Deep Learning

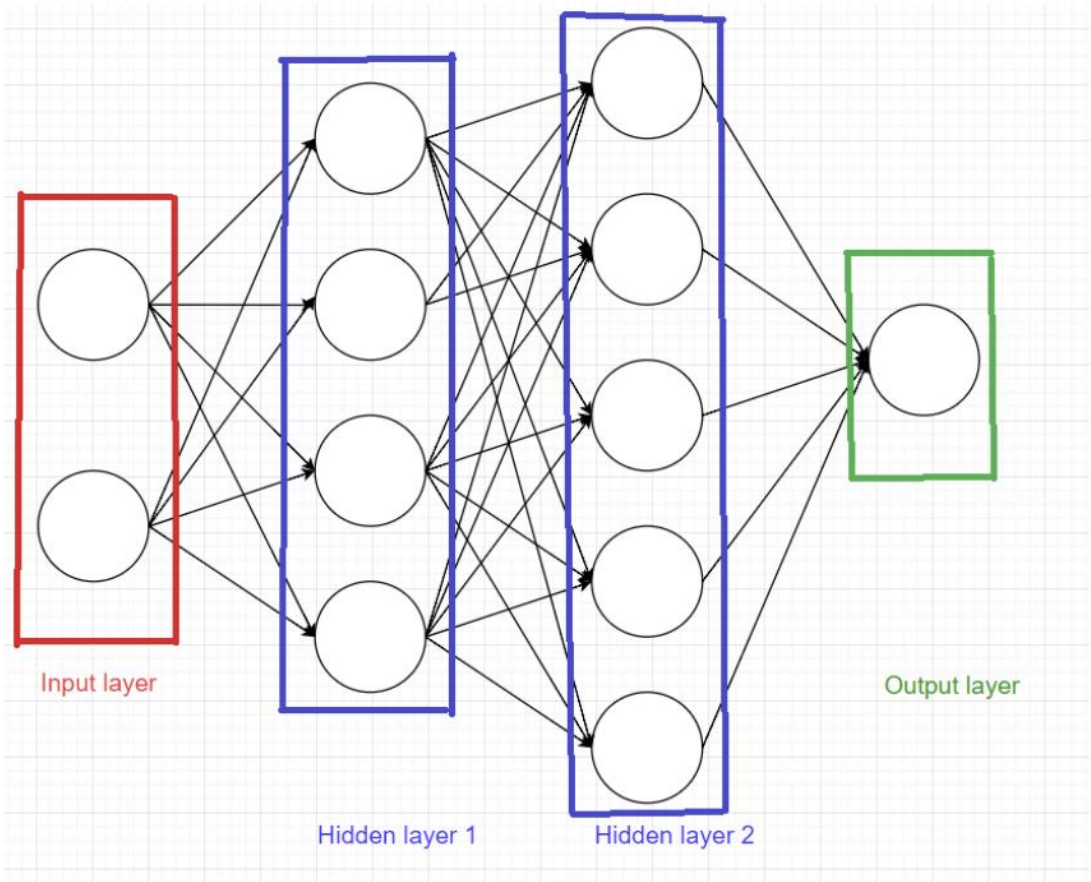
Tiếp theo

Outline

- Convolutional neural network
- Convolutional layer
- Pooling layer
- Fully connected layer
- Mạng VGG16
- Visualizing

Mô hình neural network

Mỗi hidden layer được gọi là fully connected layer, tên gọi theo đúng ý nghĩa, mỗi node trong hidden layer được kết nối với tất cả các node trong layer trước. Cả mô hình được gọi là fully connected neural network (FCN).



Hạn chế của fully connected neural network

- Với xử lý ảnh, thì ảnh màu 64*64 được biểu diễn dưới dạng 1 tensor 64*64*3. Nên để biểu thị hết nội dung của bức ảnh thì cần truyền vào input layer tất cả các pixel (64*64*3 = 12288). Nghĩa là input layer giờ có 12288 nodes.
- Giả sử số lượng node trong hidden layer 1 là 1000. Số lượng weight W giữa input layer và hidden layer 1 là 12288*1000 = 12288000, số lượng bias là 1000 => tổng số parameter là: 12289000. Đấy mới chỉ là số parameter giữa input layer và hidden layer 1, trong model còn nhiều layer nữa, và nếu kích thước ảnh tăng, ví dụ 512*512 thì số lượng parameter tăng cực kì nhanh => Cần giải pháp tốt hơn !!!

Convolutional neural network

- Nhận xét:
    - Trong ảnh các pixel ở cạnh nhau thường có liên kết với nhau hơn là những pixel ở xa.

Ví dụ như phép tính convolution trên ảnh ở bài trước. Để tìm các đường trong ảnh, ta áp dụng sobel kernel trên mỗi vùng kích thước 3*3. Hay làm nét ảnh ta áp dụng sharpen kernel cũng trên vùng có kích thước 3*3.

- Với phép tính convolution trong ảnh, chỉ 1 kernel được dùng trên toàn bộ bức ảnh. Hay nói cách khác là các pixel ảnh chia sẻ hệ số với nhau. => Áp dụng phép tính convolution vào layer trong neural network ta có thể giải quyết được vấn đề lượng lớn parameter mà vẫn lấy ra được các đặc trưng của ảnh.

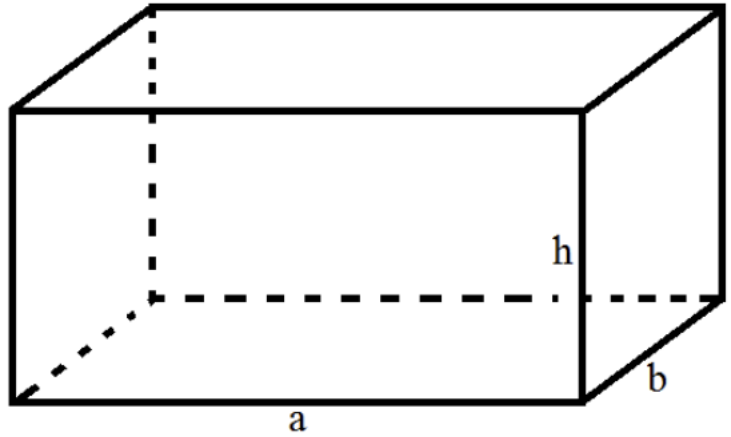
Tensor là gì?

- Khi dữ liệu biểu diễn dạng 1 chiều, người ta gọi là vector, mặc định khi viết vector sẽ viết dưới dạng cột.
- Khi dữ liệu dạng 2 chiều, người ta gọi là ma trận, kích thước là số hàng * số cột.

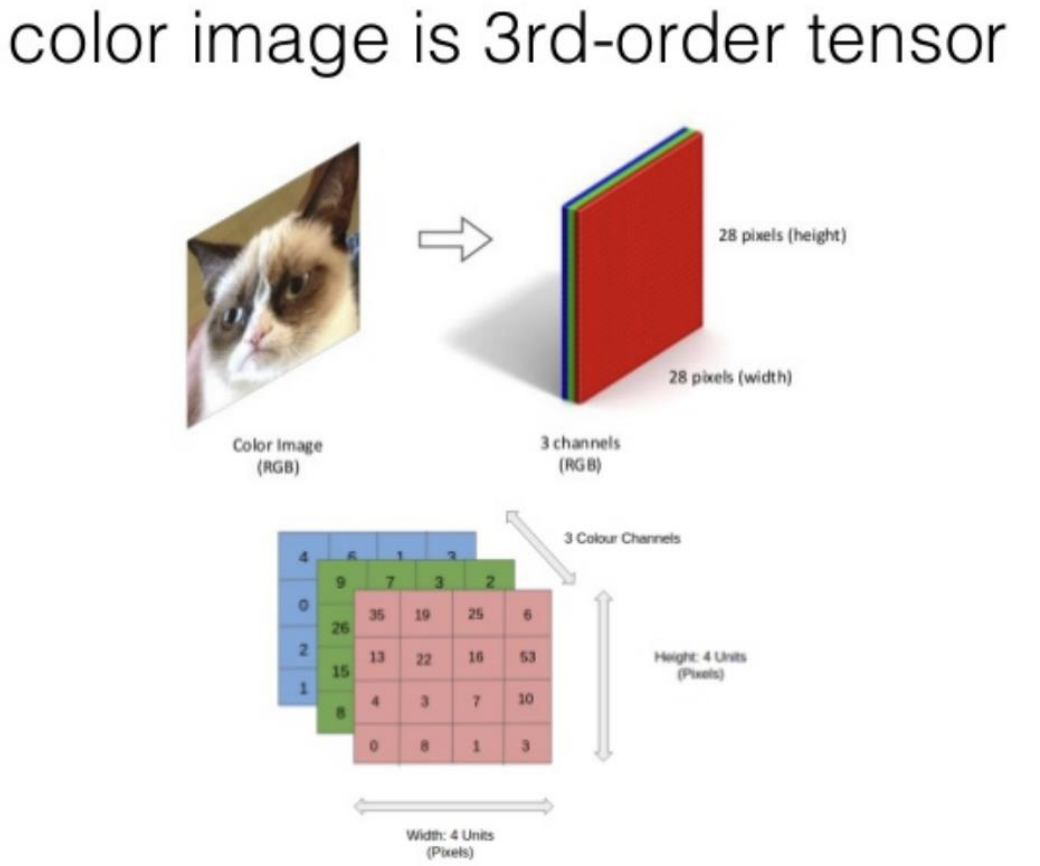
$$v = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix}, W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$

Tensor là gì?

- Khi dữ liệu nhiều hơn 2 nhiều thì sẽ được gọi là tensor, ví dụ như dữ liệu có 3 chiều.
- Tưởng tượng mặt đáy là một ma trận kích thước a * b, được tạo bởi b vector kích thước a. Cả hình hộp là tensor 3 chiều kích thước a*b*h, được tạo bởi xếp h ma trận kích thước a*b lên nhau. Do đó biểu diễn ảnh màu trên máy tính ở phần trên sẽ được biểu diễn dưới dạng tensor 3 chiều kích thước 600*800*3 do có 3 ma trận (channel) màu red, green, blue kích thước 600*800 chồng lên nhau

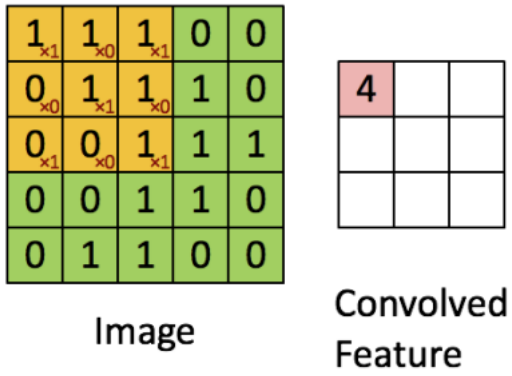


Tensor


color image is 3rd-order tensor
28 pixels (height)
28 pixels (width)
Color Image
(RGB)
3 channels
(RGB)
3 Colour Channels
Height: 4 Units
(Pixels)
Width: 4 Units
(Pixels)

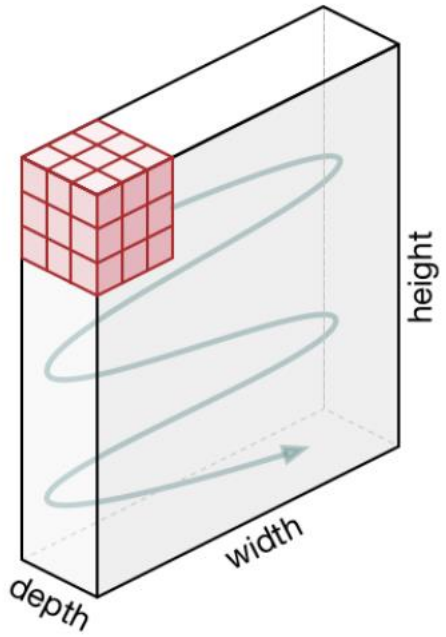
Convolutional layer

- Ở bài trước, phép tính convolution thực hiện trên ảnh xám với biểu diễn ảnh dạng ma trận



Convolutional layer

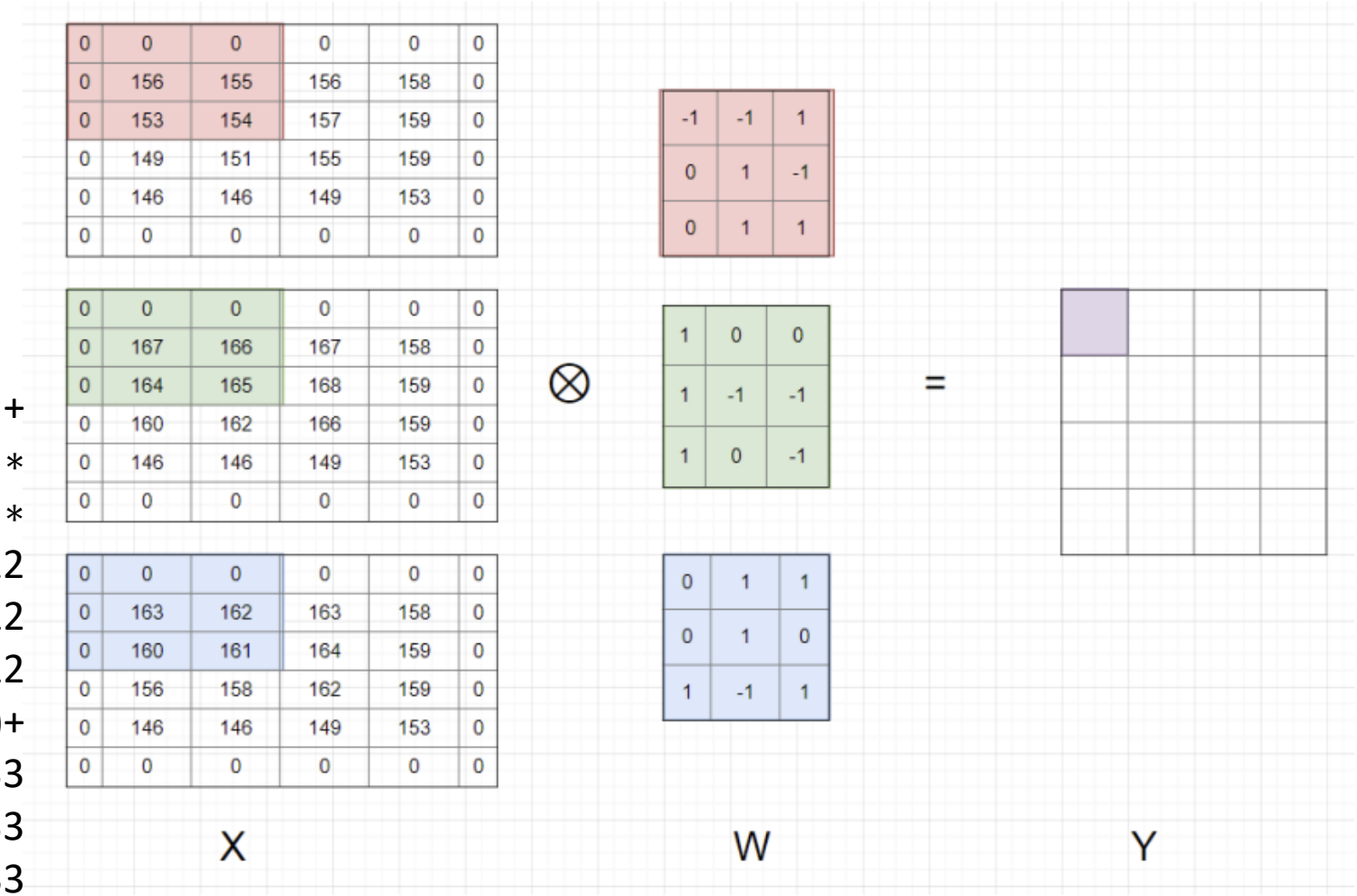
- Ảnh màu có 3 channels red, green, blue nên khi biểu diễn ảnh dưới dạng tensor 3 chiều. Ta cũng sẽ định nghĩa kernel là 1 tensor 3 chiều kích thước k*k*3



Tensor X, W 3 chiều được viết dưới dạng 3 matrix.

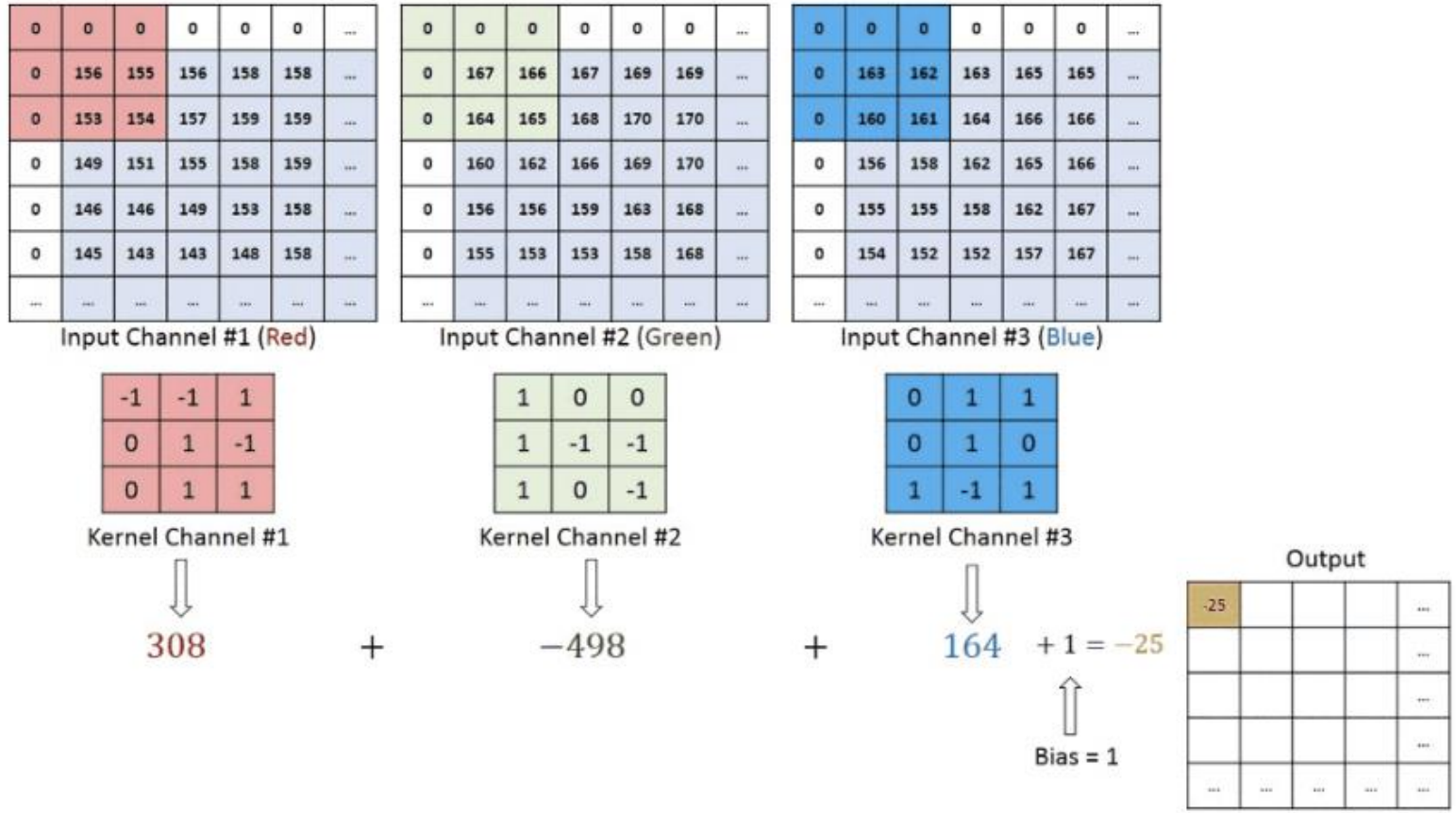
Khi biểu diễn ma trận ta cần 2 chỉ số hàng và cột: i và j, thì khi biểu diễn ở dạng tensor 3 chiều cần thêm chỉ số độ sâu k. Nên chỉ số mỗi phần tử trong tensor là x_{ijk}.

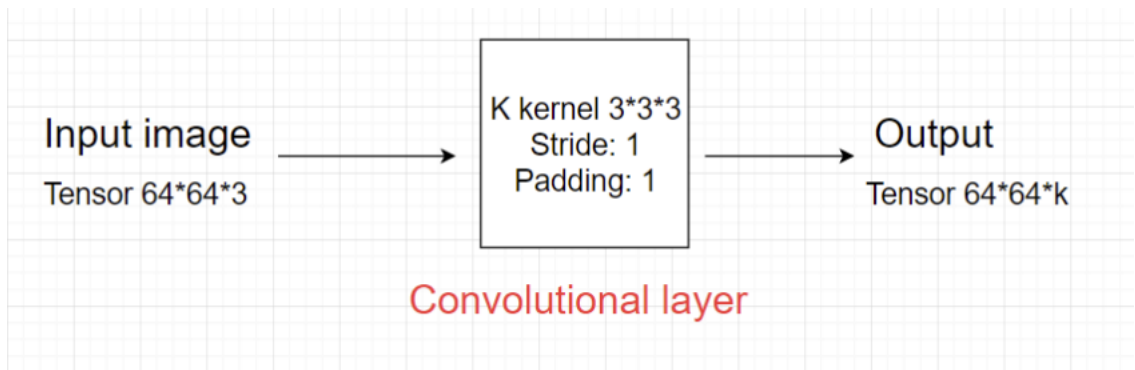
y11 = b + (x111 ∗ w111 + x121 ∗ w121 + x131 ∗ w131 + x211 ∗ w211 + x221 ∗ w221 + x231 ∗ w231 + x311 ∗ w311+x321 ∗w321+x331 ∗w331)+ (x112 ∗w112+x122 ∗w122+x132 ∗w132+x212 ∗w212+x222 ∗w222+ x232 ∗w232 +x312 ∗w312 +x322 ∗w322 +x332 ∗w332)+ (x113 ∗w113 +x123 ∗w123 +x133 ∗w133 +x213 ∗ w213 +x223 ∗w223 +x233 ∗w233 +x313 ∗w313 +x323 ∗w323 +x333 ∗w333) = −25



Convolutional layer: Nhận xét

- Output Y của phép tính convolution trên ảnh màu là 1 matrix.
- Có 1 hệ số bias được cộng vào sau bước tính tổng các phần tử của phép tính element-wise
- Các quy tắc đối với padding và stride toàn hoàn tương tự như ở bài trước.
- Với mỗi kernel khác nhau ta sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel để học được nhiều thuộc tính của ảnh. Vì mỗi kernel cho ra output là 1 matrix nên k kernel sẽ cho ra k output matrix. Ta kết hợp k output matrix này lại thành 1 tensor 3 chiều có chiều sâu k





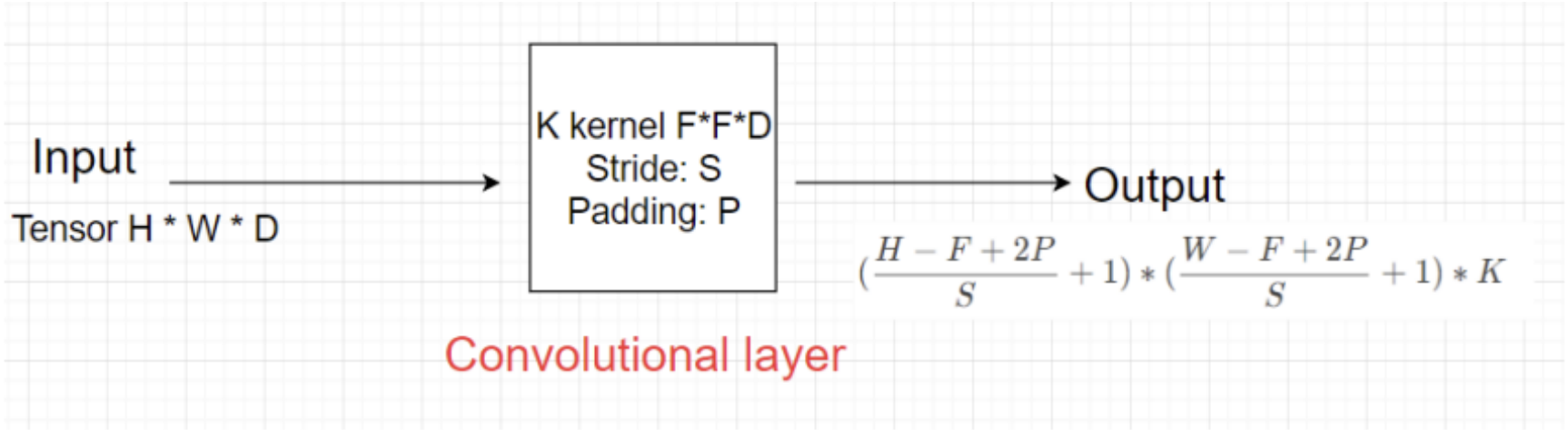
Output của convolutional layer đầu tiên sẽ thành input của convolutional layer tiếp theo.

Convolutional layer tổng quát

- Giả sử input của 1 convolutional layer tổng quát là tensor kích thước H * W * D.
- Kernel có kích thước F * F * D (kernel luôn có depth bằng depth của input và F là số lẻ), stride: S, padding: P. Convolutional layer áp dụng K kernel. => Output của layer là tensor 3 chiều có kích thước:

$$(\frac{H-F+2P}{S}+1)*(\frac{W-F+2P}{S}+1)*K$$

Convolutional layer tổng quát



Lưu ý:

- Output của convolutional layer sẽ qua hàm non-linear activation function trước khi trở thành input của convolutional layer tiếp theo.
- Tổng số parameter của layer: Mỗi kernel có kích thước F*F*D và có 1 hệ số bias, nên tổng parameter của 1 kernel là F*F*D + 1. Mà convolutional layer áp dụng K kernel => Tổng số parameter trong layer này là K * (F*F*D + 1).

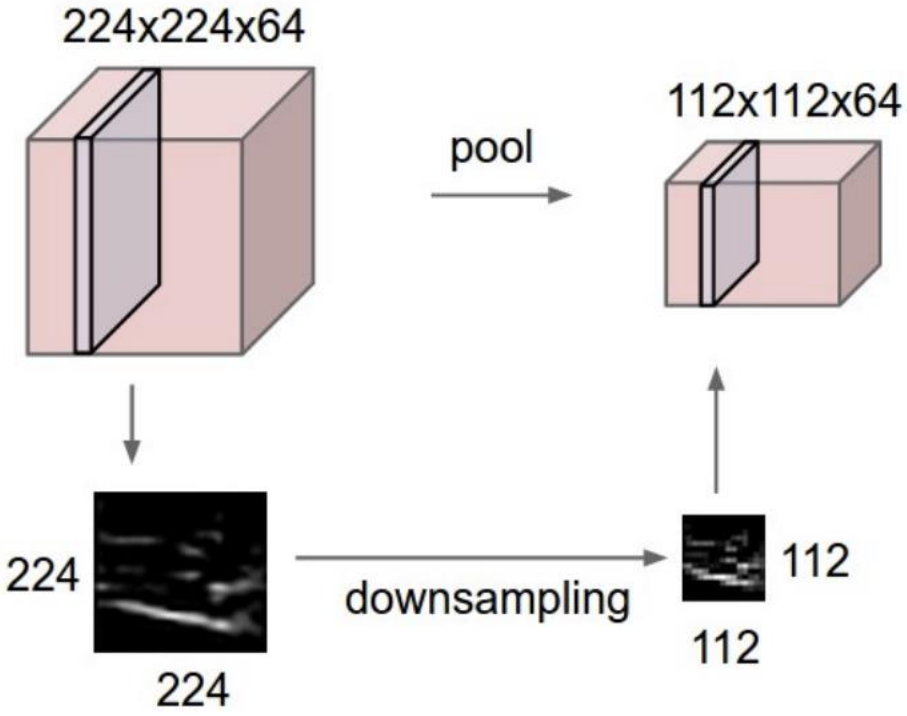
Pooling layer

- Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Việc giảm kích thước dữ liệu giúp giảm các phép tính toán trong model.
- Bên cạnh đó, với phép pooling kích thước ảnh giảm, nên lớp convolution học được các vùng có kích thước lớn hơn. Ví dụ như ảnh kích thước 224*224 qua pooling về 112*112 thì vùng 3*3 ở ảnh 112*112 tương ứng với vùng 6*6 ở ảnh ban đầu.

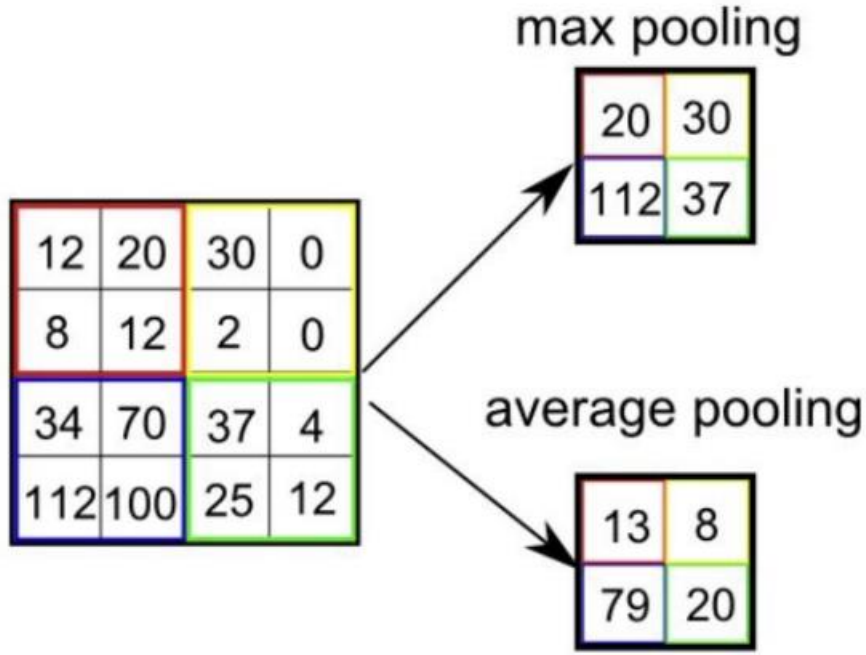
Pooling layer

- Gọi pooling size kích thước K*K. Input của pooling layer có kích thước H*W*D, ta tách ra làm D ma trận kích thước H*W. Với mỗi ma trận, trên vùng kích thước K*K trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh

Pooling layer



Có 2 loại pooling layer phổ biến là: max pooling và average pooling.

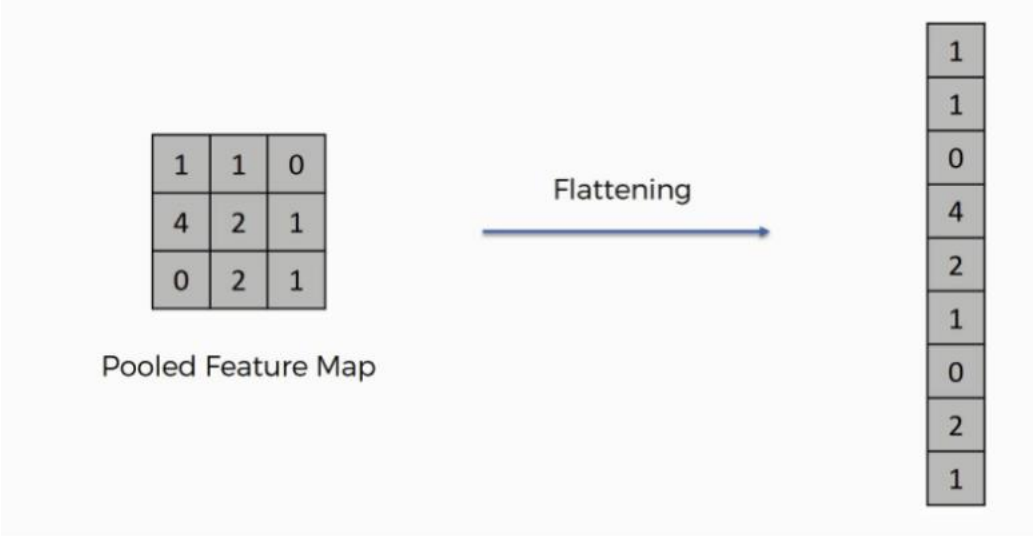


Sau pooling layer (2*2)

Trong một số model người ta dùng convolutional layer với stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.

Fully connected layer

- Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,...) thì tensor của output của layer cuối cùng, kích thước H*W*D, sẽ được chuyển về 1 vector kích thước (H*W*D, 1)
- Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model.



Mạng VGG16

- VGG16 là mạng convolutional neural network được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford. Model sau khi train bởi mạng VGG16 đạt độ chính xác 92.7% top-5 test trong dữ liệu ImageNet gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau

- Phân tích:
  - Convolutional layer: kích thước 3*3, padding=1, stride=1
  - Pool/2 : max pooling layer với size 2*2
  - 3*3 conv, 64: thì 64 là số kernel áp dụng trong layer đó, hay depth của output của layer đó.
  - Các convolutional layer về sau thì kích thước width, height càng giảm nhưng depth càng tăng.
  - Sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer

Visualizing

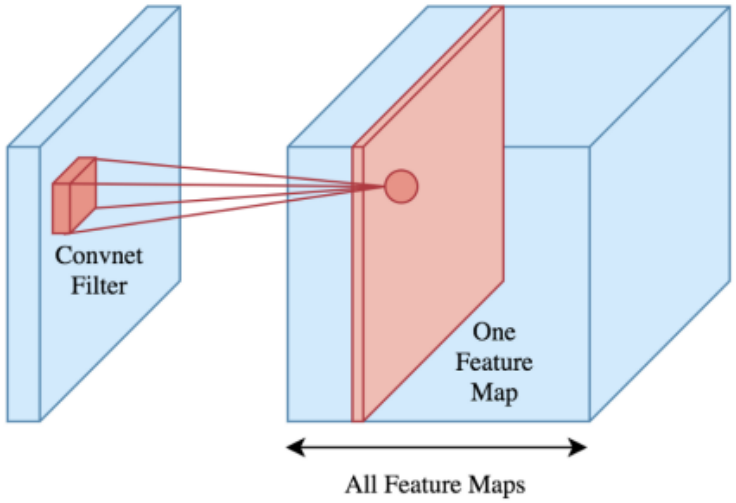
- Bên cạnh những thứ rất thú vị mà mô hình học sâu mang lại, thứ mà thực sự bên trong mô hình học sâu vẫn được coi là một chiếc hộp đen. Rõ ràng chúng ta luôn muốn giải mã chiếc hộp đen đó, rằng tại sao mô hình lại phân biệt được các chữ số, hay tại sao tìm ra vị trí của chú chó bull và xa hơn là tại sao mô hình lại có thể chuẩn đoán một tế bào là tế bào ung thư hay không. Càng ứng dụng các mô hình học sâu vào đời sống, việc giải mã chiếc hộp đen càng quan trọng.

Visualizing

- Có 3 thành phần quan trọng trong mô hình CNN mà ta có thể visualize để hiểu mô hình CNN thực hiện việc gì:
  - Features Map
  - Convnet Filters
  - Class Output

Visualizing Feature Maps

- Như đã viết ở phần trên, với một ảnh có số kênh là C, sau khi được đưa qua convolutional layer với K convnet filters, ta được một feature map có K kênh, với mỗi kênh là một feature map được trích xuất từ một convet filter tương ứng



Visualizing Feature Maps

- Như đề cập ở trên, đầu ra của convolutional layer là một tensor 3D với chiều rộng, chiều cao và chiều sâu là số kênh. Như vậy mỗi kênh của tensor là một kết quả độc lập từ phép tích chập tương ứng với mỗi filter và có 2 chiều tương ứng. Ta có thể tách từng kênh ra và xuất ra được một hình ảnh đại diện cho kênh đó. Đây chính là cách để visualize feature map.

Visualizing feature maps

- Trong kiến trúc của mạng VGG-16, ta có 5 khối (block) chính, mỗi block có các convolutional layer. Để tiện gọi tên các lớp, quy ước blockX-convY với X và Y là thứ tự tương ứng của block X và conv layer Y. Ví dụ với conv thứ 2 trong khối thứ 4 có tên là block4-conv2.
- Phần có màu sáng hơn được gọi là vùng kích hoạt, nghĩa là vùng đó đã được filter lọc ra thứ mà nó cần tìm. Như hình trên, có vẻ filter này đang muốn trích xuất ra phần mắt và mũi của chú mèo.

Ảnh gốc

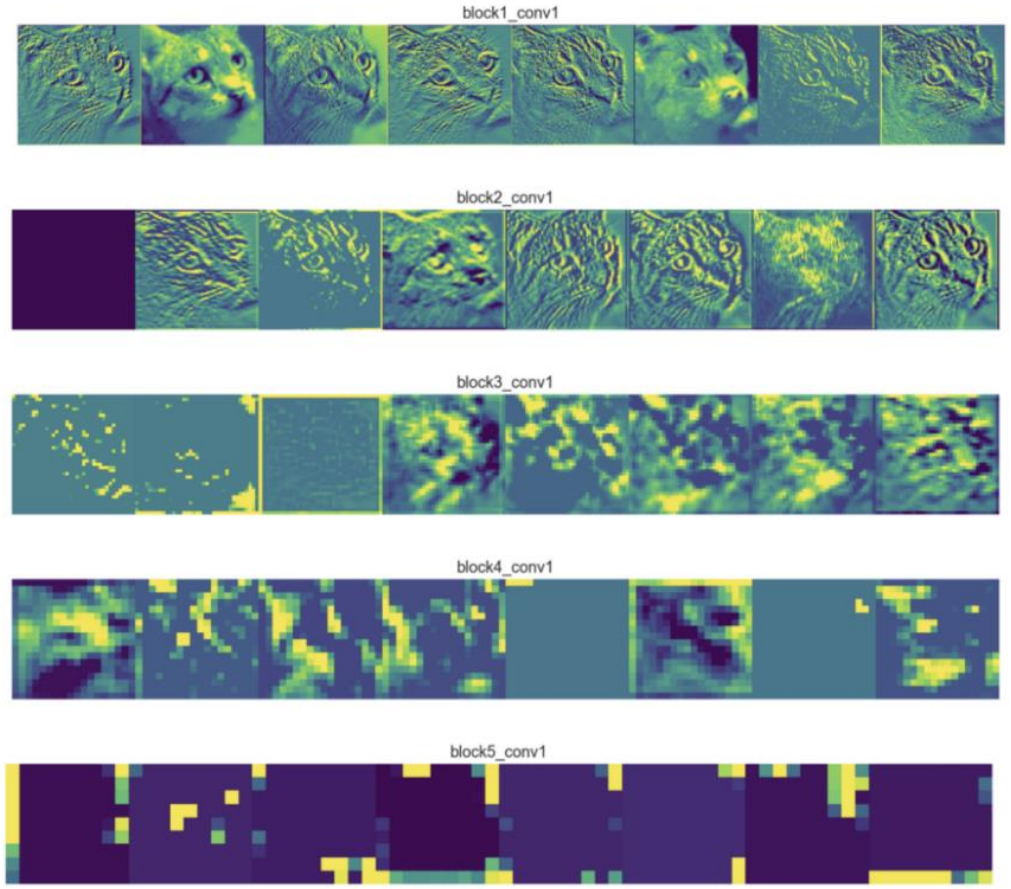
block1_conv1

Nhận xét:

• Các lớp đầu tiên (block1-conv1) chủ yêu giữ lại hầu hết thông tin của ảnh. Trong mạng CNN, các lớp đầu tiên thường là trích xuất viền cạnh của ảnh

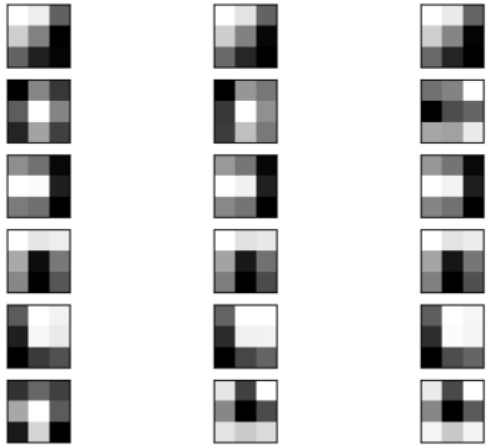
Càng đi sâu, càng khó có thể nhận ra là hình gì. Như hình ở block2-conv1, ta vẫn có thể nhận ra đó là con mèo. Tuy nhiên đến block thứ 3 thì rất khó đoán được đó là hình gì và block thứ 4 thì không thể đoán. Vì càng về sâu, các feature map biểu diễn cho những hình trừu tượng hơn. Những thông tin càng cụ thể hơn thì ta càng khó hình dung nó là gì. Chẳng hạn như các lớp về sau chỉ giữ lại thông tin của tai, mắt hay mũi của mèo mà thôi.

Tuy nhiên càng đi sâu, mô hình tìm những thứ phức tạp hơn ví dụ như cái đuôi của con mèo - thứ mà không phải hình nào cũng có. Vậy nên đó là lý do tại sao ta thấy một số hình ở trên bị trống



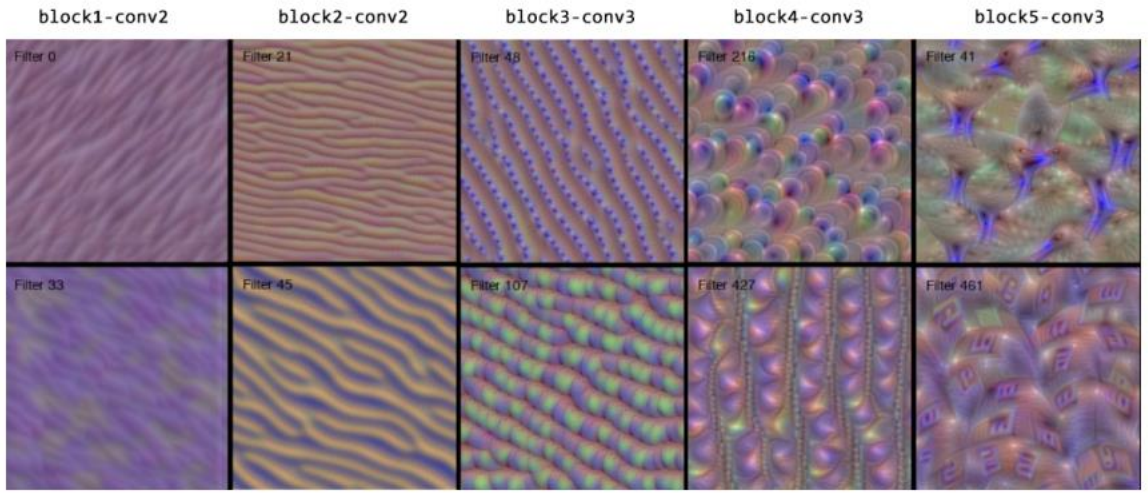
Visualizing Convolutional Filters

- Tuy nhiên ta sẽ không trực tiếp visualize conv filters bởi vì như đã biết, các filters thường có kích thước rất nhỏ (3x3 hoặc 5x5). Việc visualize nó không đem lại nhiều ý nghĩa

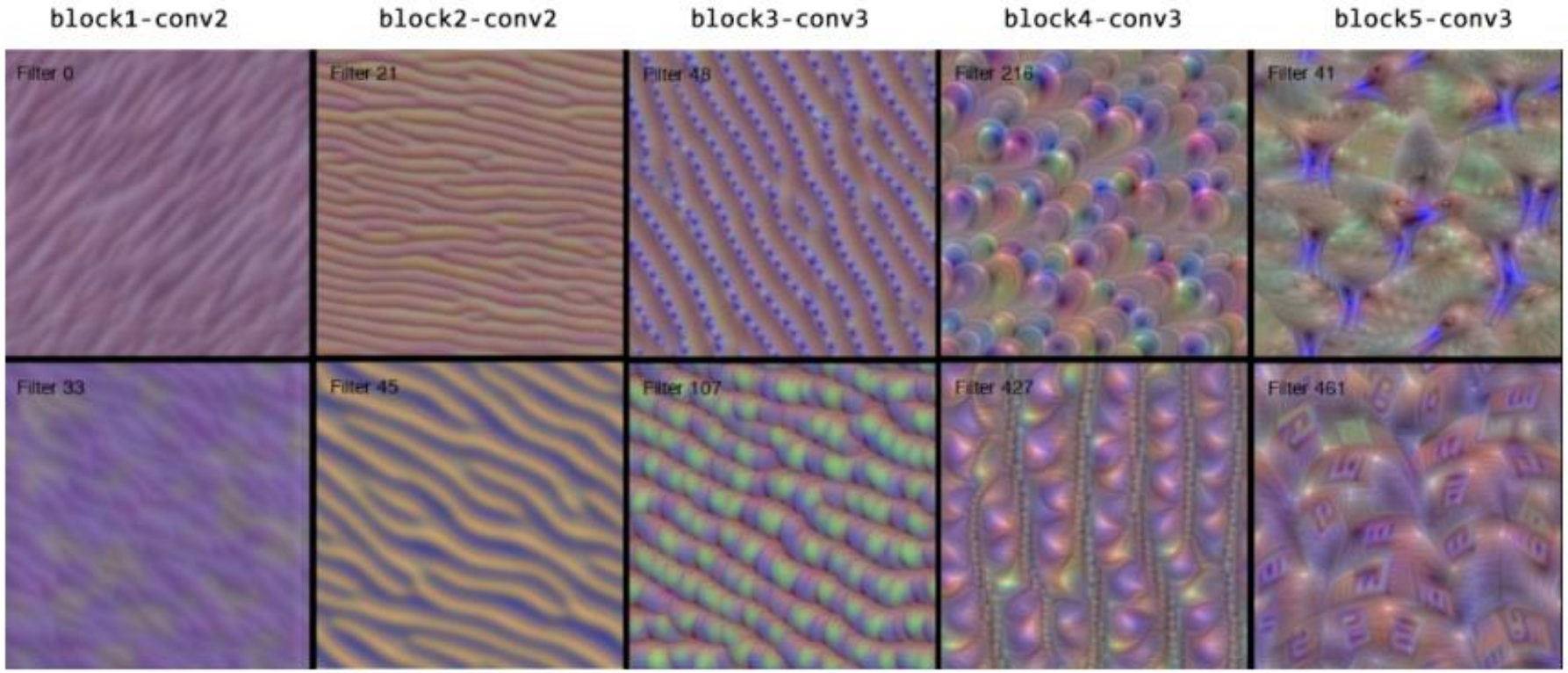


Visualizing Convolutional Filters

- Thay vào đó, ta sẽ thử biểu diễn các filters thông qua một hình đầu vào mà khi qua hàm tích chập với filters tương ứng đạt được giá trị kích hoạt trung bình cực đại. Có một số kỹ thuật nâng cao, được đề xuất bởi Erhan et al. 2009, nhưng tựu chung gồm các bước cơ bản như sau:
  - Một hình ảnh với giá trị các pixel ngẫu nhiên và một pre-trained model CNN.
  - Ta sẽ tìm các giá trị của pixel sao cho hàm loss là để cực đại trung bình giá trị kích hoạt của filter bằng cách thực hiện gradient ascent.
  - Lặp lại đến khi đạt giá trị cực đại



Càng về các layer cuối, hình ảnh càng trông rõ ràng hơn:
• Các filter đầu tiên chủ yếu dùng để phát hiện những cạnh, viền, màu sắc và những hình dạng đơn giản.
• Các filter về sau mã hoá những hình ảnh phức tạp hơn.
Như ta thấy filter 41 ở layer block5- conv3 có vẻ dùng để nhận ra các con chim. Khi mà rất nhiều hình con chim với dáng khác nhau ở đó, nó khiến cho giá trị trung bình hàm kích hoạt gần đến cực đại.
Có thể thấy mô hình CNN cũng giống con người. Khi còn là trẻ sơ sinh, ta tập trung vào những hình đơn giản trước rồi qua quá trình luyện tập những hình ảnh phức tạp dễ dàng được nhận ra bởi não bộ con người.



Một số khái niệm

- Epoch: Một Epoch được tính là khi chúng ta đưa tất cả dữ liệu trong tập train vào mạng neural network 1 lần. Ví dụ, bạn có 10 triệu bức hình trong tập train, bạn cho toàn bộ số hình đó là input của mô hình 3 lần, suy ra bạn đã train được 3 epoch.

- Khi dữ liệu quá lớn, chúng ta không thể đưa hết mỗi lần tất cả tập dữ liệu vào để huấn luyện được, vì bạn cần một siêu máy tính có lượng RAM và GPU RAM cực lớn để lưu trữ toàn bộ hình ảnh trên, điều này là bất khả thi đối với người dùng bình thường, phòng lab nhỏ. Buộc lòng chúng ta phải chia nhỏ tập dữ liệu ra, và khái niệm batch hình thành.

Batch size

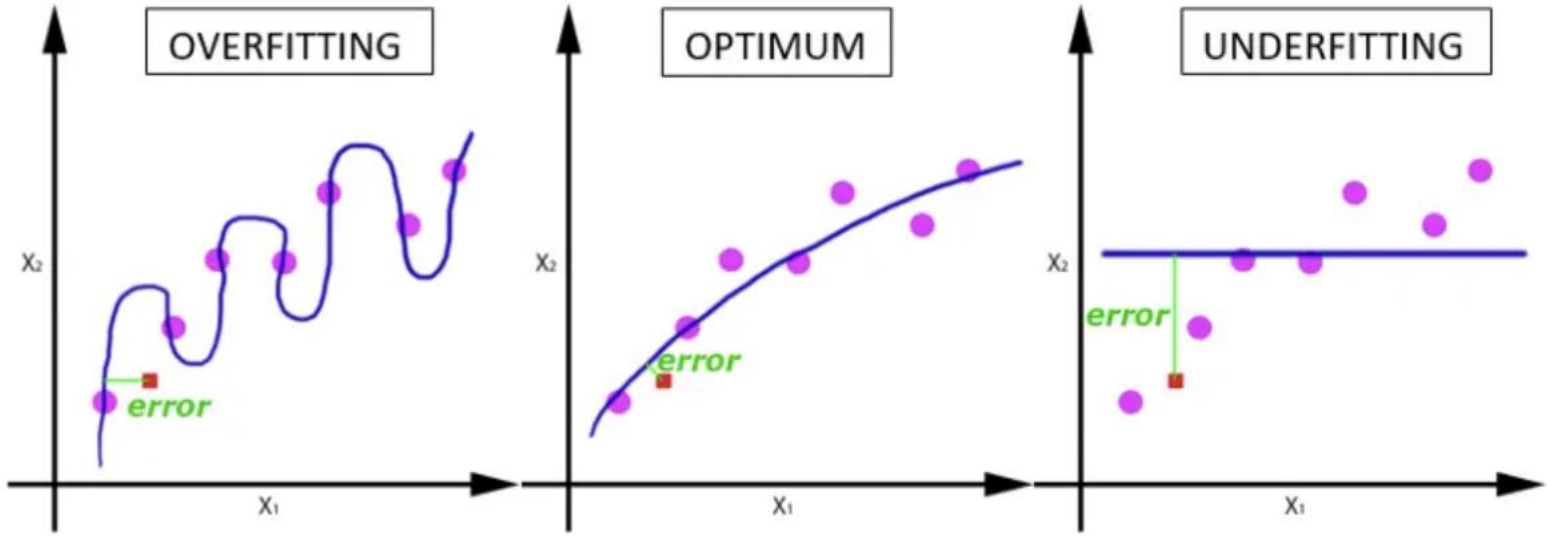
- Batch size là số lượng mẫu dữ liệu trong một lần huấn luyện. Ví dụ, trong bài toán phân loại chó mèo, chọn batch size =32, nghĩa là 1 lần lặp ta sẽ cho ngẫu nhiên 32 bức hình chó hoặc mèo chạy lan truyền tiến trong mạng neural network. tiếp theo bạn cho tiếp 32 hình ngẫu nhiên, không lặp với các hình trước đó, vào mạng, đến khi nào không còn hình nào có thể cho vào nữa -> bạn hoàn thành 1 epoch.

Iterations

- Iterations là số lượng batchs cần để hoàn thành 1 epoch.
- Ví dụ chúng ta có tập dữ liệu có 20,000 mẫu, batch size là 500, vậy chúng ta cần 40 lần lặp (iteration) để hoàn thành 1 epoch.

Tại sao phải dùng nhiều hơn 1 epoch

- Vì chúng ta đang dùng thuật toán tối ưu là Gradient Descent. Thuật toán này đòi hỏi chúng ta phải đem toàn bộ dữ liệu qua mạng một vài lần để tìm được kết quả tối ưu. Vì vậy, dùng 1 epoch thật sự không đủ để tìm được kết quả tốt nhất.
- Với việc chỉ sử dụng 1 lần lặp, xác suất rất cao là dữ liệu sẽ bị underfitting(như hình mô tả bên dưới).



Vậy số lần lặp tối ưu là bao nhiêu?

- Tiếc rằng không có câu trả lời cho câu hỏi này. Phụ thuộc hoàn toàn vào nhiều yếu tố. Mục tiêu chung là ta sẽ lặp đến khi nào hội tụ. Có một số phương pháp giúp chúng ta xác định mô hình đã đứng ở ngưỡng cực tiểu cục bộ rồi, không thể xuống hơn được nữa.

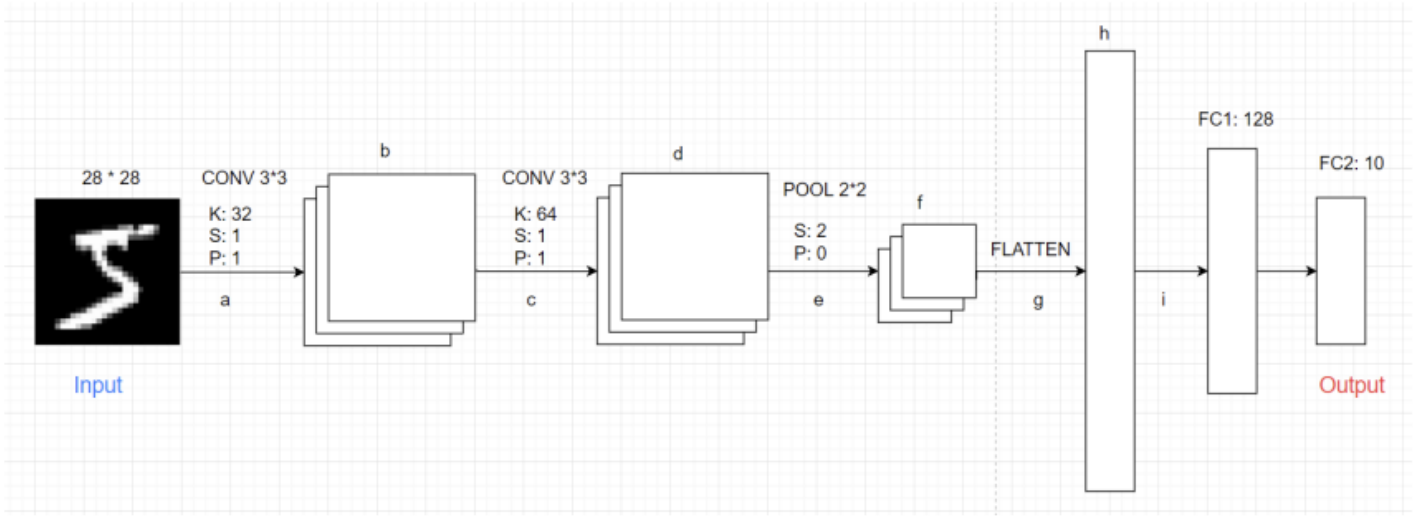
- Các bạn có thể tìm hiểu với từ khóa early stopping.

Bài tập

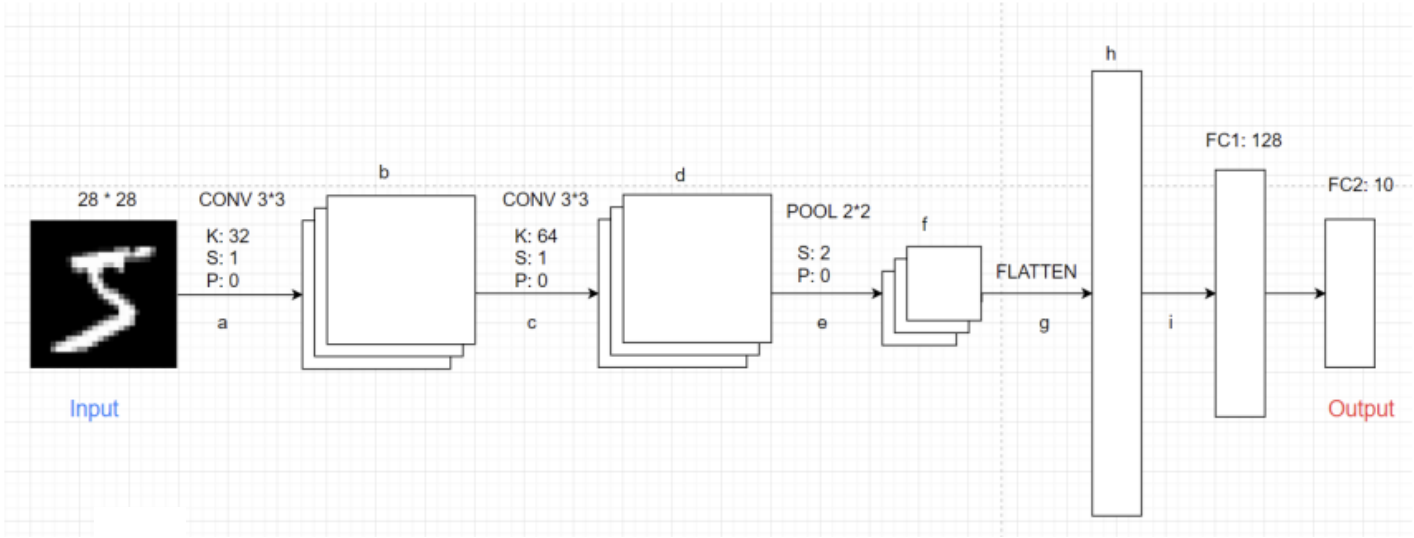
1. Convolutional layer để làm gì? Lại sao mỗi layer cần nhiều kernel?
2. Hệ số của convolutional layer là gì?
3. Tự tính lại số lượng parameter, output size của convolutional layer với stride và padding trong trường hợp tổng quát.

Bài tập

4. Hình dưới là mô hình nhận diện chữ số MNIST, K: số kernel, S: stride, P: padding. Tính số lượng parameter ở layer và output tương ứng (Tìm a, b, c, d, e, f, g, h, i)



(a)



(b)

Bài tập

- 5. Tại sao cần flatten trong CNN?
- 6. Tại sao trong model VGG16, ở layer càng sâu thì wight, height giảm nhưng depth lại tăng.