



Module 5

Convolutional Neural Networks



DL: Convolutional Nets

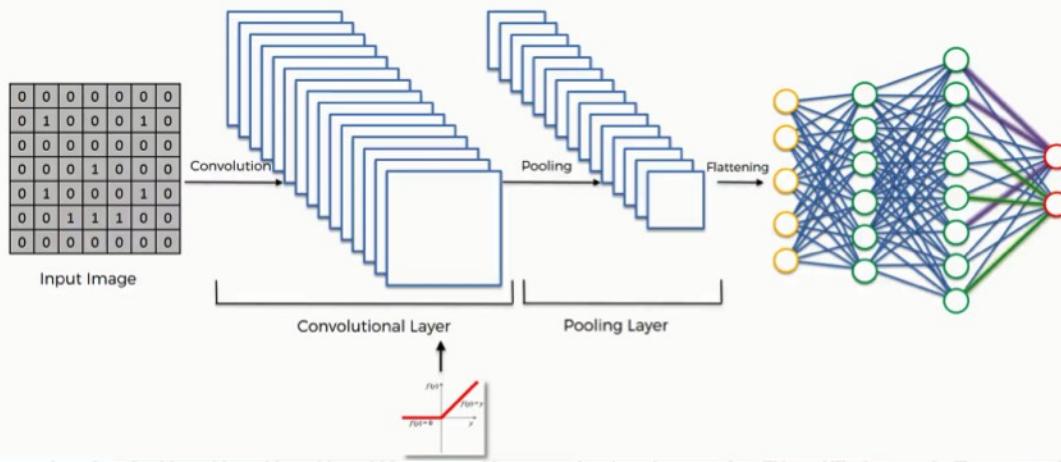


Image Representation

- 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.

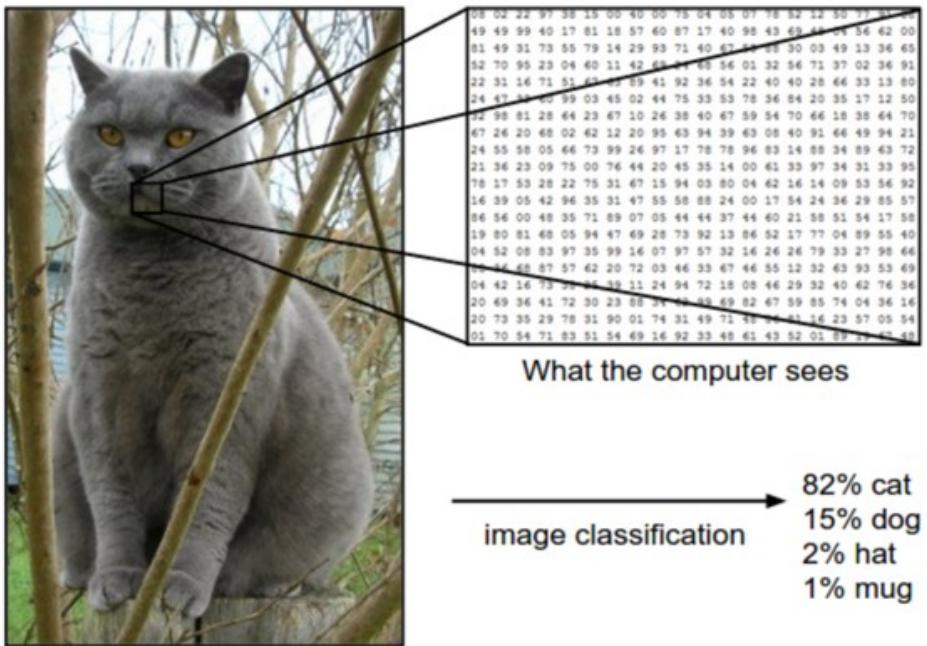


Image Representation

- ☐ Channels can be flattened to create a single vector.

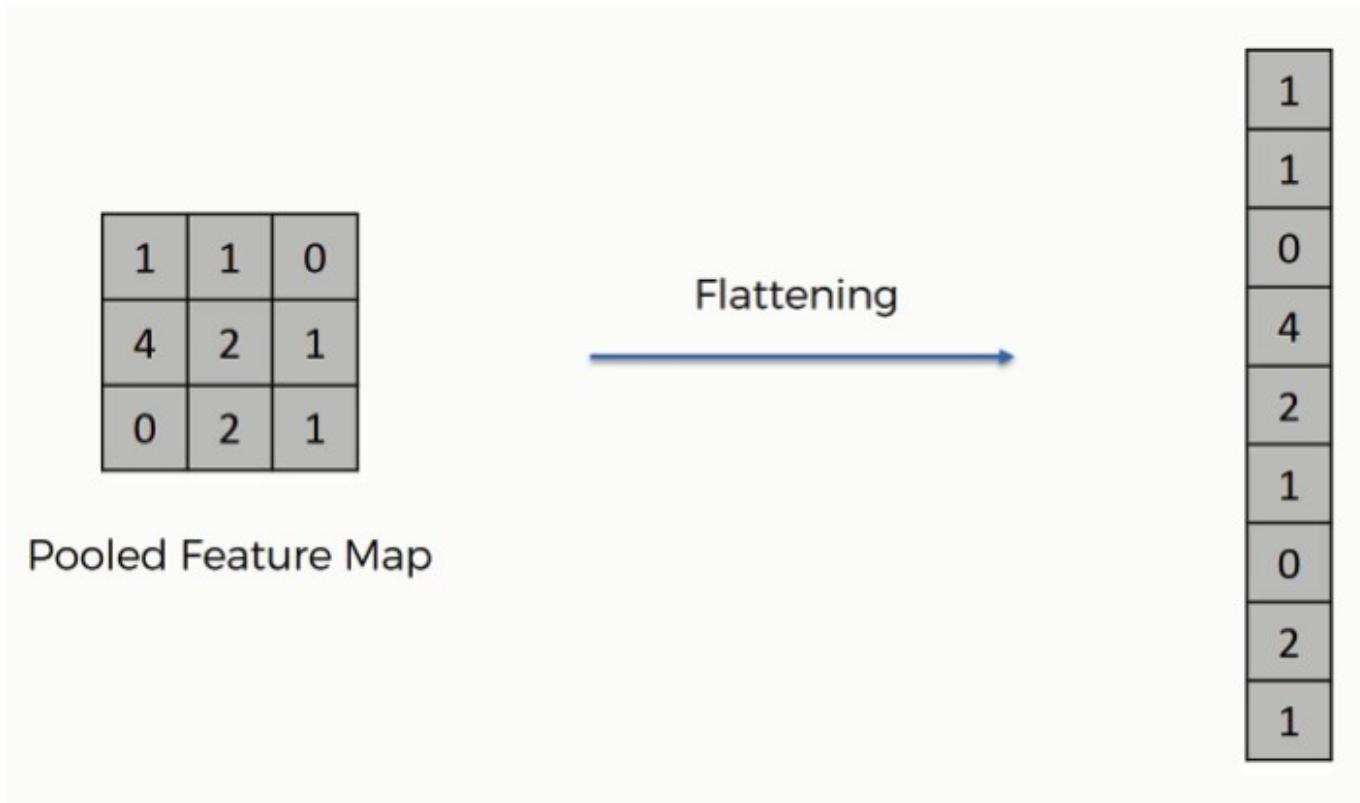
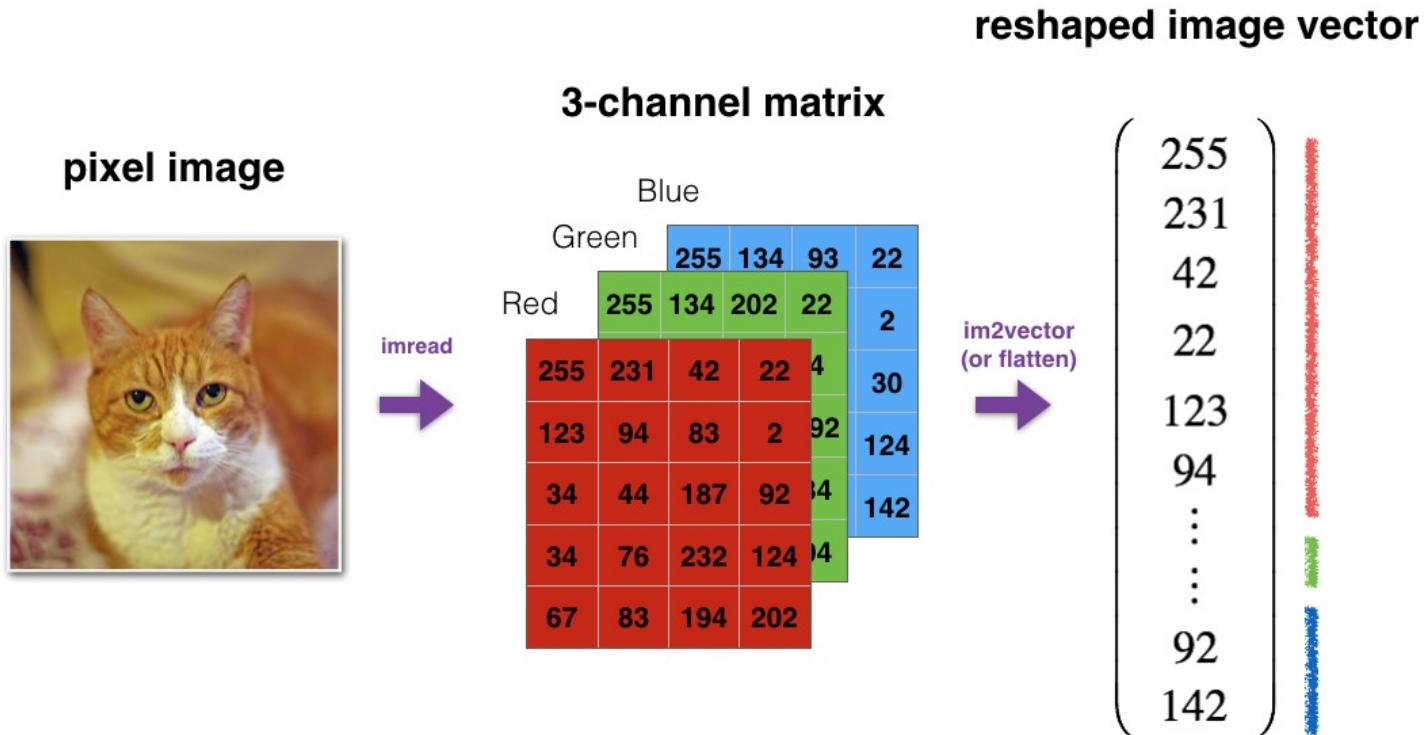


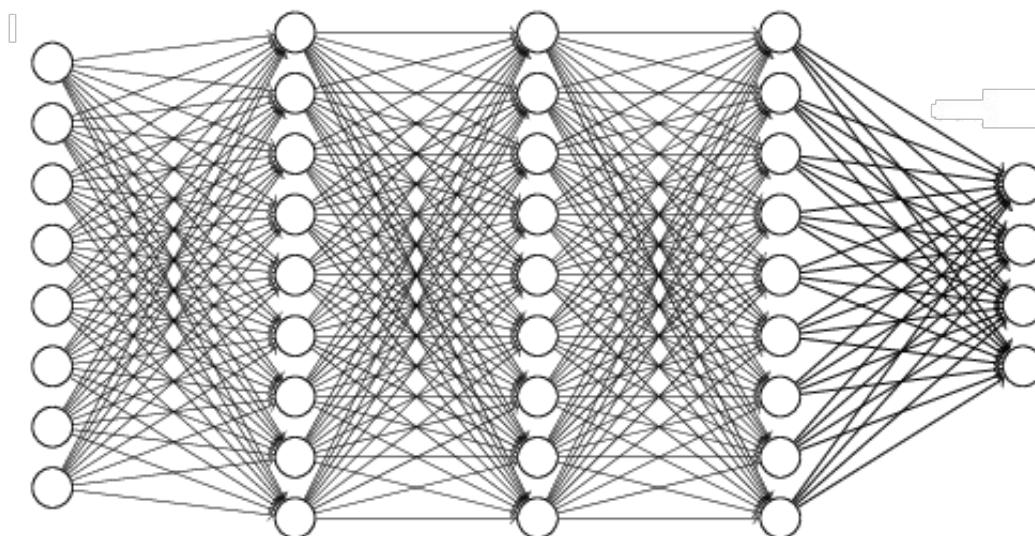
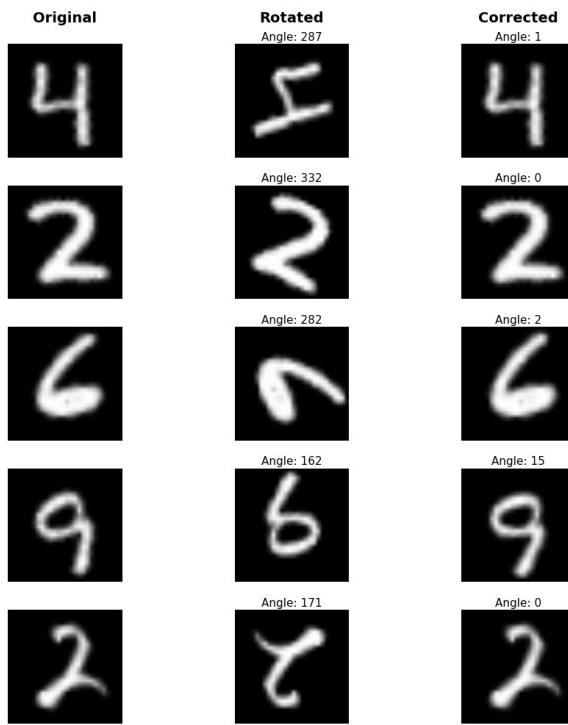
Image Representation

- Channels can be flattened to create a single vector.



Using Feed-Forward NNs for Images

- Feed forward NNs are not very effective with images

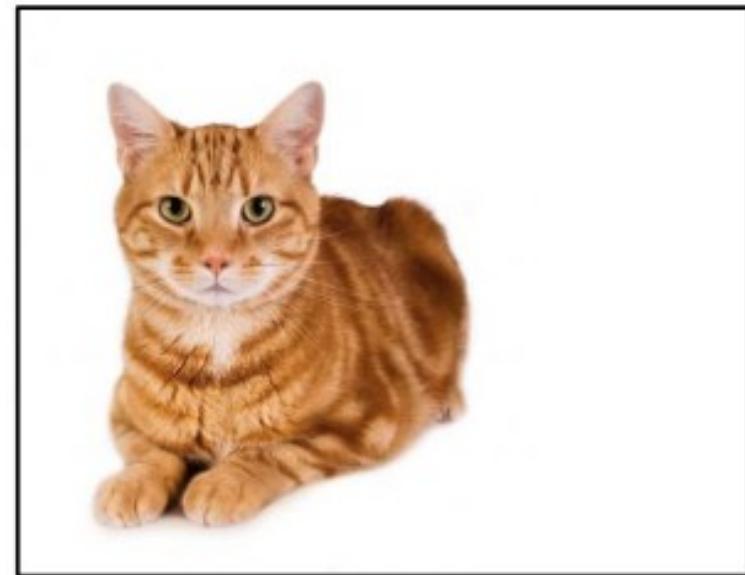


Using Feed-Forward NNs for Images

- Translation Invariance



Cat

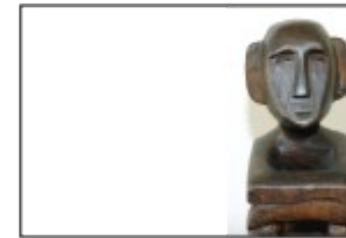


Cat

Using Feed-Forward NNs for Images

- Other Invariances cause issues

Translation Invariance



Rotation/Viewpoint Invariance



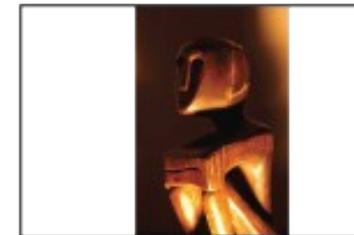
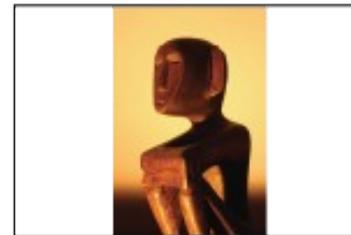
Using Feed-Forward NNs for Images

- Other Invariances cause issues

Size Invariance

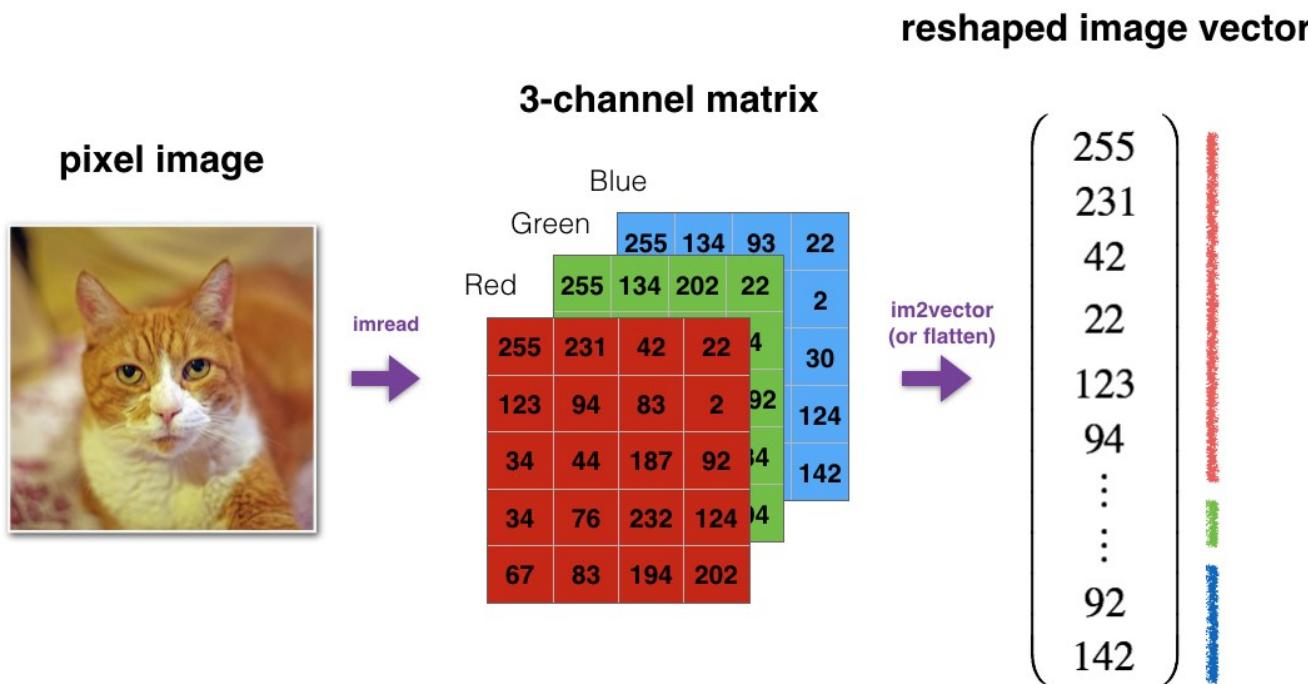


Illumination Invariance



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

Input	Kernel	Output																	
<table border="1"><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	$*$	<table border="1"><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table> $=$ <table border="1"><tr><td>19</td><td>25</td></tr><tr><td>37</td><td>43</td></tr></table>	0	1	2	3	19	25	37	43
0	1	2																	
3	4	5																	
6	7	8																	
0	1																		
2	3																		
19	25																		
37	43																		

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$

Convolutions

- Convolutions go over the image and produce an output

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

*

1	0	-1
1	0	-1
1	0	-1

=

6		

$$\begin{aligned} & 7 \times 1 + 4 \times 1 + 3 \times 1 + \\ & 2 \times 0 + 5 \times 0 + 3 \times 0 + \\ & 3 \times -1 + 3 \times -1 + 2 \times -1 \\ & = 6 \end{aligned}$$

Convolutions

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

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

*

1	0	-1
1	0	-1
1	0	-1

=

6		

$$\begin{aligned} & 7 \times 1 + 4 \times 1 + 3 \times 1 + \\ & 2 \times 0 + 5 \times 0 + 3 \times 0 + \\ & 3 \times -1 + 3 \times -1 + 2 \times -1 \\ & = 6 \end{aligned}$$

$I_{5 \times 5}$

$K_{3 \times 3}$

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

Convolutions Use

- Convolutions can blur (downsample) images

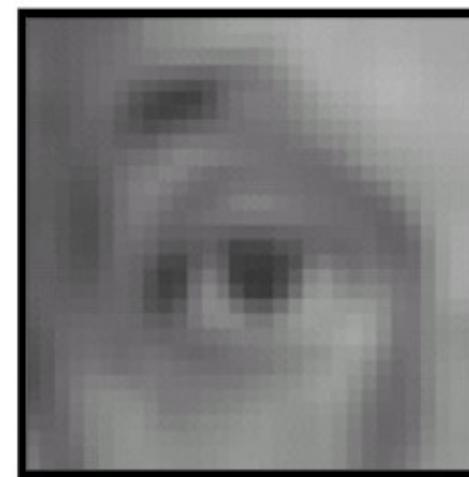


Input $f(x,y)$

$$\text{Input } f(x,y) * \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = \text{Output } g(x,y)$$

Convolution operator,
not multiplication!

A diagram illustrating a convolution operation. On the left is the input image $f(x,y)$. In the center is a convolution operator represented as a 3x3 matrix of ones divided by 9. To the right is the output image $g(x,y)$, which is a blurred version of the input. Red arrows indicate the operation: one from the input to the operator, another from the operator to the output, and a third from the operator to the text "Convolution operator, not multiplication!".



Output $g(x,y)$

Convolutions Use

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

Input Image



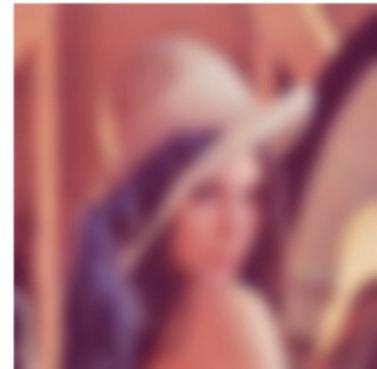
*

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)

$$G_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$G_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

=

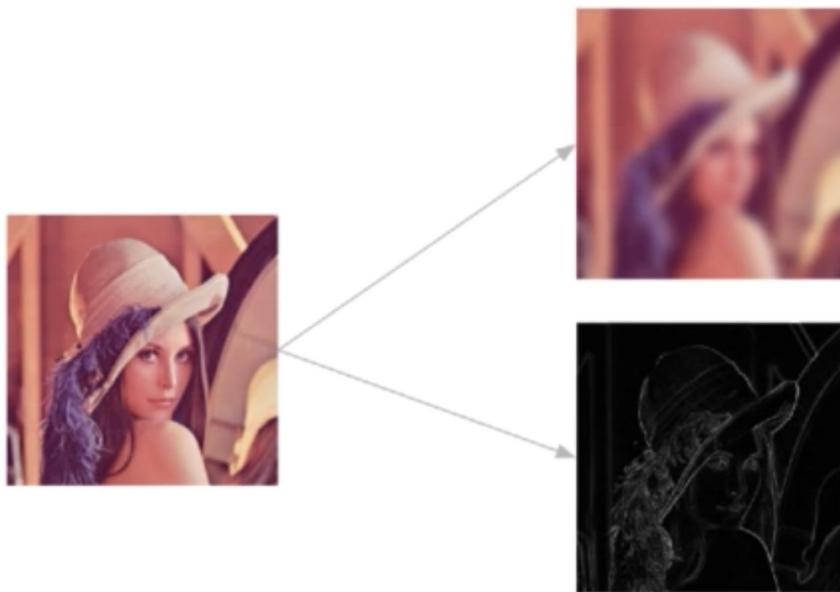
Output Image

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



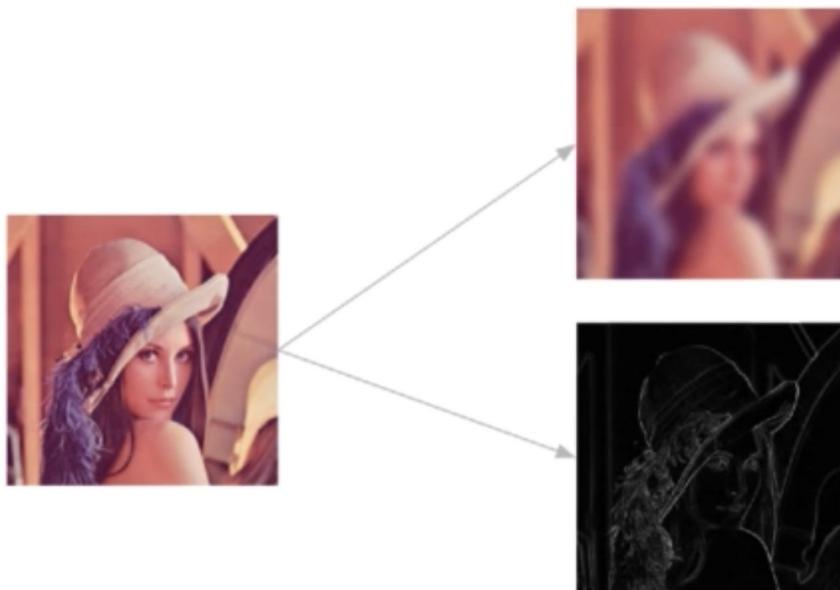
Convolutions Use

- Convolutions work to create new features.



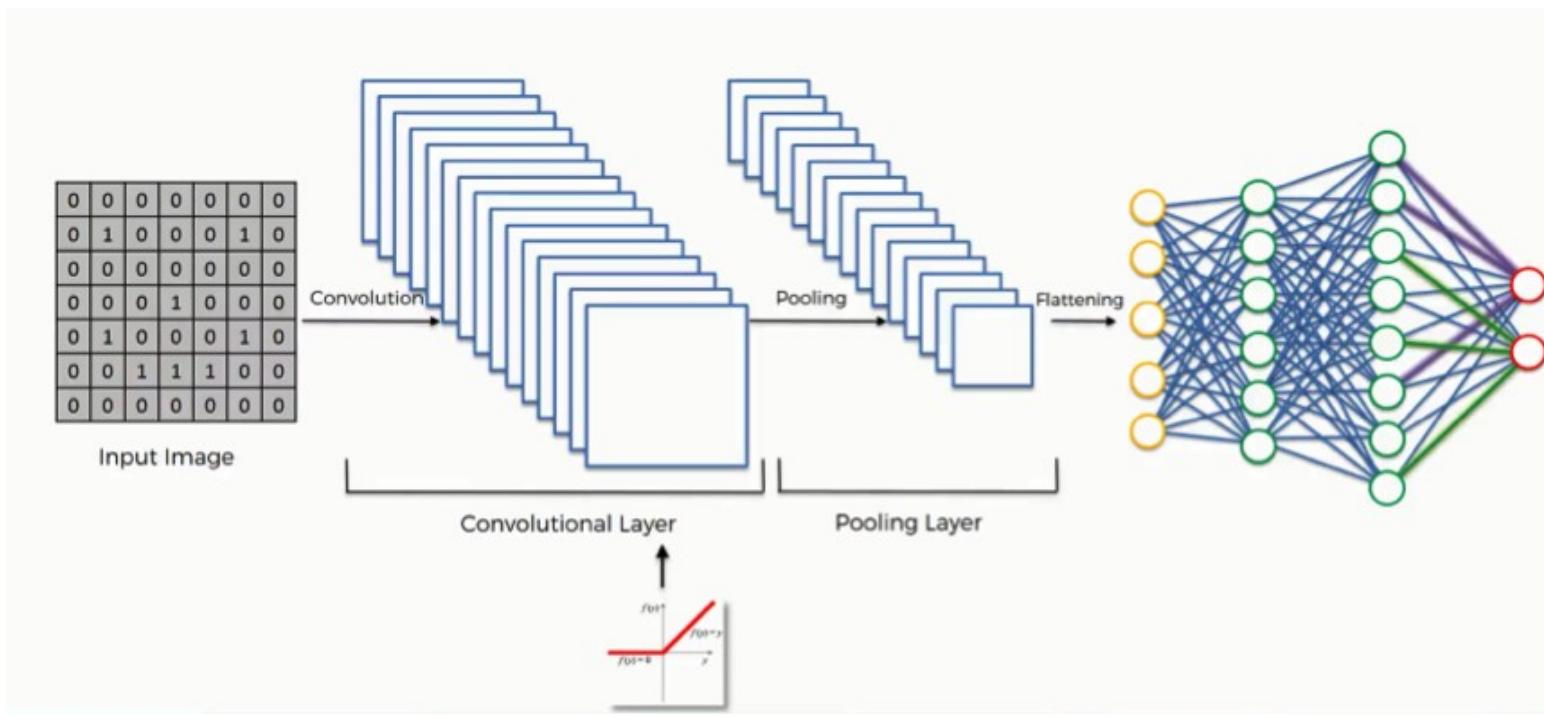
Convolutions Use

- 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

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

*

Filter

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.

$$\begin{array}{c} \text{Image} \\ \begin{array}{|c|c|c|c|} \hline 0 & 10 & 10 & 0 \\ \hline 20 & 30 & 30 & 20 \\ \hline 10 & 20 & 20 & 10 \\ \hline 0 & 5 & 5 & 0 \\ \hline \end{array} \end{array} * \begin{array}{c} \text{Filter} \\ \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 2 \\ \hline \end{array} \end{array} = \begin{array}{c} \text{Output} \\ \begin{array}{|c|c|c|} \hline X & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{array}$$

The diagram illustrates a convolution operation. The input image is a 4x4 matrix with values: [0, 10, 10, 0], [20, 30, 30, 20], [10, 20, 20, 10], and [0, 5, 5, 0]. A 2x2 filter with weights [1, 0] and [0, 2] is applied. The result is an output matrix where the top-left element is marked with a large red 'X', indicating a failure or an error in the convolution process.

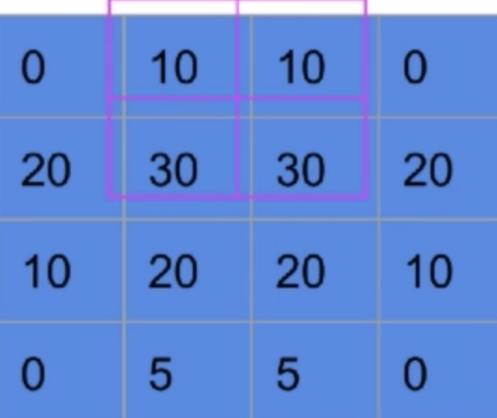
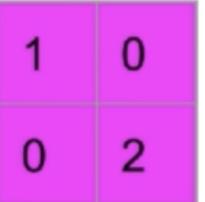
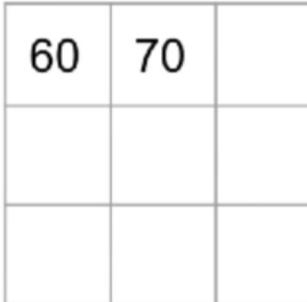


What is the value of X?

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

Example

- Suppose we have the following image and convolution.

Image	*	Filter	=	Output
	*		=	

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

Example

- Suppose we have the following image and convolution.

Image	*	Filter	=	Output																													
<table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>0</td><td>10</td><td>10</td><td>0</td></tr><tr><td>20</td><td>30</td><td>30</td><td>20</td></tr><tr><td>10</td><td>20</td><td>20</td><td>10</td></tr><tr><td>0</td><td>5</td><td>5</td><td>0</td></tr></table>	0	10	10	0	20	30	30	20	10	20	20	10	0	5	5	0	*	<table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>1</td><td>0</td></tr><tr><td>0</td><td>2</td></tr></table>	1	0	0	2	=	<table border="1" style="border-collapse: collapse; text-align: center;"><tr><td>60</td><td>70</td><td>50</td></tr><tr><td>60</td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	60	70	50	60					
0	10	10	0																														
20	30	30	20																														
10	20	20	10																														
0	5	5	0																														
1	0																																
0	2																																
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

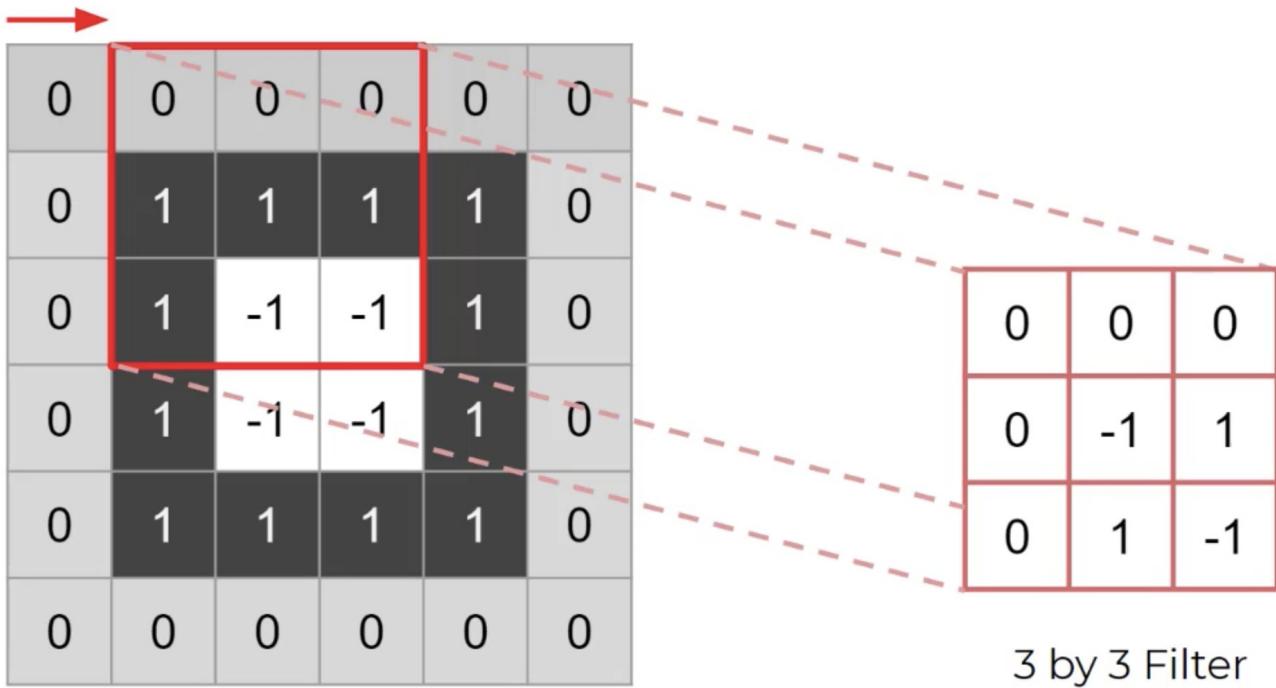
0	0	1	2	2	1	0
0	0	2	2	3	1	0
1	1	3	4	3	2	1
0	1	3	1	3	1	0
1	2	2	5	2	2	1
0	1	2	2	2	1	0
0	1	1	2	1	1	0

Max
→

0		

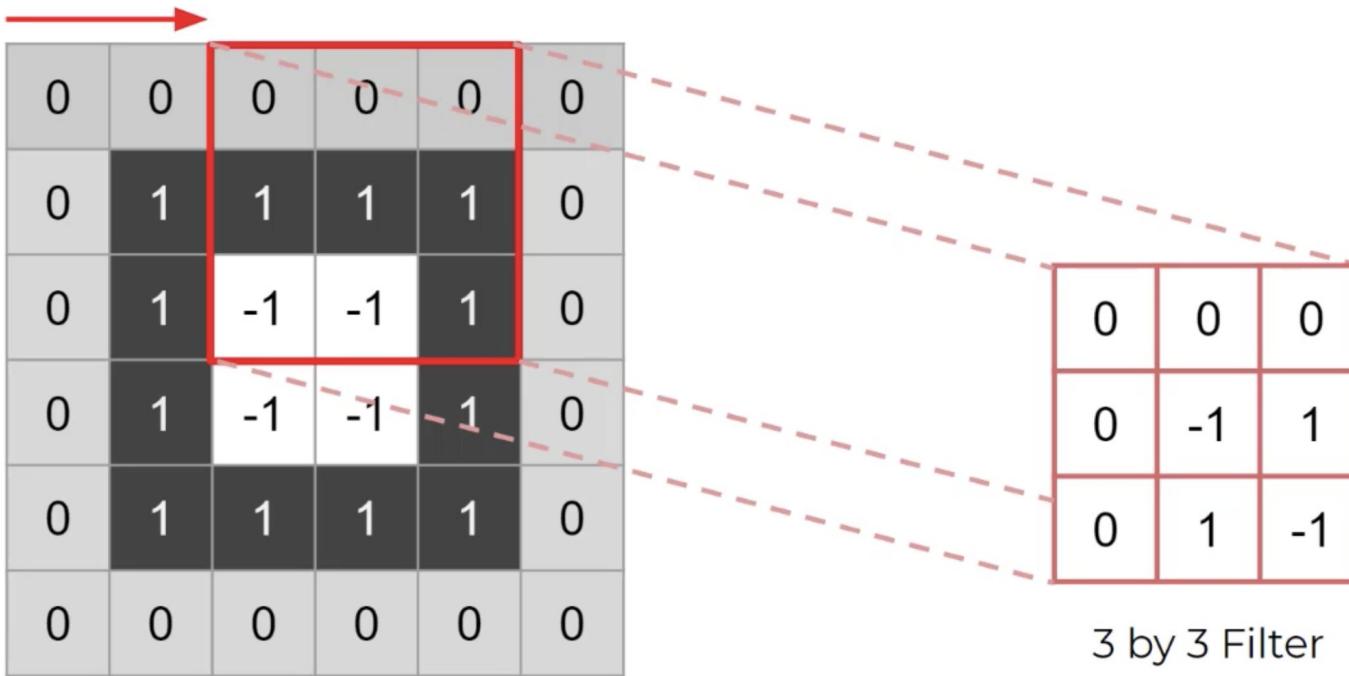
Stride

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



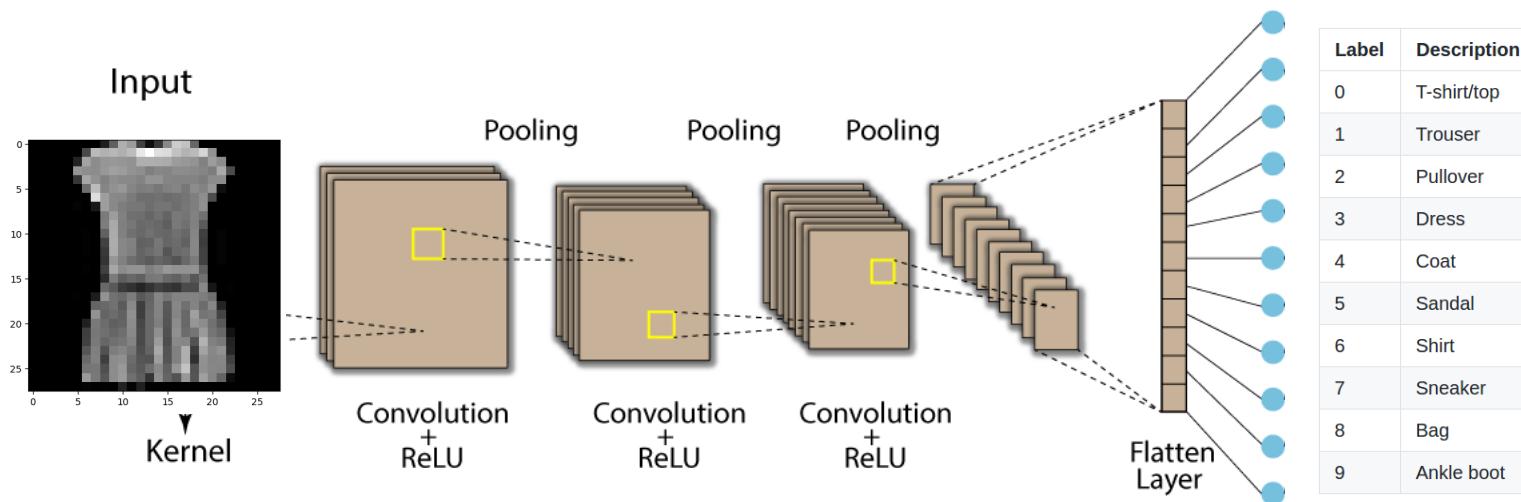
Stride

- 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.



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.

