

# Convolution neural network

## 1. Phép nhân tích chập convolution

Phép nhân tích chập convolution là một kỹ thuật quan trọng xử lý ảnh số (digital image processing). Nó xuất hiện trong các thuật toán xử lý ảnh như làm mờ (blur), làm nét (sharpen), làm rõ đường nét (edge detection).

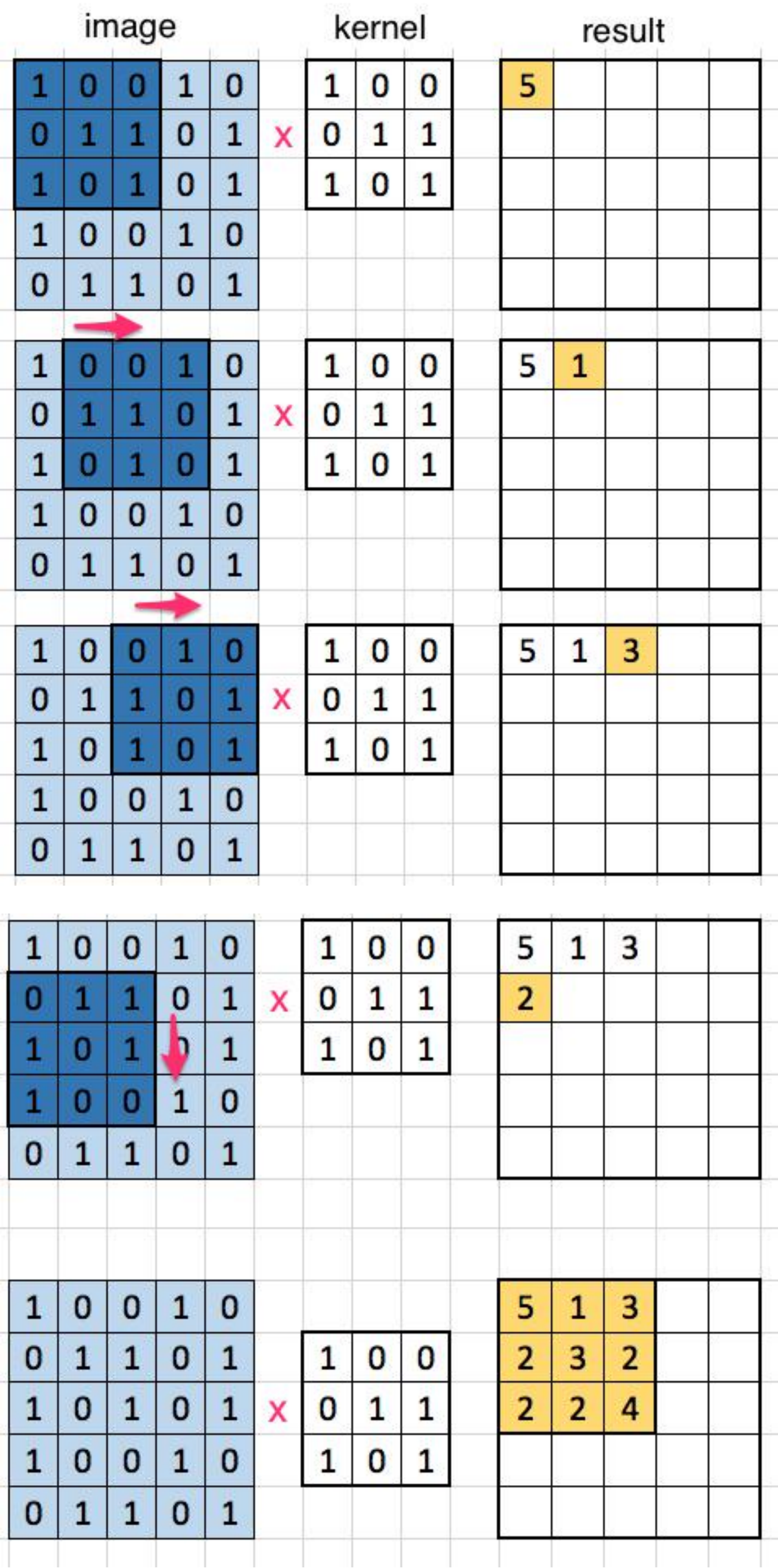
Ngoài ra, với sự phát triển của deep learning, các mô hình xử lý ảnh computer vision cũng sử dụng rất nhiều các mạng nơ ron được xây dựng từ phép convolution. Convolution layer là một tầng biến đổi ma trận đầu vào để làm rõ và tách ra các đặc tính của hình ảnh mà vẫn bảo toàn tính tương quan không gian giữa đầu ra và đầu vào.

[Đây](#) là một bài blog khá hay về trực quan hoá phép convolution trên ảnh.

### 1.1. Công thức của phép nhân tích chập convolution

Công thức của phép convolution được tính như sau:

$$y_{11} = (x_{11} * w_{11} + x_{12} * w_{12} + x_{13} * w_{13}) + (x_{21} * w_{21} + x_{22} * w_{22} + x_{23} * w_{23}) + (x_{31} * w_{31} + x_{32} * w_{32} + x_{33} * w_{33})$$

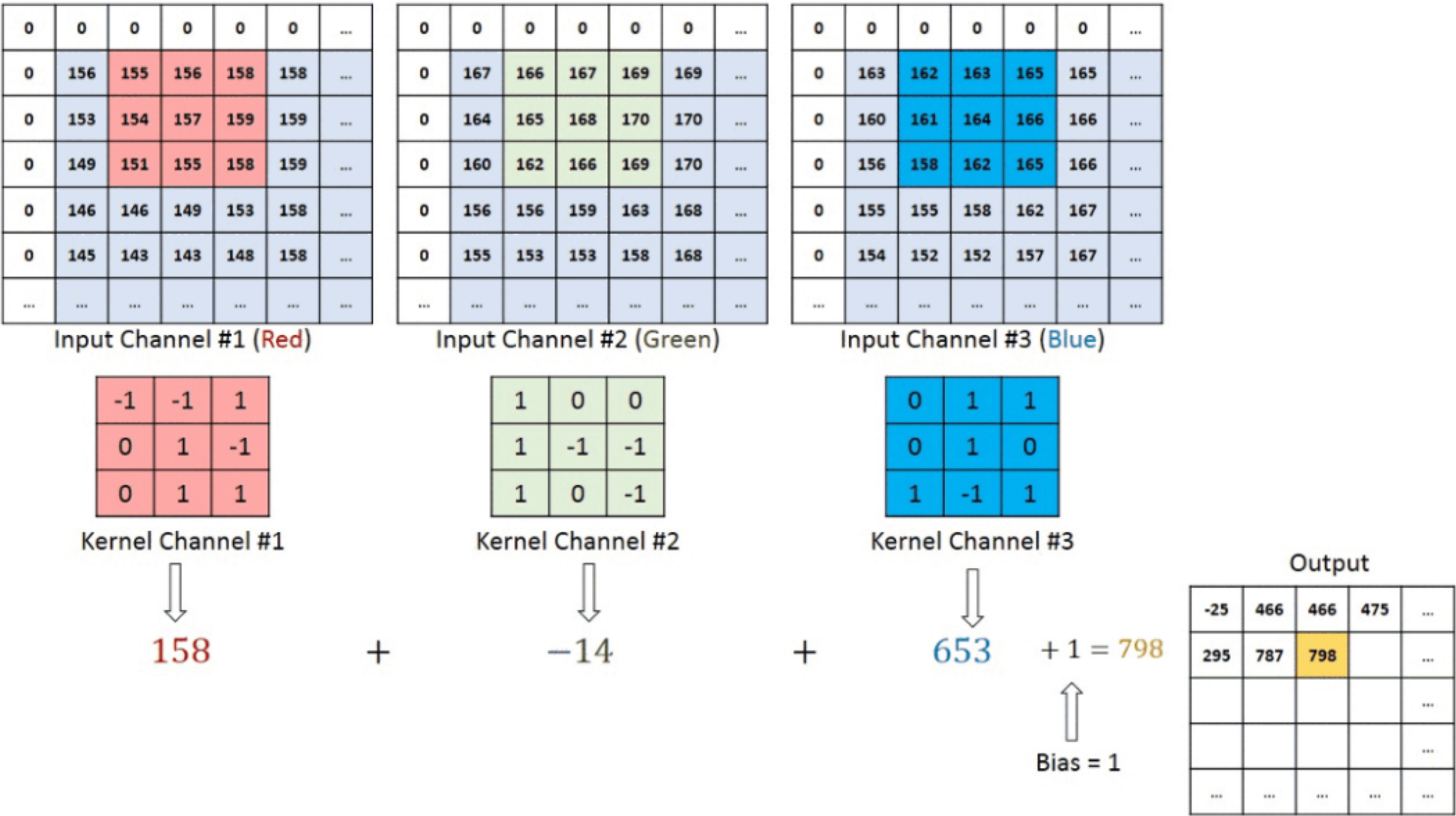


### 1.2. Tính toán tích chập đối với ảnh RGB



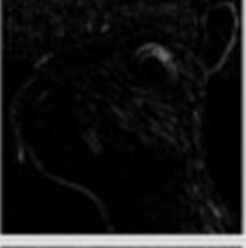
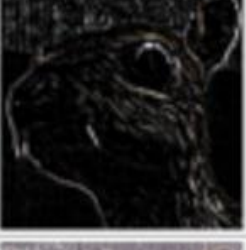



Đối với ảnh RGB, thay vì được đại diện bởi 1 con số, mỗi pixel được đại diện bởi 3 con số đại diện cho giá trị tương ứng lần lượt của màu đỏ (Red), xanh lá (Green) và xanh dương (Blue).

Do đó, phép convolution cũng phức tạp hơn, công thức của phép convolution đối với ảnh RGB được tính như sau:

$$y_R = (x_{R11} * w_{R11} + x_{R12} * w_{R12} + x_{R13} * w_{R13}) + (x_{R21} * w_{R21} + x_{R22} * w_{R22} + x_{R23} * w_{R23}) + (x_{R31} * w_{R31} + x_{R32} * w_{R32} + x_{R33} * w_{R33})$$
$$y_G = (x_{G11} * w_{G11} + x_{G12} * w_{G12} + x_{G13} * w_{G13}) + (x_{G21} * w_{G21} + x_{G22} * w_{G22} + x_{G23} * w_{G23}) + (x_{G31} * w_{G31} + x_{G32} * w_{G32} + x_{G33} * w_{G33})$$
$$y_B = (x_{B11} * w_{B11} + x_{B12} * w_{B12} + x_{B13} * w_{B13}) + (x_{B21} * w_{B21} + x_{B22} * w_{B22} + x_{B23} * w_{B23}) + (x_{B31} * w_{B31} + x_{B32} * w_{B32} + x_{B33} * w_{B33})$$
$$y = y_R + y_G + y_B + b$$



1.3. Một số kernel đặc biệt trong convolution

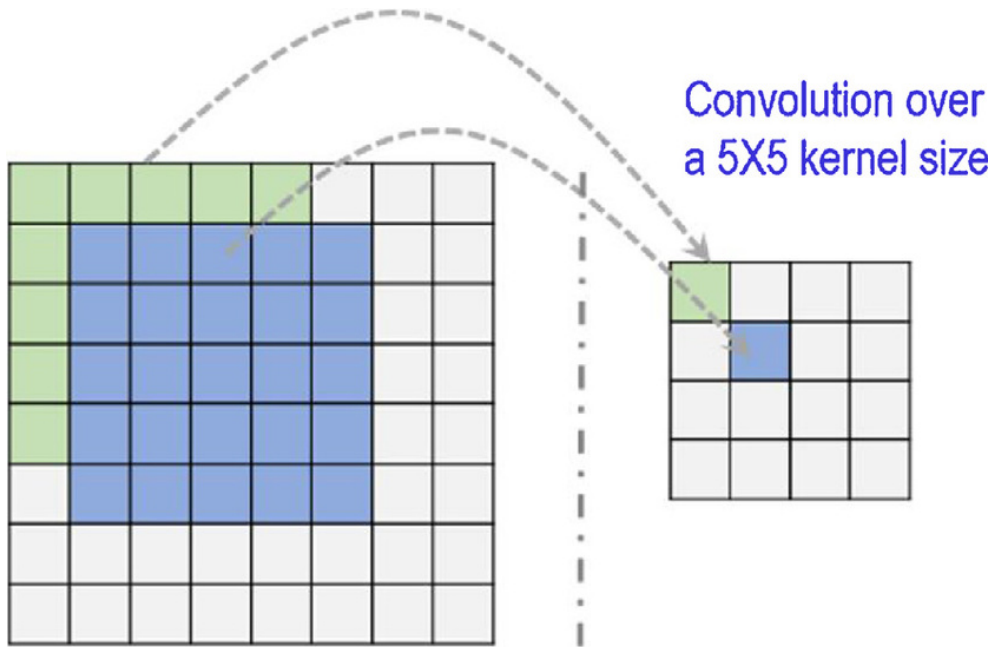
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

1.4. Các tham số quan trọng của phép convolution

1.4.1. Kernel size

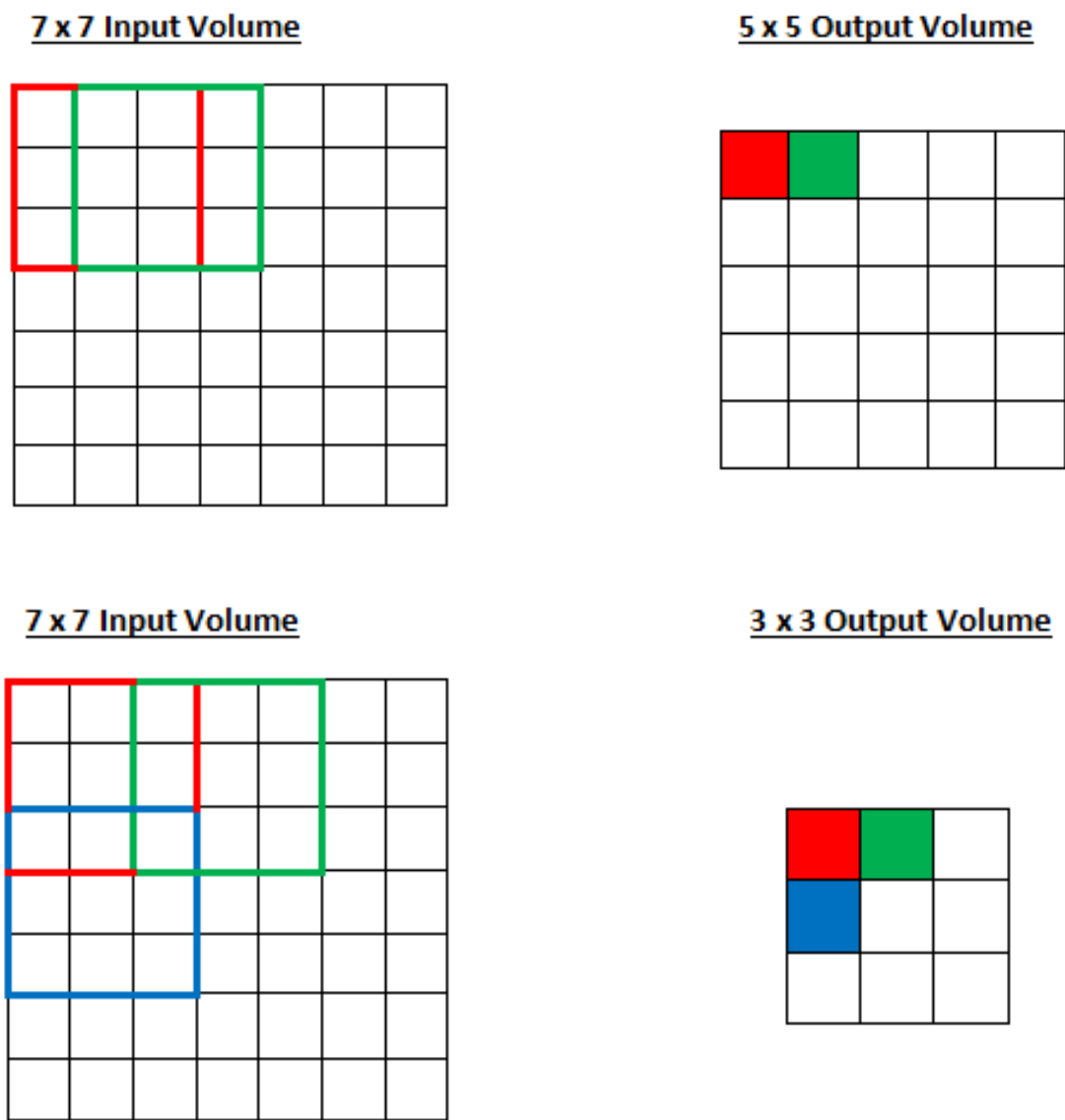
Kernel size là kích thước của ma trận được sử dụng làm kernel. Thông thường kernel là ma trận vuông, có kích thước là  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$  ... Trong một số trường hợp đặc biệt, kernel có thể là ma trận chữ nhật.





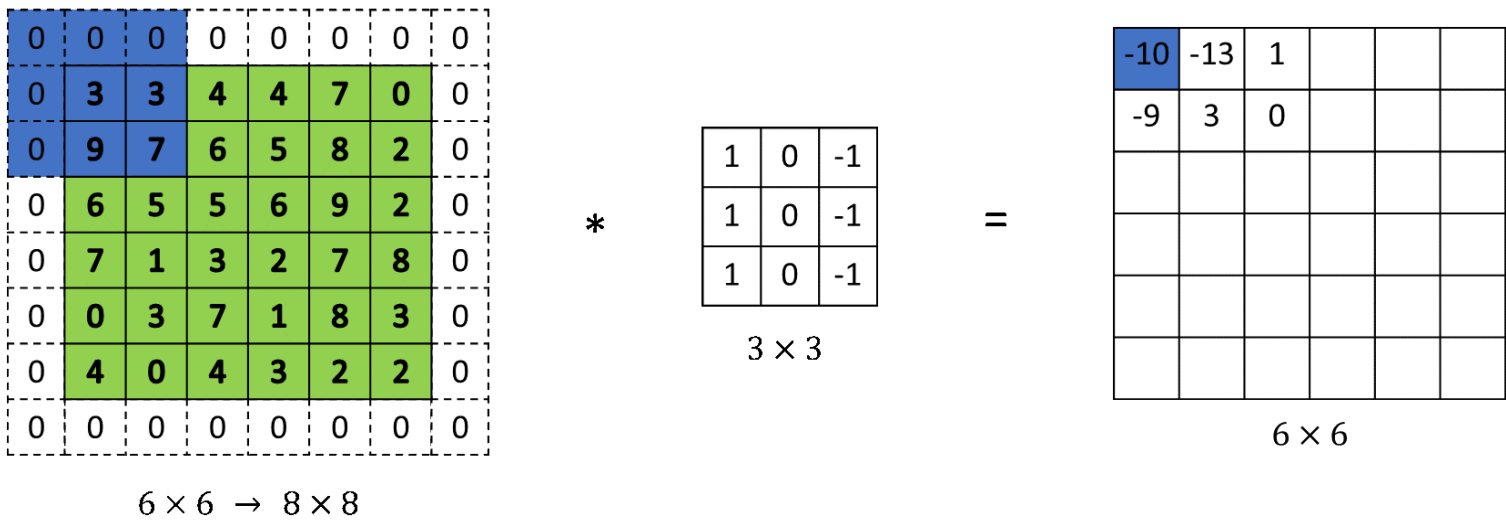
#### 1.4.2. Stride

Stride có thể được gọi là bước nhảy trong phép tính convolution. Với stride bằng 1, ta dịch chuyển kernel đến pixel ngay tiếp theo để tiếp tục tính toán, trong khi với stride bằng 2, ta dịch chuyển 2 bước.



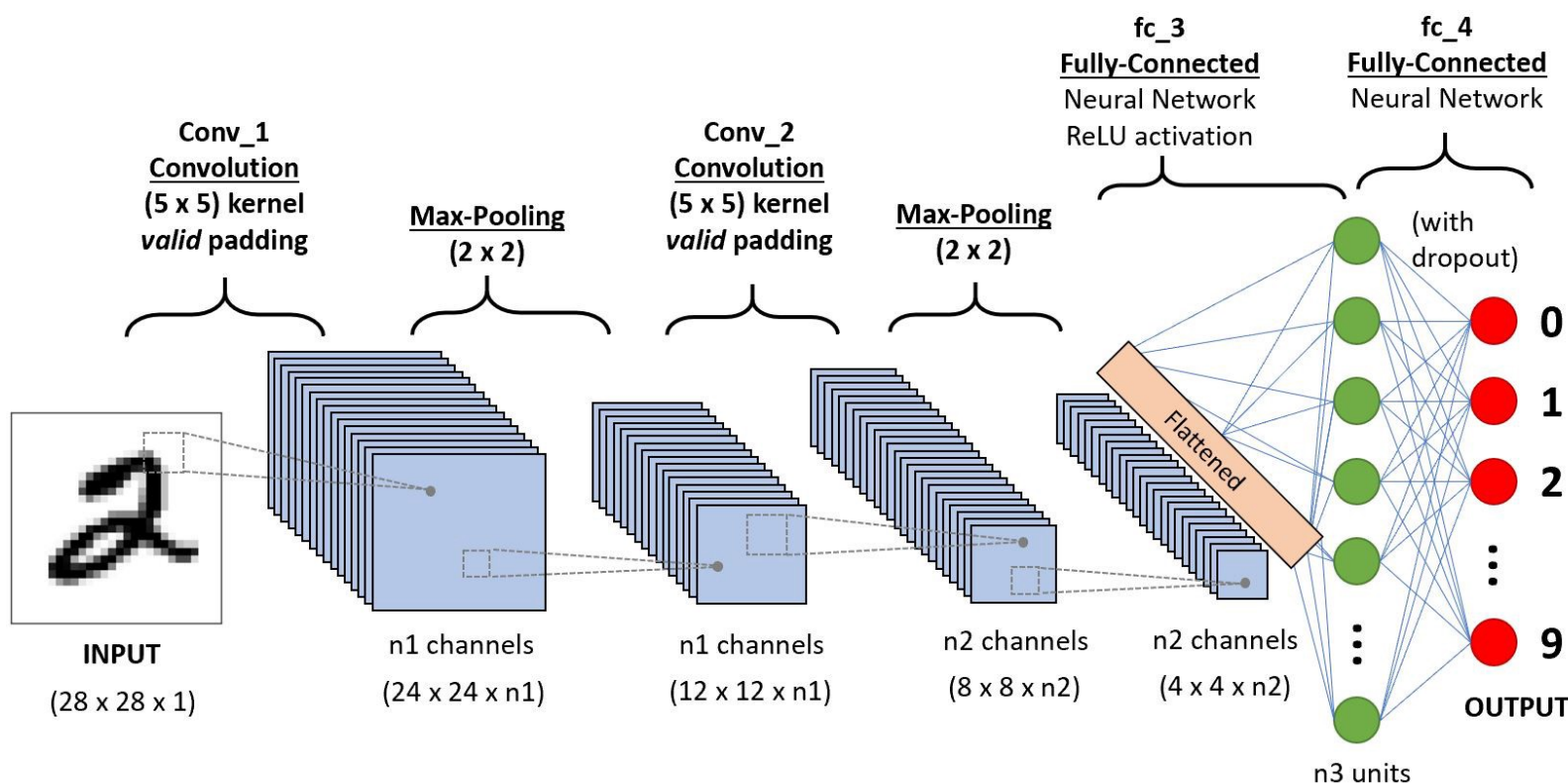
#### 1.4.3. Padding

Padding là thao tác mà ta bổ sung thêm một số pixel vào xung quanh của ma trận đầu vào trước khi tính toán convolution. Padding giúp cho kích thước của ma trận đầu ra giống với kích thước của ma trận đầu vào. Có một số kiểu padding như black padding, reflect padding ...



## 2. Các layer trong convolution neural network (CNN)

Kiến trúc khái quát của một convolution neural network CNN được mô tả như sau:



### 2.1. Convolution layer

Convolution layer là layer cơ bản nhất trong CNN, giúp thực hiện phép convolution.

Điểm khác biệt của convolution layer so với một phép convolution đơn giản là convolution layer có thể chứa rất nhiều các kernel khác nhau và thực hiện nhiều phép convolution khác nhau. Do đó, trong convolution layer, ta có thể một tham số về số lượng kernel được sử dụng trong layer convolution đó (trong một số thư

viện deep learning, tham số này được gọi là out\_channel).

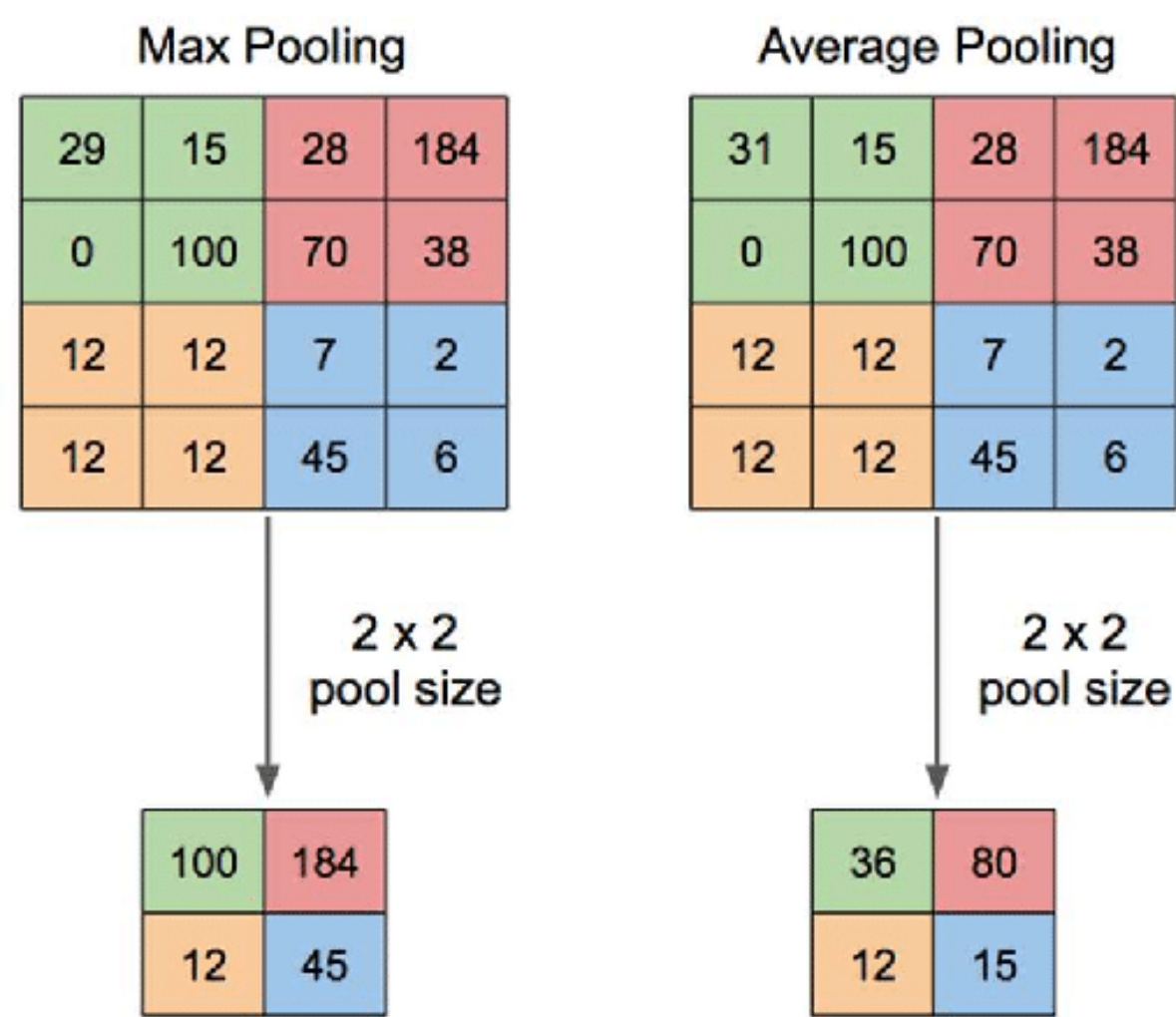
2.2. Pooling layer

Pooling layer là layer giúp giảm kích thước của feature maps trong CNN.

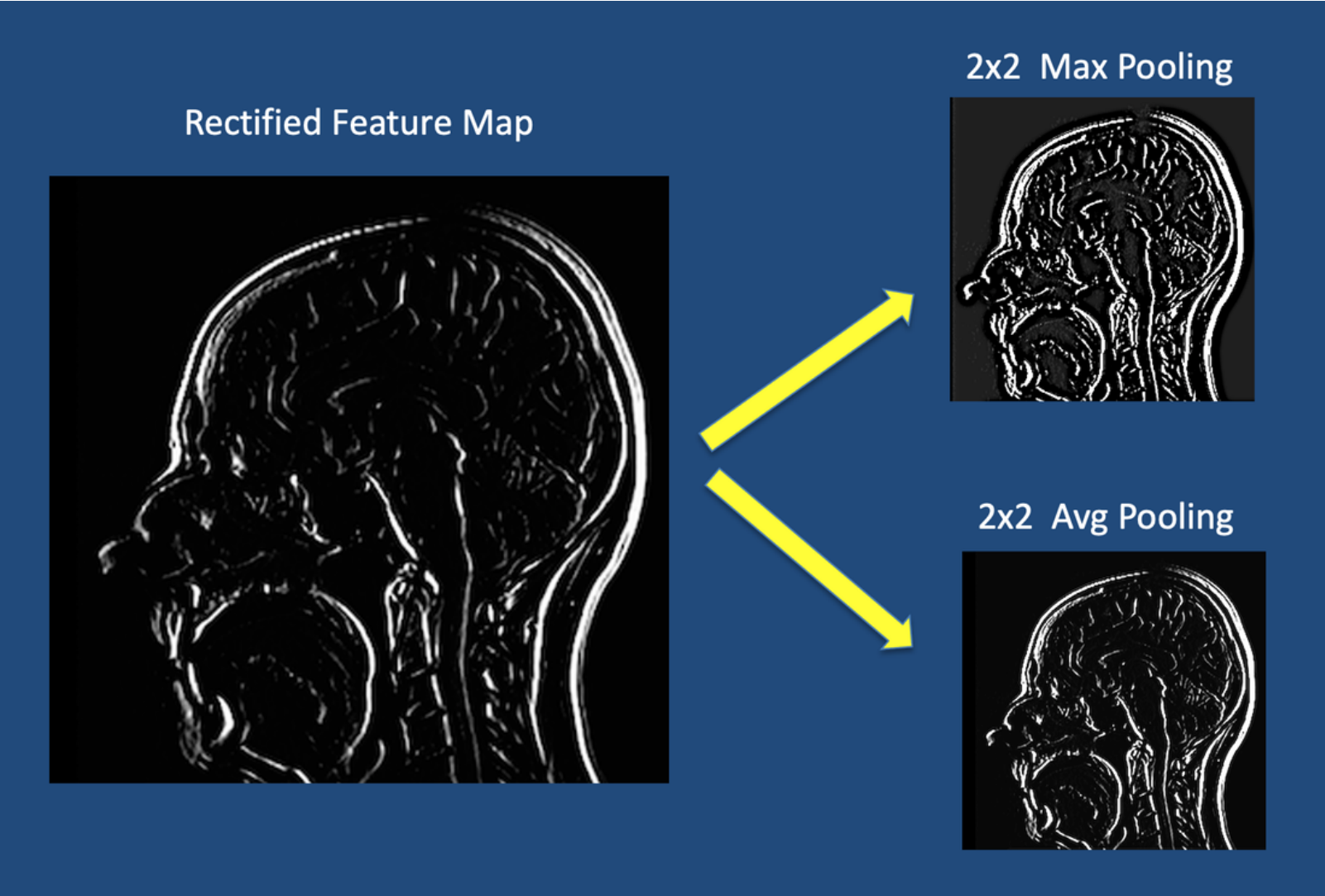
Pooling layer chia feature maps thành các ô và thực hiện phép tính toán trên từng ô.

2.2.1. Max pooling và Average pooling

Max pooling và Average pooling chia feature maps thành các ô có kích thước là một tham số được xác định trước.



Max pooling lựa chọn giá trị max của mỗi ô làm giá trị đầu ra thì Average pooling tính trung bình các giá trị của mỗi ô làm giá trị đầu ra.



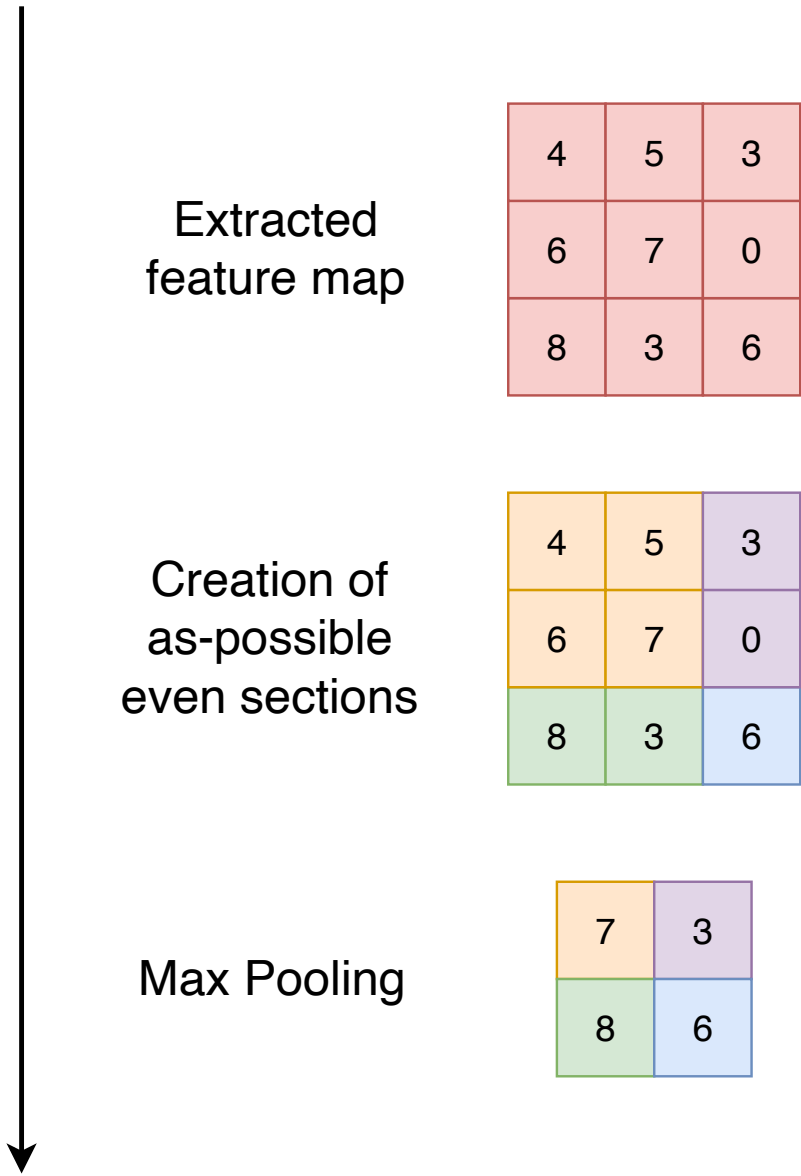
Điểm mạnh của max pooling là giúp làm rõ hơn các chi tiết sáng, tuy nhiên điều này kéo theo việc max pooling làm mất đi các chi tiết ít sáng hơn.

Trong khi đó, average pooling gần như sao chép y hệt hình ảnh input ra output nhưng với kích thước nhỏ hơn.

2.2.3. Adaptive pooling - RoI pooling

Với cách chia feature maps của max pooling và average pooling cơ bản ở trên, ta thu được output có tỷ lệ kích thước tương đối giống với tỷ lệ kích thước của input.

Adaptive pooling tiếp cận việc chia feature maps theo một cách khác. Adaptive pooling xác định trước kích thước đầu ra, sau đó chia đều feature maps input theo tỷ lệ kích thước của output. Điều này giúp cho ta luôn đảm bảo được chính xác kích thước của output với input có kích thước bất kỳ.



Sau khi thực hiện chia feature maps thành các ô, Adaptive pooling cũng sẽ thực hiện max hoặc average pooling trên từng ô.

### 2.3. Flatten layer

Trong CNN, ta thường xuyên làm việc với các ma trận input và output, tuy nhiên, để đưa ra giá trị dự đoán cuối cùng, CNN cần có một layer giúp biến đổi các ma trận nhiều chiều thành vector.

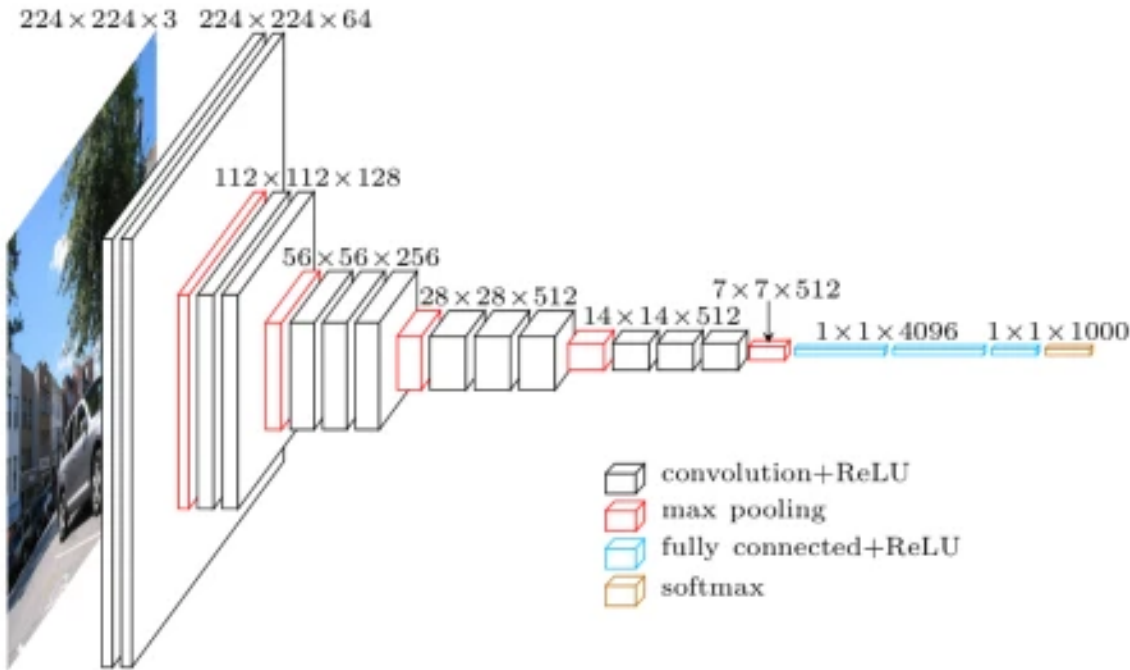
Flatten layer lần lượt nối các giá trị của ma trận output thành một vector dài nhằm đưa các giá trị này qua các linear layer và cho ra kết quả dự đoán cuối cùng của mô hình CNN.

## 3. Các kiến trúc mô hình CNN nổi tiếng

### 3.1. VGG

VGG viết tắt của Visual Geometry Group, tổ chức đã nghiên cứu và công bố mô hình VGG.

Có hai phiên bản nổi tiếng của VGG là VGG-16 và VGG-19. 16 và 19 ở đây đại diện cho số lượng các layer convolution trong mô hình.



VGG là một mô hình có kiến trúc đơn giản, phù hợp với một số các bộ dữ liệu nhỏ hoặc đơn giản.

### 3.2. ResNet

ResNet viết tắt của Residual Network, là một kiến trúc mô hình CNN đột phá ở thời điểm nó ra đời. ResNet hướng đến việc xây dựng mô hình CNN rất sâu, rất nhiều layer, rất nhiều trọng số giúp giải quyết được nhiều bài toán, nhiều bộ dữ liệu phức tạp. ResNet có một số phiên bản nổi tiếng là ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, ResNet-200 ...



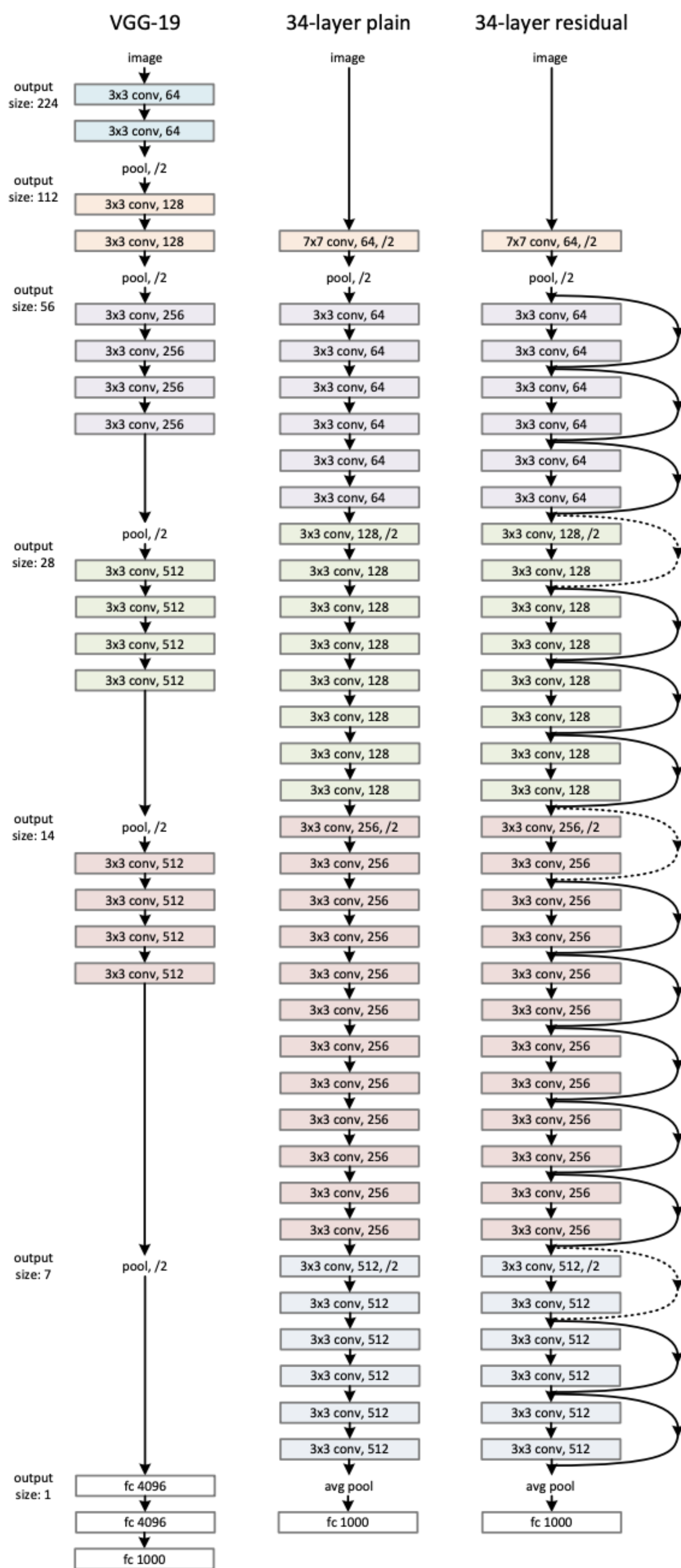
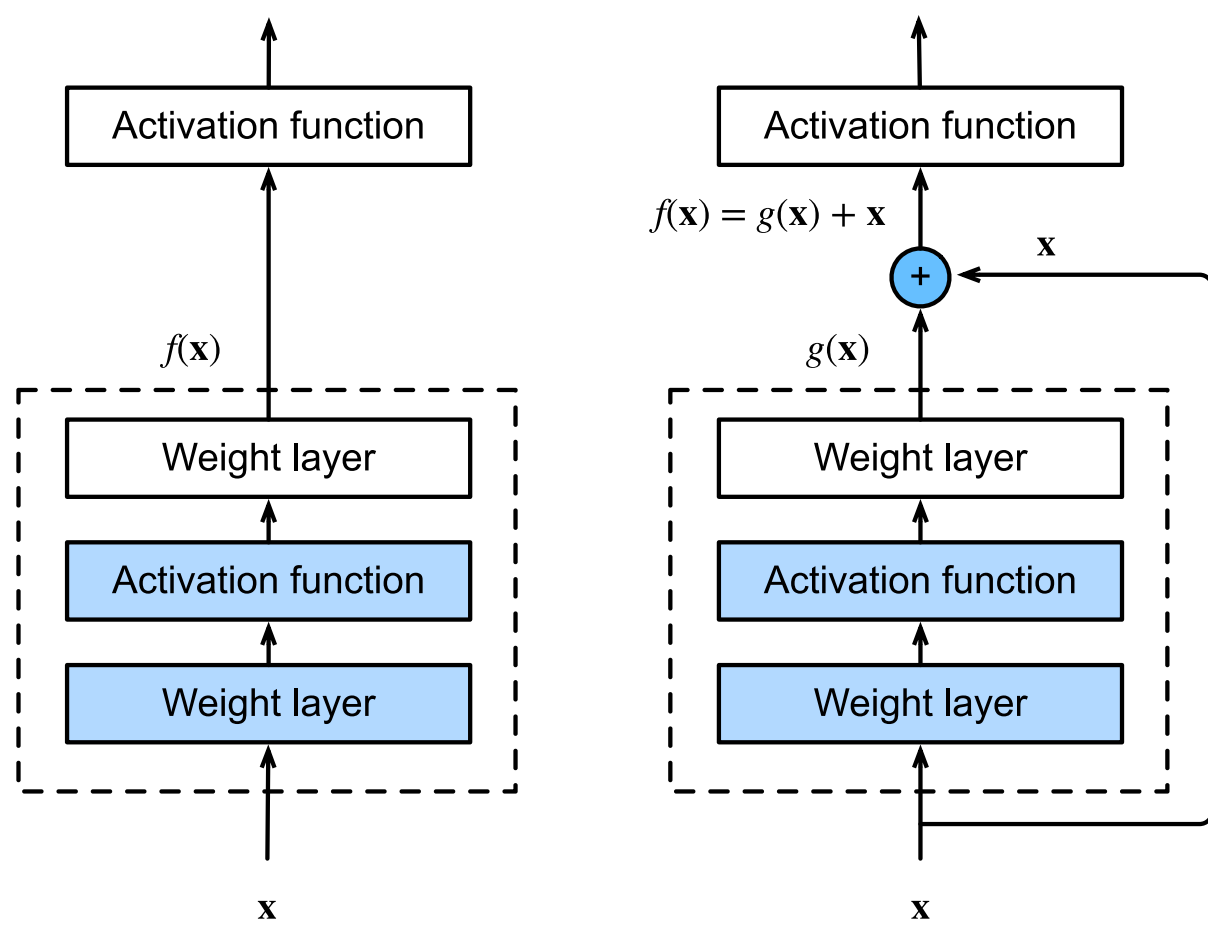


Figure 3. Example network architectures for ImageNet. **Left:** the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle:** a plain network with 34 parameter layers (3.6 billion FLOPs). **Right:** a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

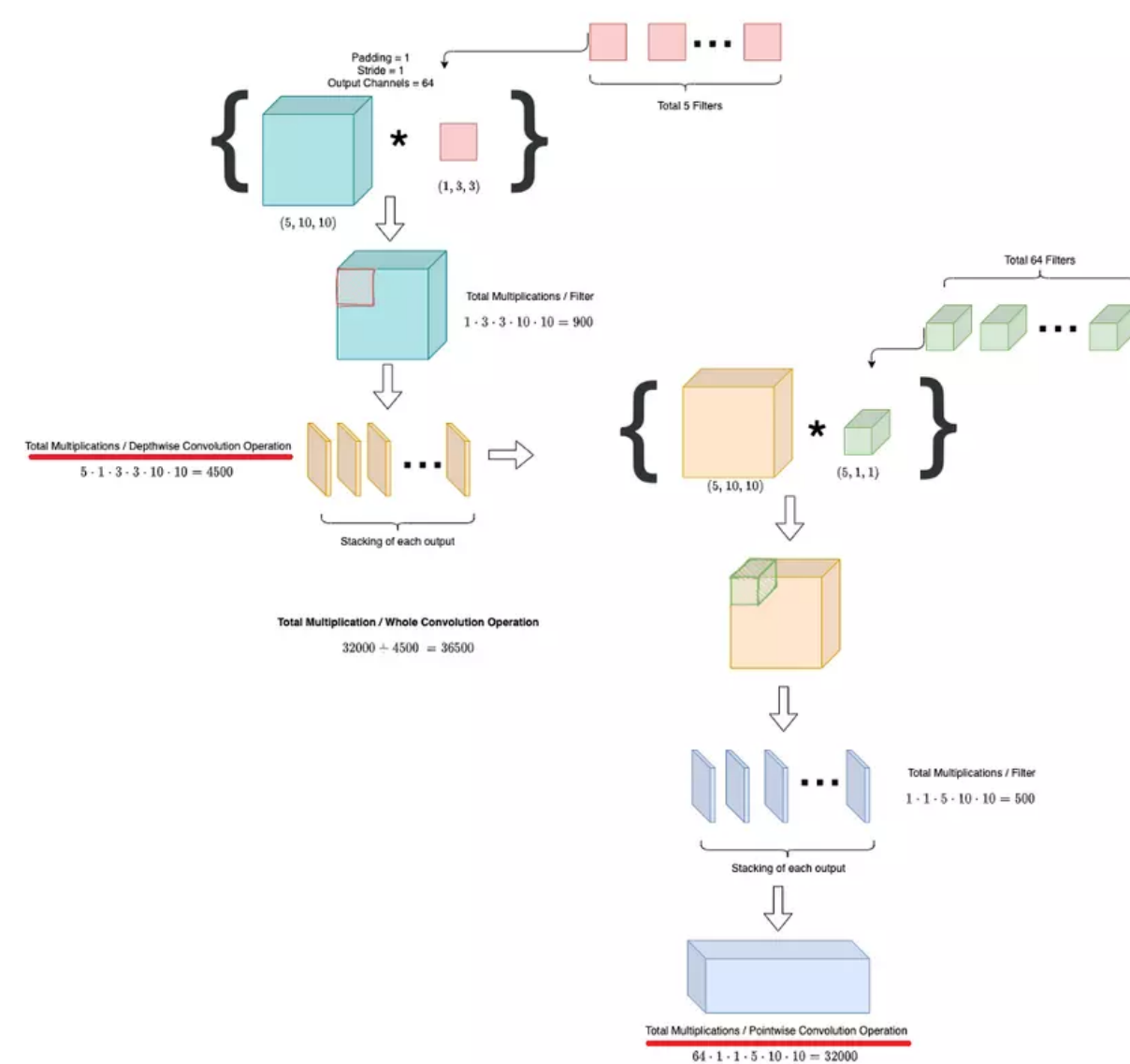
Tuy nhiên, khi xây dựng các mô hình quá sâu, trong quá trình training thường gặp phải vấn đề vanishing gradient. Do đó, nhóm tác giả của ResNet đã đề xuất ý tưởng về residual connection và residual block, giúp giải quyết vấn đề này đối với các mạng CNN có kích thước lớn.



Một số phiên bản mô hình nâng cấp hơn của ResNet là ResNext, ResNeSt ...

### 3.3. MobileNet

MobileNet, giống với cái tên của nó, là một mô hình CNN siêu gọn nhẹ có thể chạy được trên các thiết bị di động. MobileNet có số lượng trọng số của mô hình rất ít (nếu so sánh với VGG hay ResNet) nhờ sử dụng một layer convolution đặc biệt gọi là Depthwise Separable Convolution.



Tất nhiên là với kích thước mô hình nhỏ hơn nhiều, trong một số bài toán và bộ dữ liệu cụ thể, MobileNet không thể cạnh tranh được với ResNet hay VGG hay các kiến trúc CNN khác về độ chính xác, tuy nhiên, với lợi thế về tốc độ tính toán rất nhanh, MobileNet vẫn thường được sử dụng trong một số thiết bị di động hoặc trong một số trường hợp cần mô hình nhỏ trong thực tế.