I hear you want to see more kittens

CIS 522: Lecture 4T

Convolutional Neural Nets
02/04/20

Penn Engineering

Adagrad vs RMSprop

I was right when I thought I was wrong

They really only differ in the effect of time

Trick of the day: consider freezing your random seed

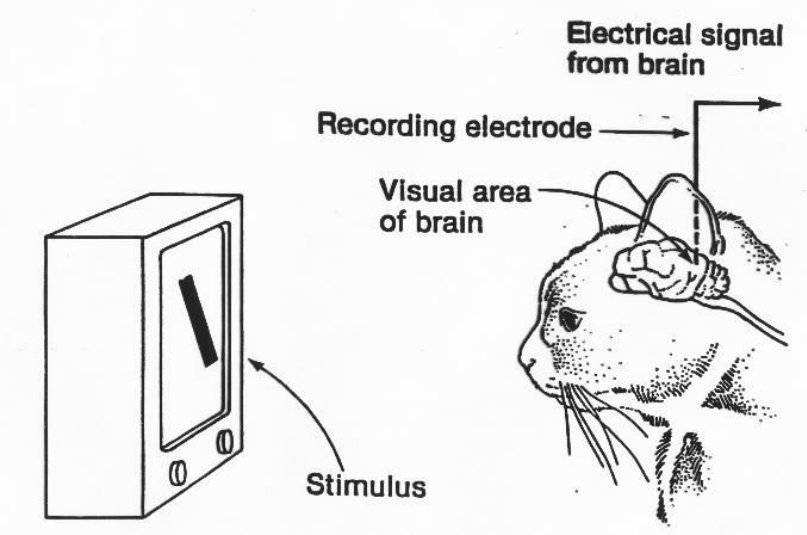
Outline

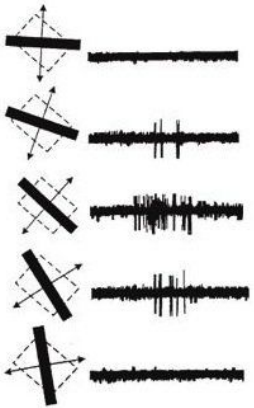
- Neuroscience motivation for convolutional neural networks (CNN)
- Definitions and intuitions
- Parts of a CNN
  - Convolutional layers
  - Pooling layers
  - Putting it together
- Properties of CNNs
- Next Week: all the application examples

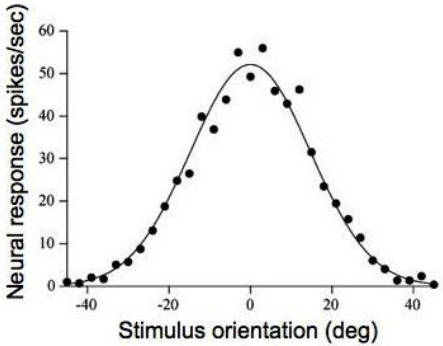
Visual processing in the brain

Penn Engineering

Hubel and Wiesel

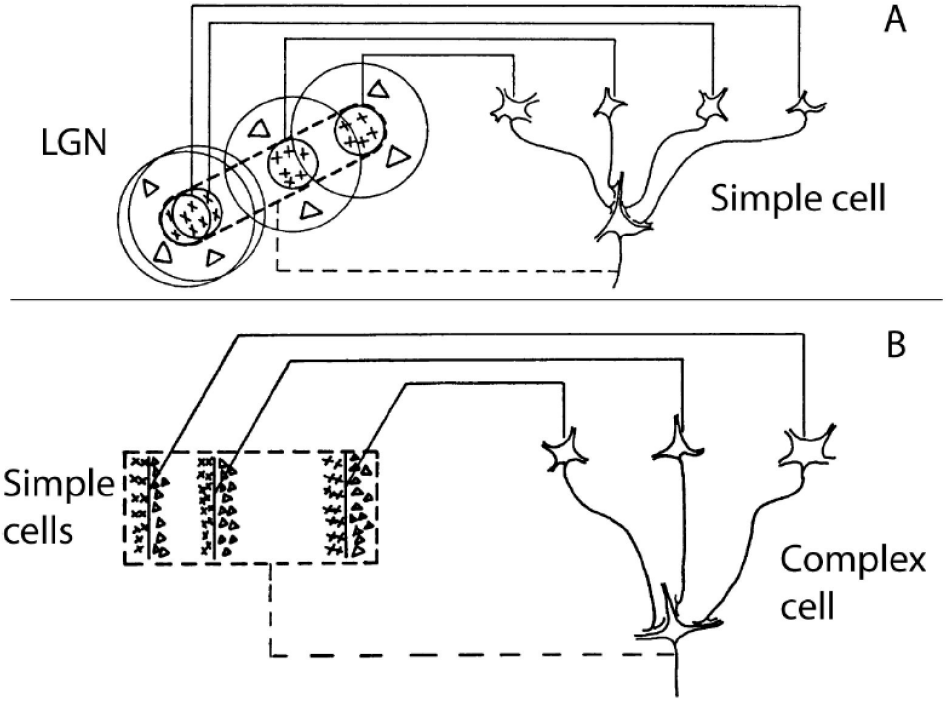




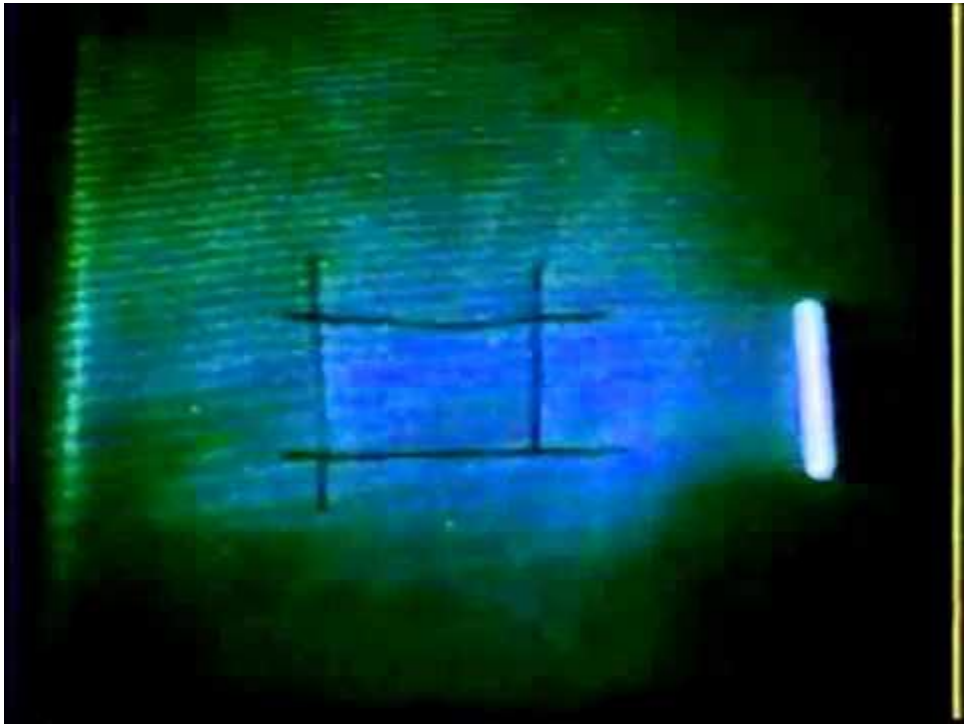


Hubel & Wiesel, 1968

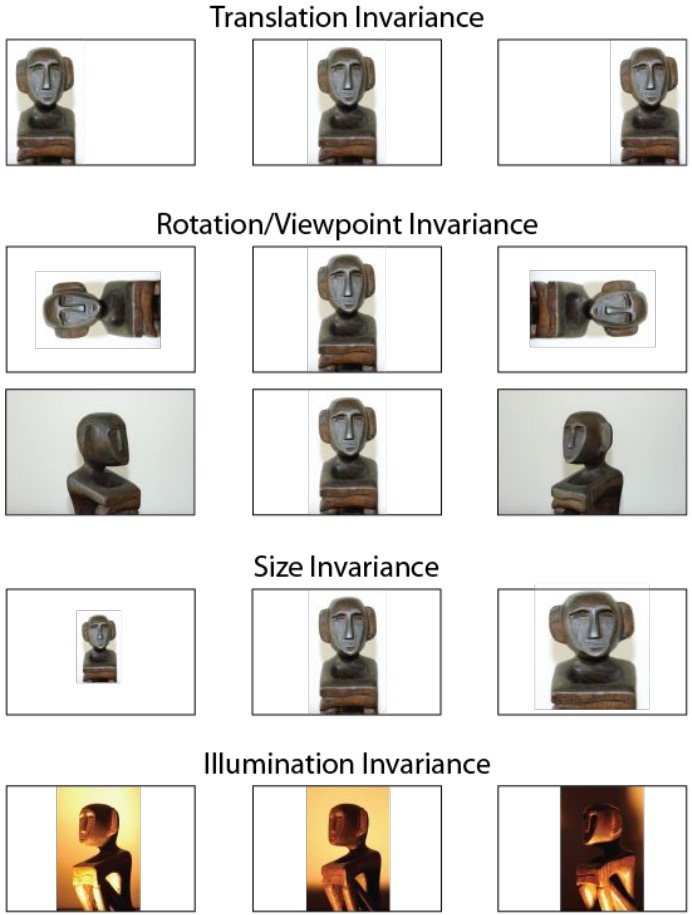
More mapping



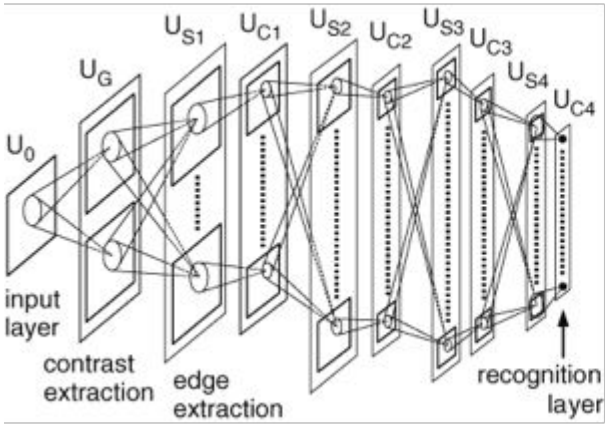
Complex cells

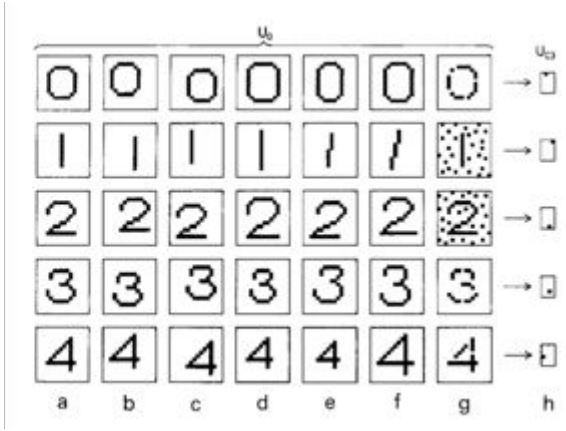


Invariances



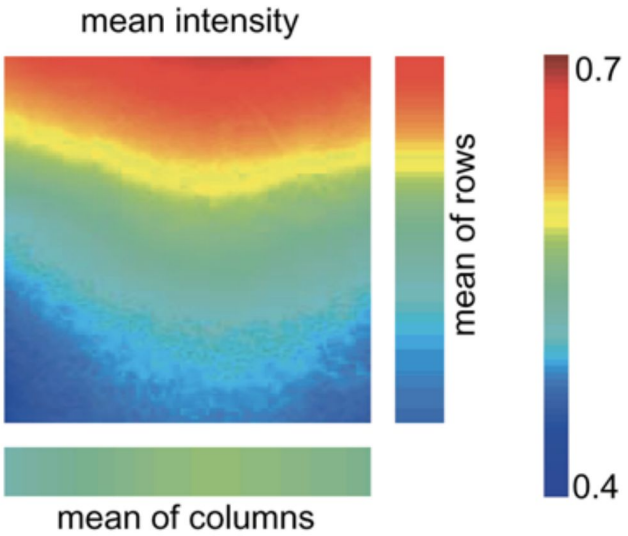
Neocognitron 1980





Fukushima Neocognitron 1980

Not entirely true

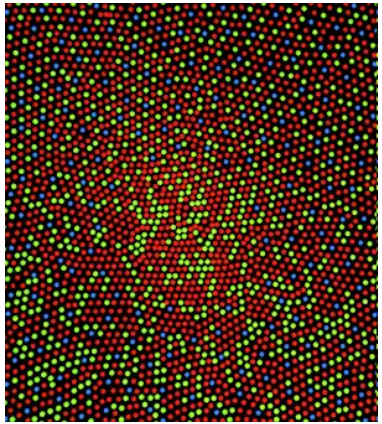


Stronger occipital cortical activation to lower than upper visual field stimuli Neuromagnetic recordings

Neuromagnetic recordings

Authors | Authors and affiliations

K. Portin, Simo Vanni, Veijo Virsu, Riitta Hari



Betsch et al 2004

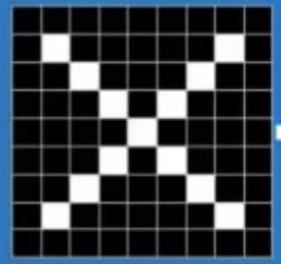
What is a CNN?

Penn Engineering

What is a CNN?

All following (blue) slides from Brandon Rohrer

Setting



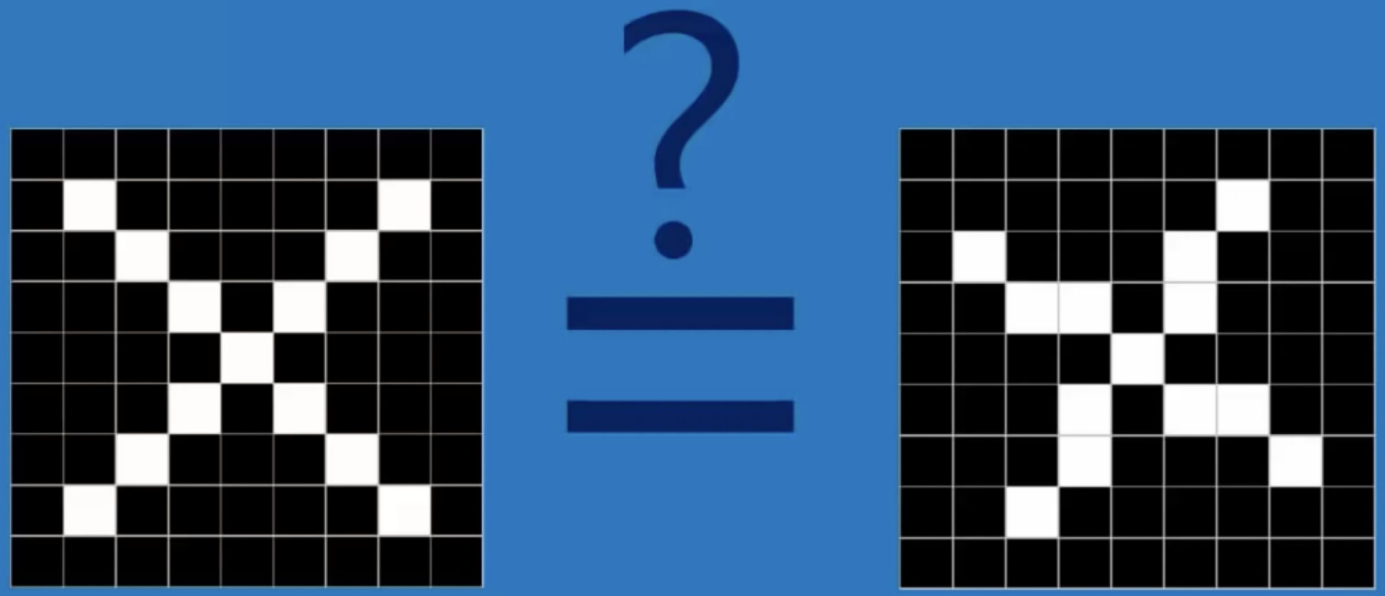
CNN

X

CNN

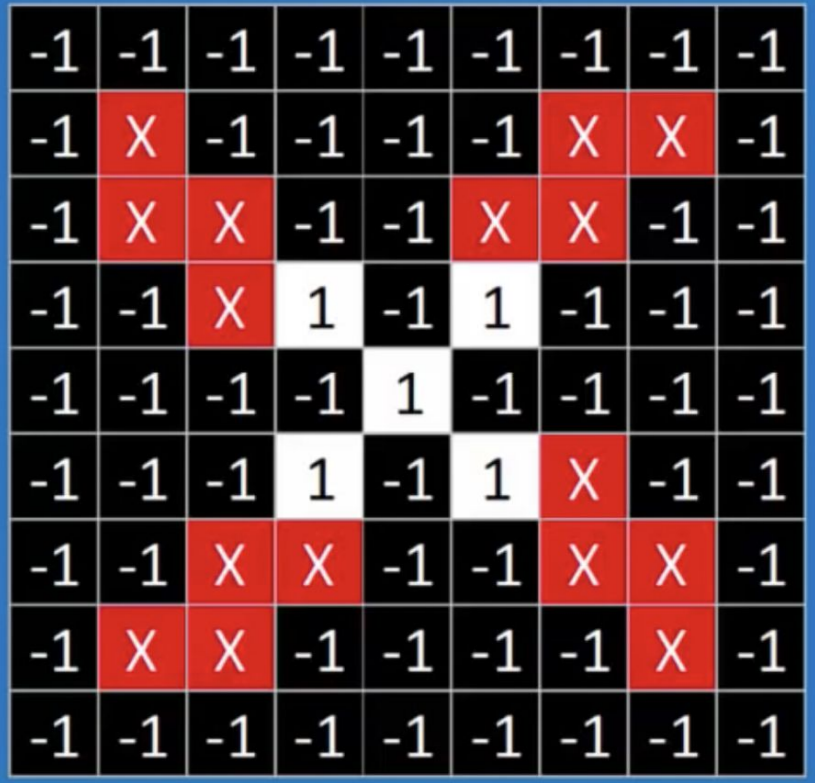
O

Nontrivial



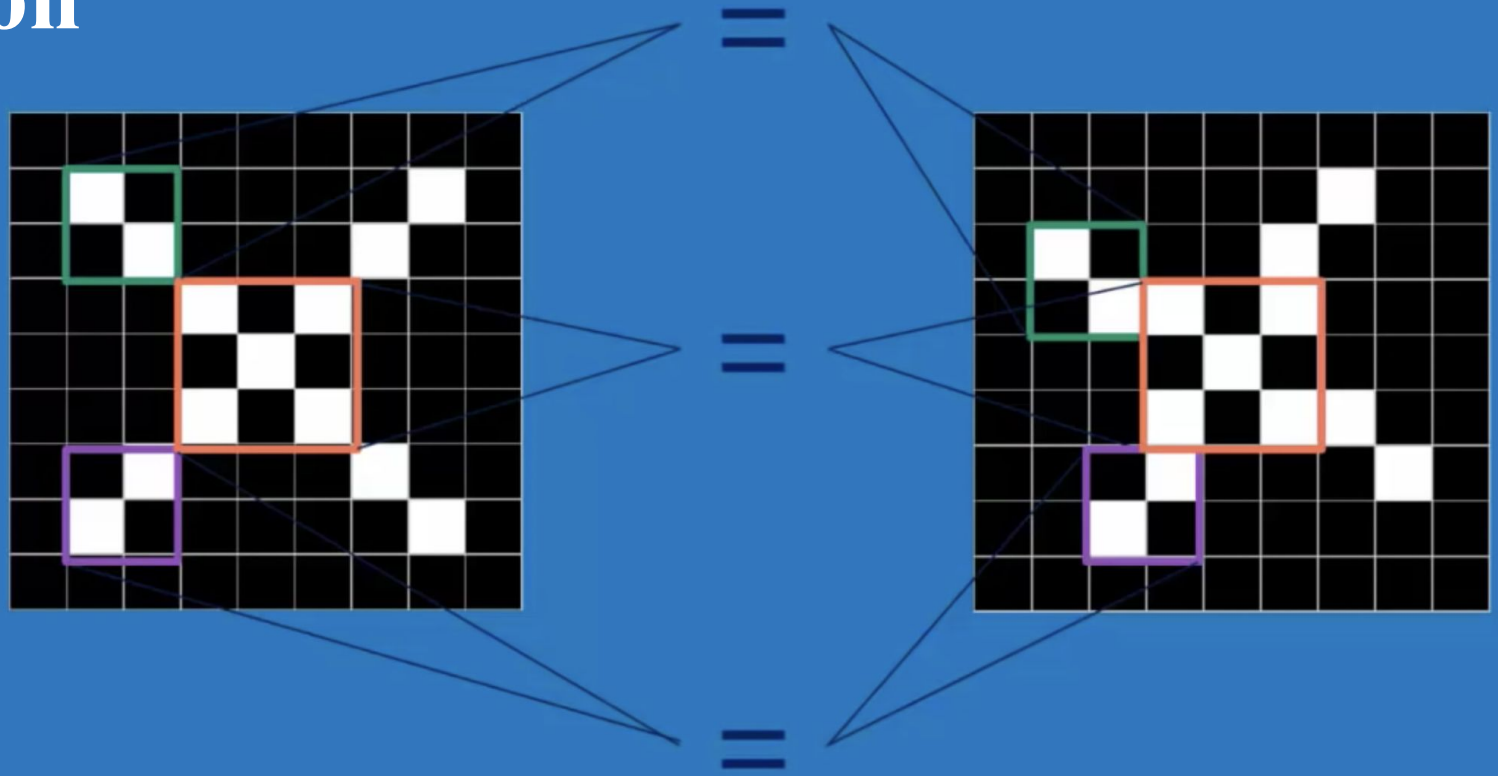
Brandon Rohrer

Naive comparison

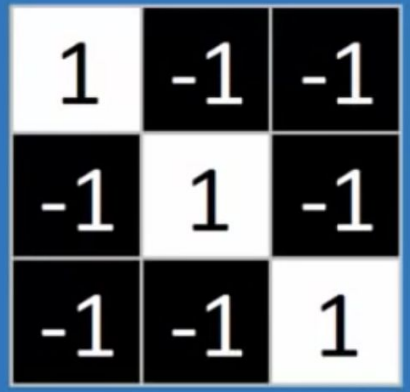


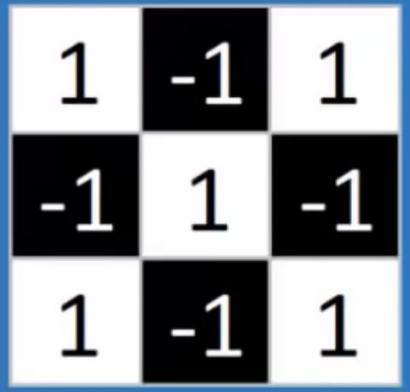
Brandon Rohrer

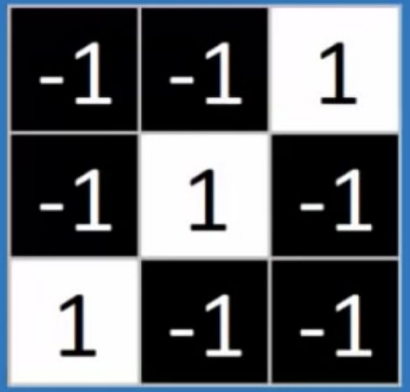
Intuition



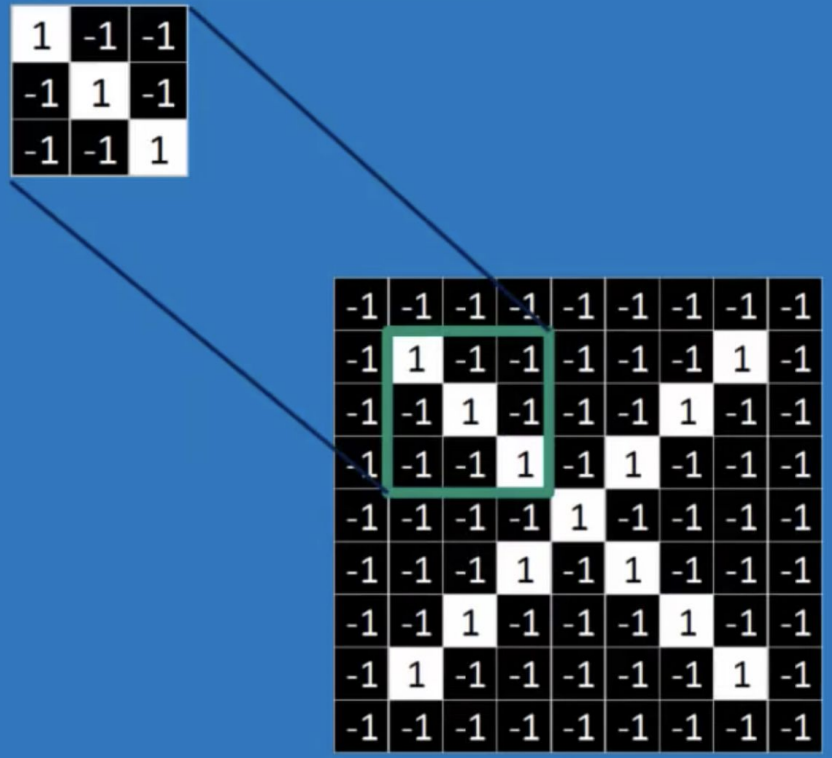
Potential local features



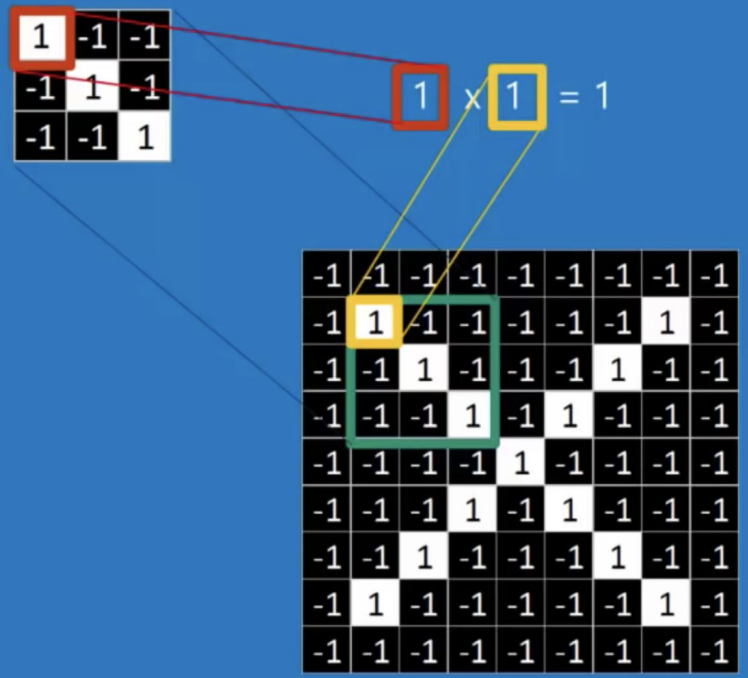




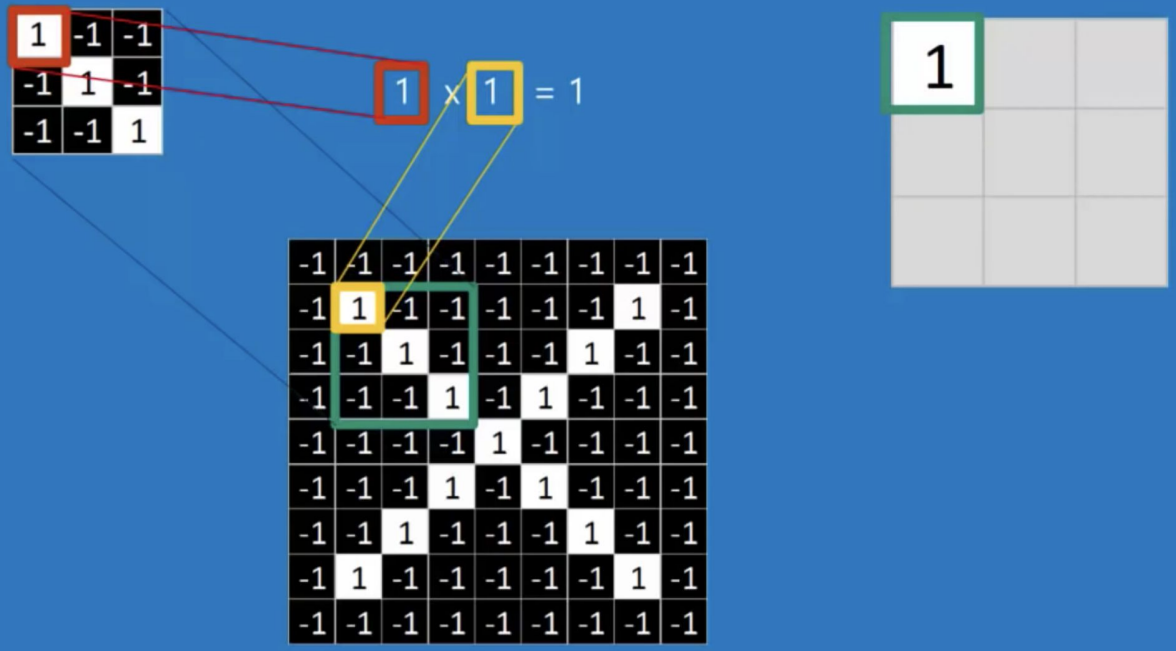
Setting


1 -1 -1
-1 1 -1
-1 -1 1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1

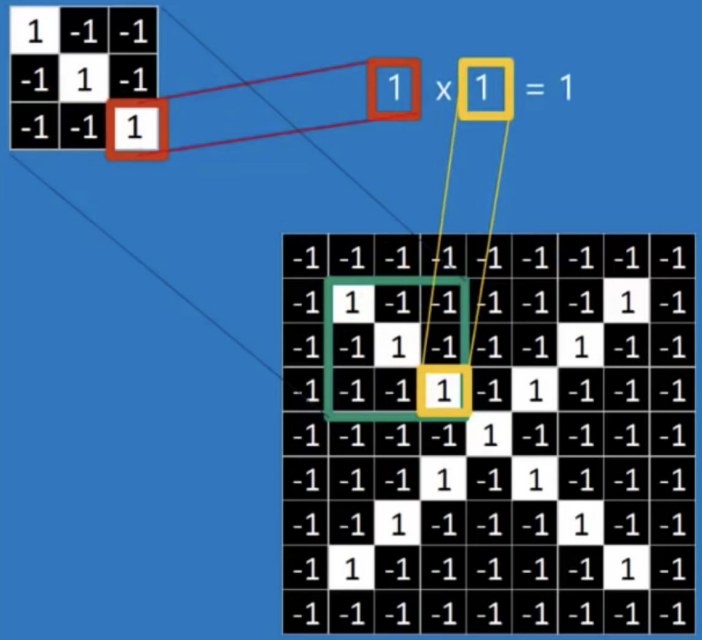
Local filtering



Multiplying the image with the filter (local)



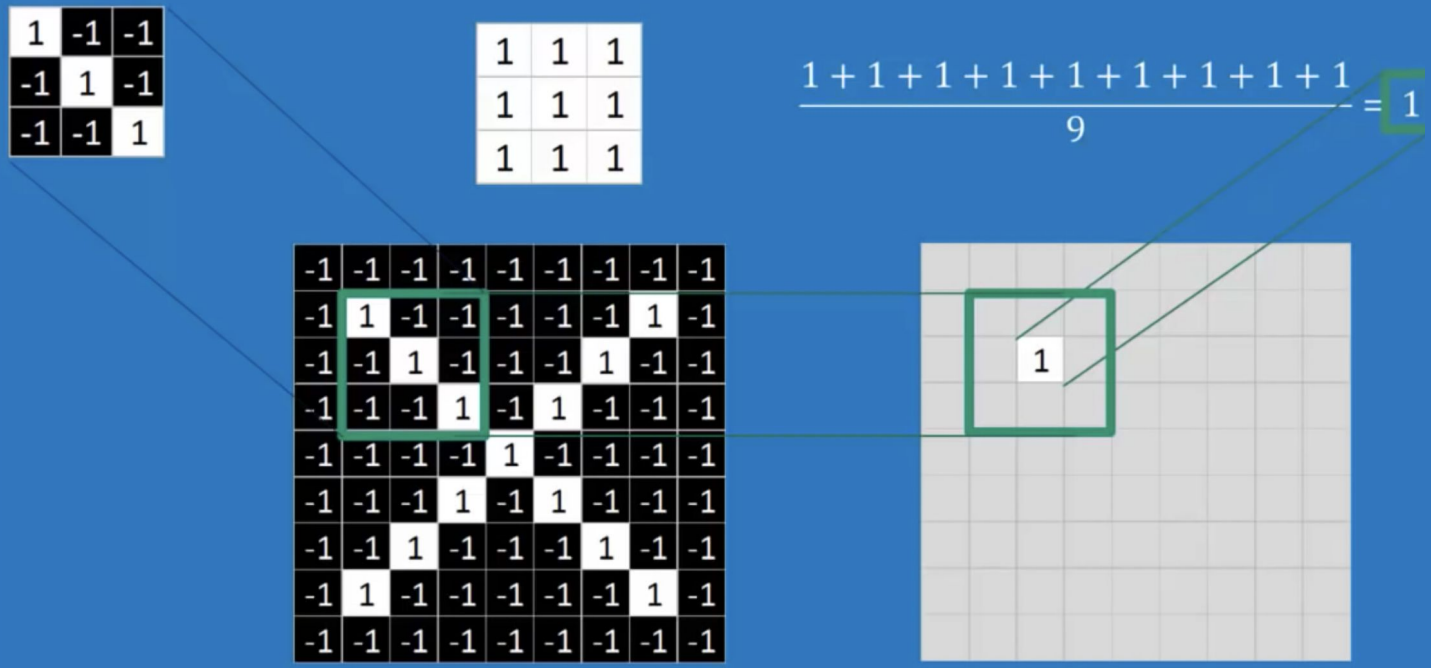
Continuing the filtering



1	1	1
1	1	1
1	1	1

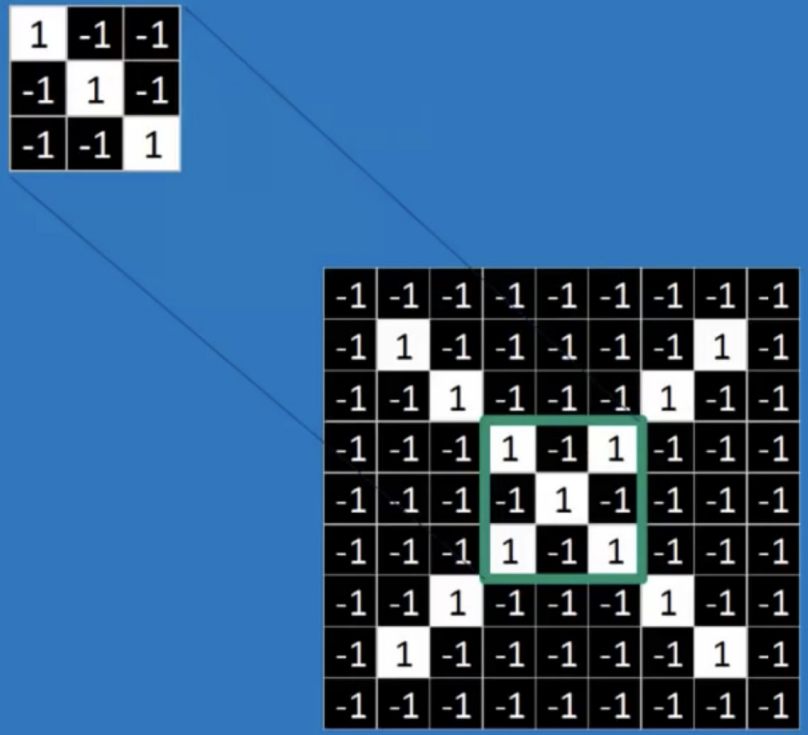
Brandon Rohrer

Result of local filtering

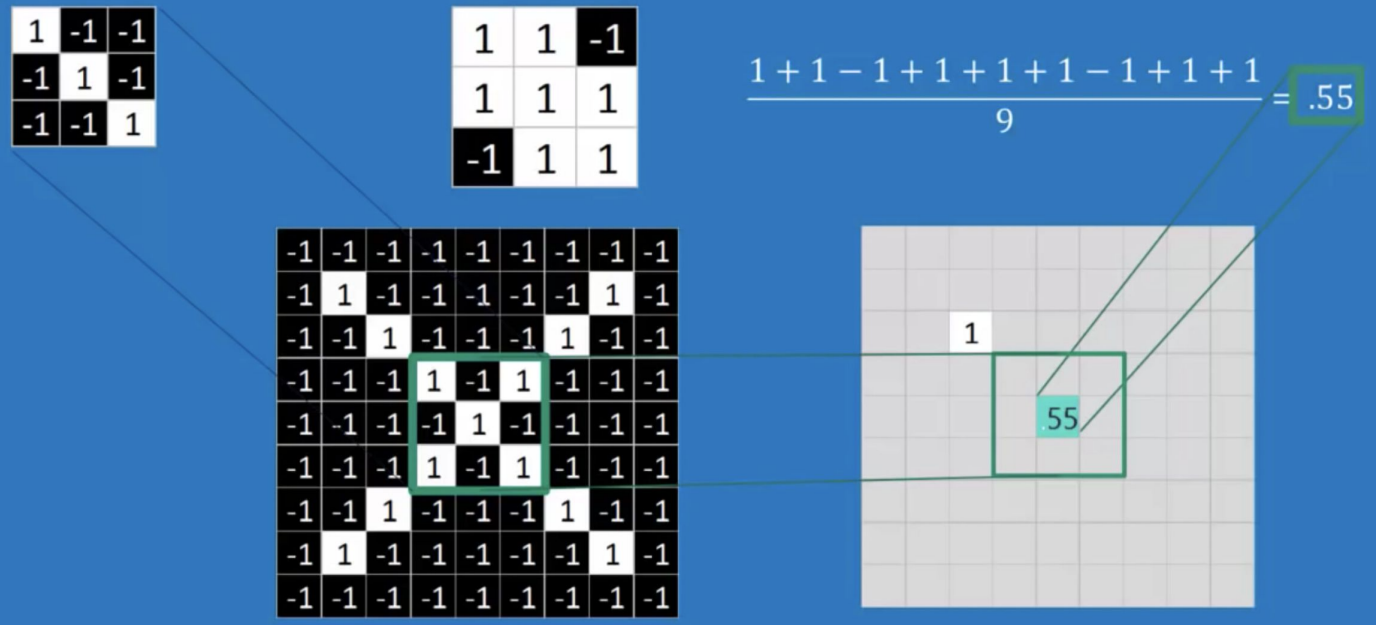


Brandon Rohrer

And at different locations

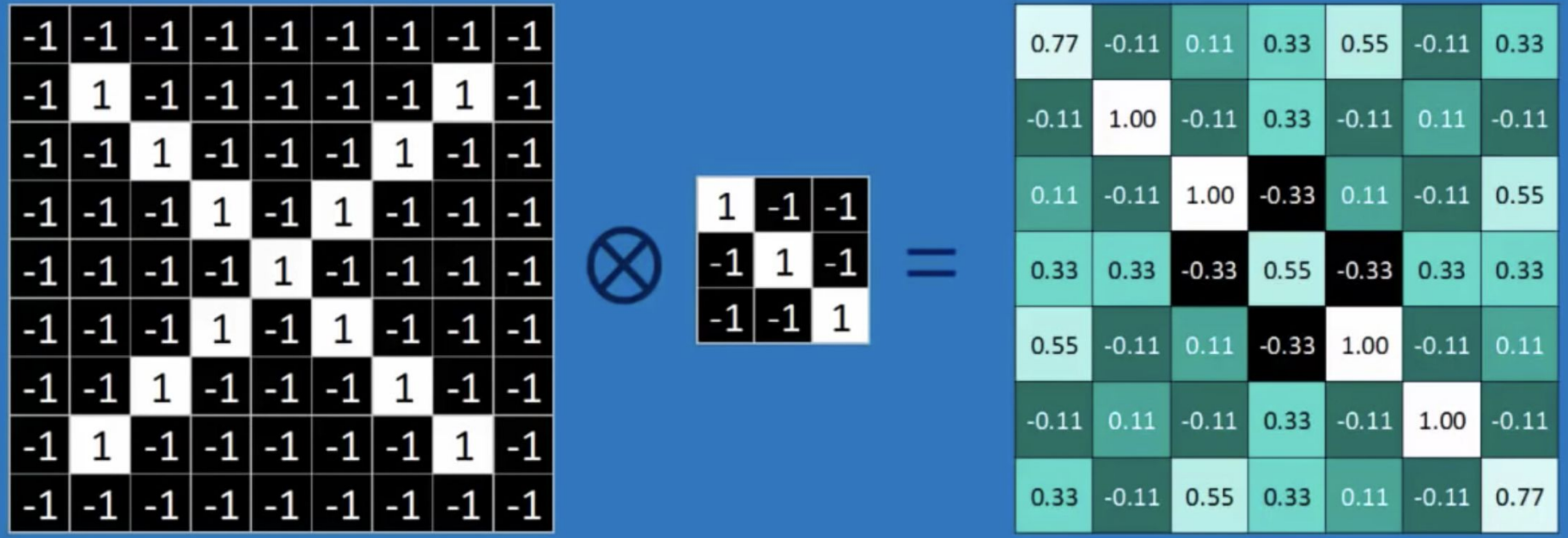


Distinct filtering results



Brandon Rohrer

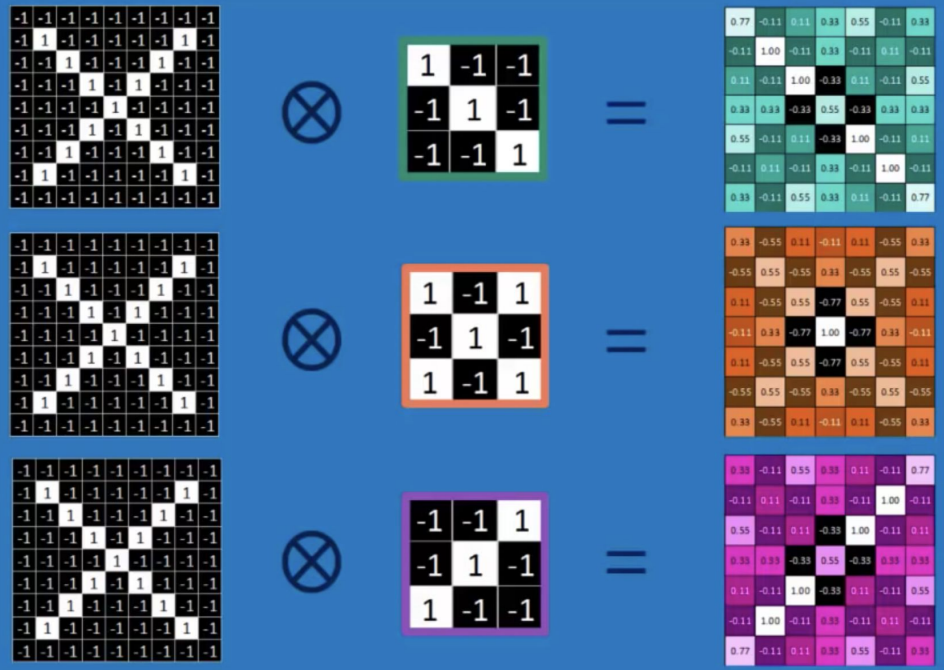
Convolution



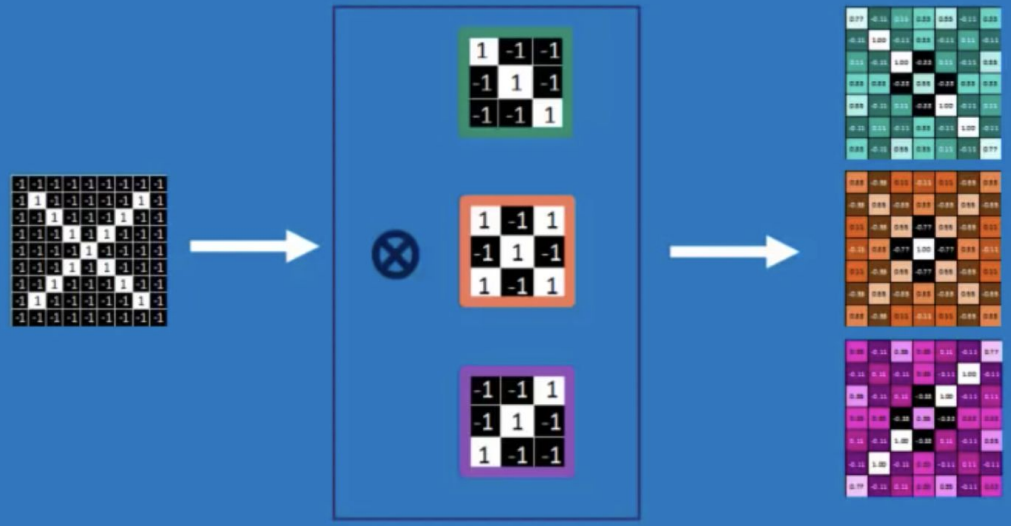
There is also stride, e.g. in layer 1 of resnet

Brandon Rohrer

Multiple filters



A convolution layer



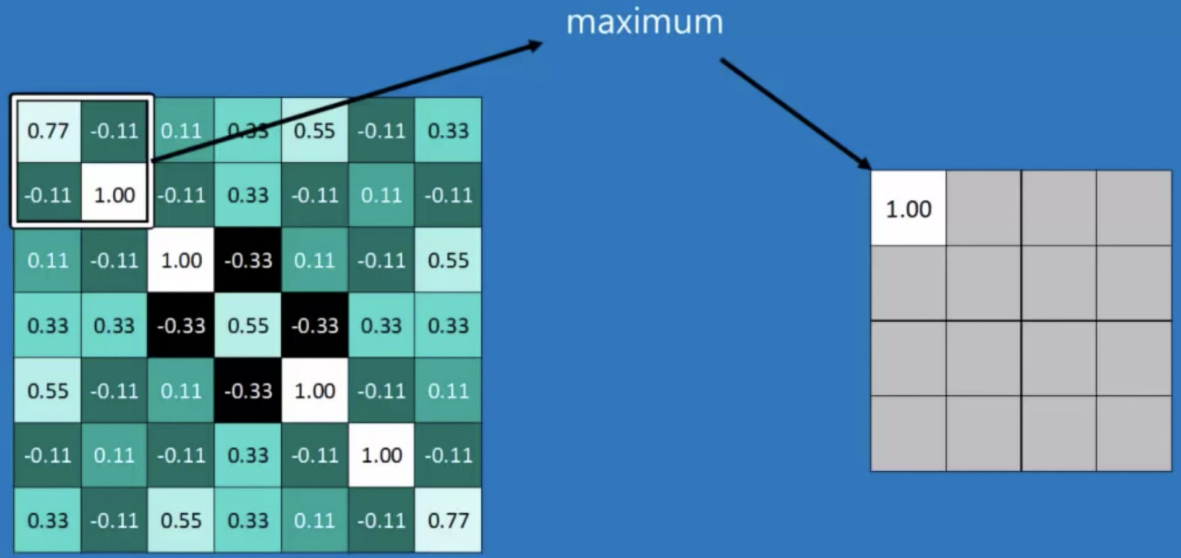
Brandon Rohrer

PollEV: how many parameters

Too many features!

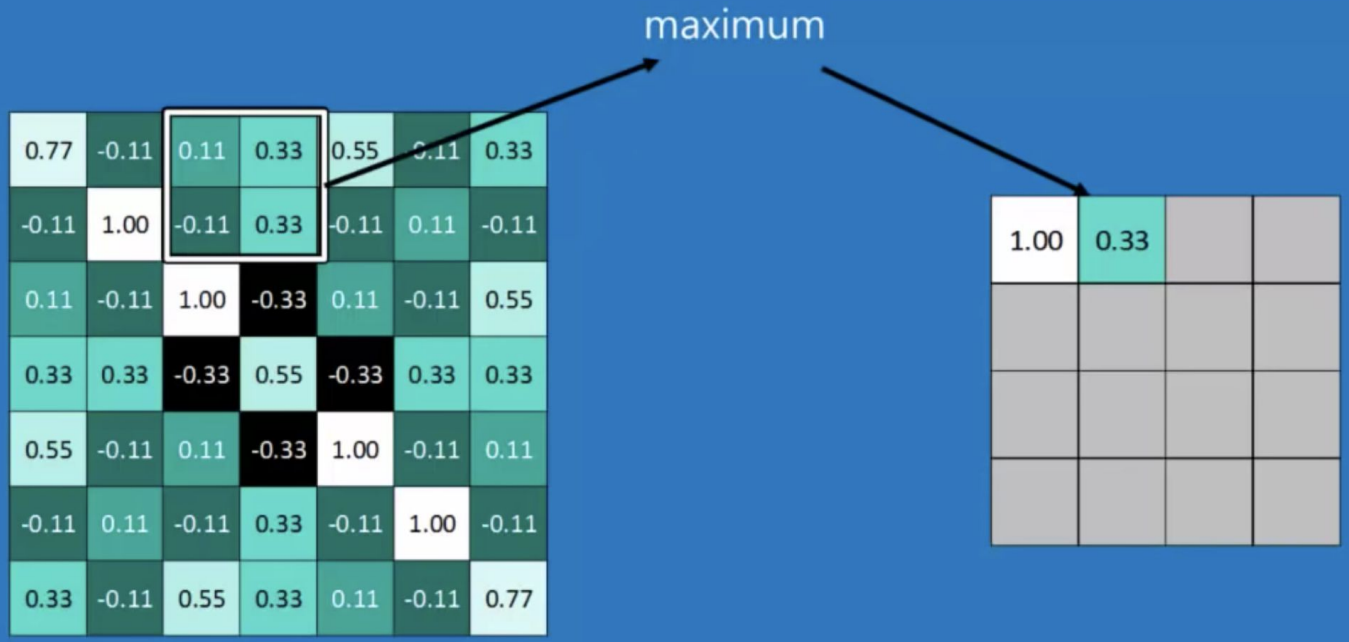
Gotta shrink that layer!
Intuition: we do not care exactly where a feature occurs

Max-pooling



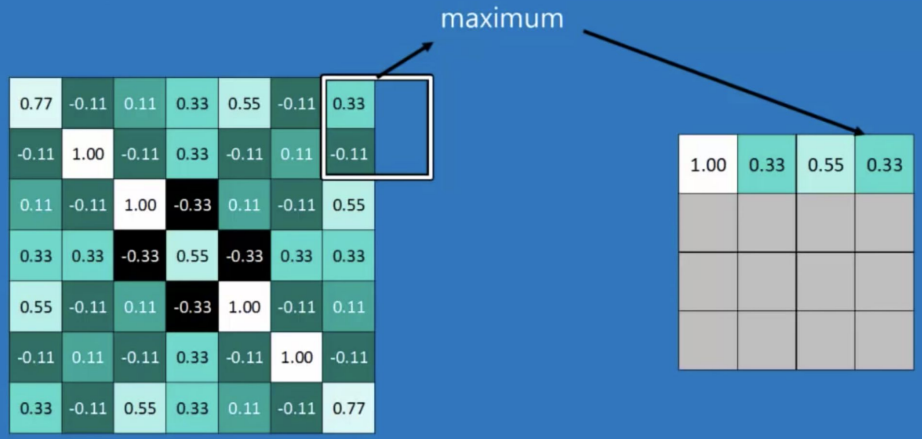
Brandon Rohrer

Across space



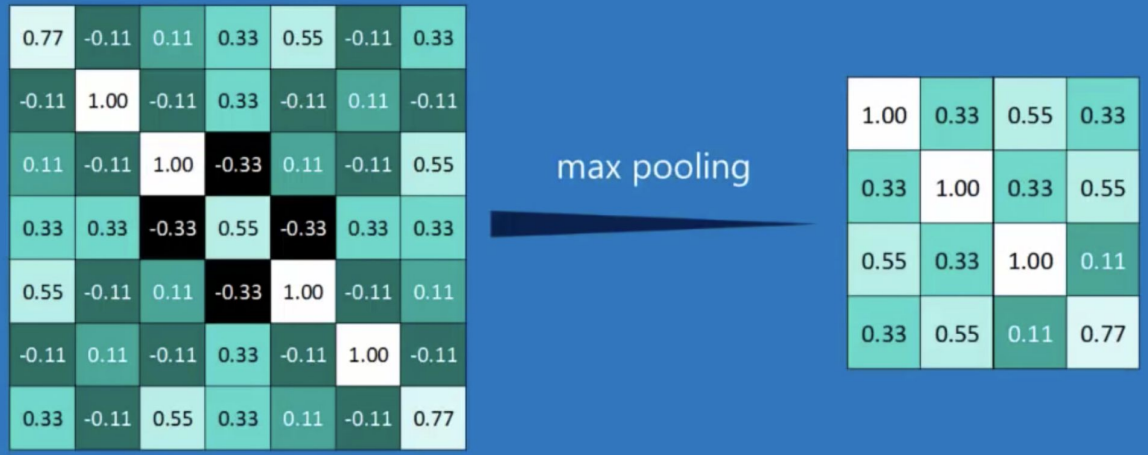
All following (blue) slides from Brandon Rohrer

Padding problem!



Brandon Rohrer

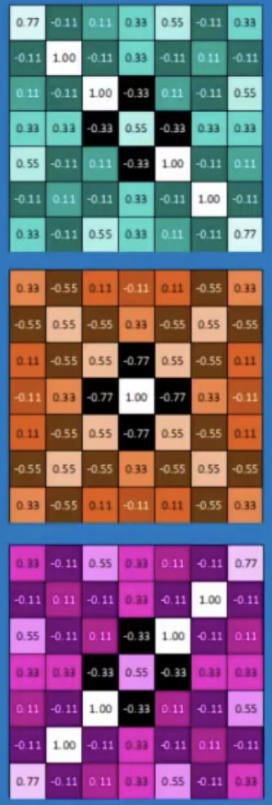
And a whole max pooling operation

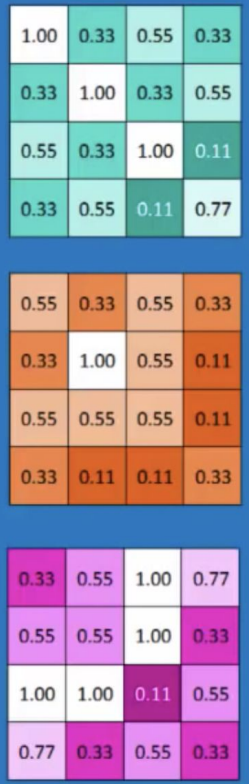


How many parameters?

Brandon Rohrer

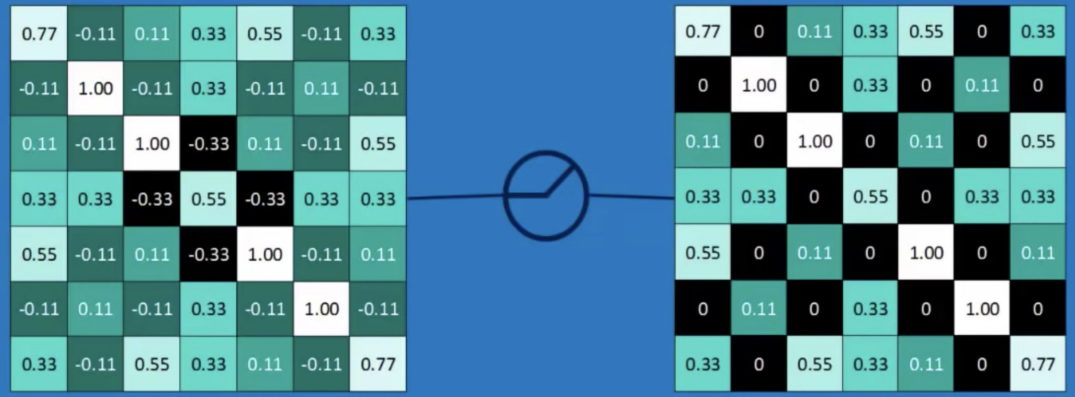
Bunch of Filters





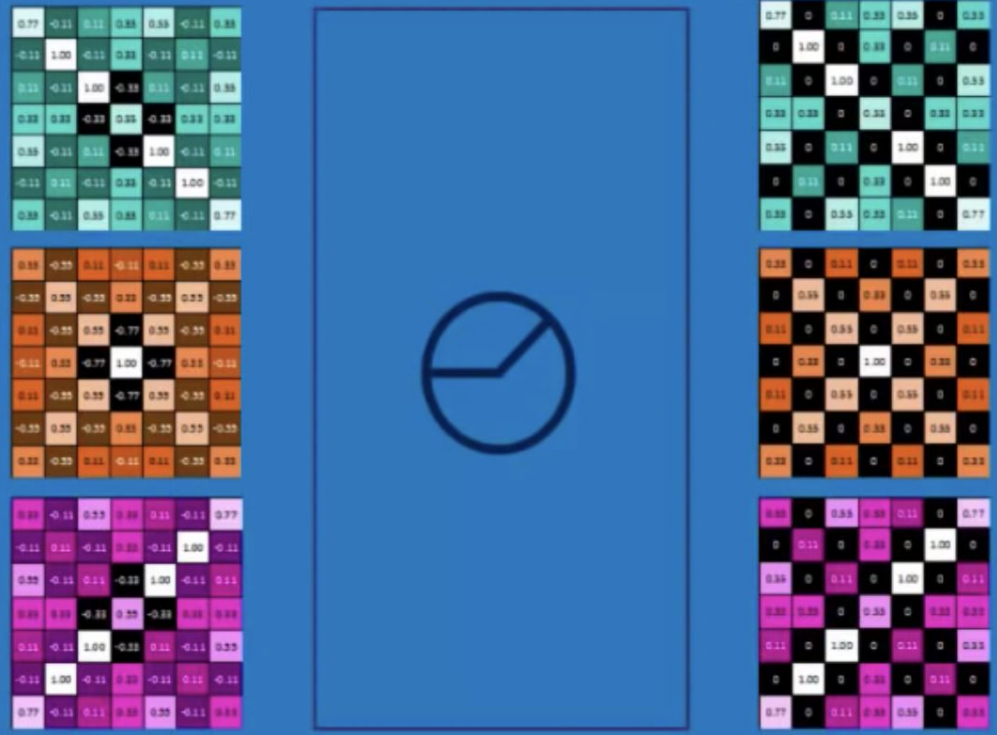
Brandon Rohrer

Potential to add a ReLU

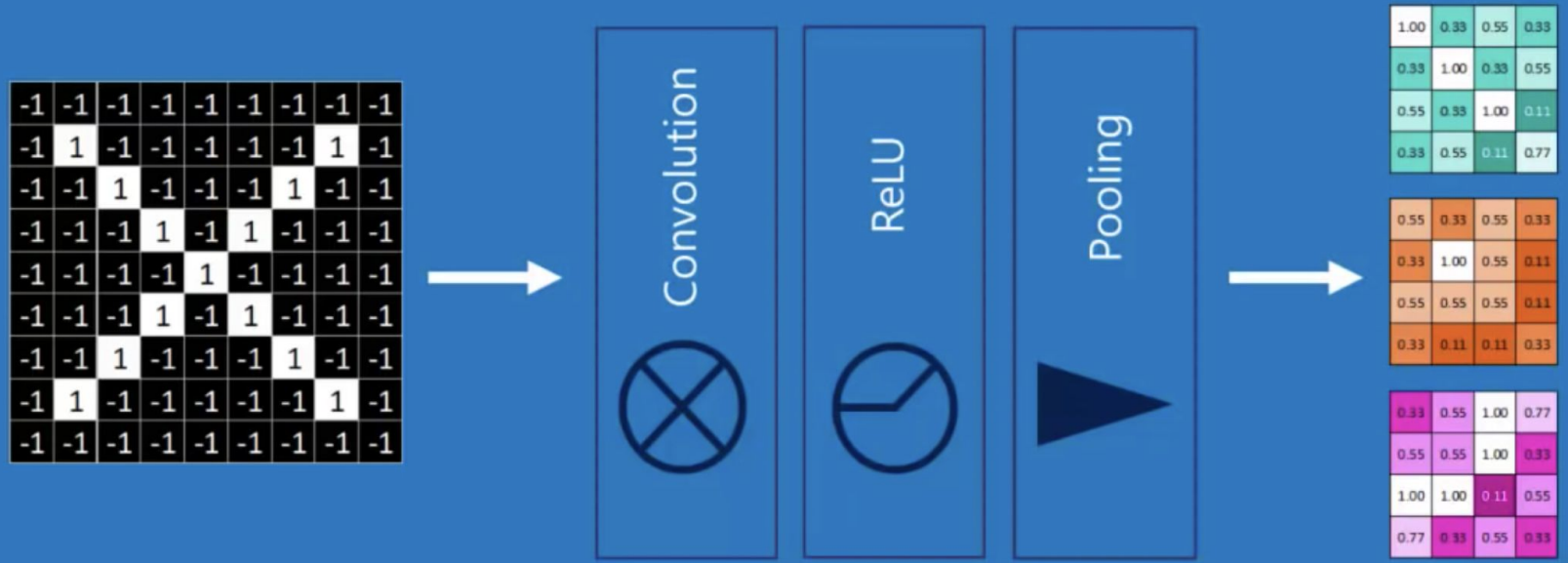


Brandon Rohrer

And multiple channels



And a hierarchy

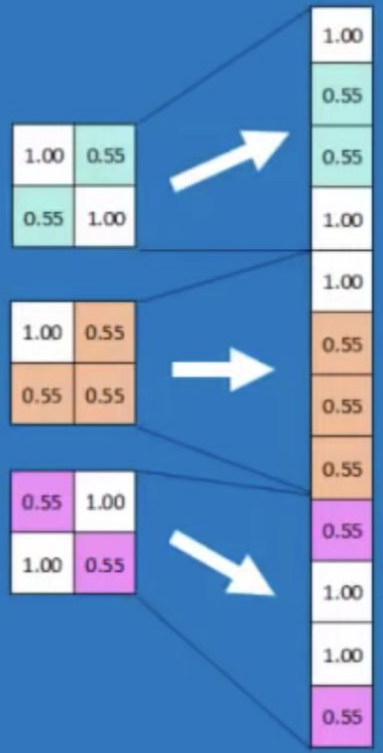


Brandon Rohrer

Chain it

Brandon Rohrer

How to transition to fully connected

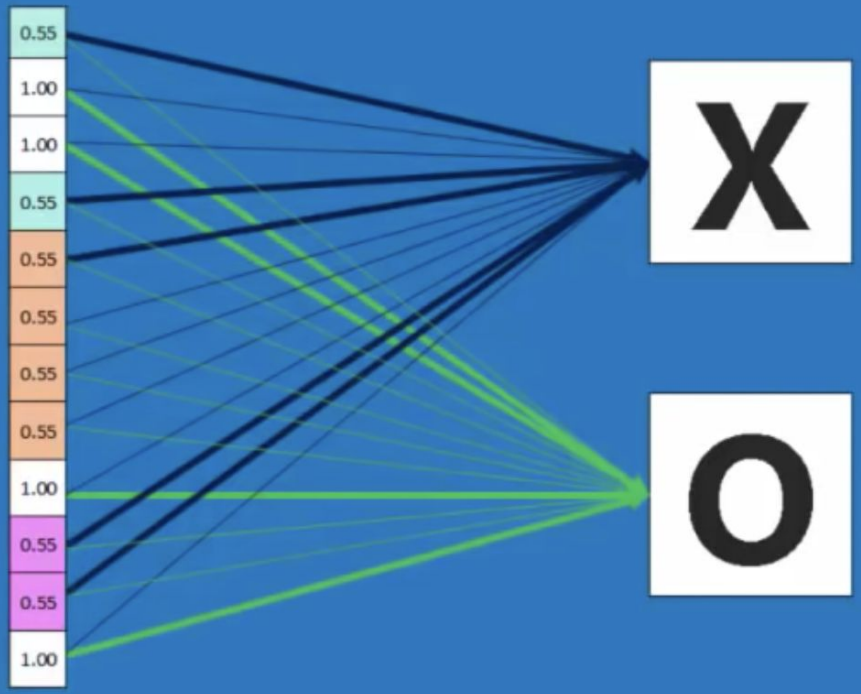


Brandon Rohrer

Fully connected read out

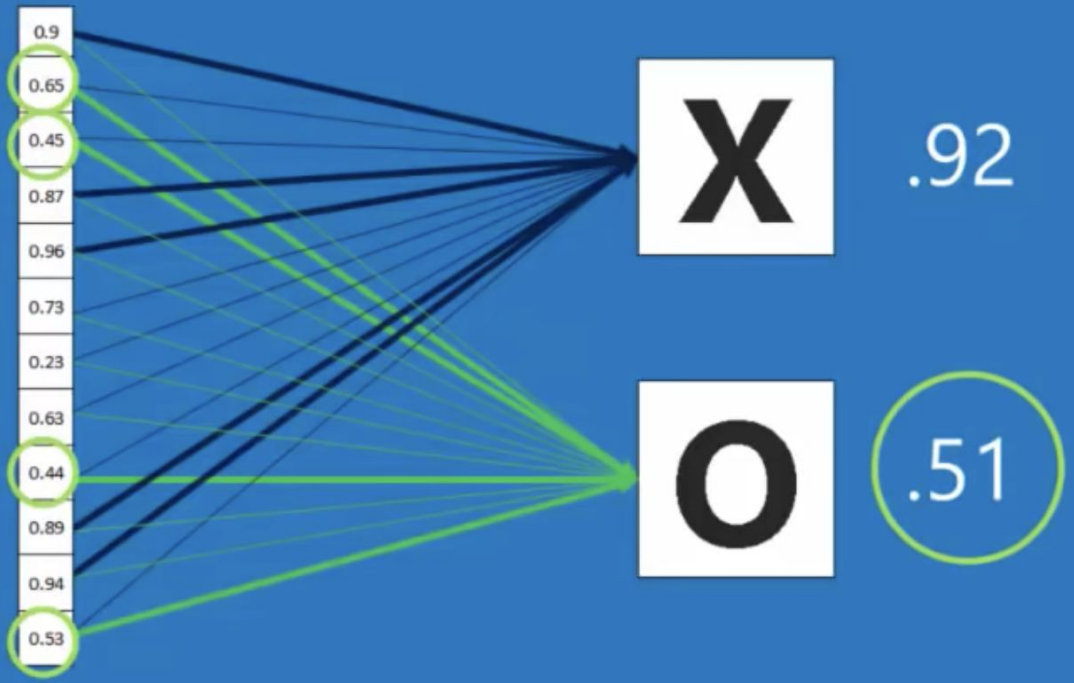
Brandon Rohrer

Predict both



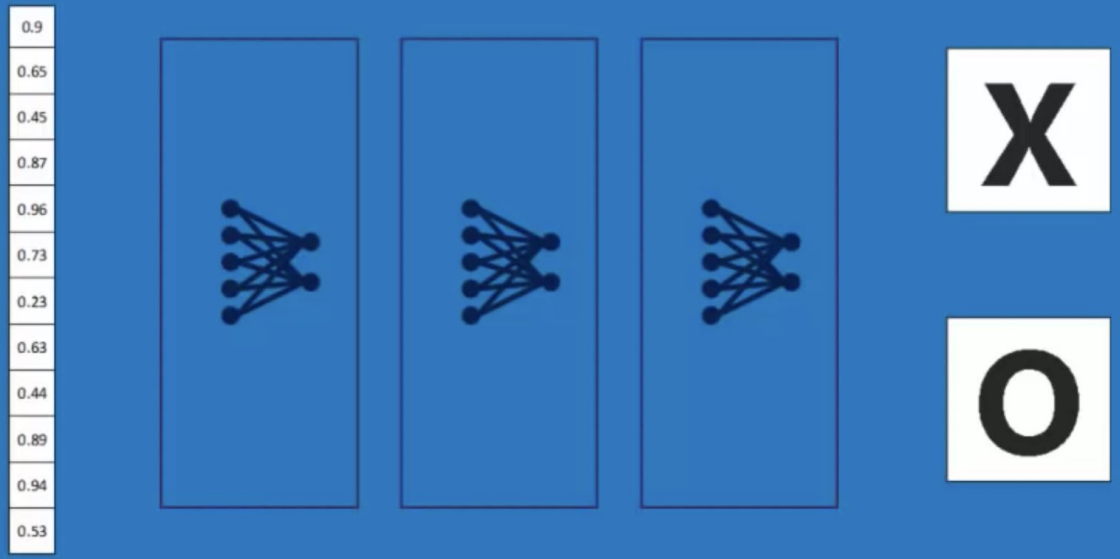
Brandon Rohrer

Get activations



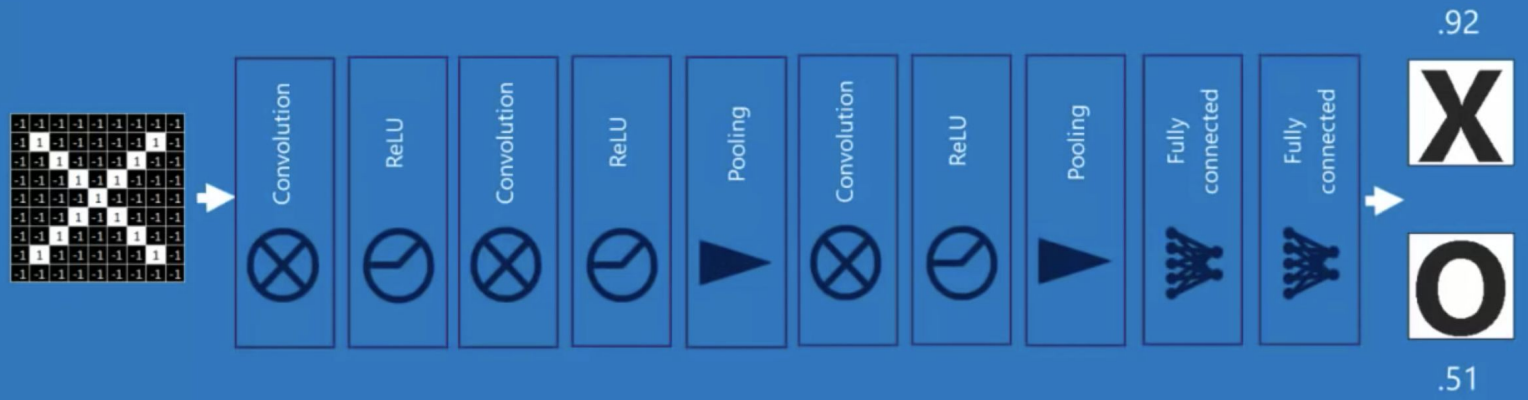
Brandon Rohrer

Have multiple layers of fully connected



Brandon Rohrer

Now stack it all together



Now what is weird here?

Brandon Rohrer

Try it ;)

https://www.cs.ryerson.ca/~aharley/vis/conv/flat.html

What is a convolution (in mathematics)?

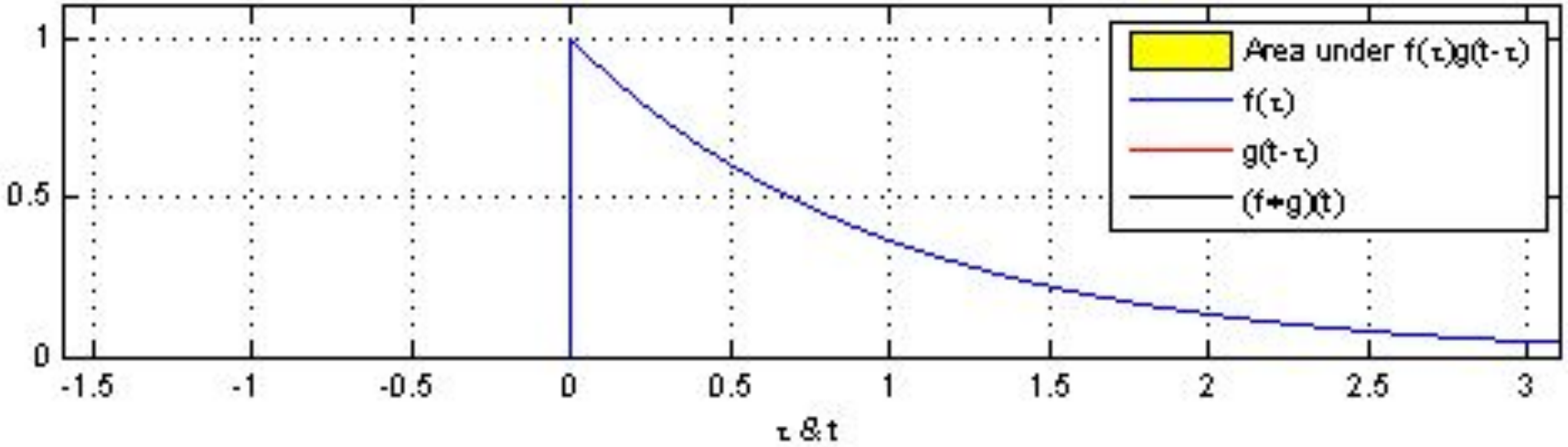
$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n - m]$$

Procedure to compute:

1. Express f, g in terms of dummy variable $\boldsymbol{\tau}$
2. Reflect g into $g(-\boldsymbol{\tau})$
3. Add phase t to get $g(t-\boldsymbol{\tau})$
4. For each phase t, compute the integral of product $f(\boldsymbol{\tau})g(t-\boldsymbol{\tau})$.

What is a convolution



A related concept: cross-correlation.

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} \overline{f(t)} g(t+\tau)\, dt$$

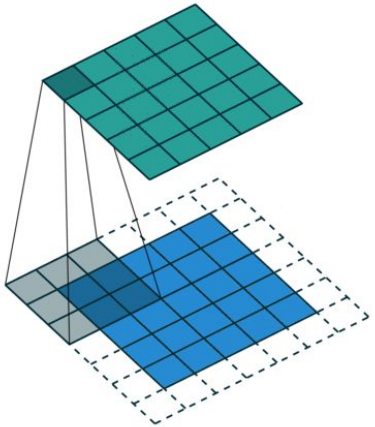
$$(f \star g)[n] \triangleq \sum_{m=-\infty}^{\infty} \overline{f[m]} g[m+n]$$

Procedure to compute:

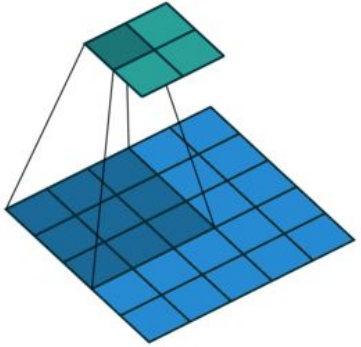
1. Express *f*, *g* in terms of dummy variable **t**
2. Take the conjugate of *f*.
3. Add phase τ to get $g(t+\tau)$
4. For each phase τ, compute the integral of product $f(t)g(t+\tau)$.

Convolution

- *Padding* - Adding fake pixels (usually 0) to the edge of the image. This may allow pixels at the edge of the image to be at the center of the sliding kernel.
- *Stride* - The number of pixels skipped when sliding the kernel over the image
- Depending on *padding* and *stride*, image may decrease slightly in size.



Padding = 1



Stride = 2

Motivation: conv for edge detection

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal lines

-1	2	-1
-1	2	-1
-1	2	-1

Vertical lines

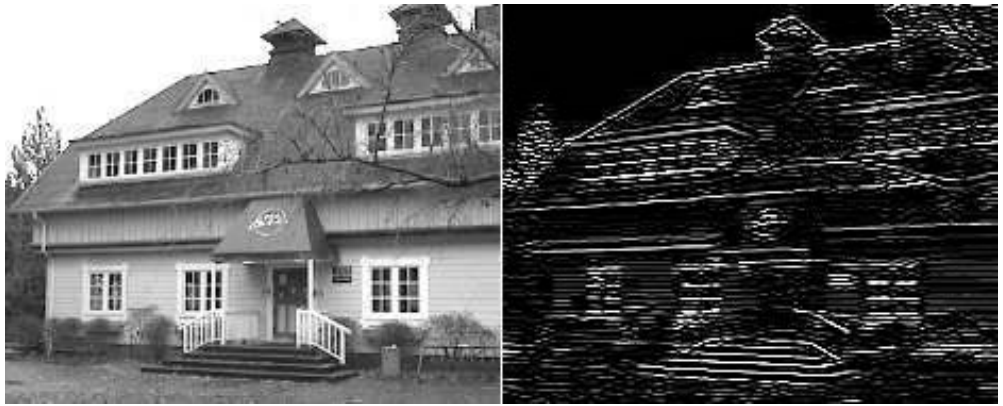
-1	-1	2
-1	2	-1
2	-1	-1

45 degree lines

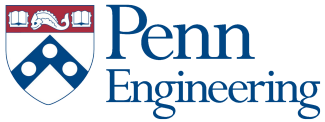
2	-1	-1
-1	2	-1
-1	-1	2

135 degree lines

Horizontal lines:



A convolution exercise



A convolution exercise (PollEV)

Suppose we want to find out if an image has horizontal or vertical lines.

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

We do so by convolving the image with two filters (no padding, stride of 1).

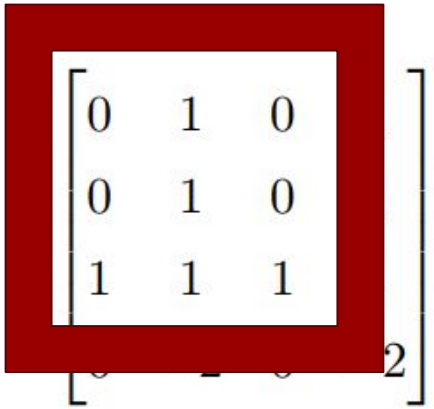
Compute the output by hand.

$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}, \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 1 & 1 & 1 \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

A convolution exercise

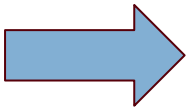
$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & -2 & 0 & -2 \end{bmatrix} \qquad \begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

A convolution exercise



$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

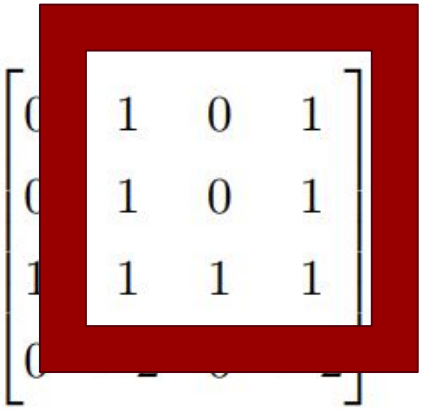
$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$



$$\begin{bmatrix} 2 & \cdot \\ \cdot & \cdot \end{bmatrix}$$

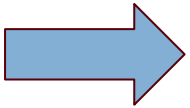
$$\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right)$$
$$\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right)$$
$$\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right) = 2$$

A convolution exercise



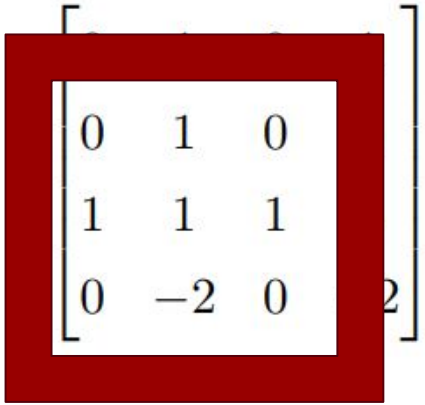
$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$



$$\begin{bmatrix} 2 & -2 \\ . & . \end{bmatrix}$$

A convolution exercise

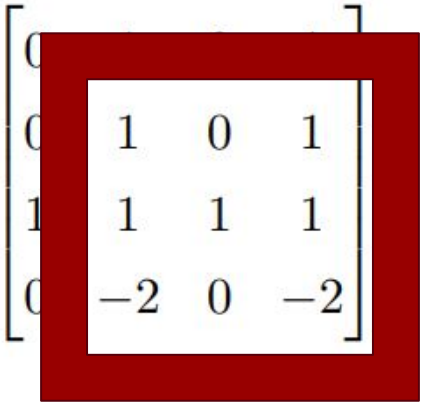


$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & -2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

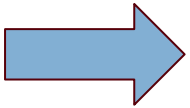
$$\begin{bmatrix} 2 & -2 \\ -1 & \cdot \end{bmatrix}$$

A convolution exercise



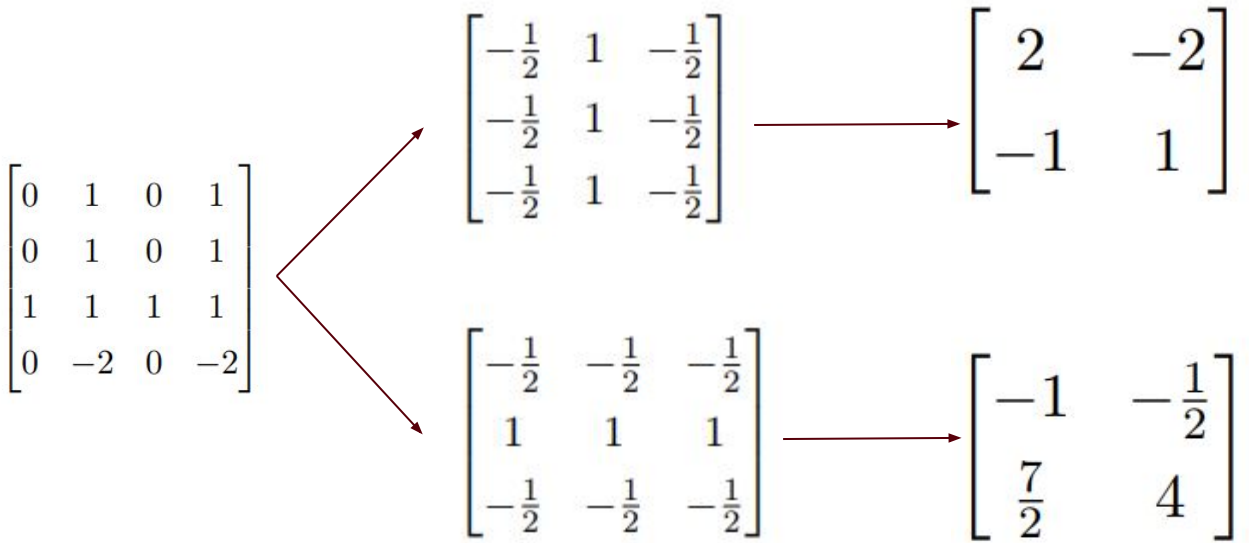
$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ -2 & 0 & -2 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$



$$\begin{bmatrix} 2 & -2 \\ -1 & 1 \end{bmatrix}$$

Convolution exercise solution



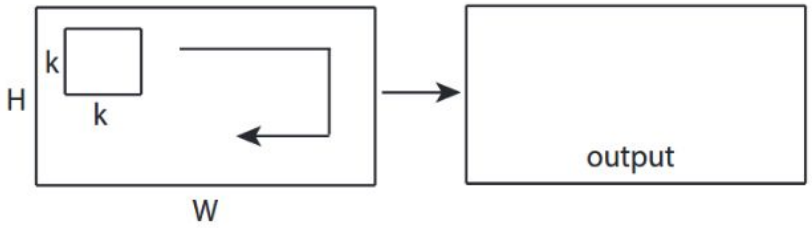
Grayscale image

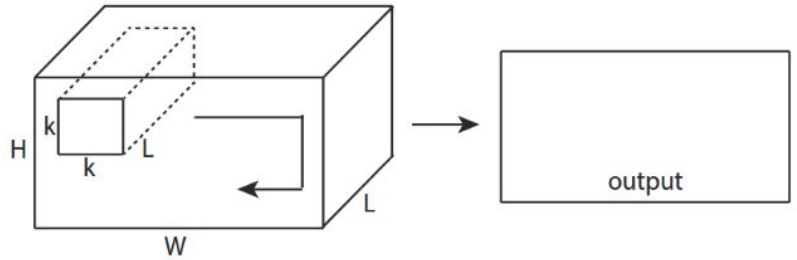
Filters

Output of filters

2D Convolution

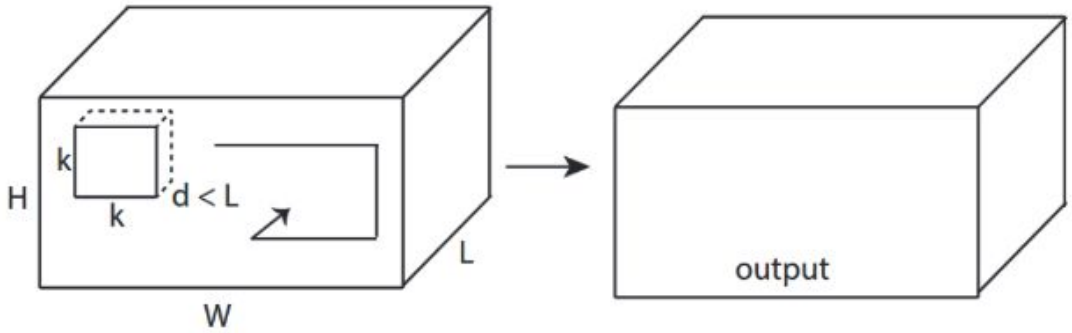
- 2D Matrix -> 2D Matrix
- The kernel moves in 2D
- If the input is 3D, the kernel should also have three dimensions where one dimension of the kernel should be equal to the respective dimension of the input (typically z dimension)





3D Convolution

- Output of this convolution is a 3D Matrix
- Needs 3D Inputs
- Kernel is three dimensional



Calculate output dimension (PollEV)

What will be the output dimension of a single 2D convolution operation on:

input of size 300 x 400

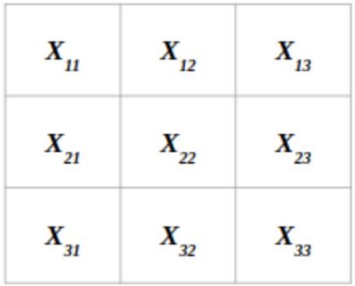
a kernel of size (4, 5)

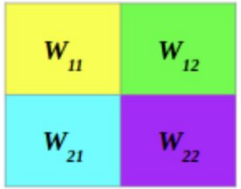
stride = 1

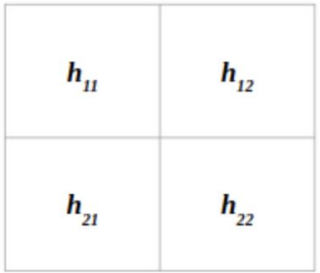
padding = 2

Backprop through convolution layer

Suppose the forward pass is expressed as,







$$h_{11} = W_{11}X_{11} + W_{12}X_{12} + W_{21}X_{21} + W_{22}X_{22}$$

$$h_{12} = W_{11}X_{12} + W_{12}X_{13} + W_{21}X_{22} + W_{22}X_{23}$$

$$h_{21} = W_{11}X_{21} + W_{12}X_{22} + W_{21}X_{31} + W_{22}X_{32}$$

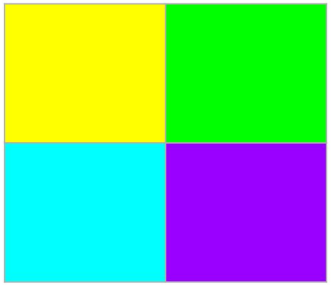
$$h_{22} = W_{11}X_{22} + W_{12}X_{23} + W_{21}X_{32} + W_{22}X_{33}$$

Our goal is to compute the local gradients $\partial h/\partial W$ and $\partial h/\partial X$

Example: Backprop through convolution layer

∂h/∂W

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}



∂h_{11}	∂h_{12}
∂h_{21}	∂h_{22}

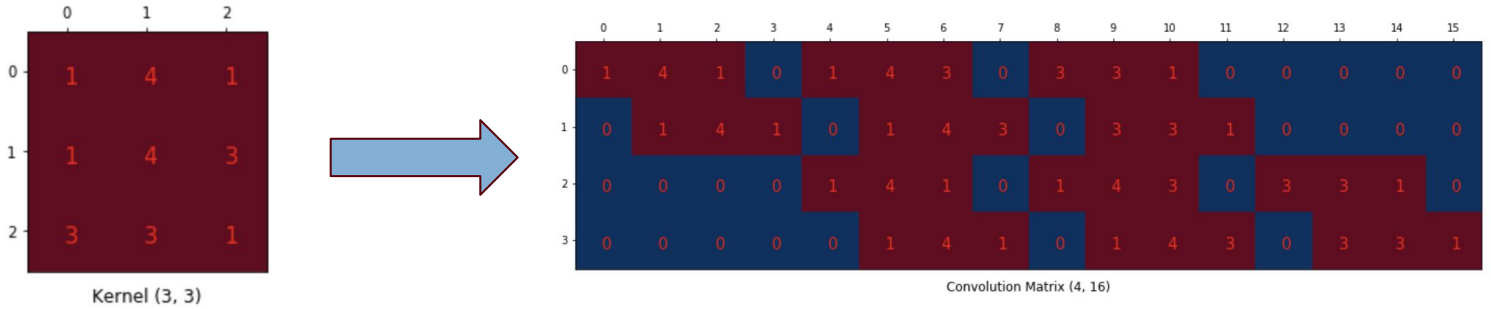
∂h/∂X

$$
\begin{aligned}
\delta X_{11} &= W_{11}\delta h_{11} \\
\delta X_{12} &= W_{12}\delta h_{11} + W_{11}\delta h_{12} \\
\delta X_{13} &= W_{12}\delta h_{12} \\
\delta X_{21} &= W_{21}\delta h_{11} + W_{11}\delta h_{21} \\
\delta X_{22} &= W_{22}\delta h_{11} + W_{21}\delta h_{12} + W_{12}\delta h_{21} + W_{11}\delta h_{22} \\
\delta X_{23} &= W_{22}\delta h_{12} + W_{12}\delta h_{22} \\
\delta X_{31} &= W_{21}\delta h_{21} \\
\delta X_{32} &= W_{22}\delta h_{21} + W_{21}\delta h_{22} \\
\delta X_{33} &= W_{22}\delta h_{22}
\end{aligned}
$$

Express convolution as matrix multiplication

Linear operation: Convolution is simply a dot product between the kernel and ‘local regions’ of the input

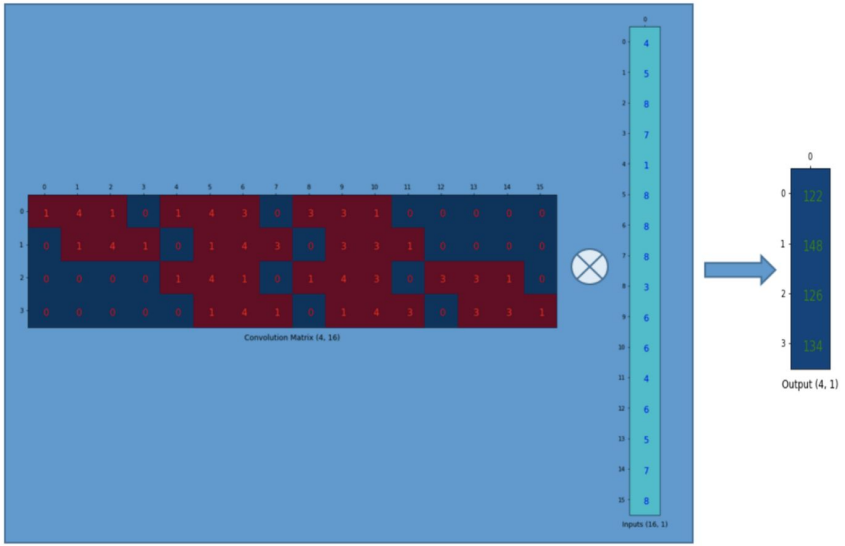
We can take advantage of this and stretch the kernel as,



Stride = 1, Padding = 0

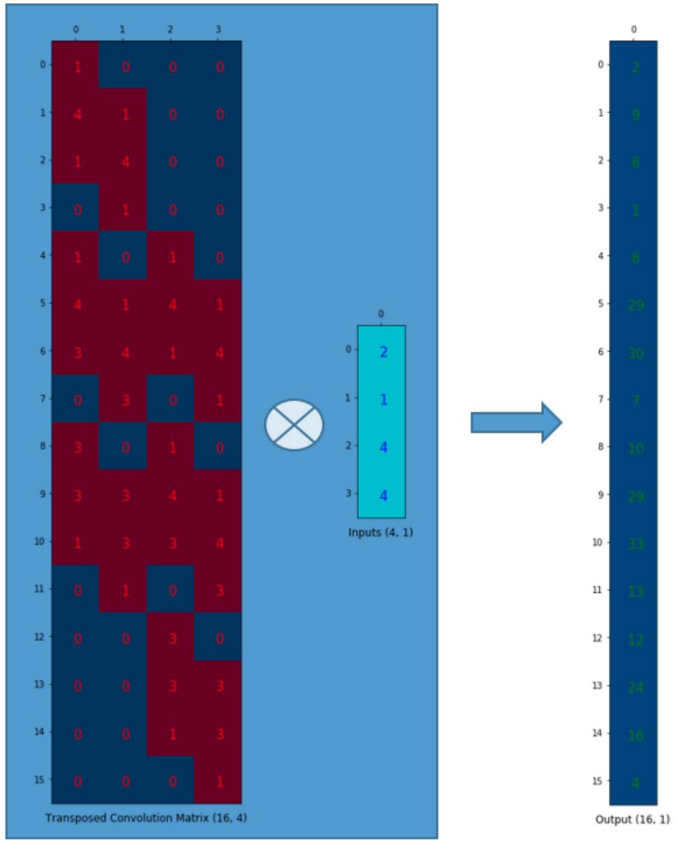
Express convolution as matrix multiplication

- Flatten the input matrix into a column matrix.
- A simple matrix multiplication between the stretched kernel matrix and the flattened input matrix will give a flattened output



Deconvolution or Transposed Convolution

- What if we want to upscale (2 x 2 to 4 x 4) across the spatial dimension?
  - Instead of stretching the kernel as a 4 x 16 matrix as in the previous slide, stretch it to a 16 x 4 matrix (or simply take its transpose)
  - A matrix multiplication between the stretched kernel (transposed) and the flattened input will result in an upscaled output



Question

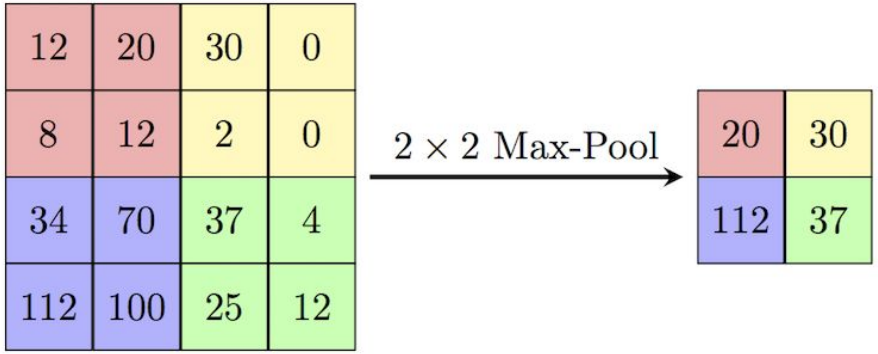
- What is convolution like with a 1x1 kernel?

Pooling layers

- Want to:
  - Reduce dimensions of data, without additional parameters
  - Introduce some translation invariance (compare w complex cells in V1)
- Pool over a window of image:
  - Maximum
  - Average

Max pooling

- MaxPool confers invariance
- Translation/rotation etc
- Over the course of a network, pooling shifts layers from spatial dimensions to features



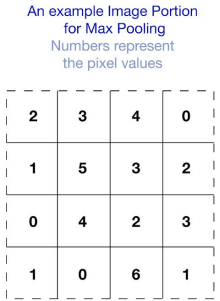
E.g.:

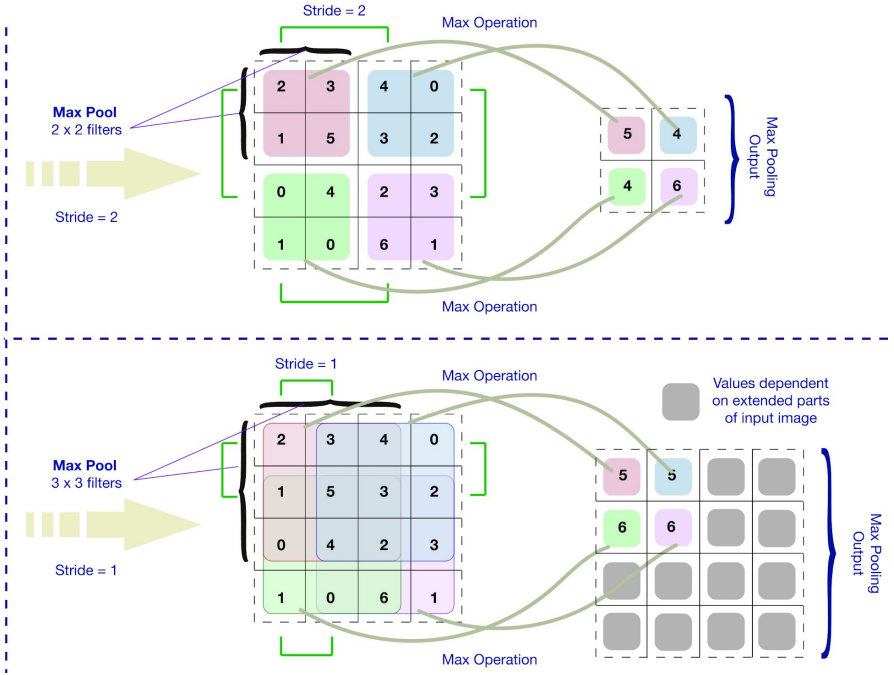
Conv **Pool** **Conv** **Pool**

28 x 28 x 1 => 26 x 26 x 16 => 13 x 13 x 16 => 11 x 11 x 64 => 6 x 6 x 64 => ...

Max pooling

- Stride doesn't need to match pool size





Backprop through a max pooling layer

Locally, max is linear with respect to the maximum value, and zero otherwise:

$$y = \max(x_1, \ldots, x_n)$$

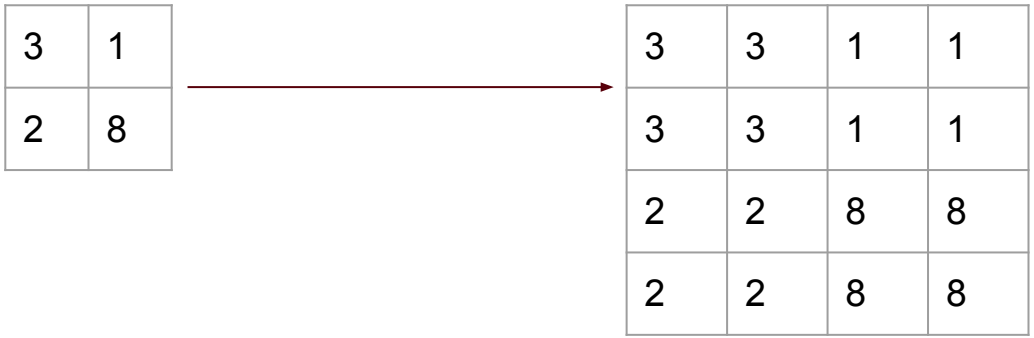
$$\frac{\partial y}{\partial x_i} = \begin{cases} 1, & x_i = \max(\mathbf{x}) \\ 0, & \text{otherwise} \end{cases}$$

In addition to storing the max value, autograd stores the index corresponding to that value

Unpooling

Upsampling: restore original image dimensions after relevant features extracted by CNN layers (e.g. image segmentation)

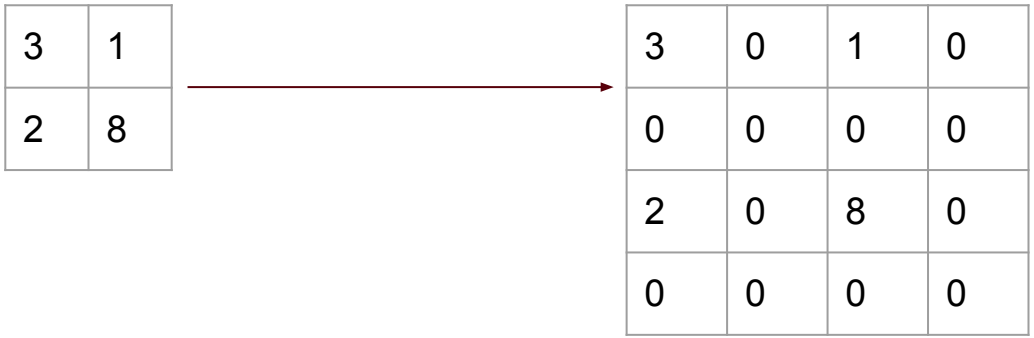
E.g. nearest neighbor:



Unpooling

Upsampling: restore original image dimensions after relevant features extracted by CNN layers (e.g. image segmentation)

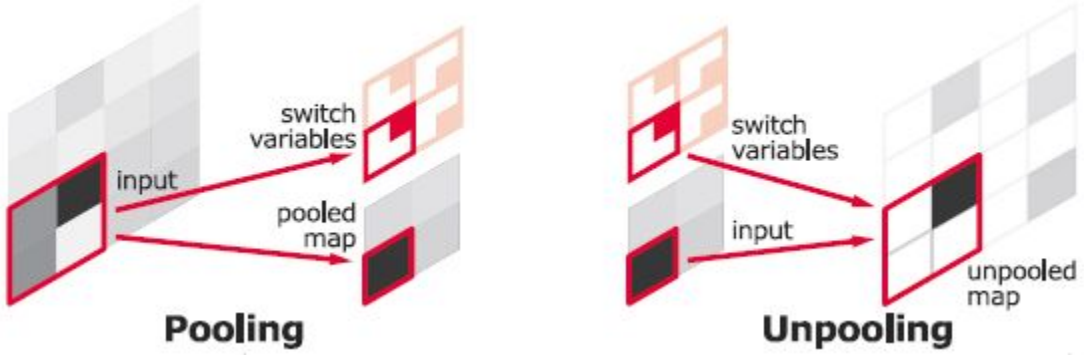
E.g. bed of nails:



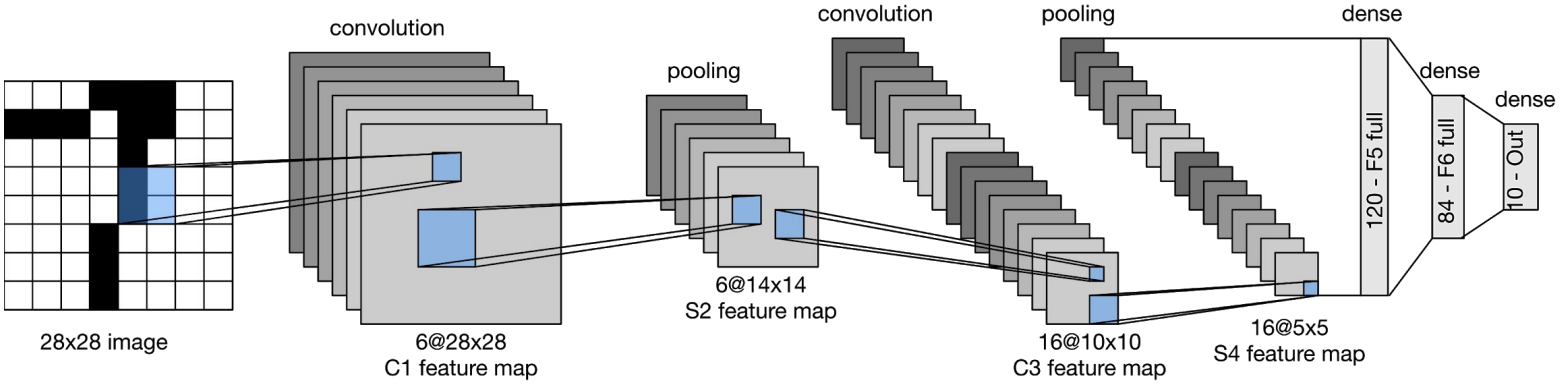
Unpooling

Upsampling: restore original image dimensions after relevant features extracted by CNN layers (e.g. image segmentation)

E.g. max unpooling. Now have to store which element was max in pooling layer!



Putting things together



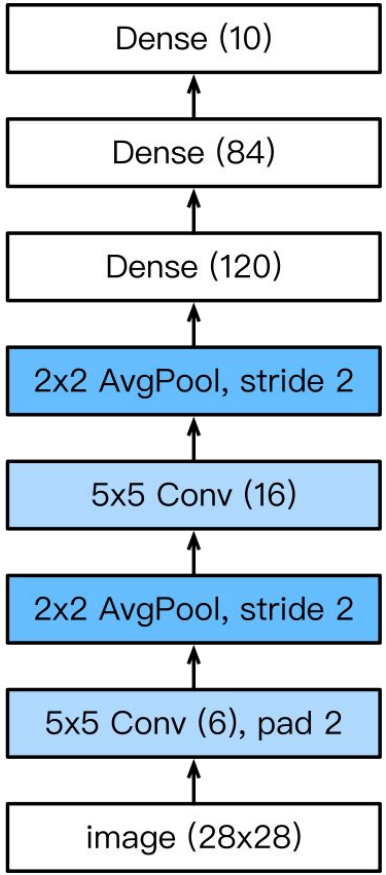
'Dive into deep learning' 6.6 Zhang et al

Input→(Conv→ReLU→Pooling)→...→Fully connected

Putting things together

Compact representation

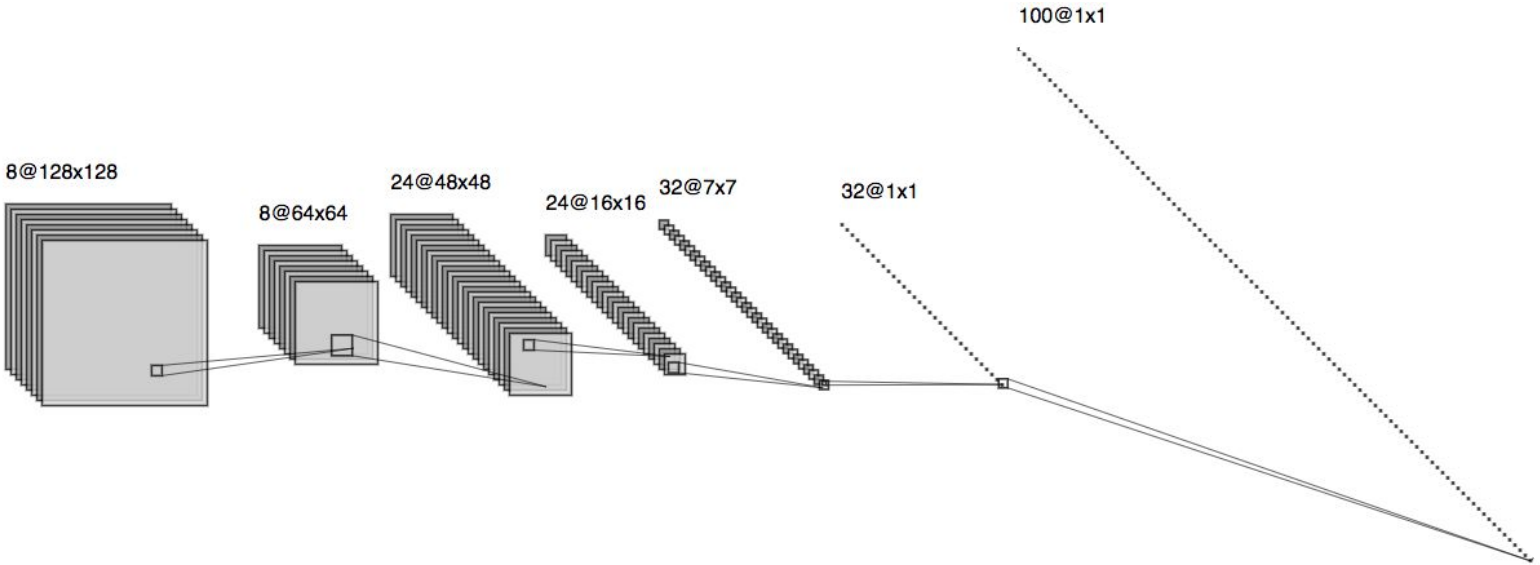
‘Dive into deep learning’ 6.6 Zhang et al



Replace FC layer by a convolution layer

- Suppose a CNN produces an output of size 7 x 7 x 32 after a series of convolution and pool layers.
- How do we use this output for a classification task with say 100 output classes?
  - Flatten the output and add fully connected layer(s)
  - Use a kernel of size 7 which produces an output of size 1 x 1 x 32 followed by another convolution layer with filter of size 1 with 100 such filters. This produces an output of shape 1 x 1 x 100.

Replace FC layer by a convolution layer



Why use a CNN?

- Shared weights => reduced number of spatial parameters - focus on features
- Computing a convolution is quick - Winograd FFT:
  - Fourier transform of convolution is the product of the Fourier transforms
  - Winograd FFT allows fast Fourier transforms
- Can often be applied to any size of image with same parameters

Also see https://www.youtube.com/watch?v=j4CIomZfsdM

Inductive biases associated with convnets

- Translation invariance
- Rotational invariance (much less common)
    - e.g. Cheng et al. (2016), literally just rotate and enforce similar representations
- Less expressive models often overfit less

Today we learned

- Neuroscience provided the motivation for convolutional neural networks (CNNs)
- Definitions and intuitions from mathematics
- Parts of a CNN
  - Convolutional layers
  - Pooling layers
  - How to gradient descent
- Next Week: many application examples
- Later: CNNs well beyond images

End of lecture

PollEV.

You know the drill ;)