Module 5

Convolutional Neural Networks

DL: Convolutional Nets

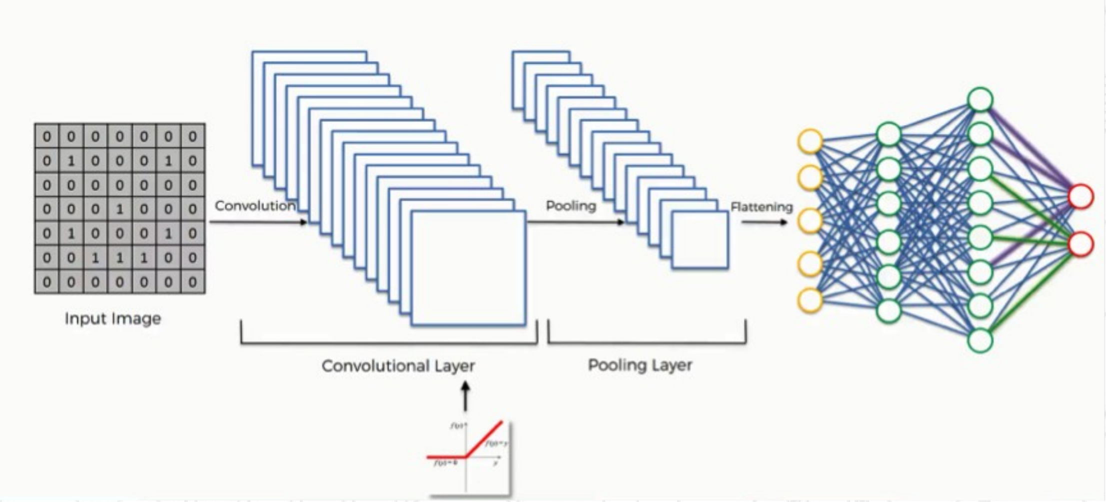


Image Representation

- We know images are represented by tensors
- A colored pixel is represented as three separate values (RGB).
- 3 matrices (tensor) represent red (R), green (G), blue (B) channels.

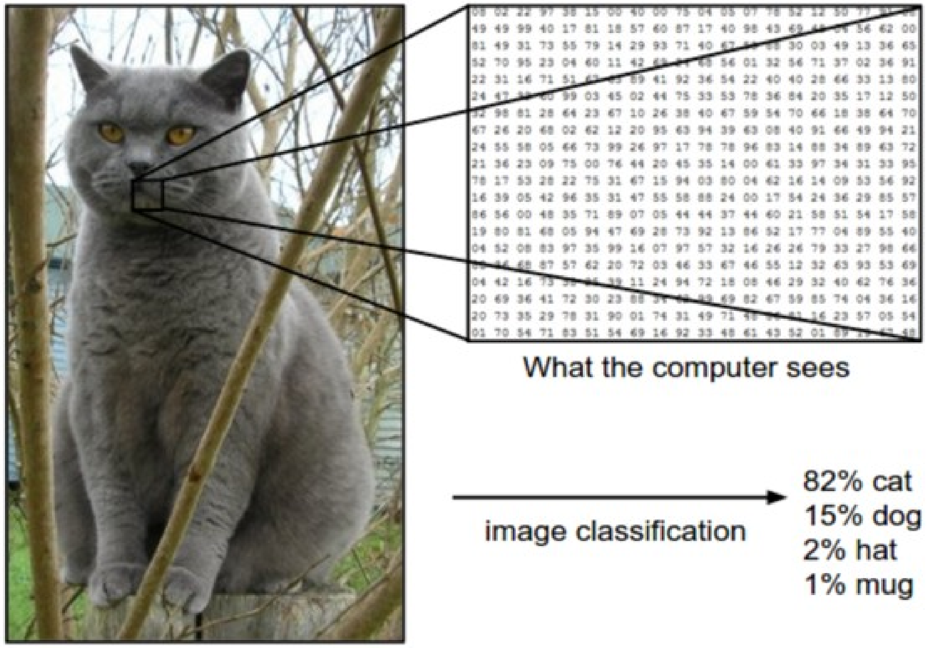


Image Representation

- Channels can be flattened to create a single vector.

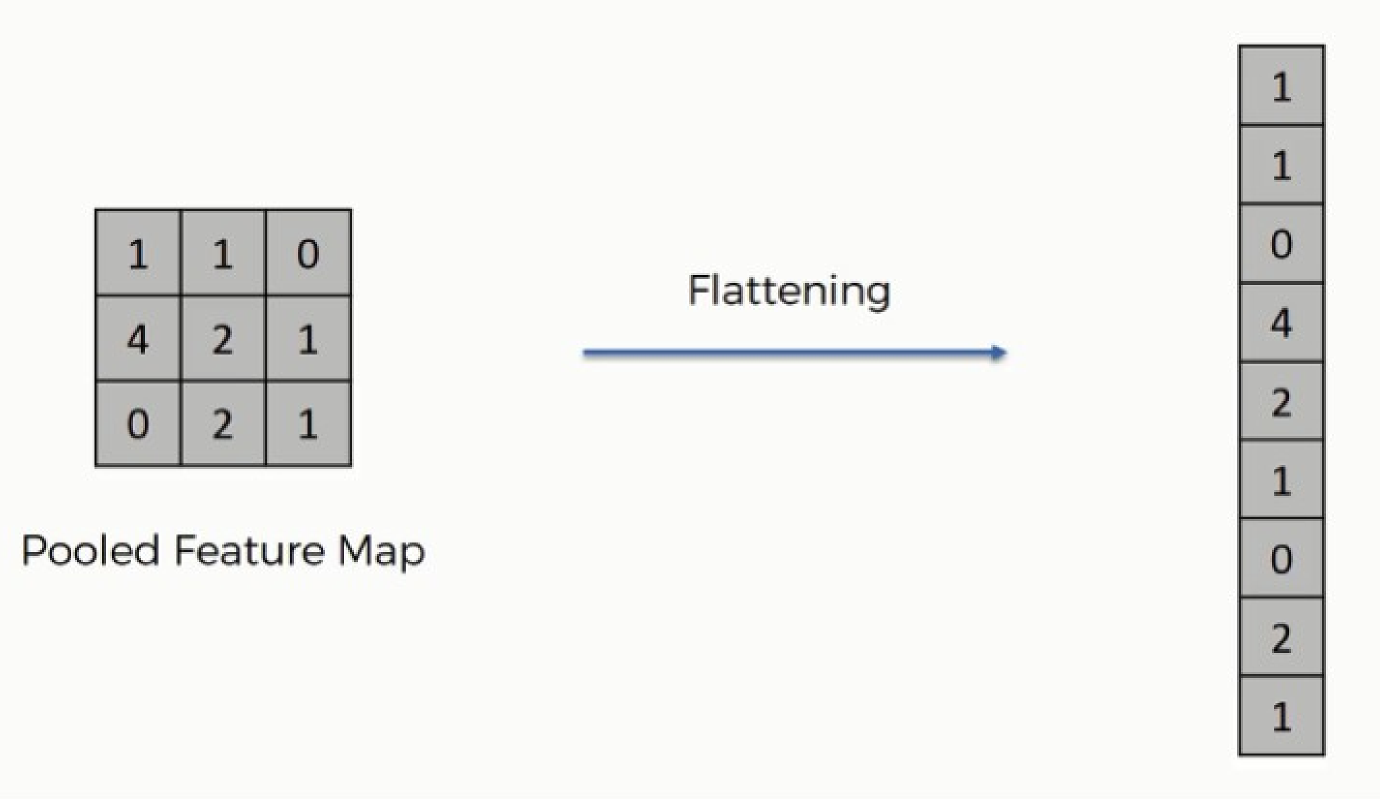
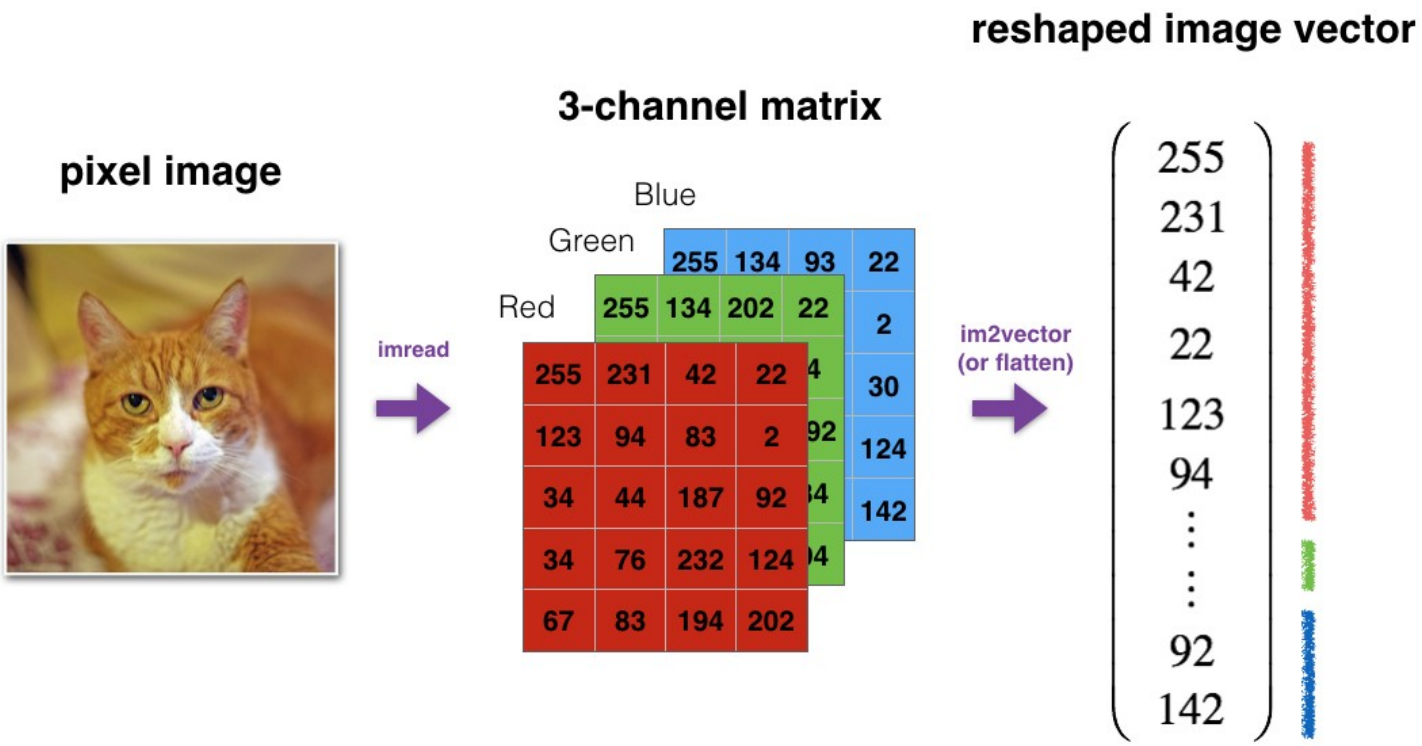


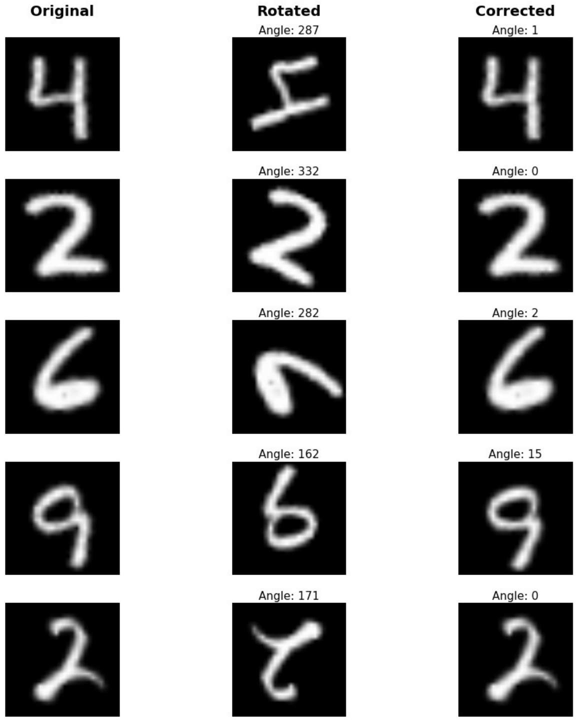
Image Representation

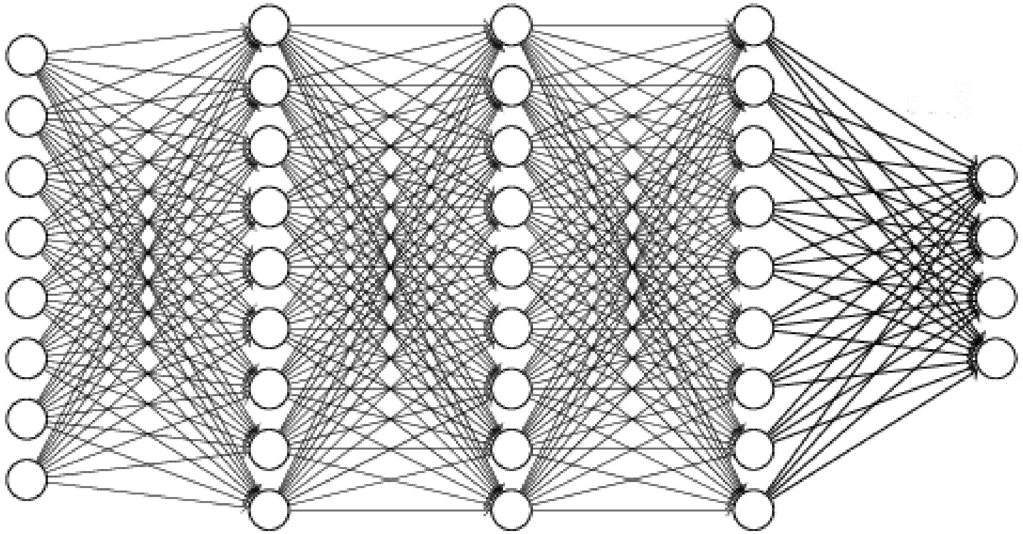
- Channels can be flattened to create a single vector.



Using Feed-Forward NNs for Images

- Feed forward NNs are not very effective with images





Using Feed-Forward NNs for Images

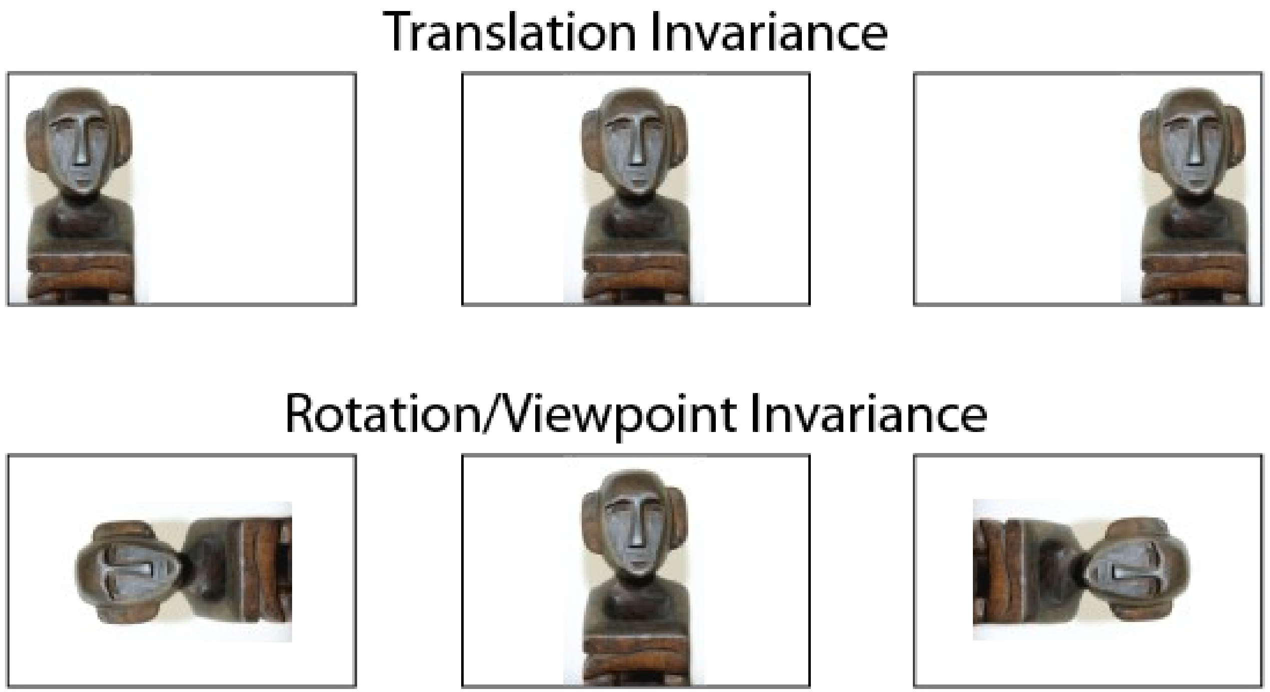
- Translation Invariance

Cat

Cat

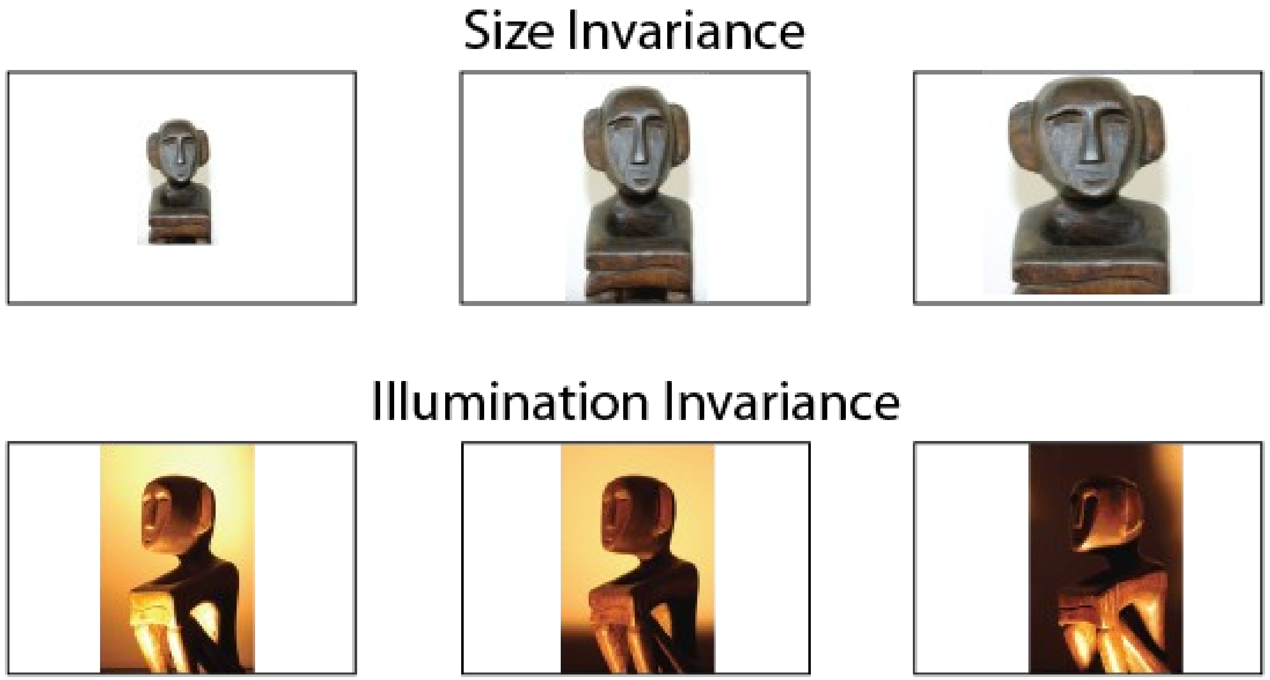
Using Feed-Forward NNs for Images

- Other Invariances cause issues



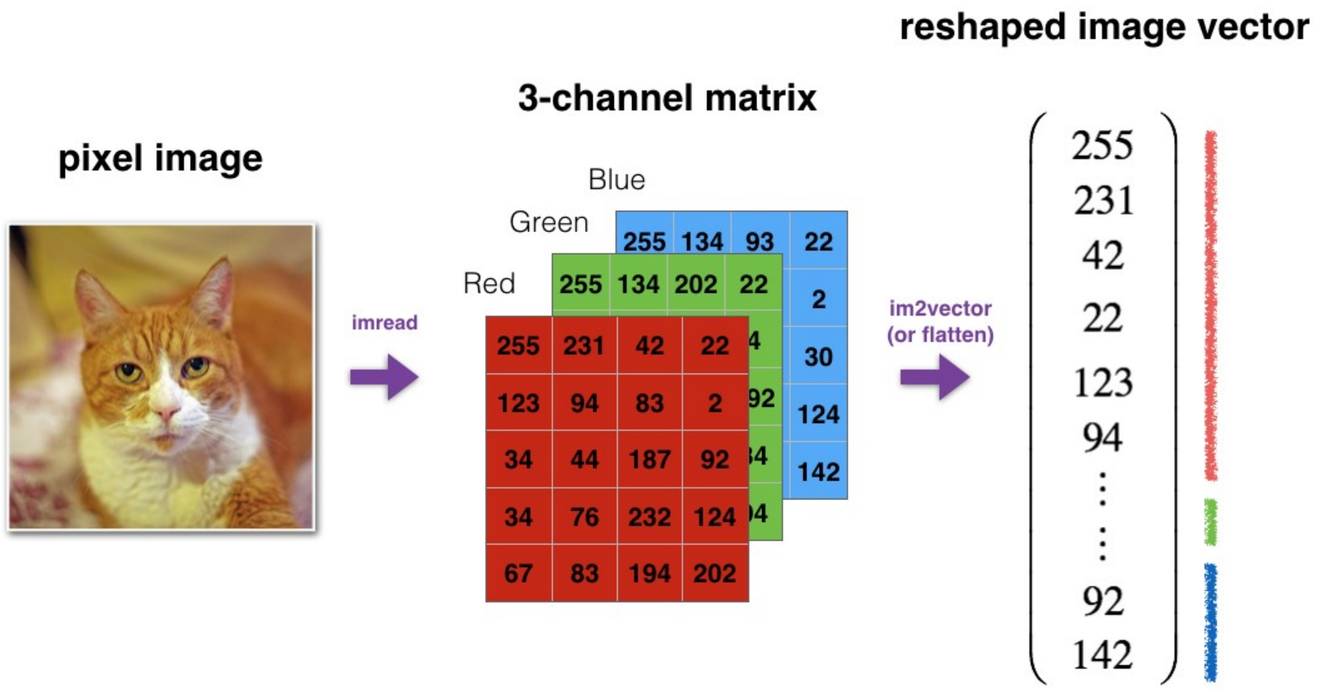
Using Feed-Forward NNs for Images

- Other Invariances cause issues



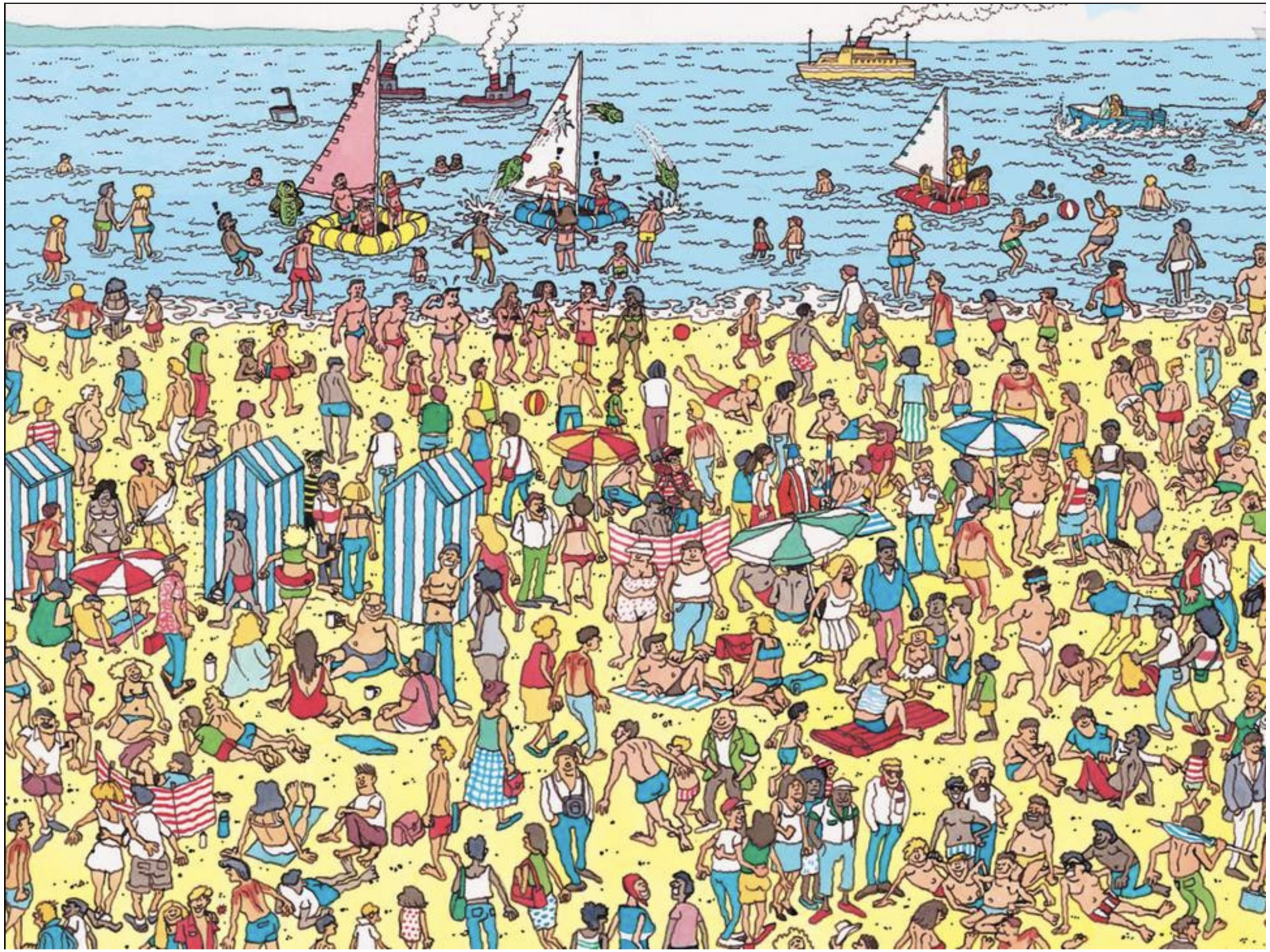
Using Feed-Forward NNs for Images

- Flattening tensors create an enormous amount of features.
- Predictive power declines with the added parameters estimated.



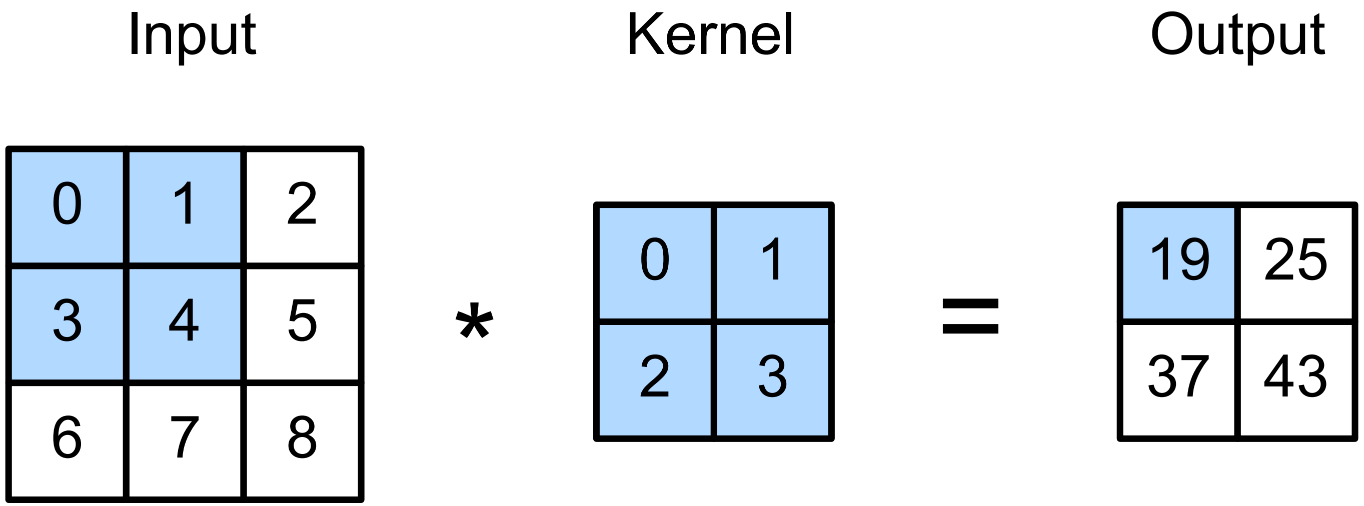
Python

Convolutions (think about Waldo)



Convolutions

- A convolution (kernel, mask, filter) is a small matrix used for blurring, sharpening, embossing, edge detection



$$
\begin{aligned}
0\times0+1\times1+3\times2+4\times3&=19,\\
1\times0+2\times1+4\times2+5\times3&=25,\\
3\times0+4\times1+6\times2+7\times3&=37,\\
4\times0+5\times1+7\times2+8\times3&=43.
\end{aligned}
$$

Convolutions

- Convolutions go over the image and produce an output

$$\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 5 & 3 & 8 & 4 \\ 3 & 3 & 2 & 8 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 6 & & \\ & & \\ & & \end{bmatrix}$$

7x1+4x1+3x1+
2x0+5x0+3x0+
3x-1+3x-1+2x-1
= 6

Convolutions

- Suppose we have an image $I_{n\times p}$ and a kernel $K_{k\times l}$
- The resulting image $O_{n-k+1 \ \times \ p-l+1}$

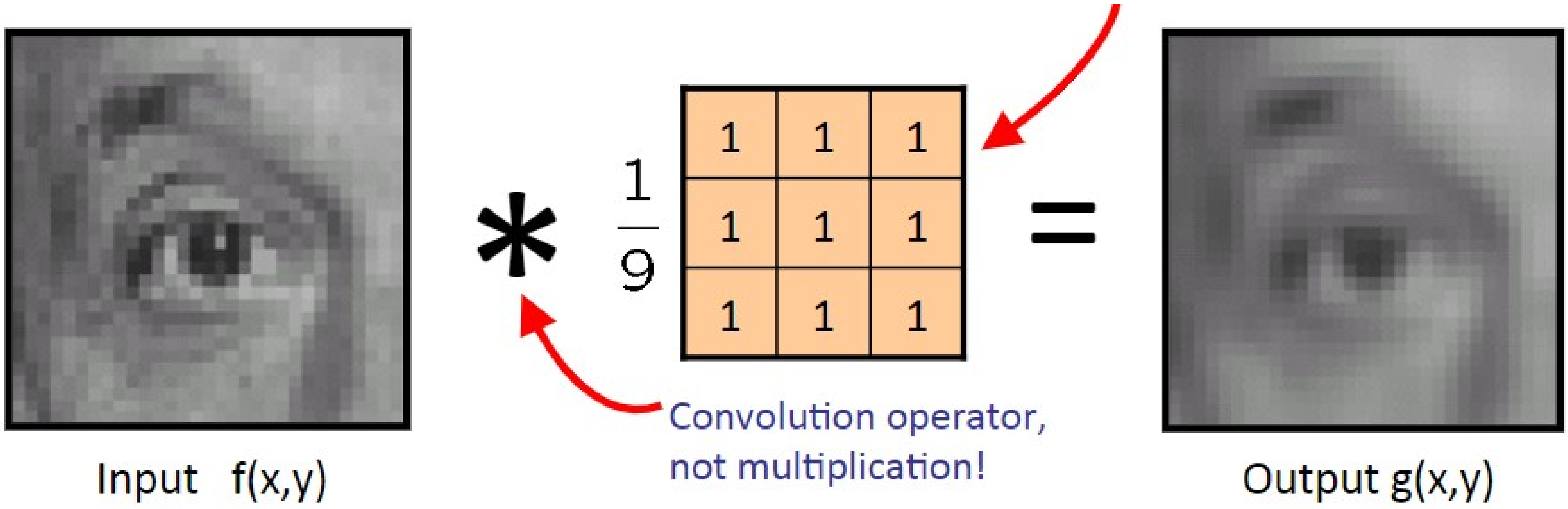
$$\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 5 & 3 & 8 & 4 \\ 3 & 3 & 2 & 8 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 6 & & \\ & & \\ & & \end{bmatrix}$$

7x1+4x1+3x1+
2x0+5x0+3x0+
3x-1+3x-1+2x-1
= 6

$I_{5\times 5}$ $K_{3\times}$ $O_{5-3+1 \times 5-3+1} = O_{3\times 3}$

Convolutions Use

- Convolutions can blur (downsample) images

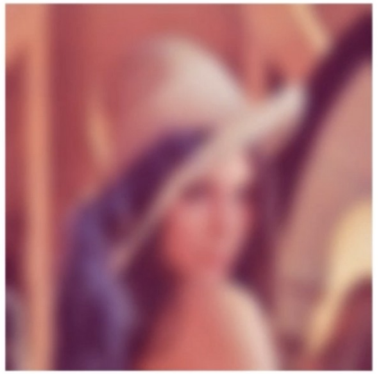


Convolutions Use

- ❑ We can create multiple levels of blurring
- ❑ In other words, create multiple images of the same one.

Input Image * Filter (Kernel) = Output Image

*



$$\begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix}$$

Convolutions Use

- We can also better detect edges.

Input Image * Filter (Kernel) = Output Image

*

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$\sqrt{G_x^2 + G_y^2}$$

Convolutions Use

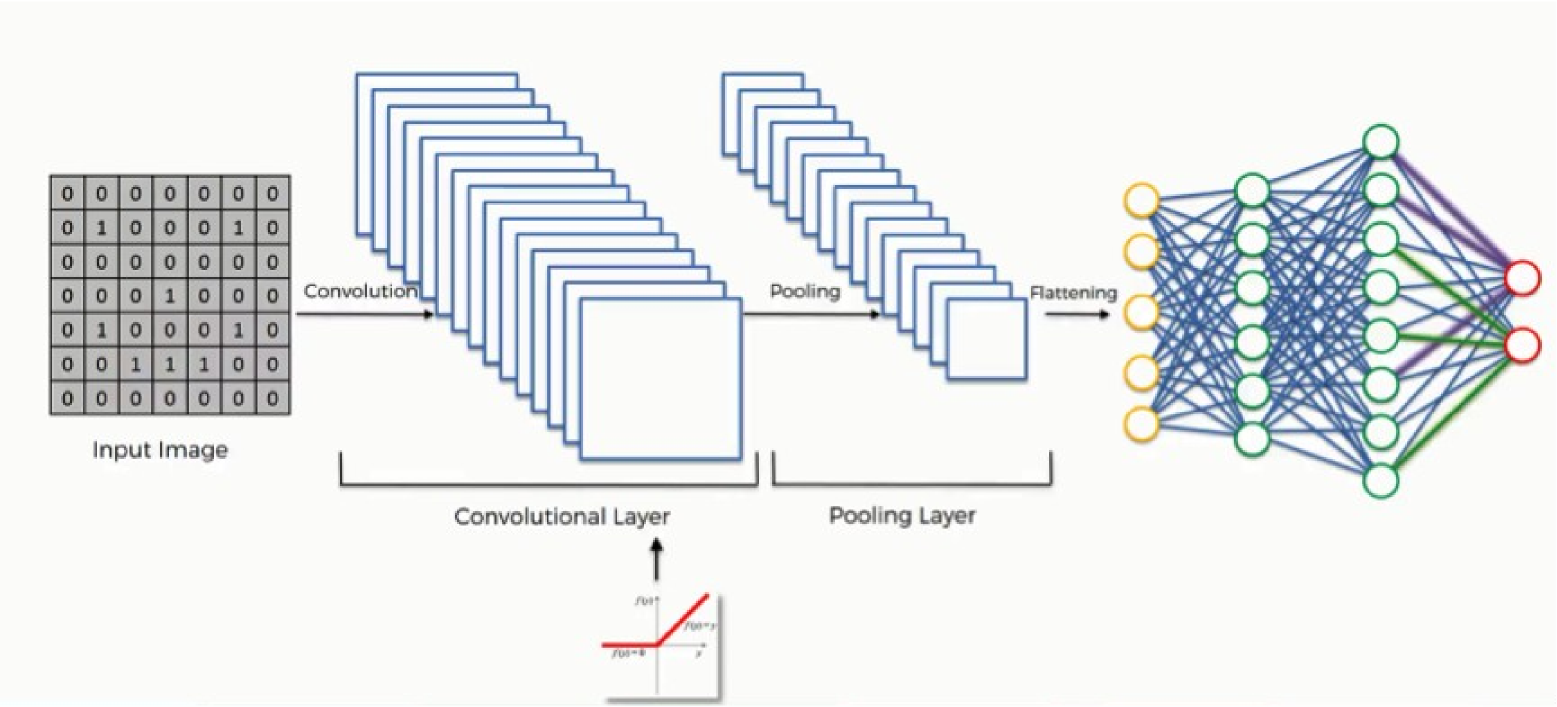
- Convolutions work to create new features.

Convolutions Use

- When we combine multiple convolutions we create a convolutional layer.

Convolutions Use

- When we combine multiple convolutions we create a convolutional layer.



Example

- Suppose we have the following image and convolution.

Image * Filter

0	10	10	0
20	30	30	20
10	20	20	10
0	5	5	0

1	0
0	2

What is the resulting output image shape

□ Start presenting to display the poll results on this slide.

Example

- Suppose we have the following image and convolution.

Image * Filter = Output

Image:

0	10	10	0
20	30	30	20
10	20	20	10
0	5	5	0

Filter:

1	0
0	2

Output:

X		

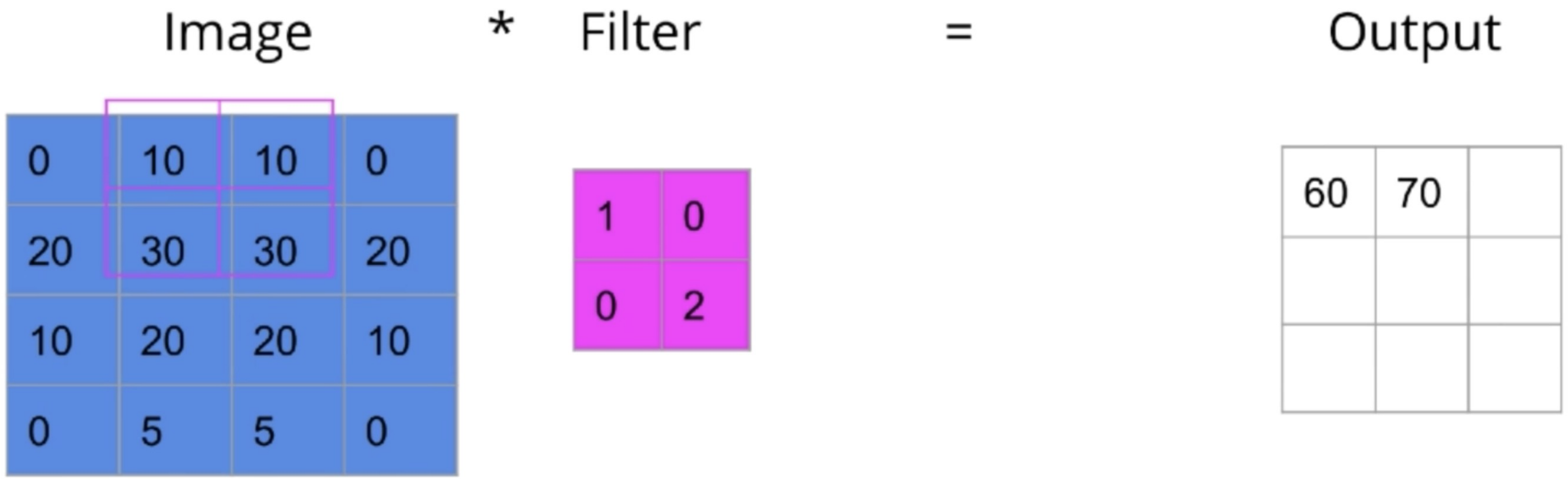
slido

What is the value of X?

Start presenting to display the poll results on this slide.

Example

- Suppose we have the following image and convolution.



$$1*10 + 0*10 + 0*30 + 2*30 = 70$$

Example

- Suppose we have the following image and convolution.

Image * Filter = Output

Image

0	10	10	0
20	30	30	20
10	20	20	10
0	5	5	0

Filter

1	0
0	2

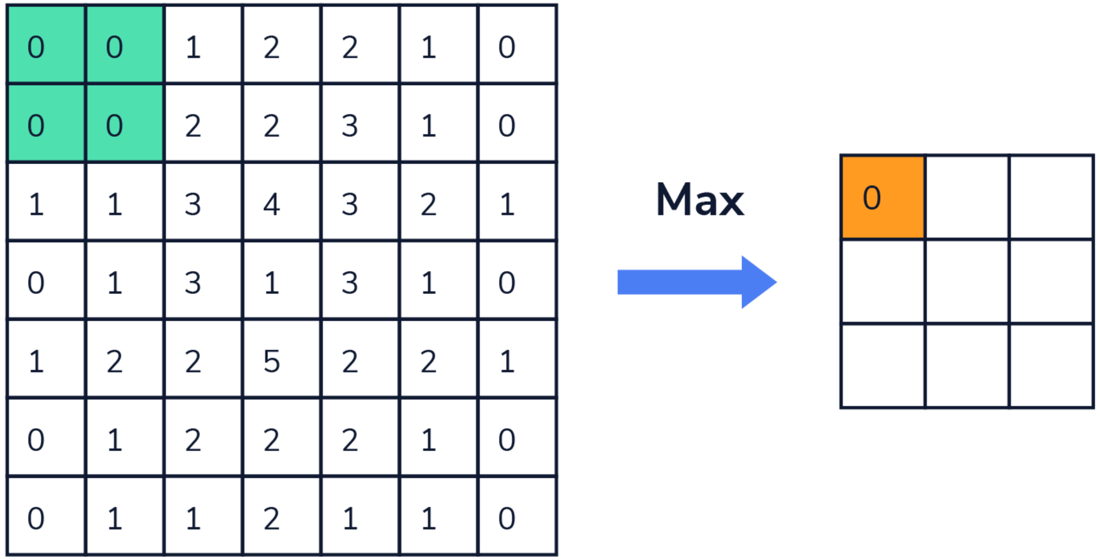
Output

60	70	50
60		

$$1*20 + 0*30 + 0*10 + 2*20 = 60$$

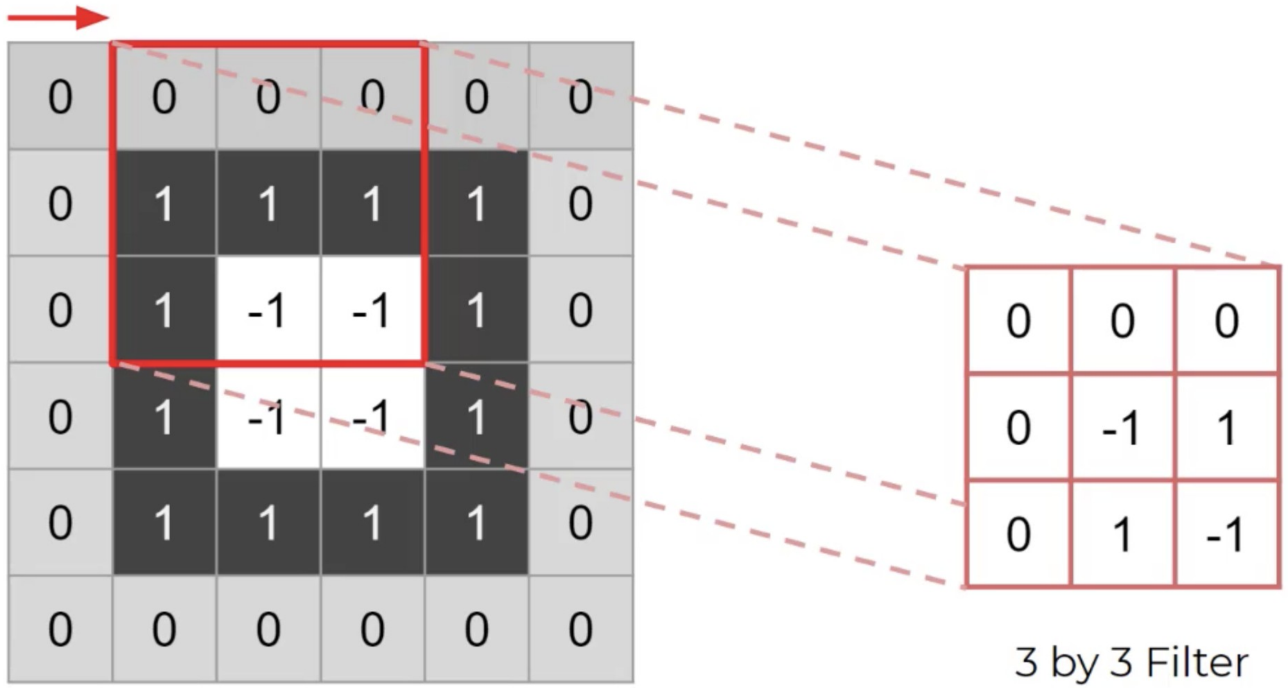
Pooling Filter

- Pooling is also used to downsample the images
- Pooling filters keep the important parts of the images.
- Max, Min and Average Pooling Filters are the most common



Stride

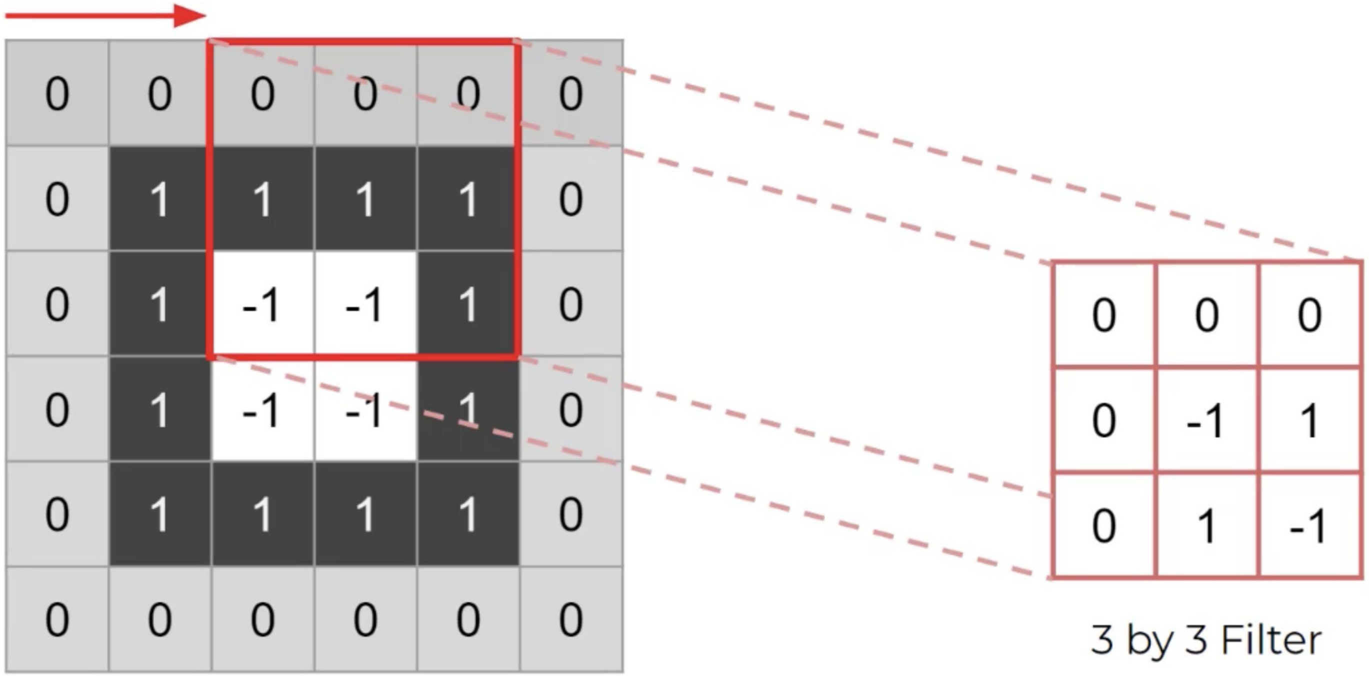
- ❑ Dictates the movement of the kernel, or filter, across the image
- ❑ Example of (1,1) stride (most common)



3 by 3 Filter

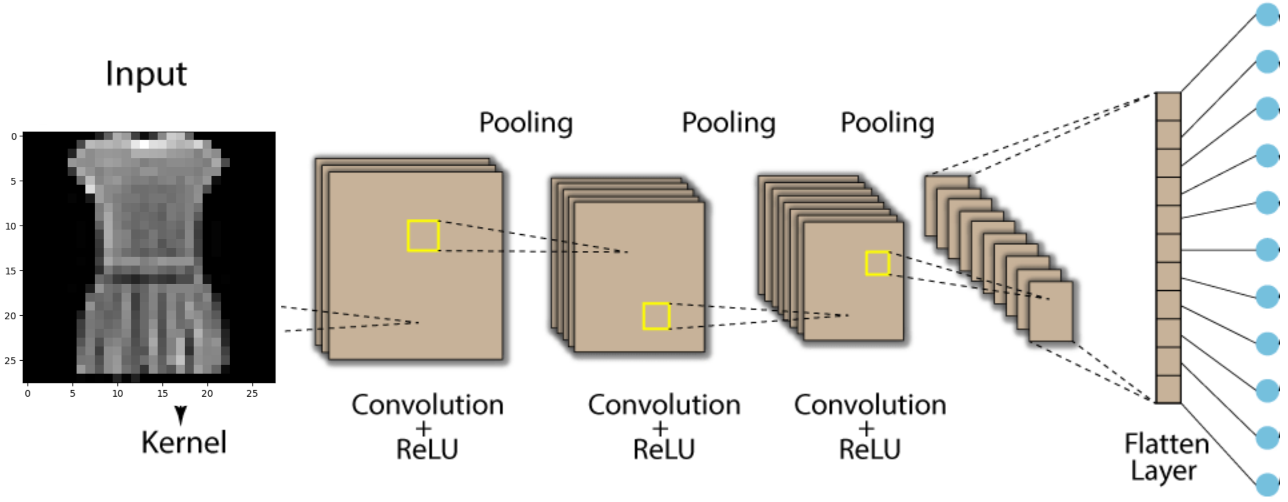
Stride

- Example of a (2,2) stride



Final CNN Product

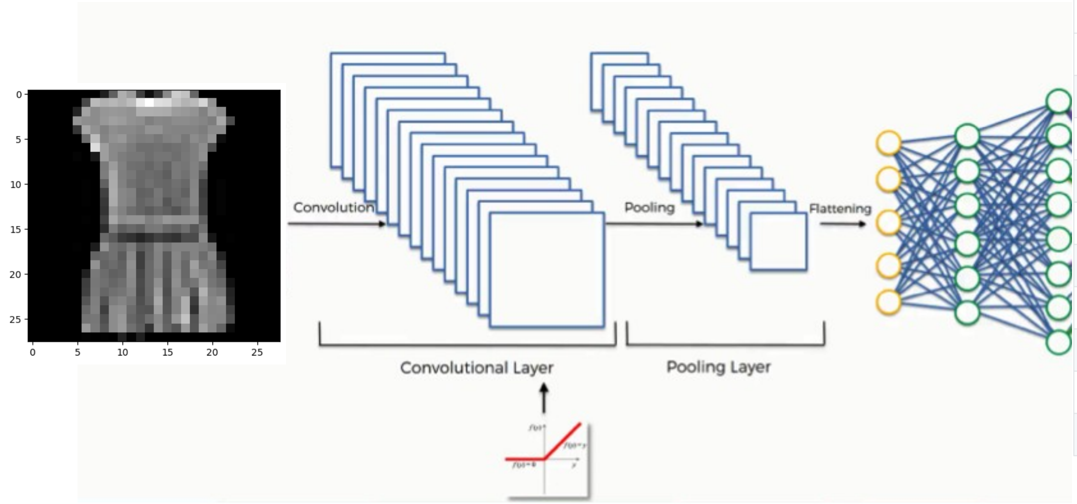
- We combine multiple convolutional layers and pooling layers
- We just set the size of the kernels
- We let the network optimize values of the filters
- We flatten and add a feed-forward neural net for further accuracy.



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Final CNN Product

- ❑ We combine multiple convolutional layers and pooling layers
- ❑ We just set the size of the kernels
- ❑ We let the network optimize values of the filters
- ❑ We flatten and add a feed-forward neural net for further accuracy.



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot