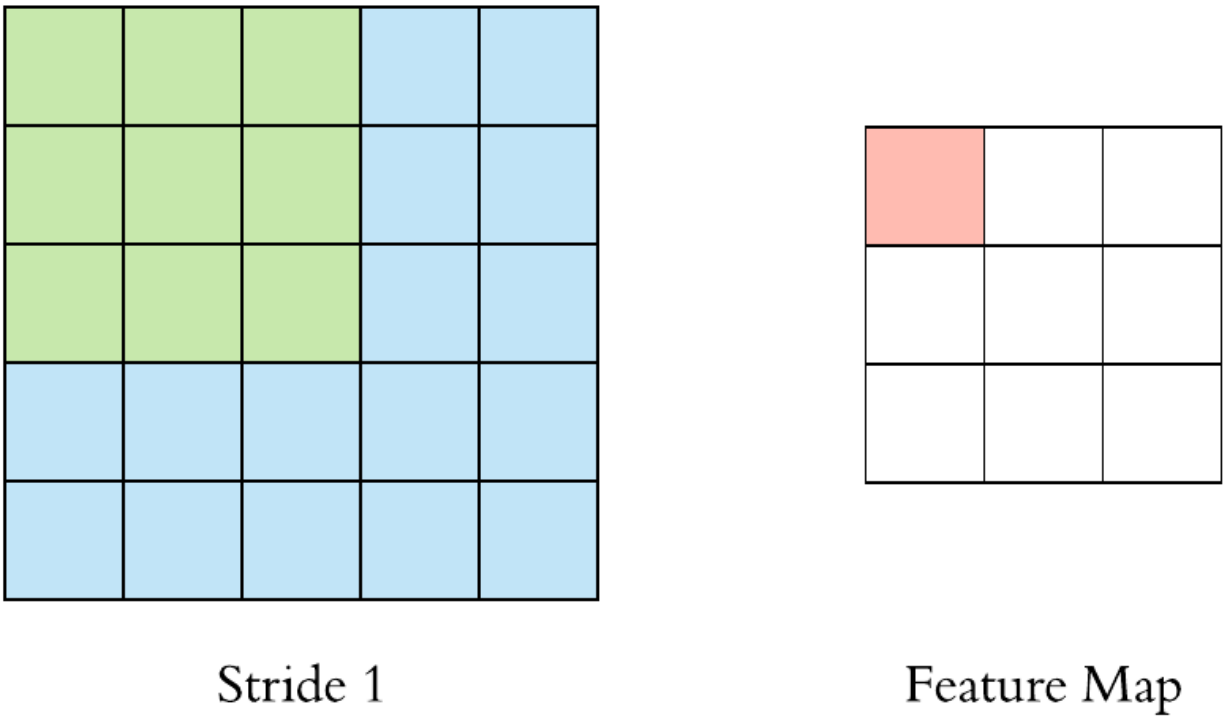
- Giải thích hình 3, tức là ta "cầm" **Filter** đặt lên ma trận con 3x3 trong Input. Sau đó nhân tương ứng với nhau và cộng tổng tất cả các tích đó lại và đem tổng đó sang **Feature Map.**
- Bước tiếp theo, ta tiếp tục "cầm" **Filter** lướt sang ma trận con 3x3 tiếp theo trong input và cứ như vậy cho tới khi **Feature Map** "đầy đủ số liệu".



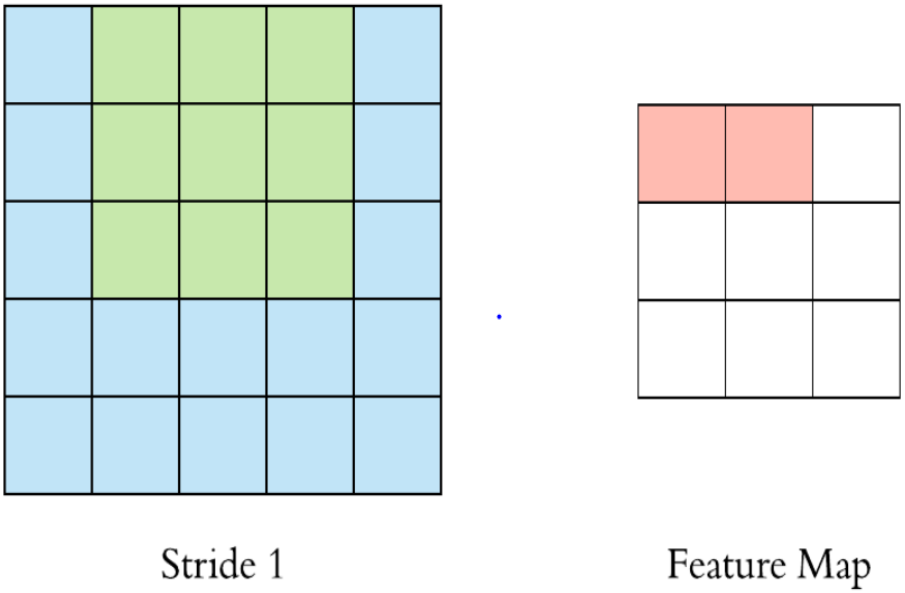
Hình 4: Cách hoạt động của toán tử Convolution (tiếp theo)

- Trên là ví dụ về toán tử **Convolution 2D** sử dụng ma trận **Filter 3x3**. Trong thực tế, toán tử 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) và chiều sâu (Depth), và chiều sâu (Depth) đại diện cho màu sắc (RGB).

- **Stride:**
  - **Stride** chỉ rõ mỗi bước thì **Filter** di chuyển bao nhiêu. Mặc định là 1.
  - VD: Stride = 1

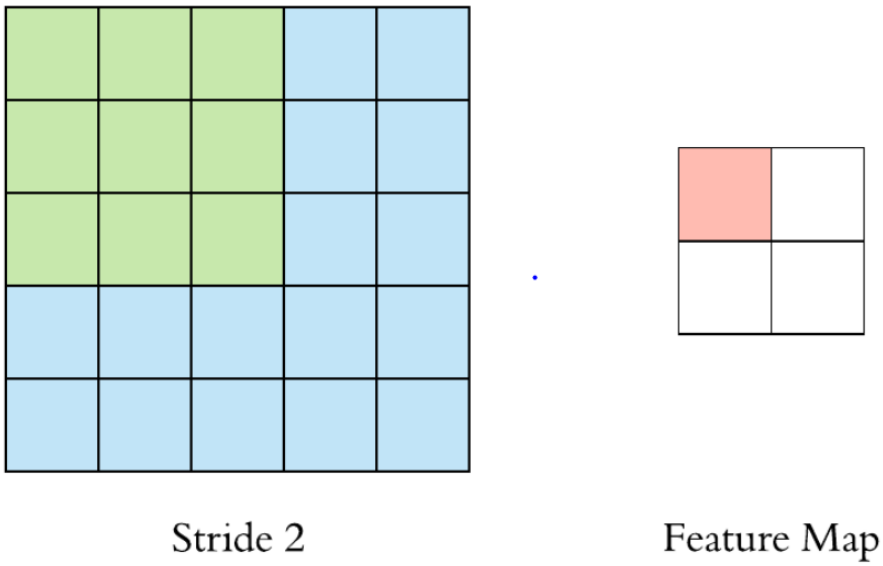


Hình 5: Stride = 1 (1)

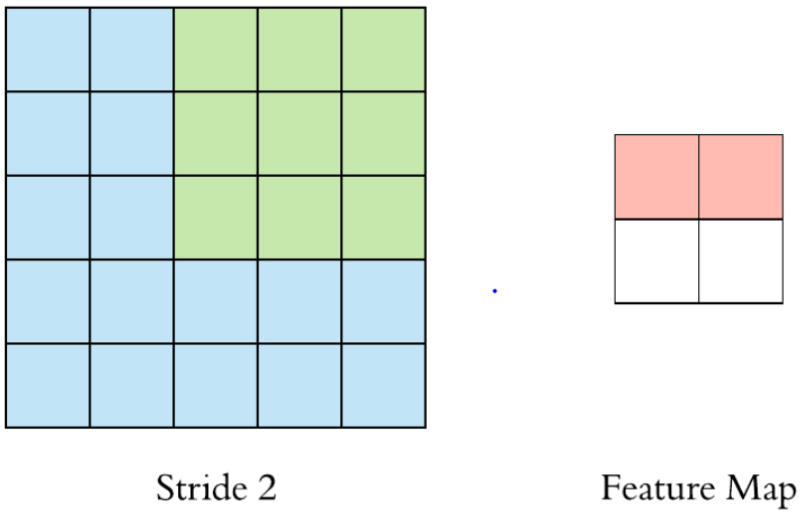


Hình 6: Stride = 1 (2)

- Ta có thể chọn **Stride** lớn hơn nếu ta không muốn có sự trùng lặp các giá trị ở các ô bị quét => Điều này dẫn tới việc **Feature Map** được tạo ra sẽ nhỏ hơn như ví dụ **Stride = 2** dưới:

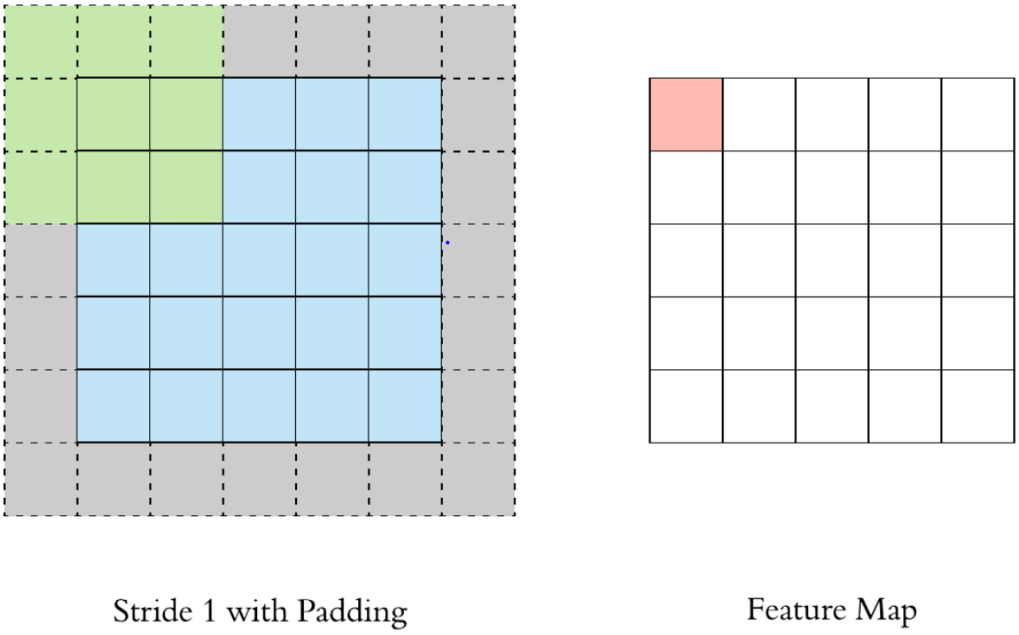


Hình 7: Stride = 2 (1)



Hình 8: Stride = 2 (2)

- **Padding:**
  - Ta thấy ở hình 7 và 8, với **Stride = 2** thì **Feature Map** nhỏ hơn input. Vậy nếu ta muốn duy trì **Feature Map** có cùng kích thước với input thì có thể dùng **Padding** để bao bọc input với các phần tử có giá trị là 0.
  - VD với **Stride = 1** và **Padding:**



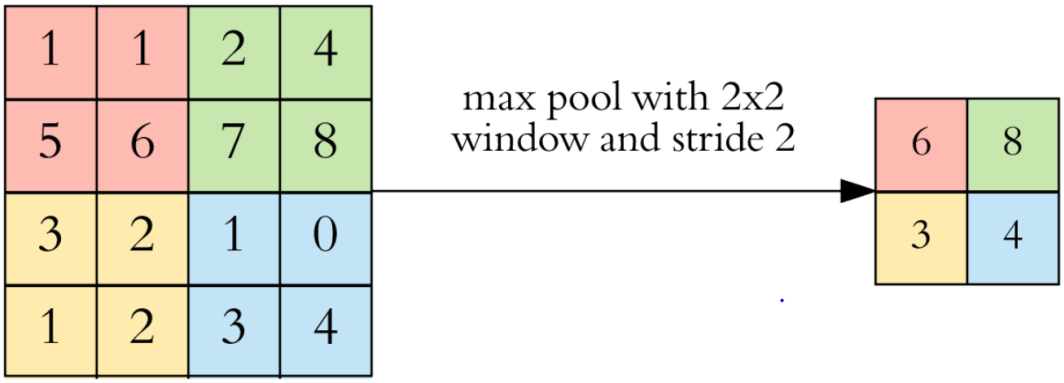
Hình 9: Stride = 1 và Padding

    - Ở ví dụ trên, ta thấy **Feature Map** tạo ra có kích thước bằng với kích thước của input.
  - **Padding** thường được sử dụng trong CNN để bào toàn kích thước của **Feature Map**.

- **Pooling:**
  - Sau mỗi phép toán **Convolution**, ta thường biểu diễn phép toán **Pooling** để giảm kích thước của **Feature Map** tạo ra sau phép toán **Convolution.** Các **Pooling Layers** làm giảm (Downsample) mỗi **Feature Map** một cách độc lập, đối với input 3D thì làm giảm

chiều cao (Height) và chiều rộng (Width) nhưng giữ nguyên chiều sâu (Depth).

- Dạng **Pooling** phổ biến nhất là **Max Pooling**. **Max Pooling** chỉ lấy giá trị lớn nhất trong cửa sổ **Pooling.** Trái ngược với **Convolution, Pooling** không có tham số (tức là trong phép toán **Convolution,** thì các tham số nằm trong **Convolution Filter**). Cửa sổ **Pooling** trượt trên input của nó và đơn giản chỉ lấy giá trị lớn nhất trong cửa sổ. Tương tự như **Convolution,** ta cần khai báo kích thước cửa sổ và **Stride.**
- VD: **Stride = 2** và cửa sổ **Pooling** là 2x2



Hình 10: Phép toán Pooling

- Để ý thấy **Feature Map** sau phép toán **Pooling** trên có kích thước bằng nửa kích thước của input => Đây là tác dụng chính của **Pooling**, làm giảm kích thước của **Feature Map** trong khi vẫn giữ những thông tin quan trọng.

- **Fully Connected:**
  - Sau các **Convolution + Pooling Layers,** ta thêm một vài **Fully Connected** để "gói" cấu trúc của CNN lại.
  - **LƯU Ý:**
    - Output của **Pooling Layer** cuối cùng có thể là 3D hoặc 2D. Nhưng một **Fully Connected** thì chỉ nhận đầu vào là vector

1D => Nên ta sẽ "làm phẳng" (**Flattening**) output thành vector 1D.

3. **Intuition:**
    - Một mô hình CNN có thể được xem là sự kết hợp giữa hai thành phần là:
        - Phần trích feature.
        - Phần phân lớp.
    - VD: Cho một tấm hình chó
        - **Convolutional Layer:** Dò đặc điểm như hai mắt, tai dài, bốn chân, đuôi ngắn, v.v….
        - **Fully Connected Layer:** Sử dụng các feature trích được lấy từ **Convolutional Layer** để phân lớp và gắn xác suất khả năng trong hình là chó.

- Nguồn tham khảo: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2