CNNs, RNNs, and Parameter Sharing

Alona Fyshe (h/t Konrad Kording)

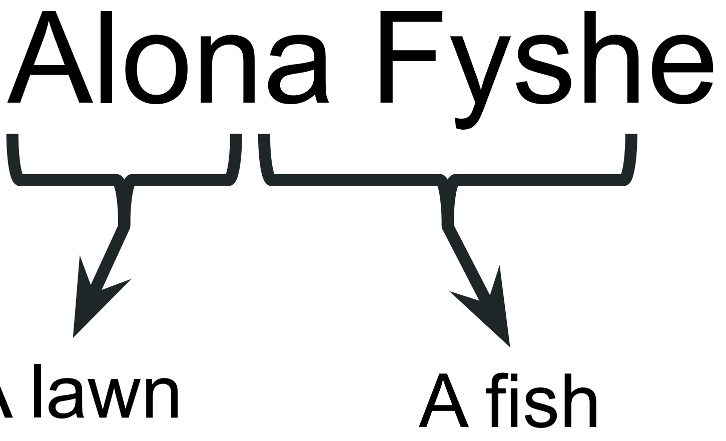
neuromatch academy

Section 0: Introduction

neuromatch
academy

Alberta Machine Intelligence Institute

UNIVERSITY OF ALBERTA



Last week: Regularization

Last week you learned about regularization

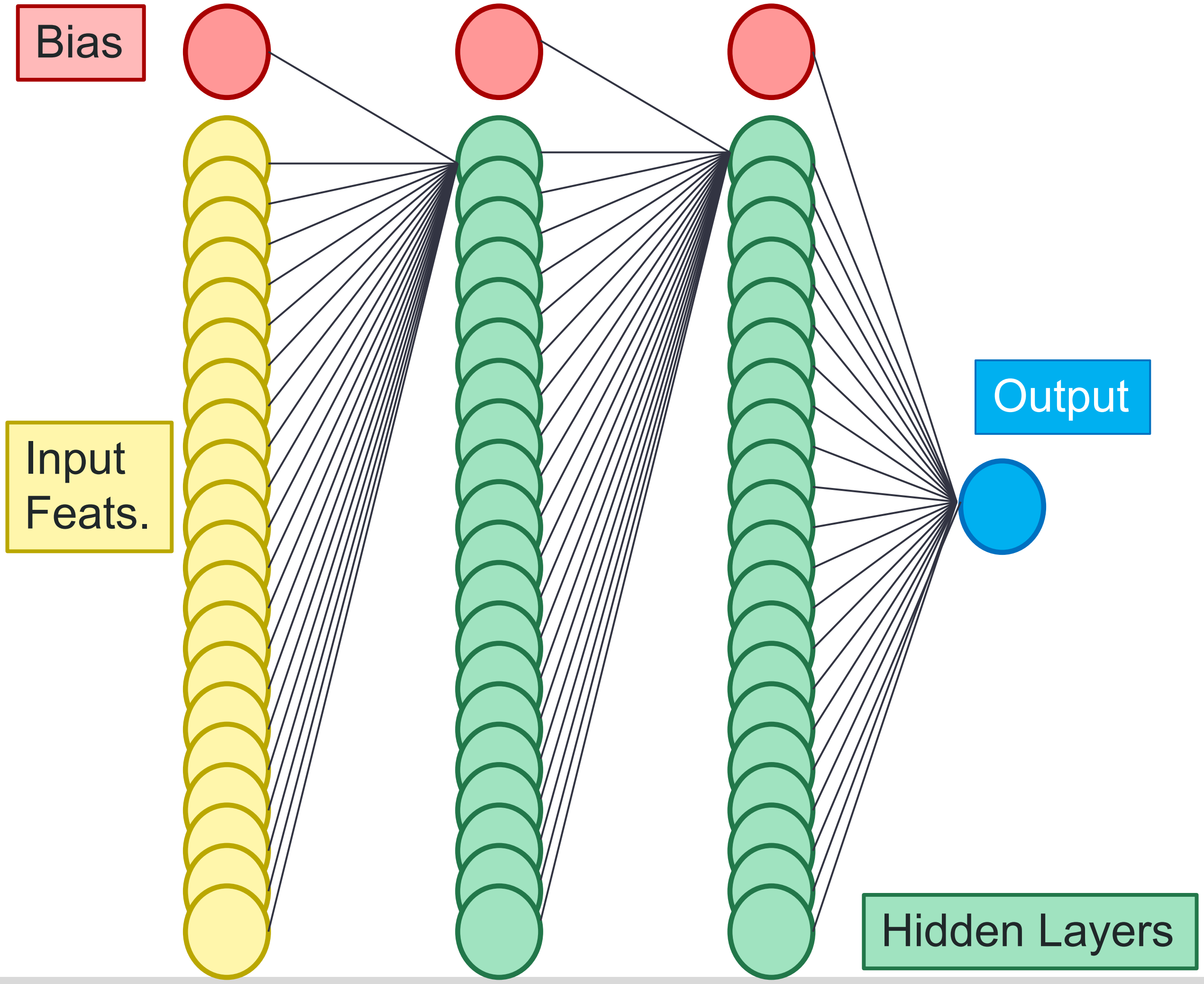
- Improves generalization
- Encourages simpler models: L1, L2, early stopping
- Data augmentation
- Dropout

Recall: regularization reduces the **effective** number of parameters in a model

Today: Parameter Sharing: CNNs and RNNs

- Convolutional Neural Networks (CNNs) operate on images
  - Parameter share over space
  - CIFAR 10 images are 32*32 = 1024 pixels, 1024 input features
  - ImageNet often pre-processed to be 256*256 = ~64K pixels, ~64k input features 😬
- Recurrent Neural networks (RNNs) operate on sequences of data
  - Often data sequences that occur over time
  - Time series can be long, but more importantly, they can vary in length

An MLP can have many many params



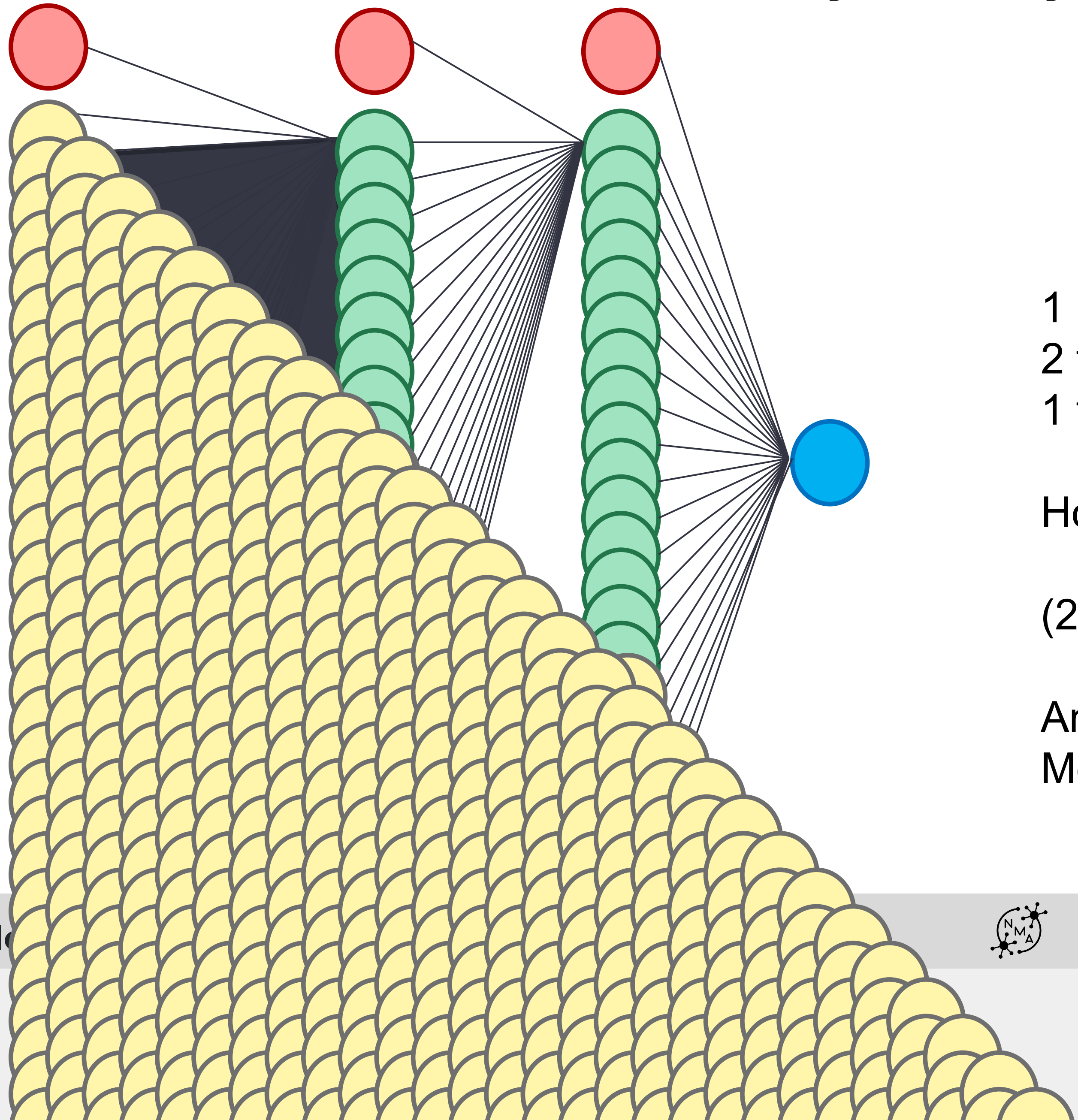
Data: 20 input features, single binary label

- 1 input layer with 20 nodes,
- 2 fully connected layers, 20 nodes each
- 1 final prediction node

How many weights is that?

(20+1)*20+(20+1)*20+(20+1)*1=861 params

An MLP can have many many params



1 input layer with 256*256 nodes,
2 fully connected layers, 20 nodes each
1 final prediction node

How many weights is that?

(256*256+1)*20+(20+1)*20+(20+1)*1 = ~1.3M params

And that's with only two fairly small hidden layers!
Modern CNNs have many more layers! 50-100 or more!

Translation invariance in images

- At the same time that the number of parameters grows very quickly, we are missing an opportunity

JOSH HAROLDSON VIA FLICKR // CC BY-NC 2.0

Outline

- Neuroscience motivation for CNNs
- Definitions and intuitions
- Parts of a CNN
    - Convolutional layers
    - Pooling layers
    - Putting it together

Regularization

E 1.1. Regularization reduces # effective params

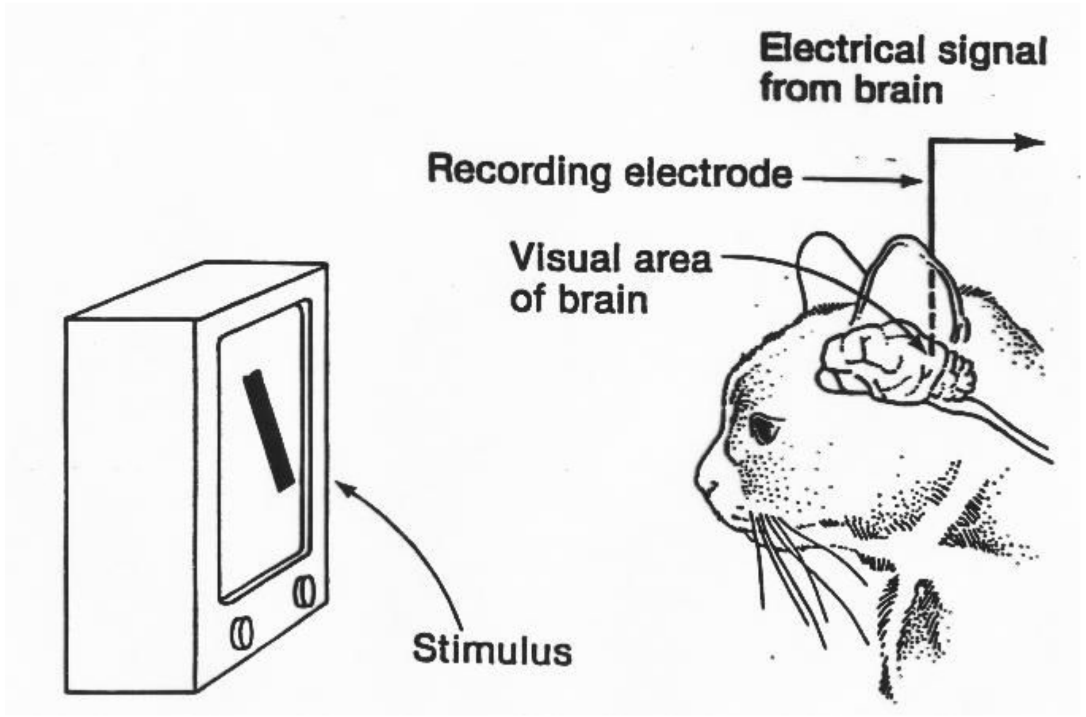
Section 1: Representations & Visual processing in the brain

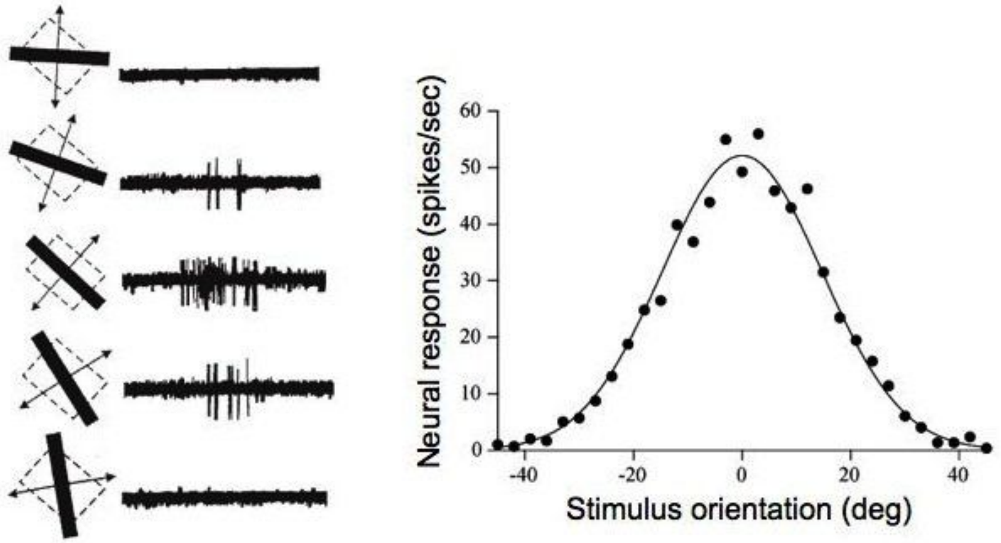
neuromatch academy

Visual Processing in the Brain

- The brain is an efficient machine
- How does it “solve” vision?

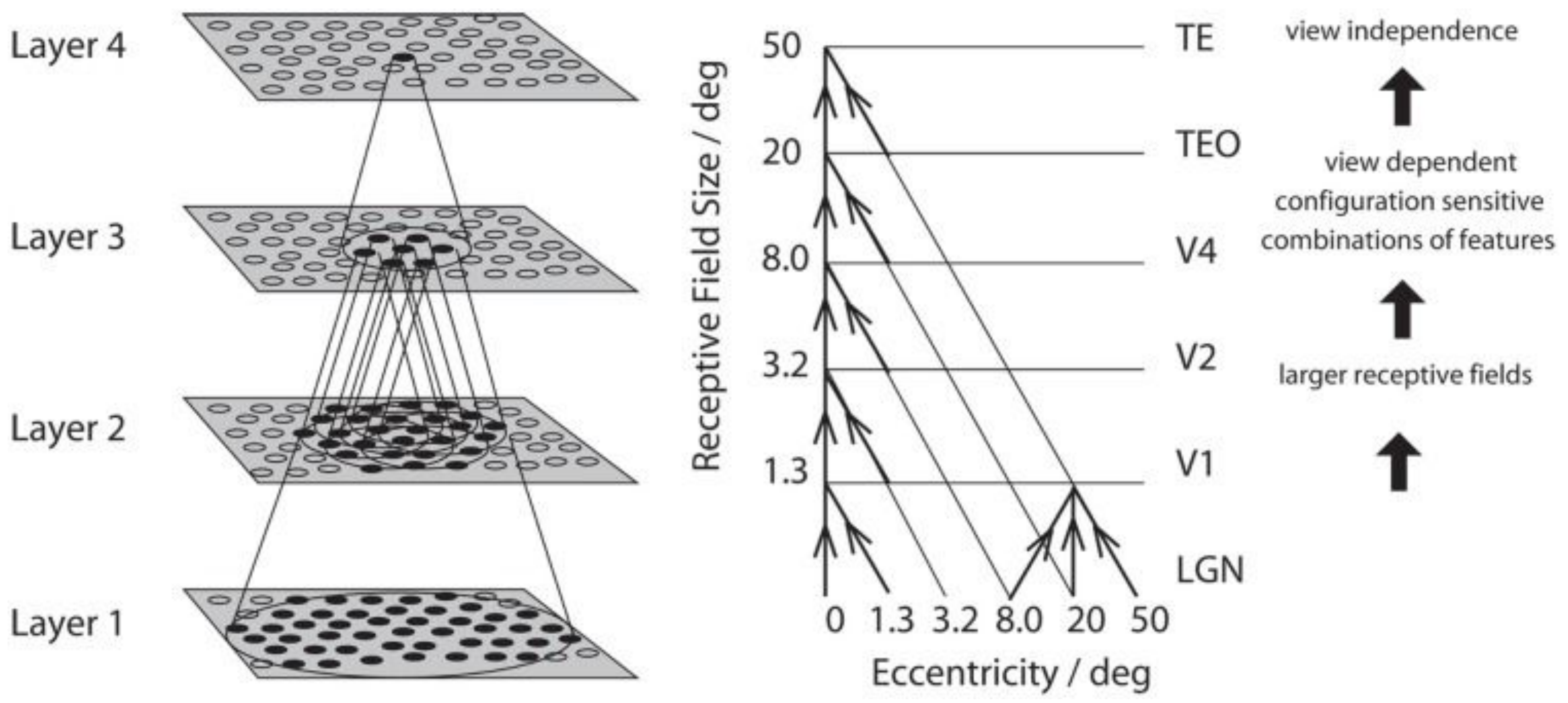
Hubel and Wiesel





Hubel & Wiesel, 1968

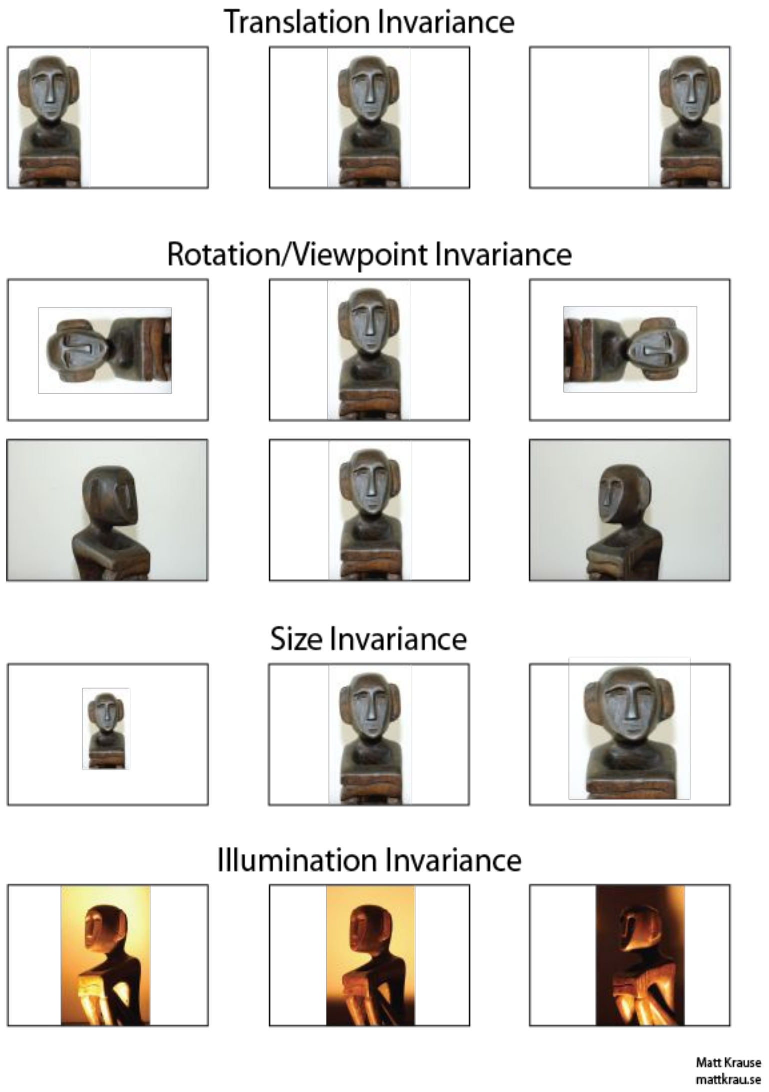
A hierarchy of processing



Rolls 2012

Invariances

JOSH HAROLDSON VIA FLICKR // CC BY-NC 2.0

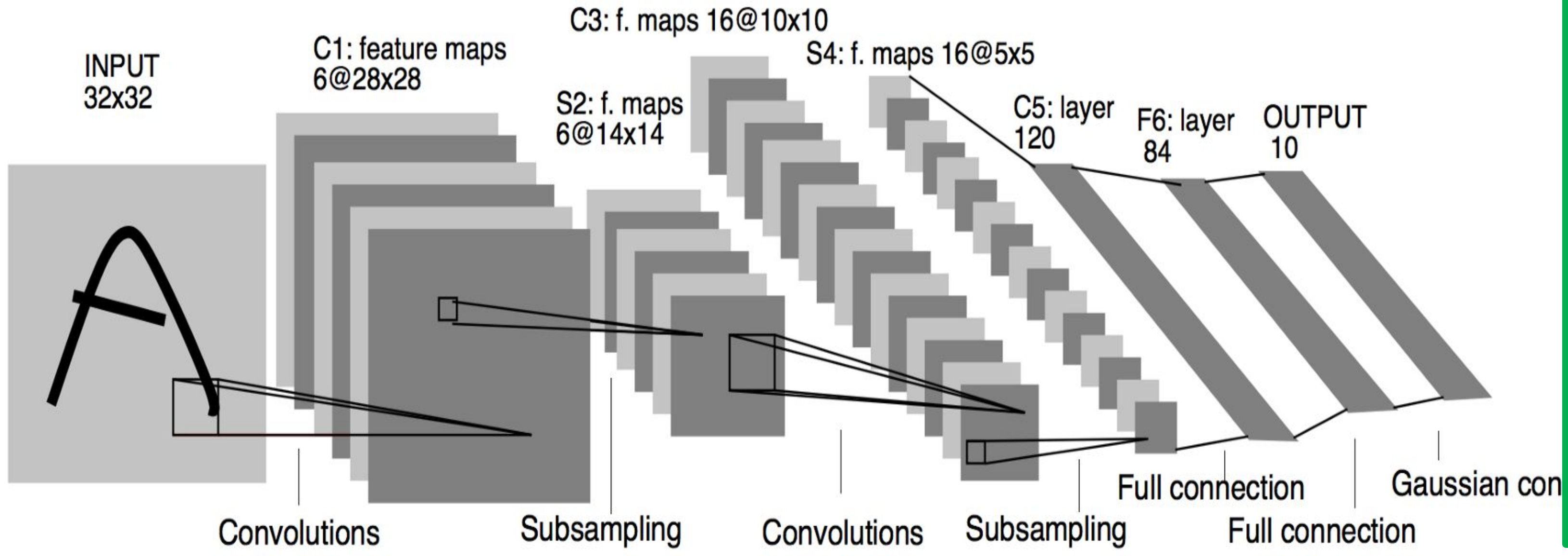


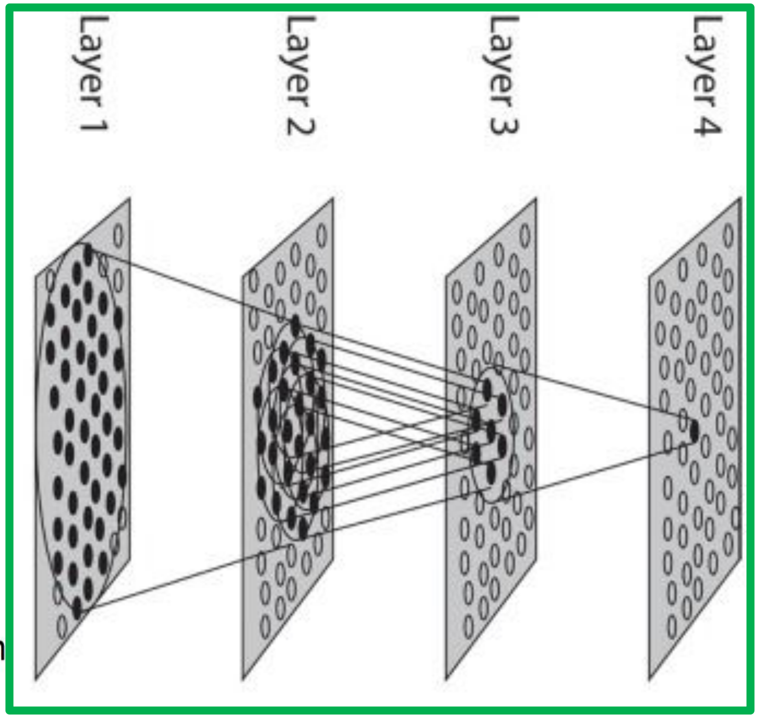
Early history of CNNs

neuromatch academy

LeNet (1998) -- Architecture

- Developed by Yann LeCun @ Bell labs





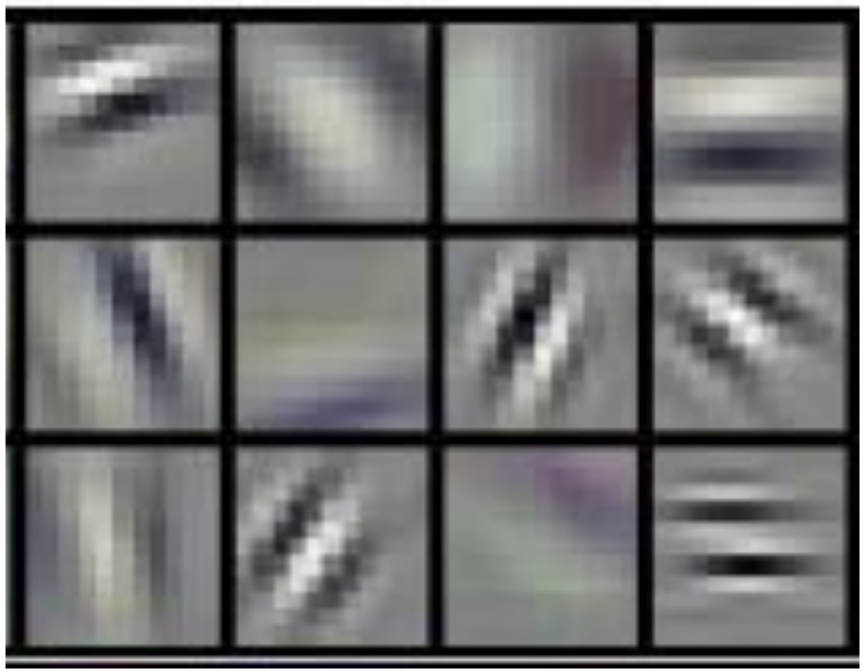
LeNet (1998) -- Results

- Successfully trained a 60K parameter neural network without GPU acceleration!
- Automatically reading zip codes on mail
- 0.8% error on MNIST; near state-of-the-art at the time.

LeNet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

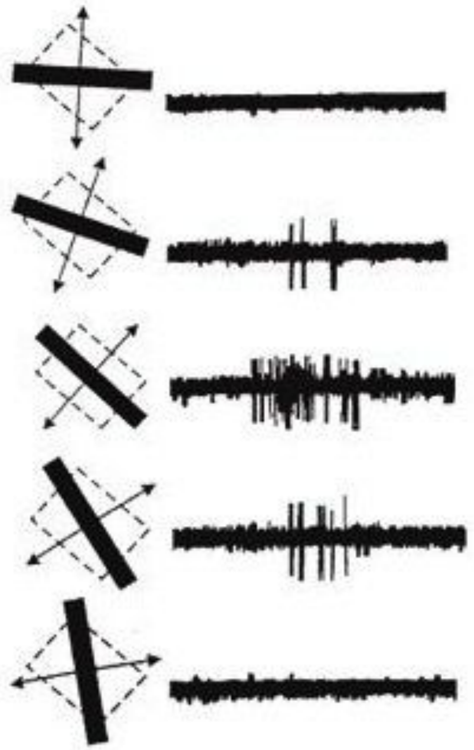
CNNs: Convolution

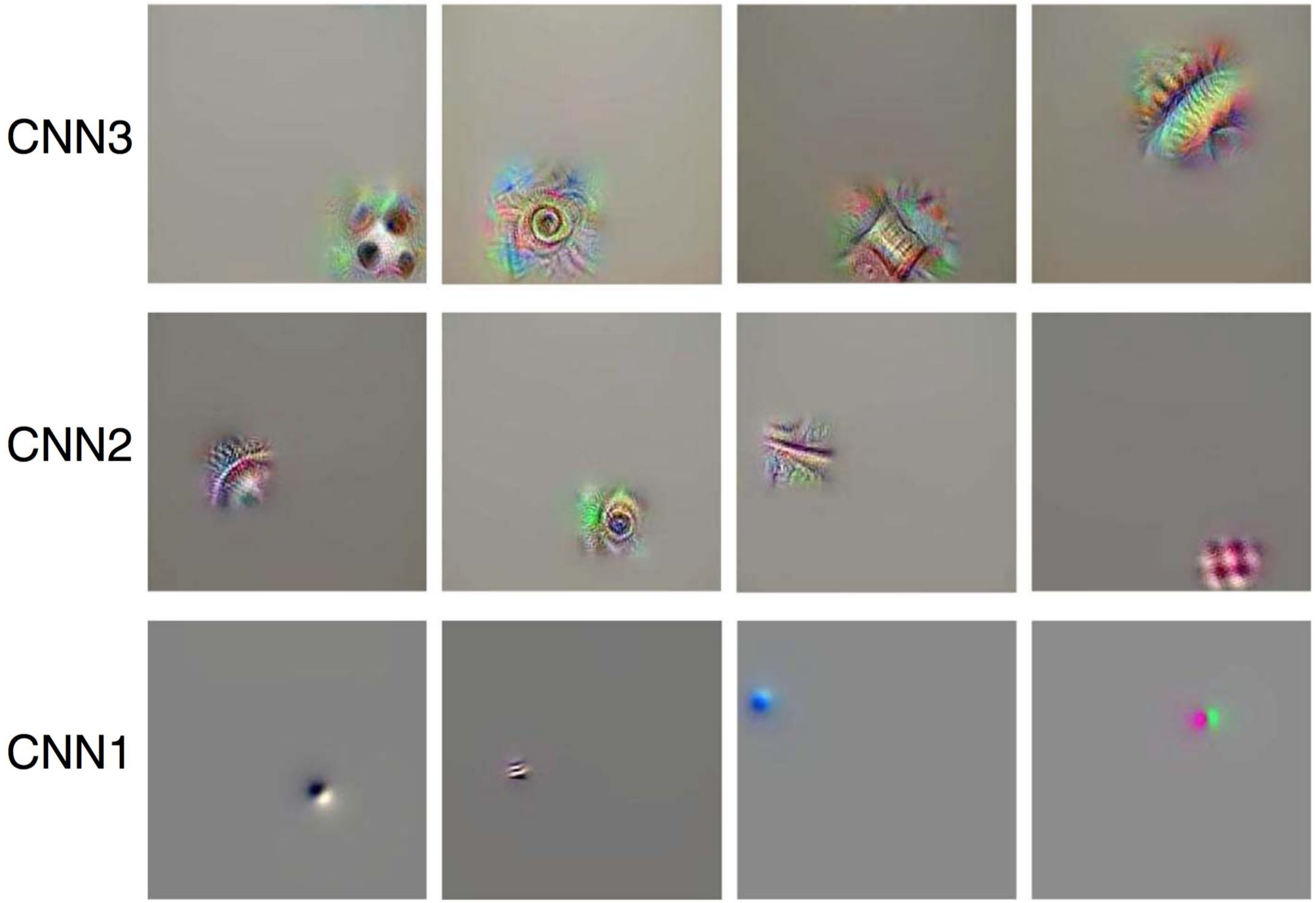
- Typical Learned Filters



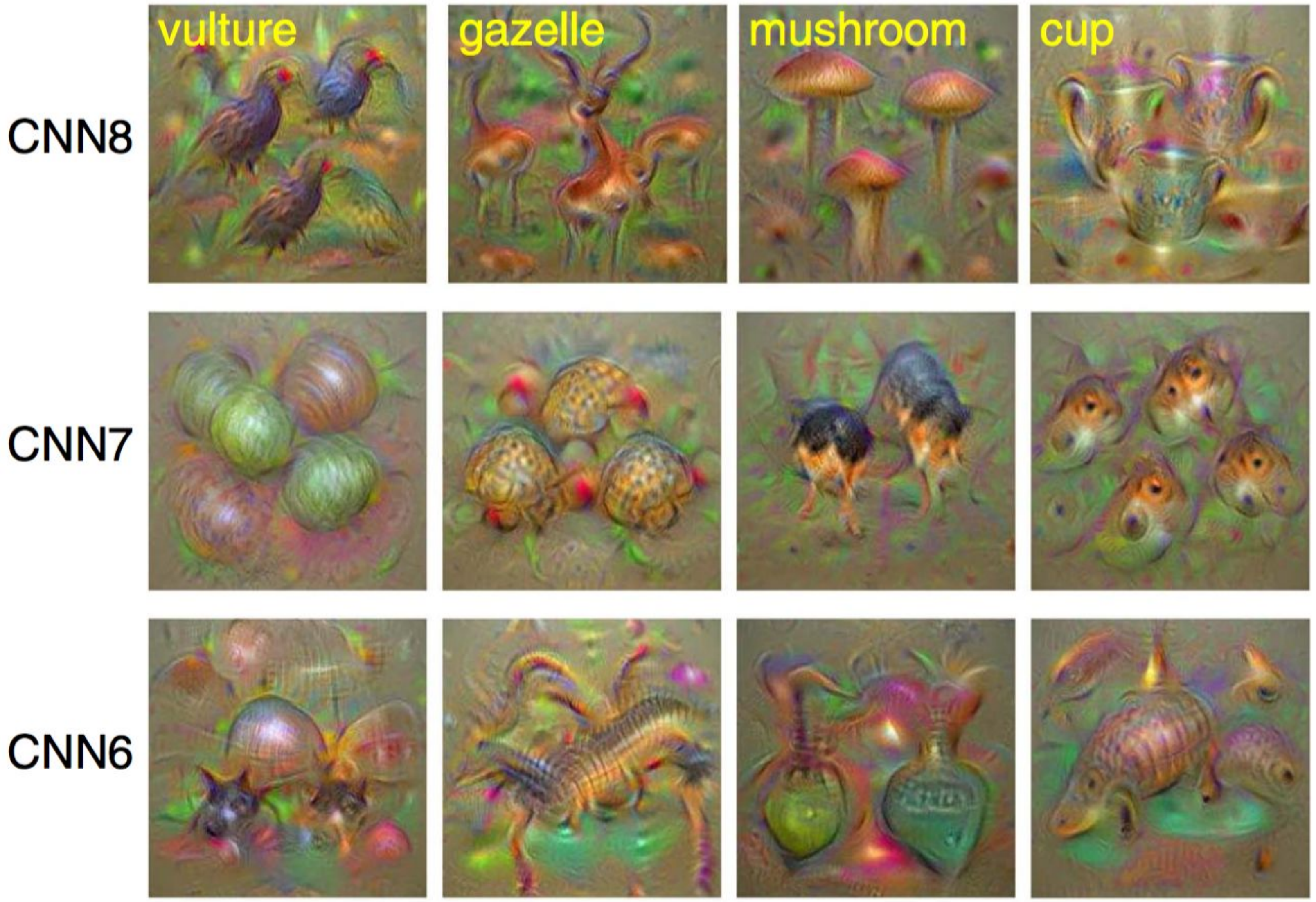
Krizhevsky et al., 2012

Hubel and Wiesel, 1968





Horikawa & Kamitani (2017)



Horikawa & Kamitani (2017)

DL is a story of representation learning

Hence ICLR

What is a representation?

A symbol or construct that approximates the entities and/or relations in the real world

Your brain has representations of the real world because it cannot actually *contain* the real world

DL is a story of representation learning

Your brain's representations allow it to approximate the real world in very useful ways.

E.g. you understand that some objects are more similar than others, you understand that objects can work together, objects can be parts of other objects, etc...

What do we want a good representation to do for us?

Regularization

Discuss with your pod...

E 1.1 What makes a representation good?

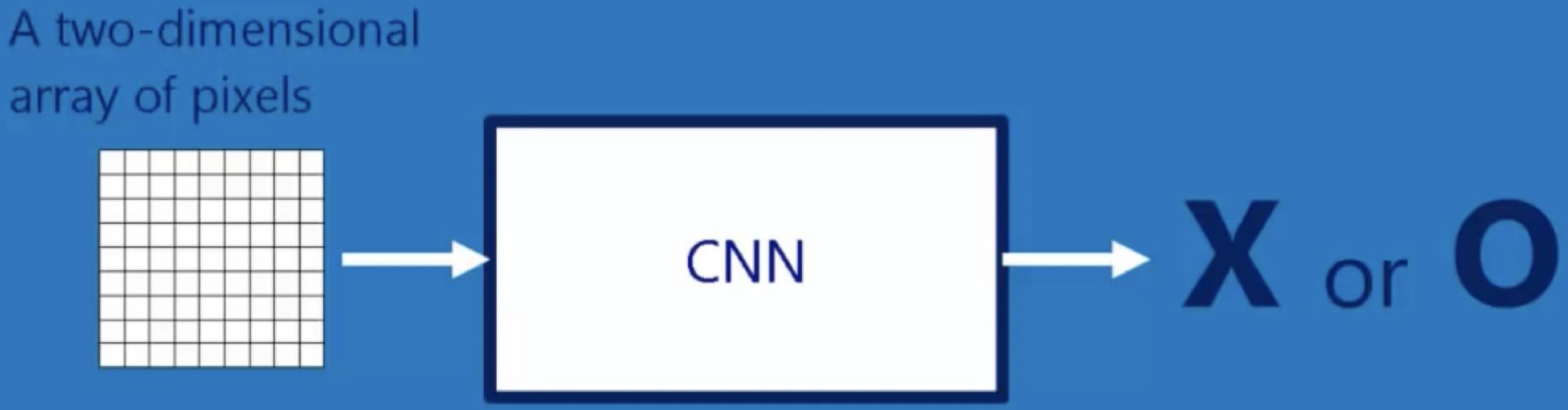
Consider: Object vs digit classification - how would the representations differ?

Section 2:
What is a CNN? Convolution?

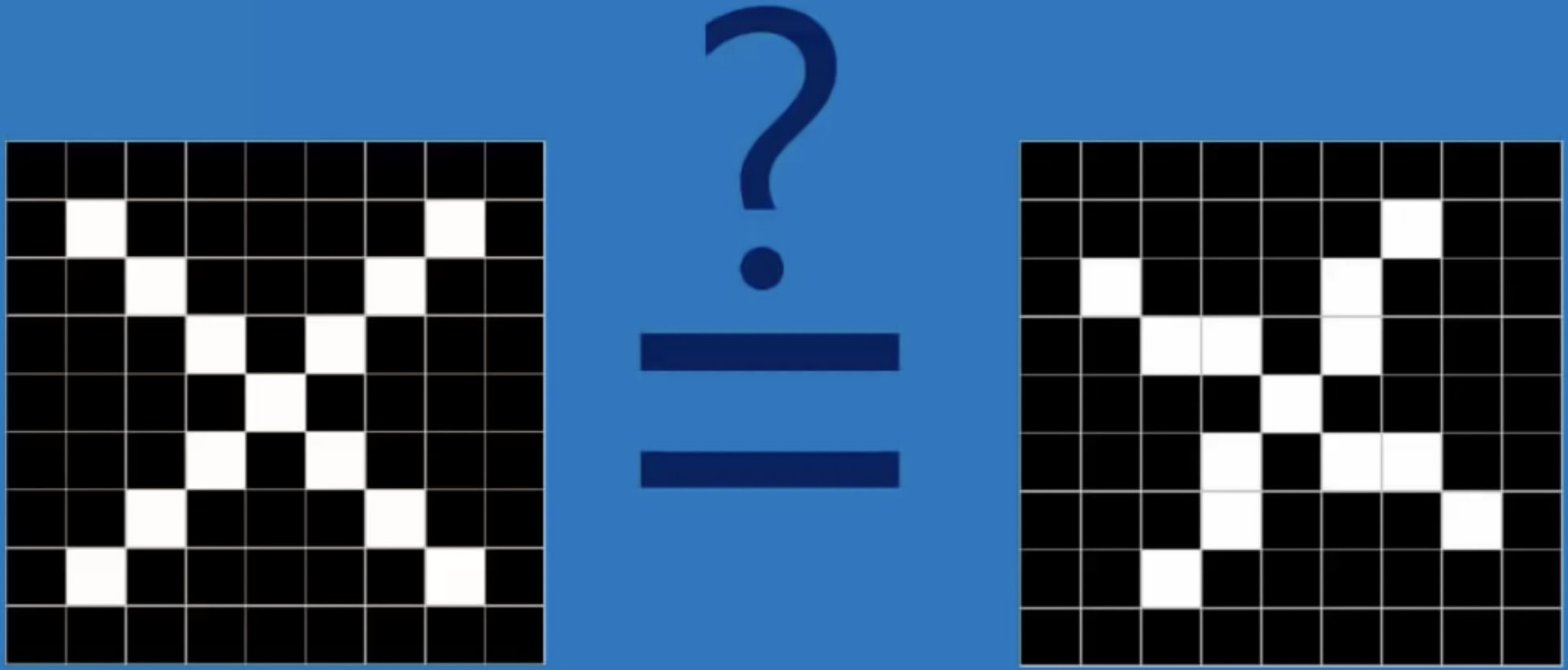
neuromatch academy

What is a CNN?

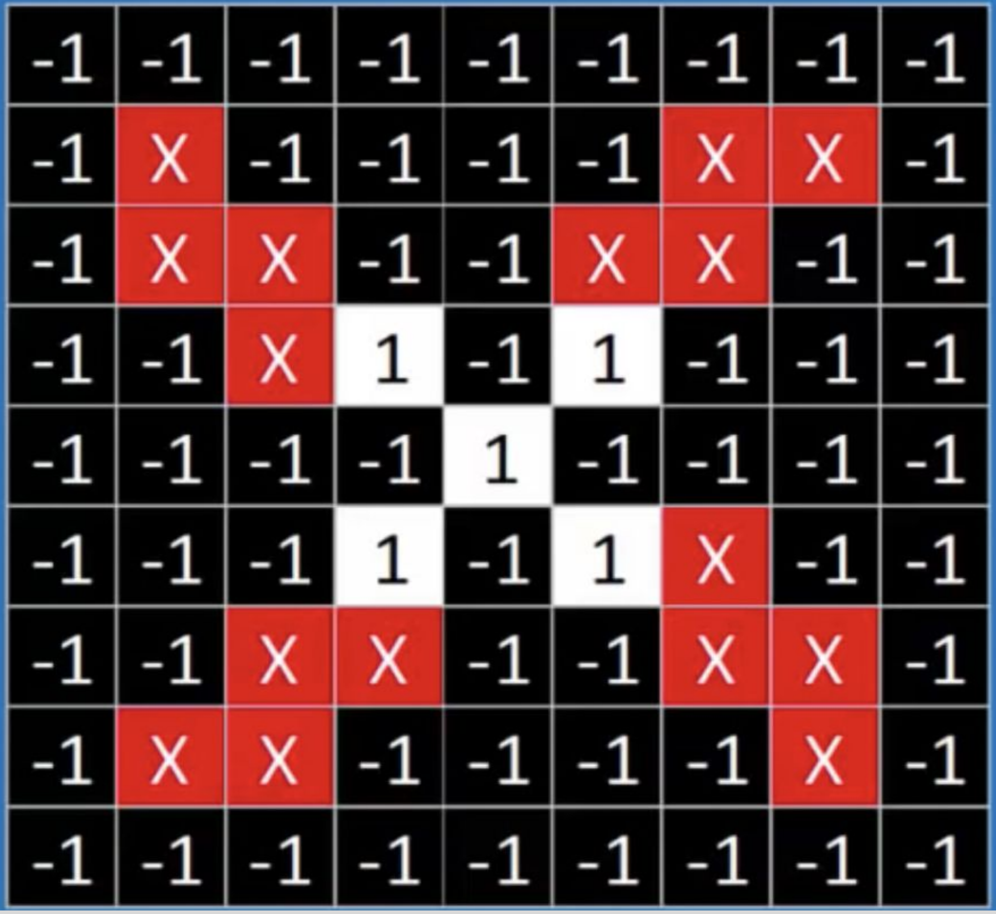
All the (blue) slides adapted from Brandon Rohrer with permission



Nontrivial

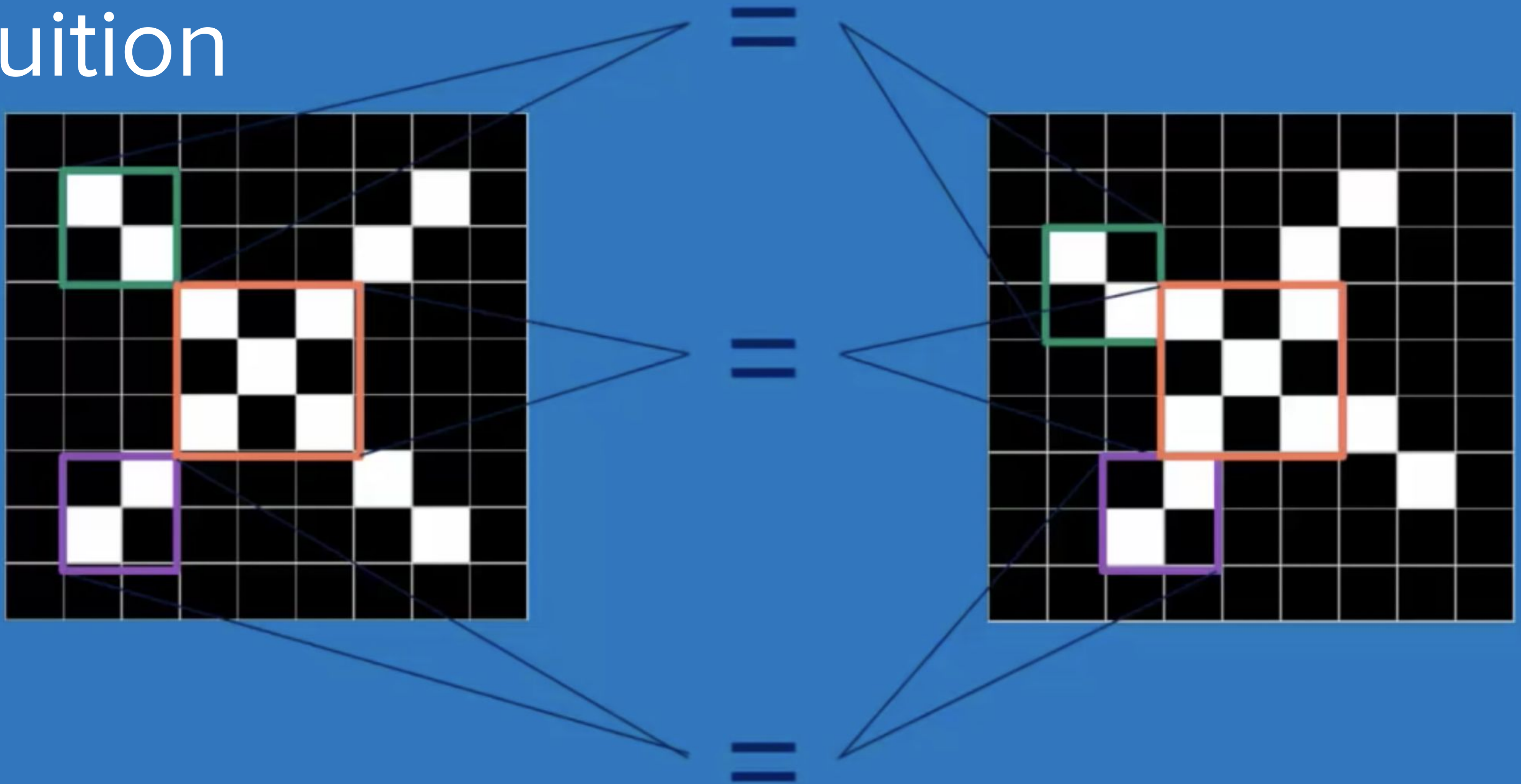


Naive comparison

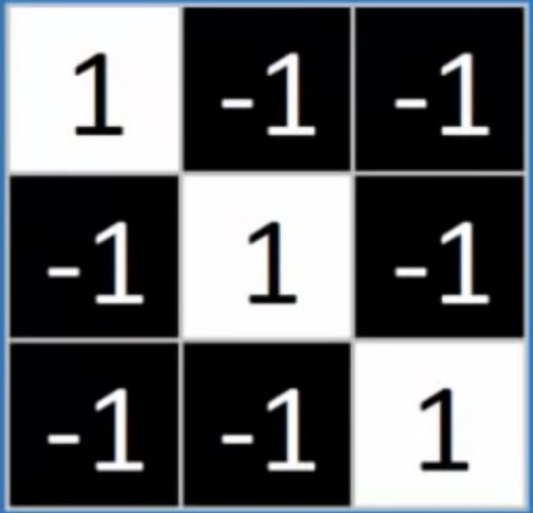


-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

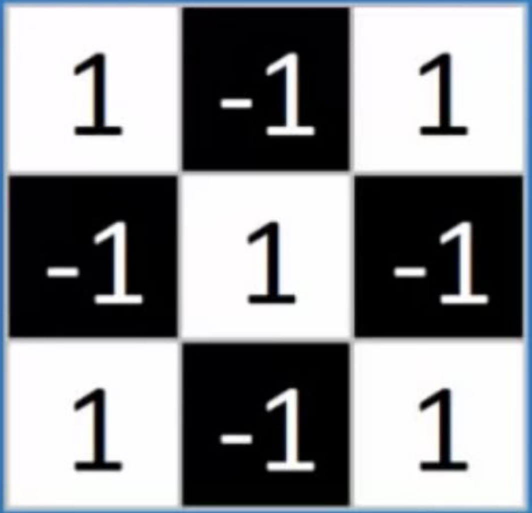
Intuition



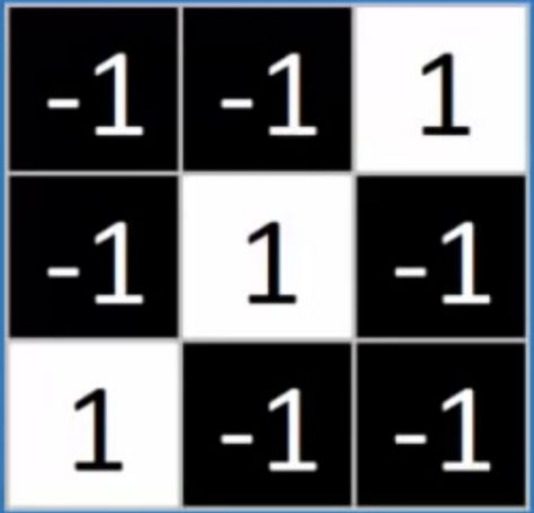
Potential local features



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

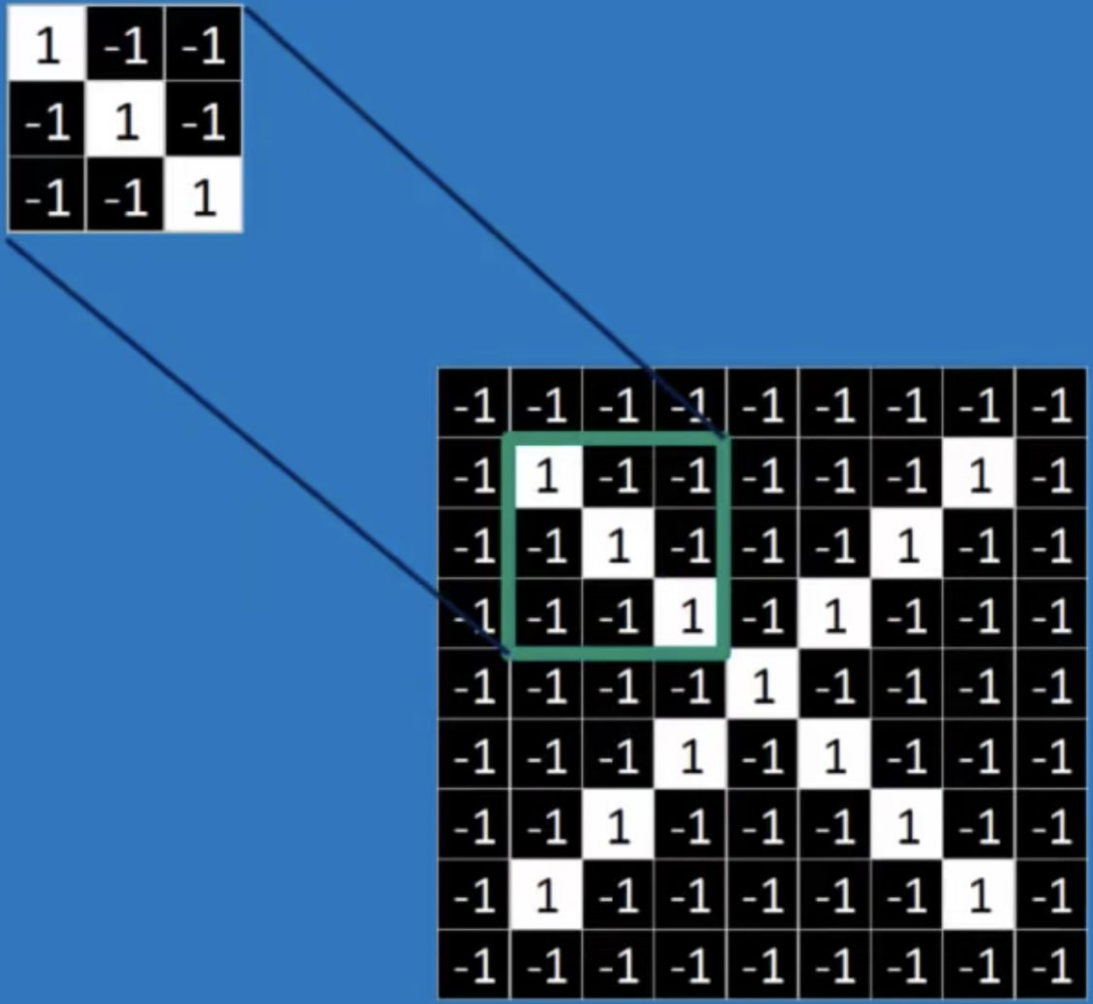


1	-1	1
-1	1	-1
1	-1	1



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

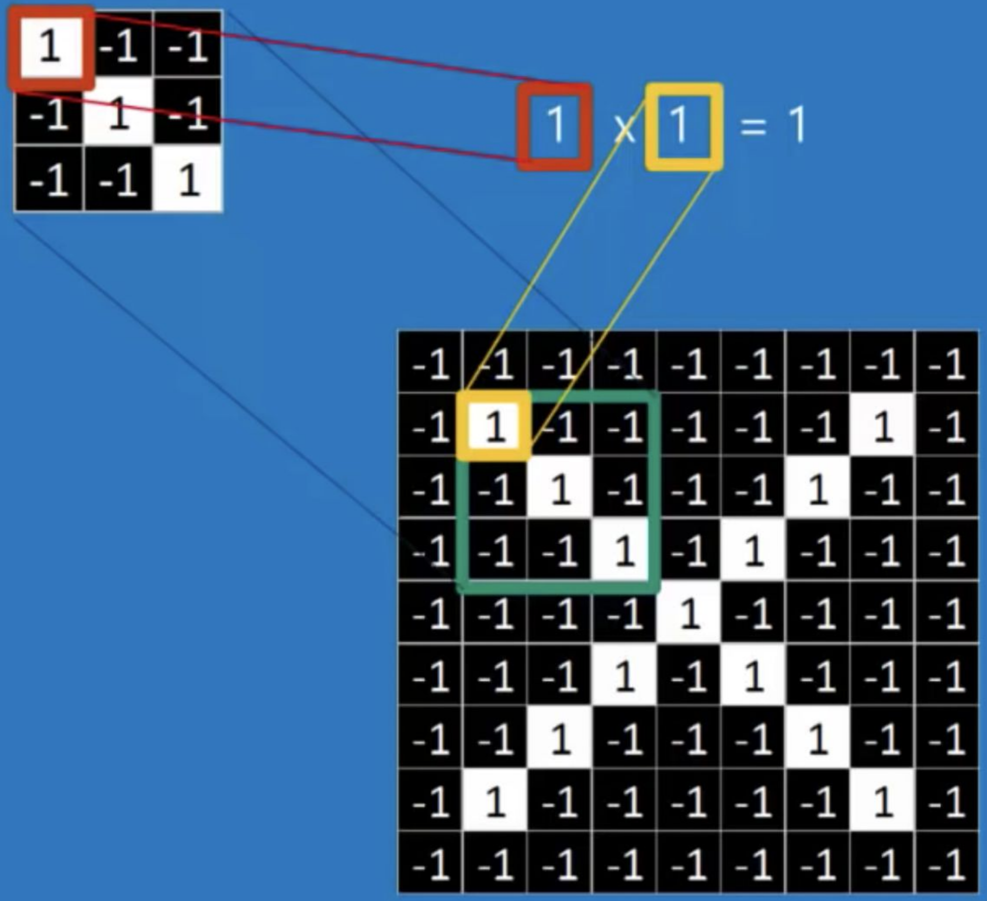
Setting



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

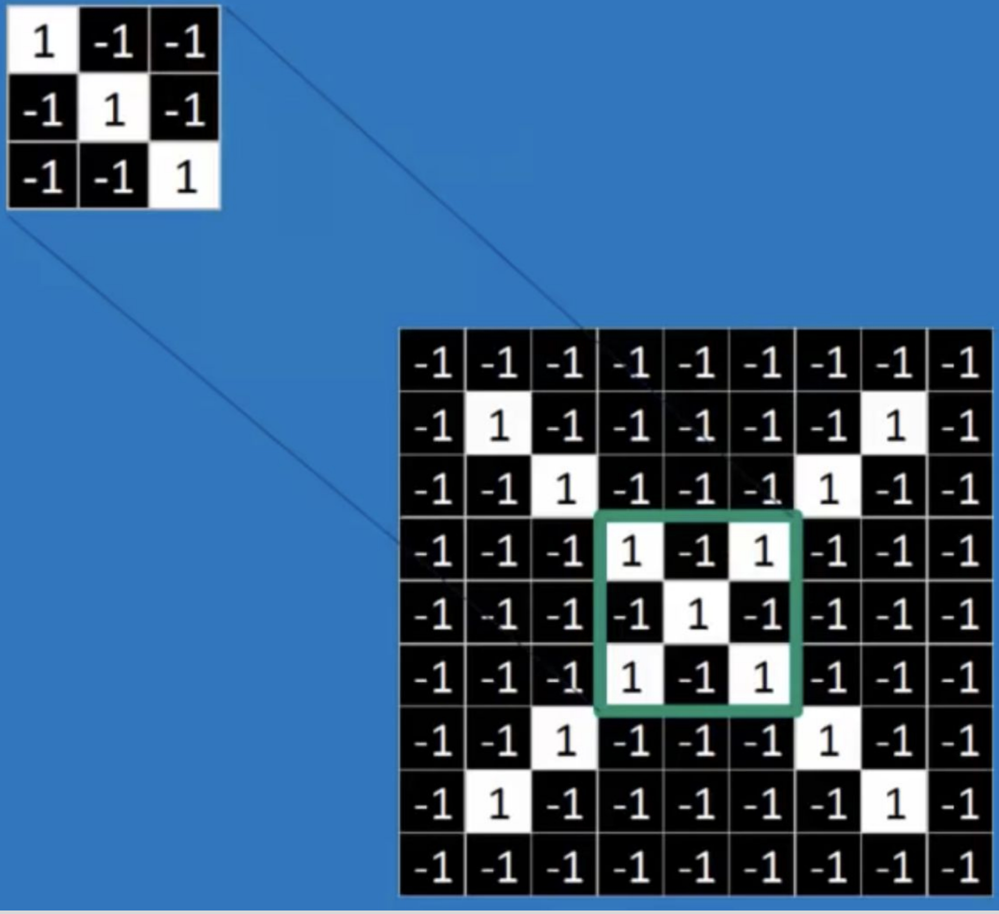
Local filtering



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

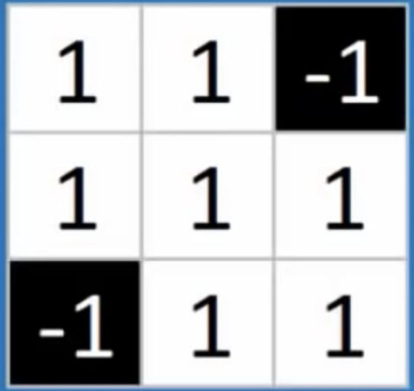
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

And at different locations



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



1	1	-1
1	1	1
-1	1	1

Convolution

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 0.77 & -0.11 & 0.11 & 0.33 & 0.55 & -0.11 & 0.33 \\ -0.11 & 1.00 & -0.11 & 0.33 & -0.11 & 0.11 & -0.11 \\ 0.11 & -0.11 & \boxed{1.00} & -0.33 & 0.11 & -0.11 & 0.55 \\ 0.33 & 0.33 & -0.33 & \boxed{0.55} & -0.33 & 0.33 & 0.33 \\ 0.55 & -0.11 & 0.11 & -0.33 & 1.00 & -0.11 & 0.11 \\ -0.11 & 0.11 & -0.11 & 0.33 & -0.11 & 1.00 & -0.11 \\ 0.33 & -0.11 & 0.55 & 0.33 & 0.11 & -0.11 & 0.77 \end{bmatrix}$$

This is 2D convolution

- Convolution can also be 1D and 3D
- Convolutions can be used in other application areas
  - Whenever there’s predictable correlation over time/space
  - Language, protein or DNA sequences

A convolution exercise

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

Here’s our “image”

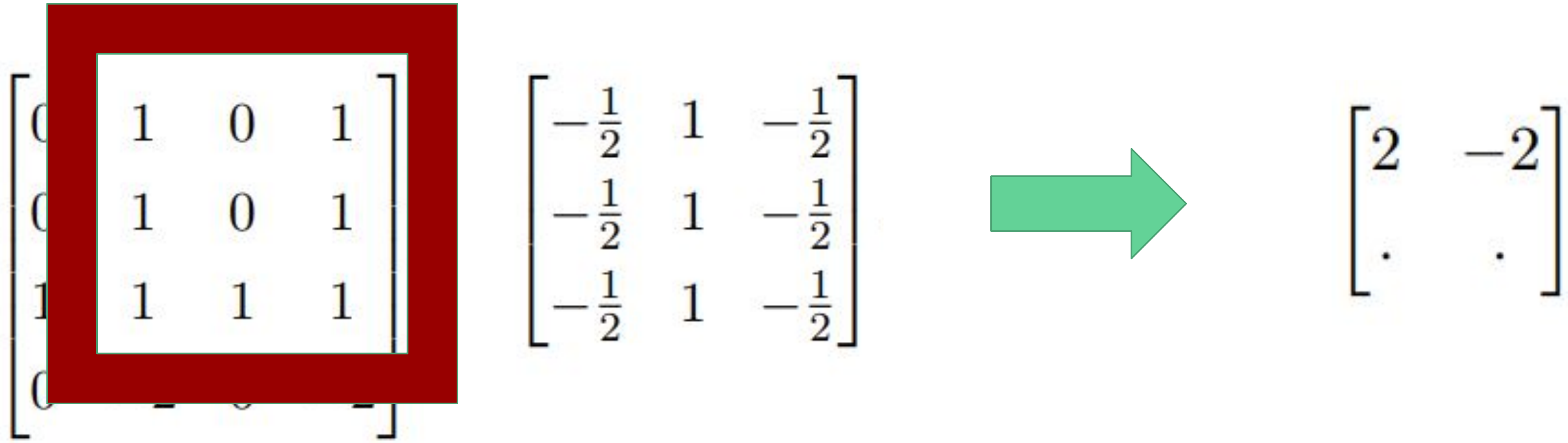
$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

Here’s our “filter”

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

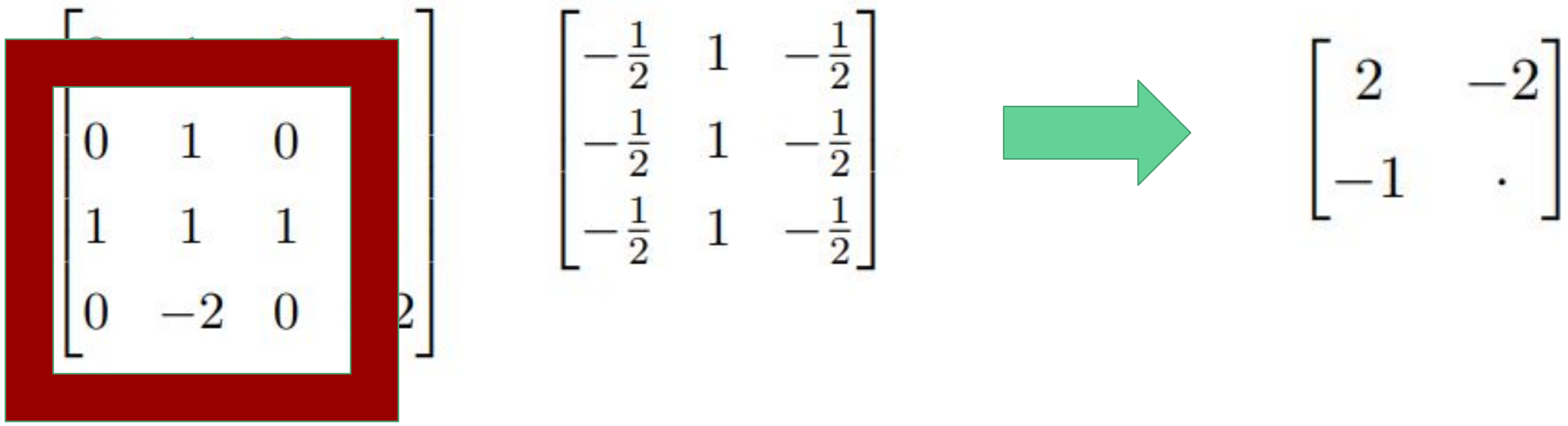
$$\begin{aligned} &\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right) \\ &\left(0 \times \frac{-1}{2}\right) + (1 \times 1) + \left(0 \times \frac{-1}{2}\right) \\ &\left(1 \times \frac{-1}{2}\right) + (1 \times 1) + \left(1 \times \frac{-1}{2}\right) = 2 \end{aligned}$$

A convolution exercise



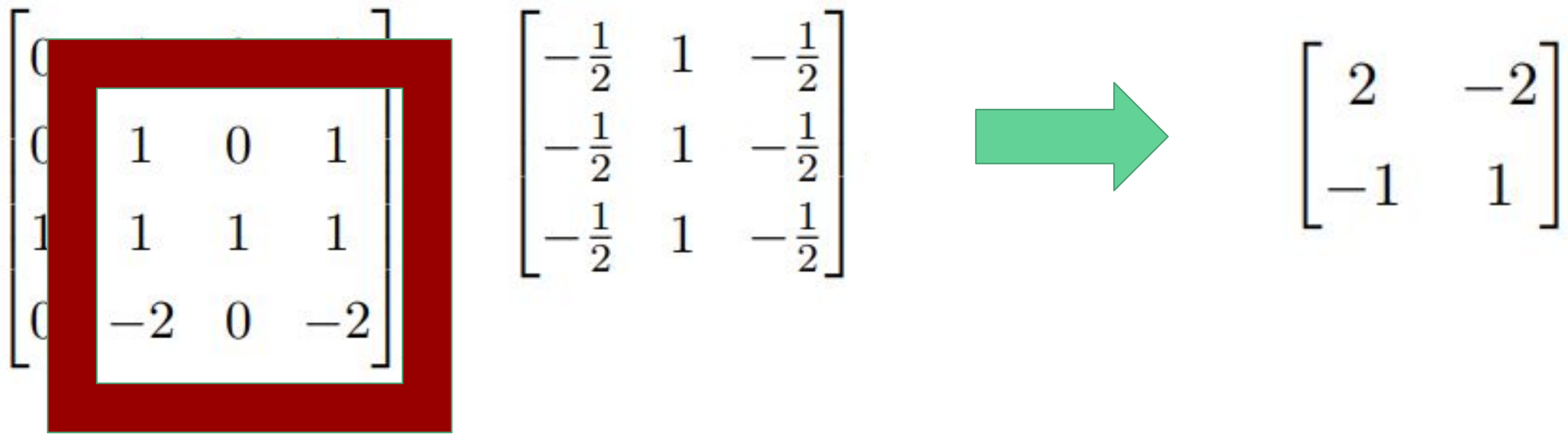
$$\begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & -2 \\ . & . \end{bmatrix}$$

A convolution exercise



$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & -2 & 0 \end{bmatrix} \quad \begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & -2 \\ -1 & \cdot \end{bmatrix}$$

A convolution exercise



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

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

E 2.1-2.2 Do a convolution by hand and implement as function

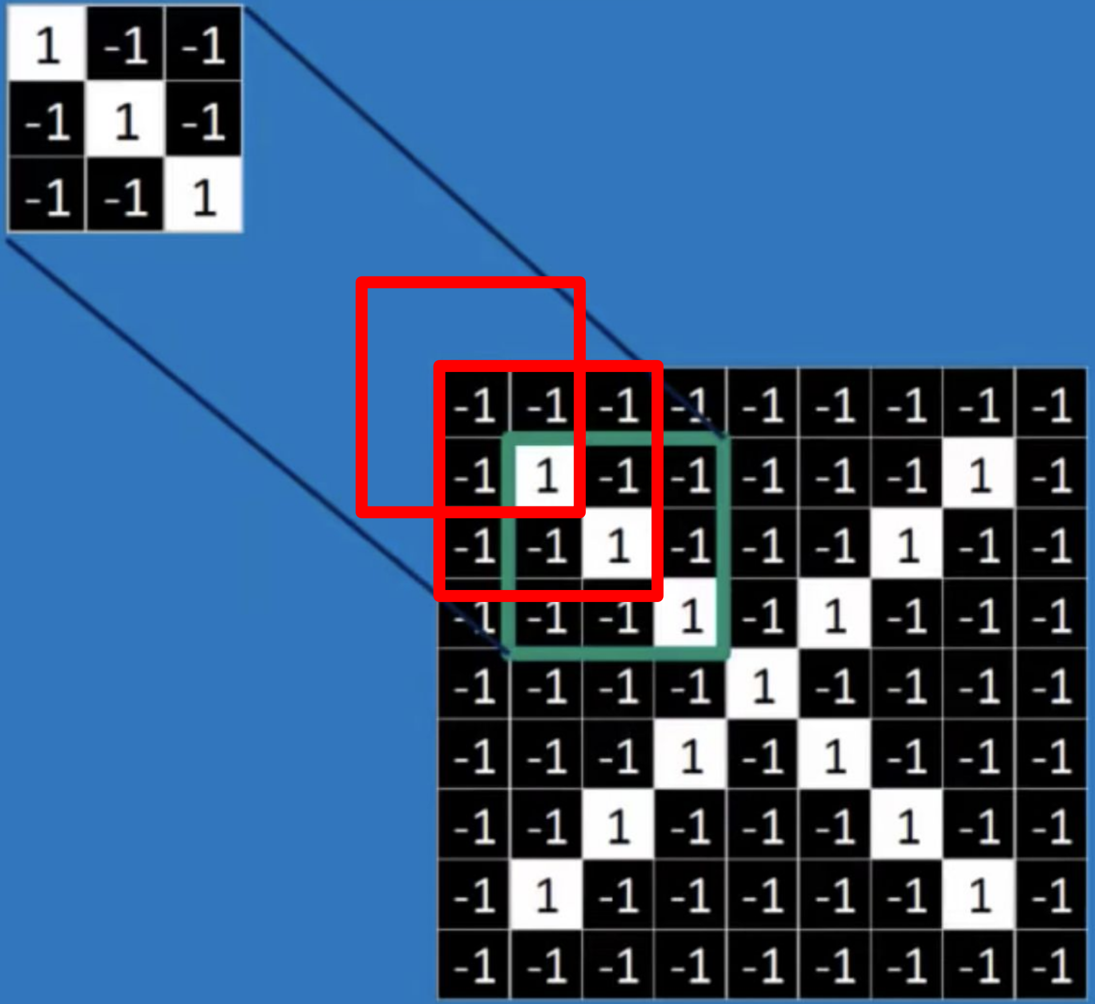
Details about convolution

neuromatch academy

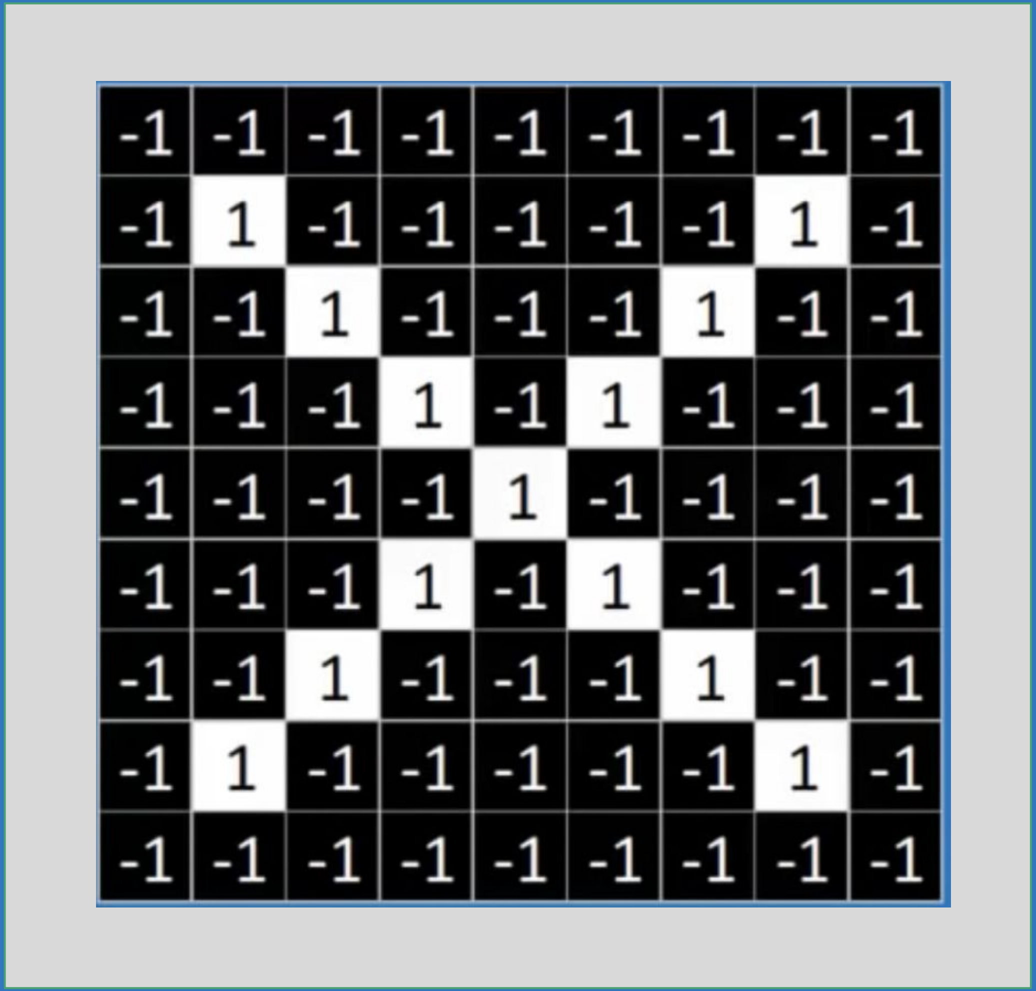
What about the edges of an image?

JOSH HAROLDSON VIA FLICKR // CC BY-NC 2.0

Padding

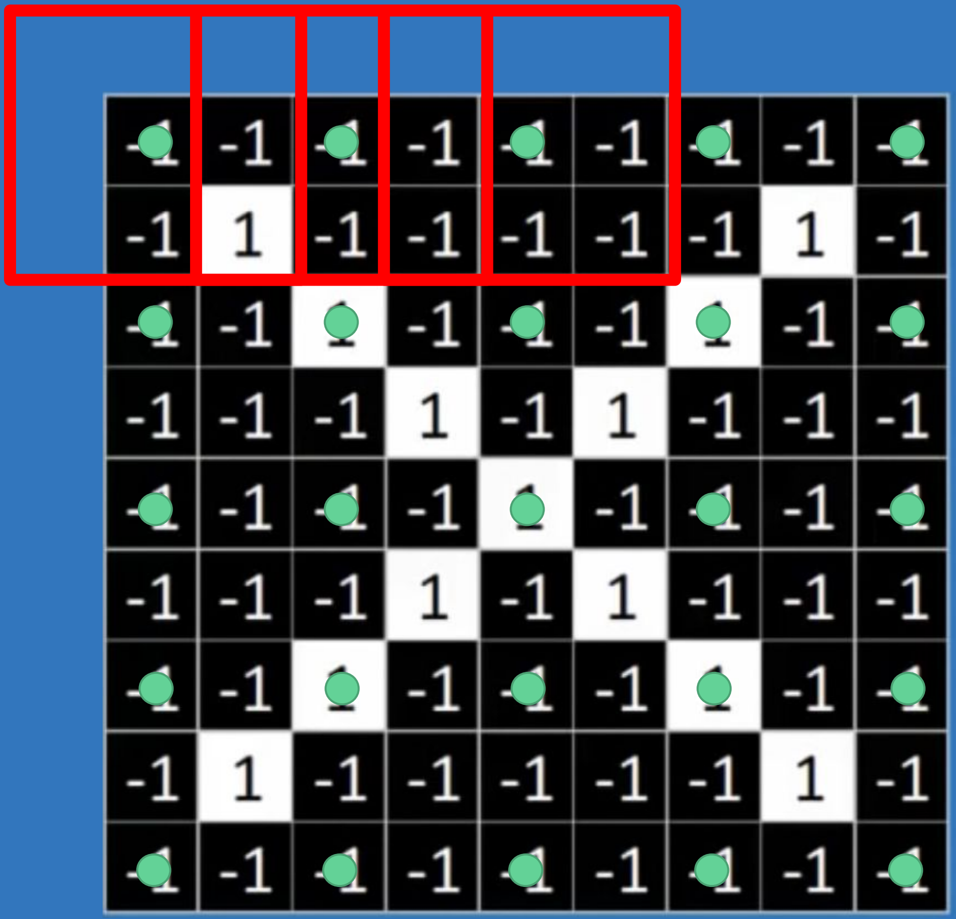


Padding

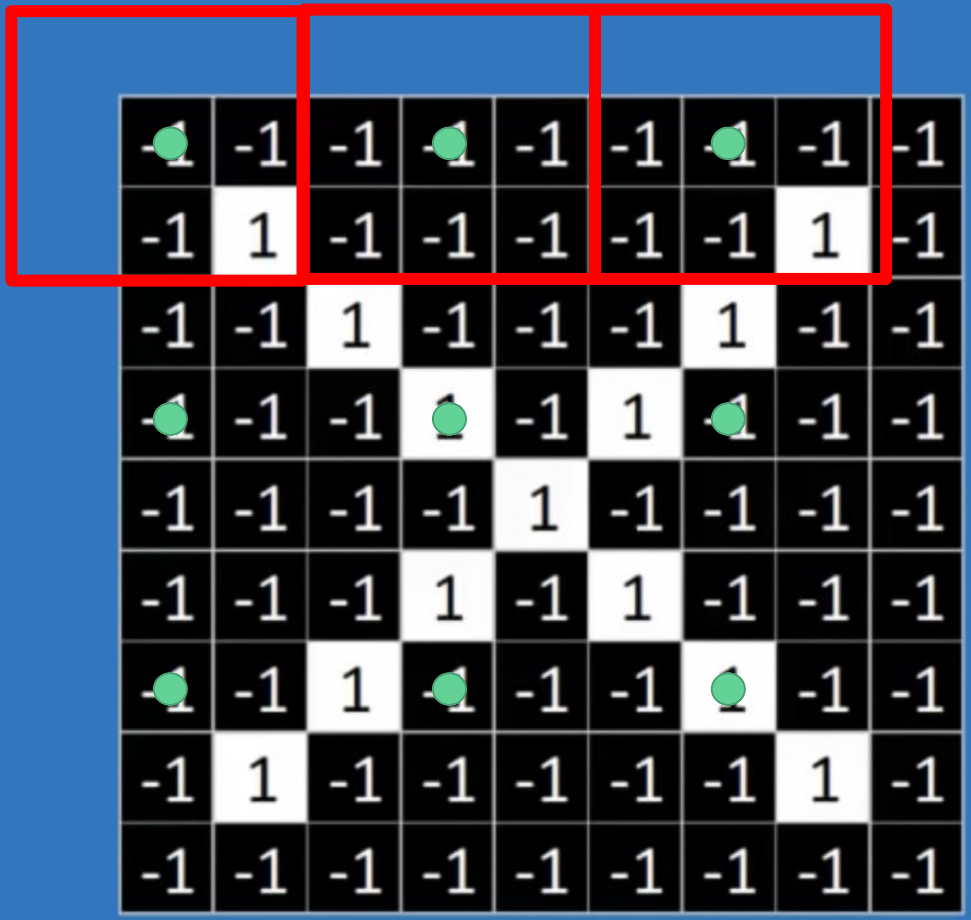


-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

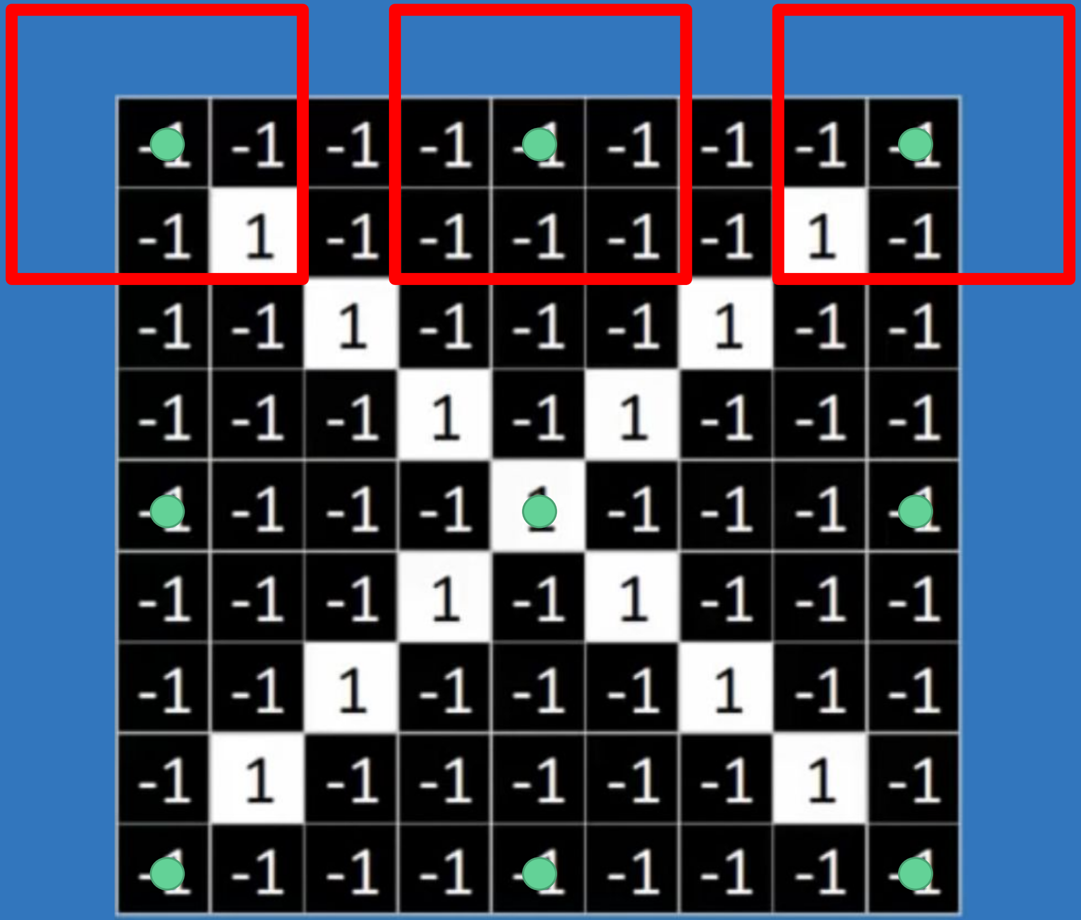
Stride 2



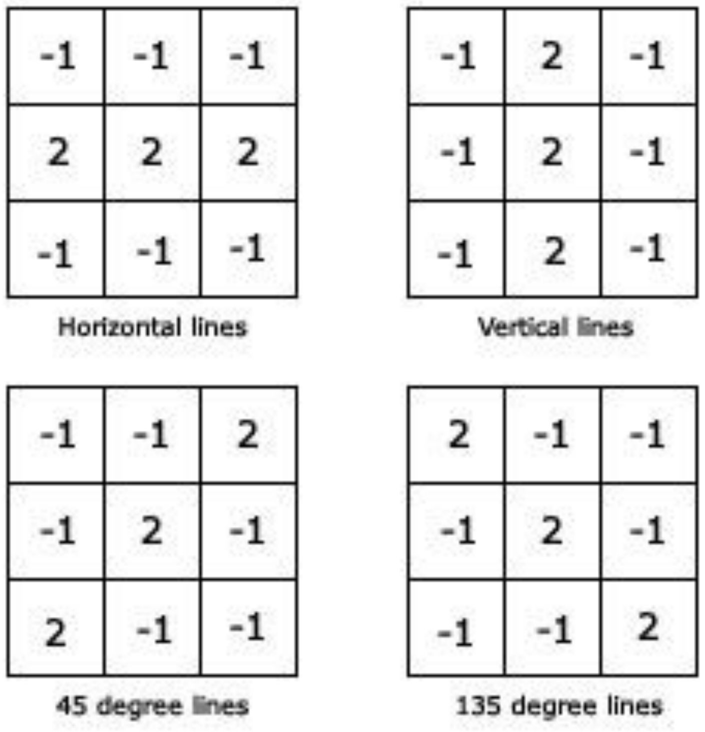
Stride 3



Stride 4


-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

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

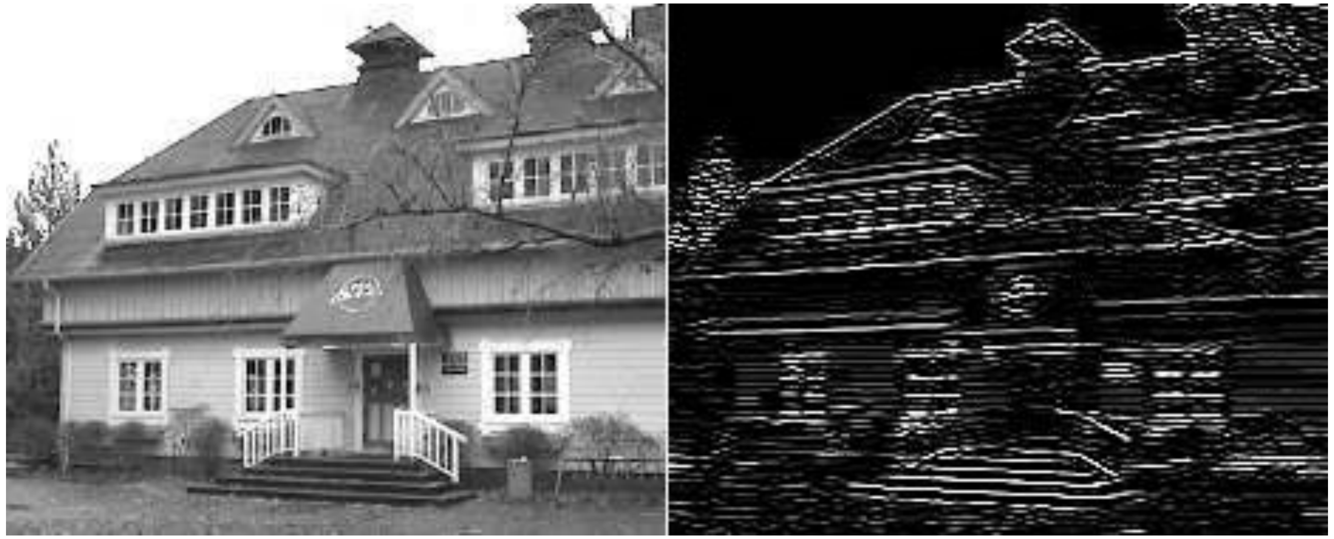
-1	-1	2
-1	2	-1
2	-1	-1

45 degree lines

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

135 degree lines

Horizontal lines:



Feature detection - edges

Filters give us global translation invariance

Run a filter across an image

How could that be useful for image recognition?

E 2.3-2.4 Convolve a local filter with an image

In what way could this be useful?

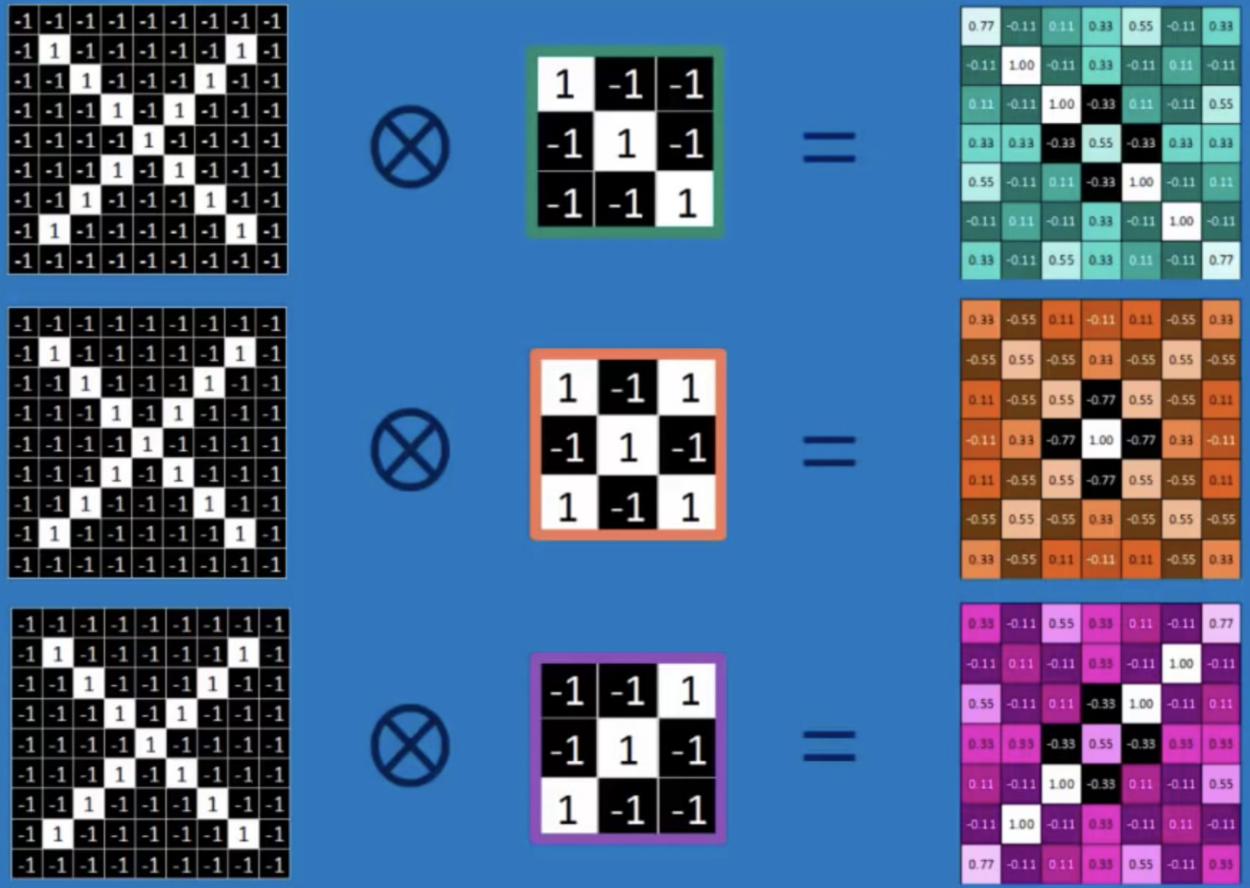
Section 3: Pooling and Subsampling

neuromatch academy

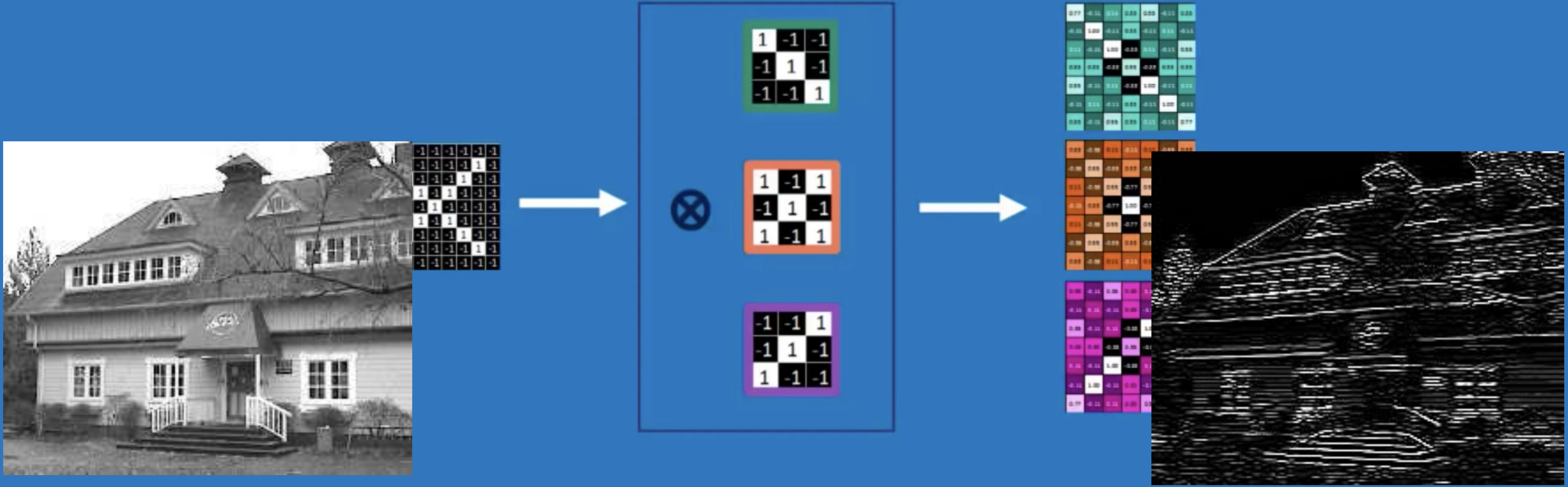
Multiple Filters and ReLU

neuromatch academy

Multiple filters

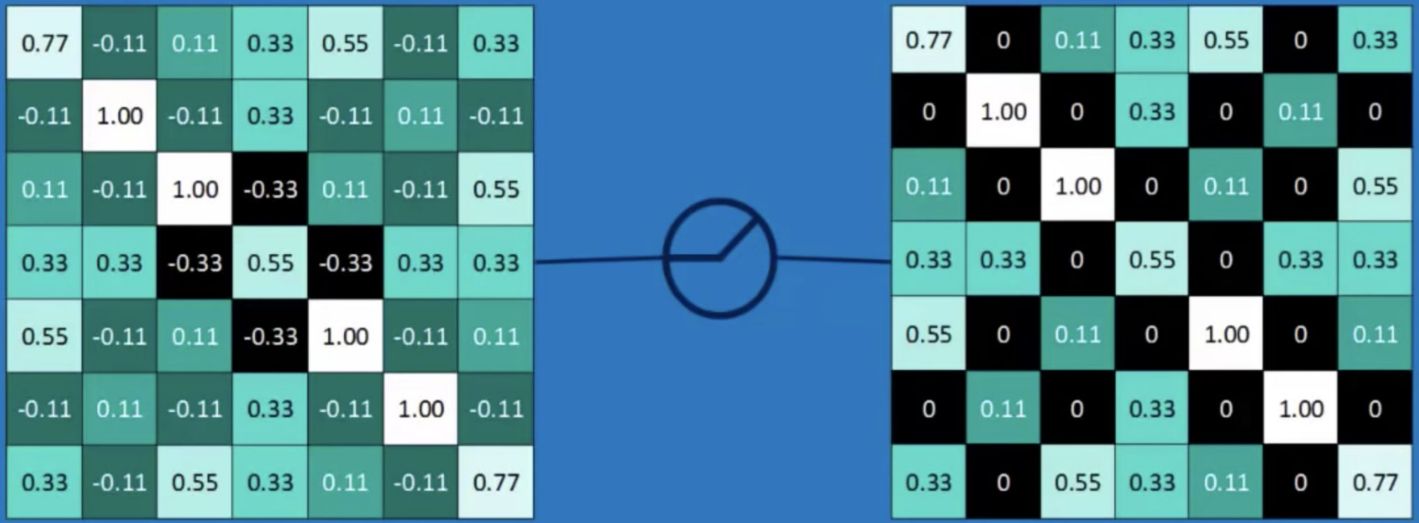


A convolution layer



Potential to add a ReLU

ReLU(x) = max(0,x)



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Combining filters

E 3.1-3.2 Experiment with multiple filters and RELU

Pooling

neuromatch academy

Pooling = Local invariance

- Recall: filters give us global invariance

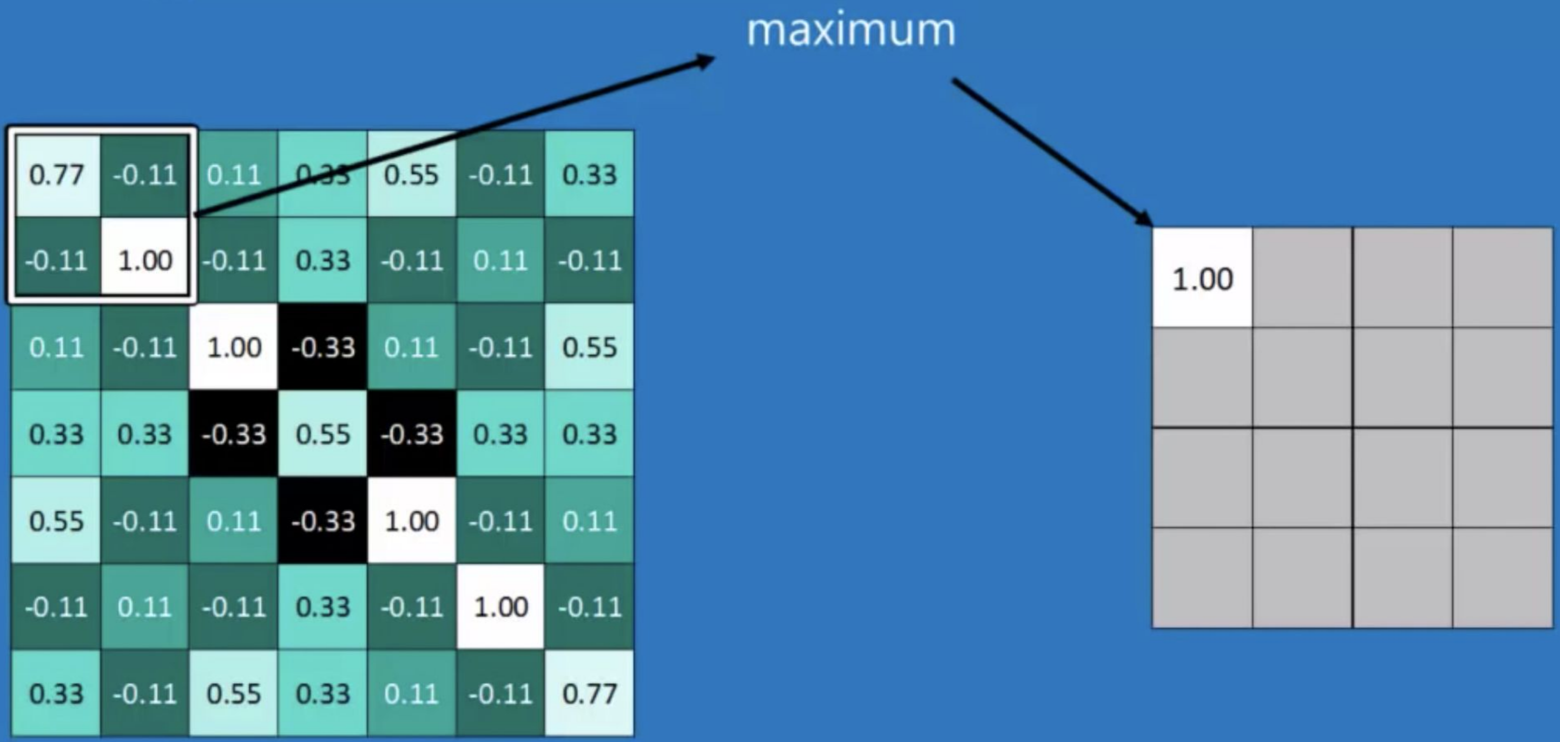
- Pooling gives local invariance

The need for some invariance

Features may appear in (slightly) different places

All following (blue) slides from Brandon Rohrer

Max-pooling

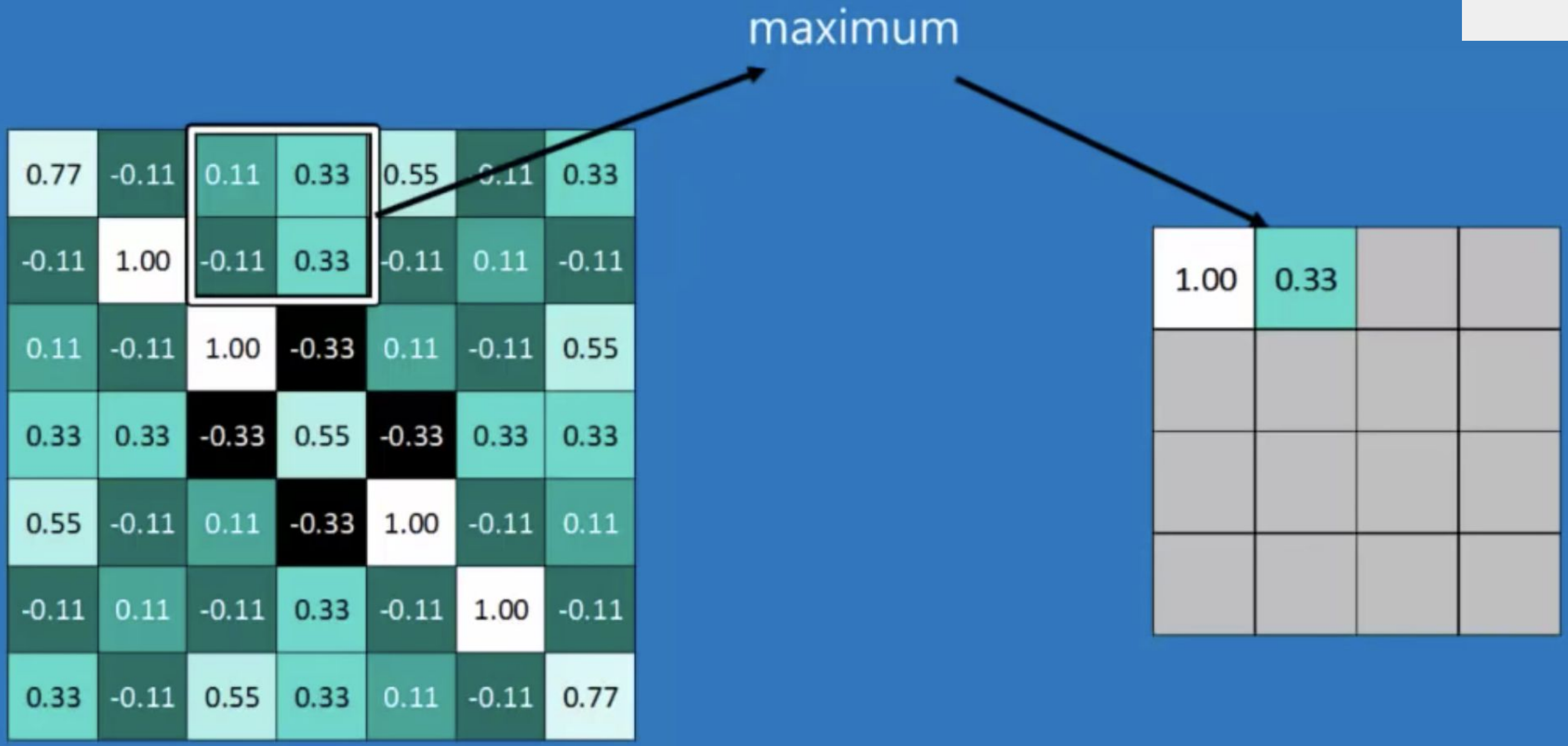


0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00			

2*2, Stride 2

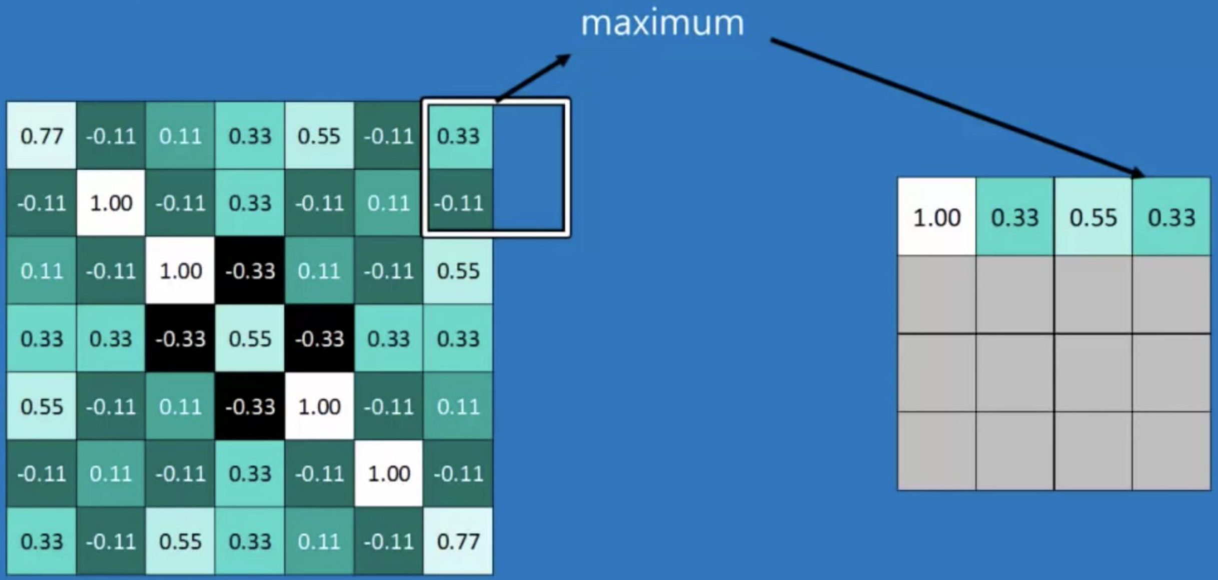
Across space



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33		

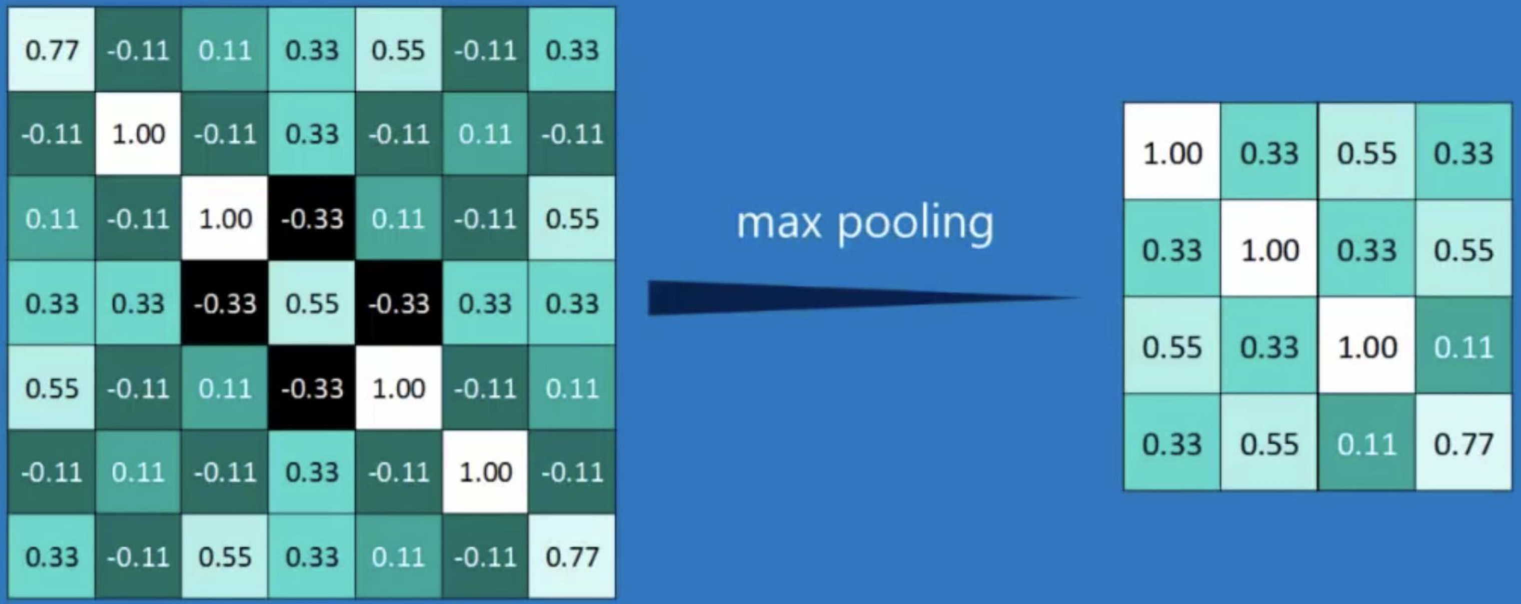
Again, Padding problem!



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33

And a whole max pooling operation

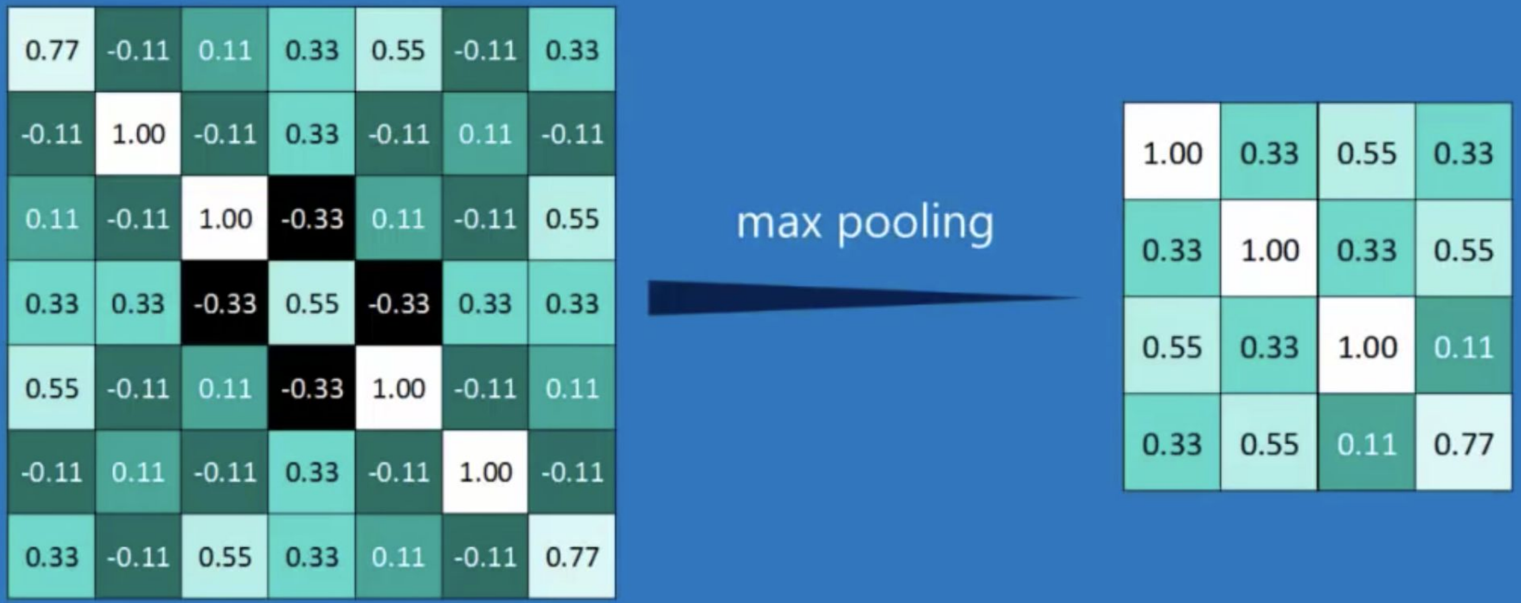


0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

How many parameters?

Hold up, how does this reduce the # of params?

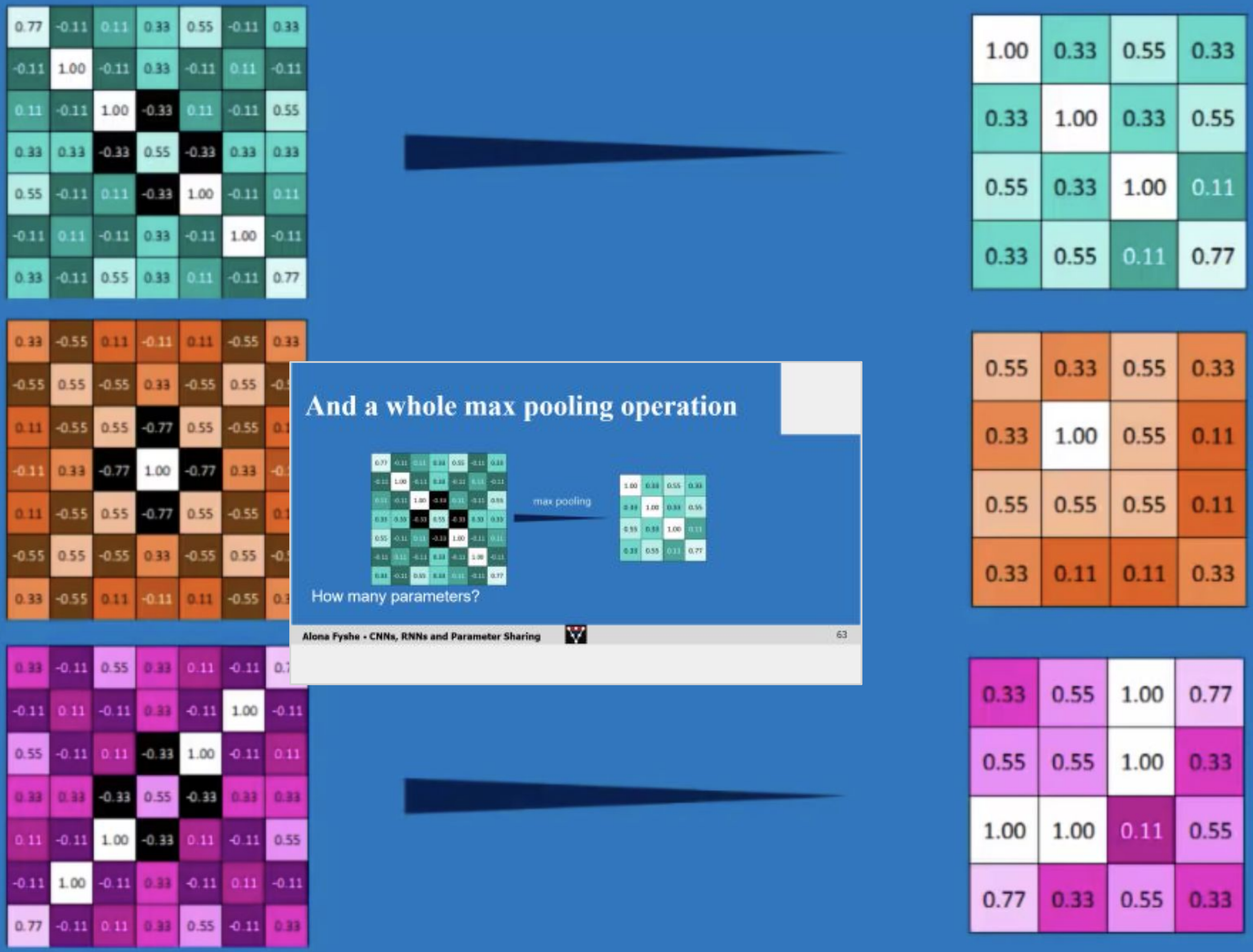


0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

Next layer has fewer connections!

Bunch of Filters



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.1
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.1
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.3

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.7
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

And a whole max pooling operation

max pooling

How many parameters?

Alona Fyshe • CNNs, RNNs and Parameter Sharing 63

Pooling layers

- Want to:
  - Reduce dimensions of data, without additional parameters
  - Introduce some ***local*** translation invariance
- Pooling operation can be max(0,x) or average(x)

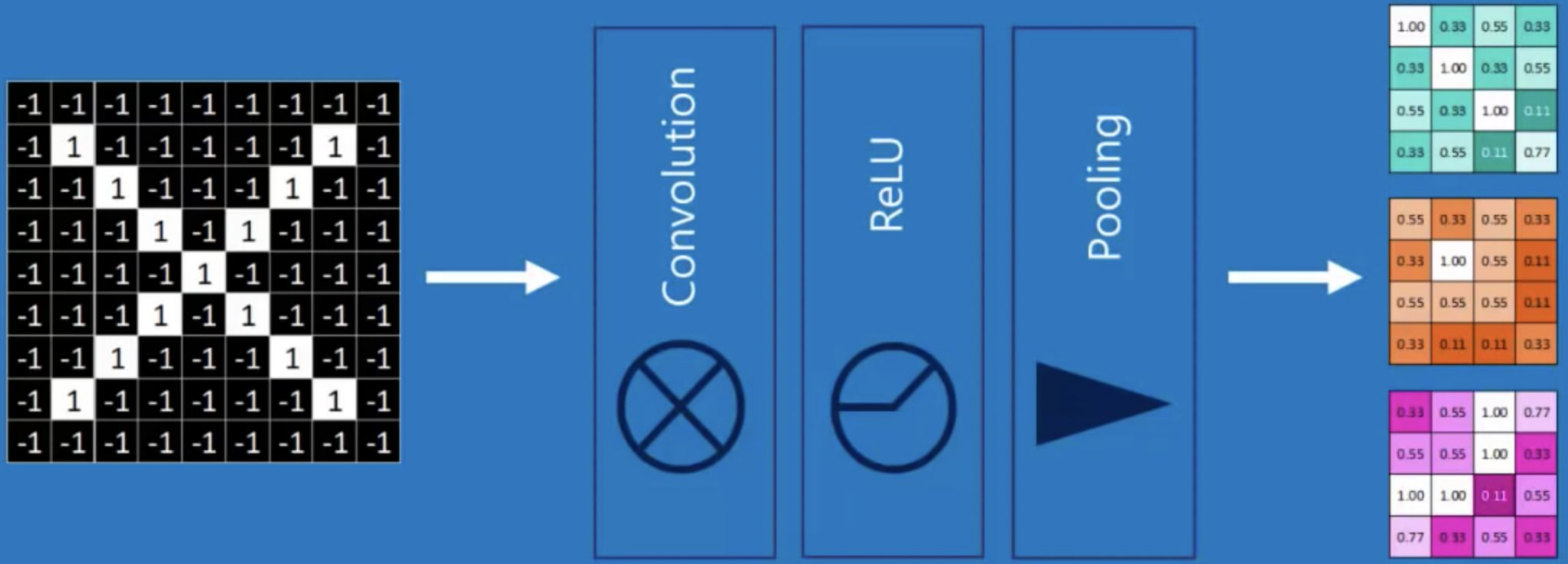
Dive into pooling 🤿

E 3.3-3.4 Experiment with and then implement max pooling

Section 4: Putting it all together

neuromatch academy

So far: not much of a hierarchy



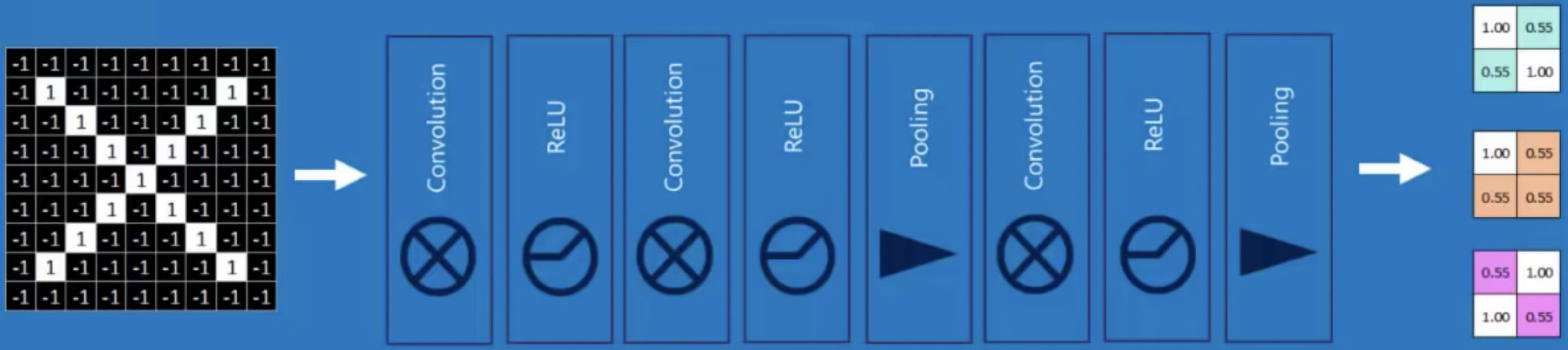
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

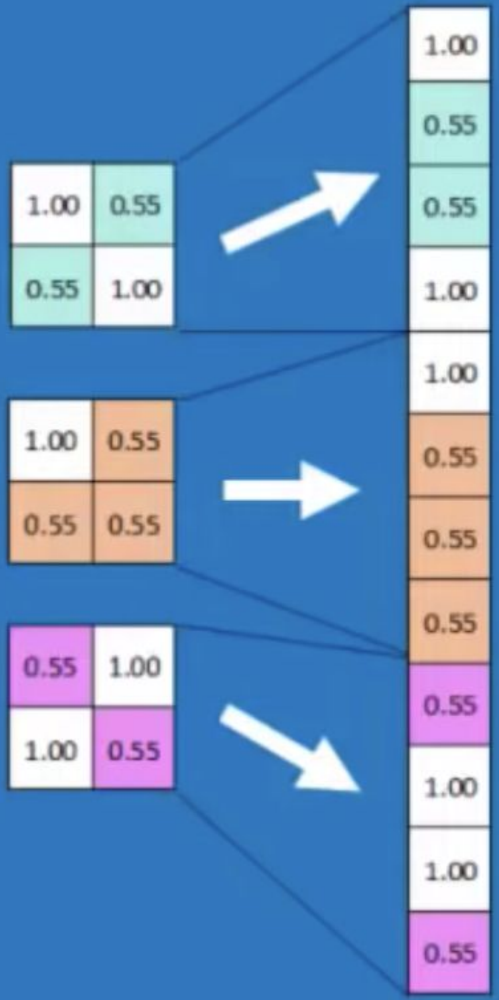
0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

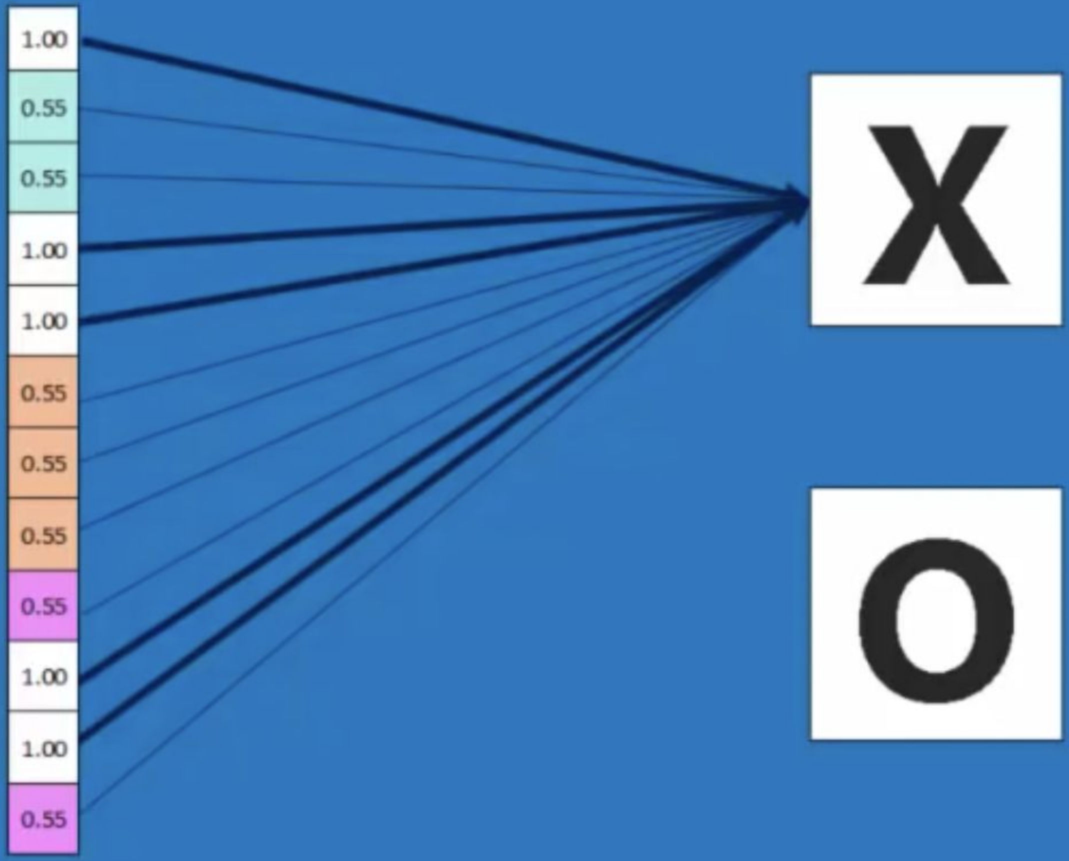
Chain it


Convolution
ReLU
Convolution
ReLU
Pooling
Convolution
ReLU
Pooling
1.00
0.55
0.55
1.00
1.00
0.55
0.55
0.55
0.55
1.00
1.00
0.55

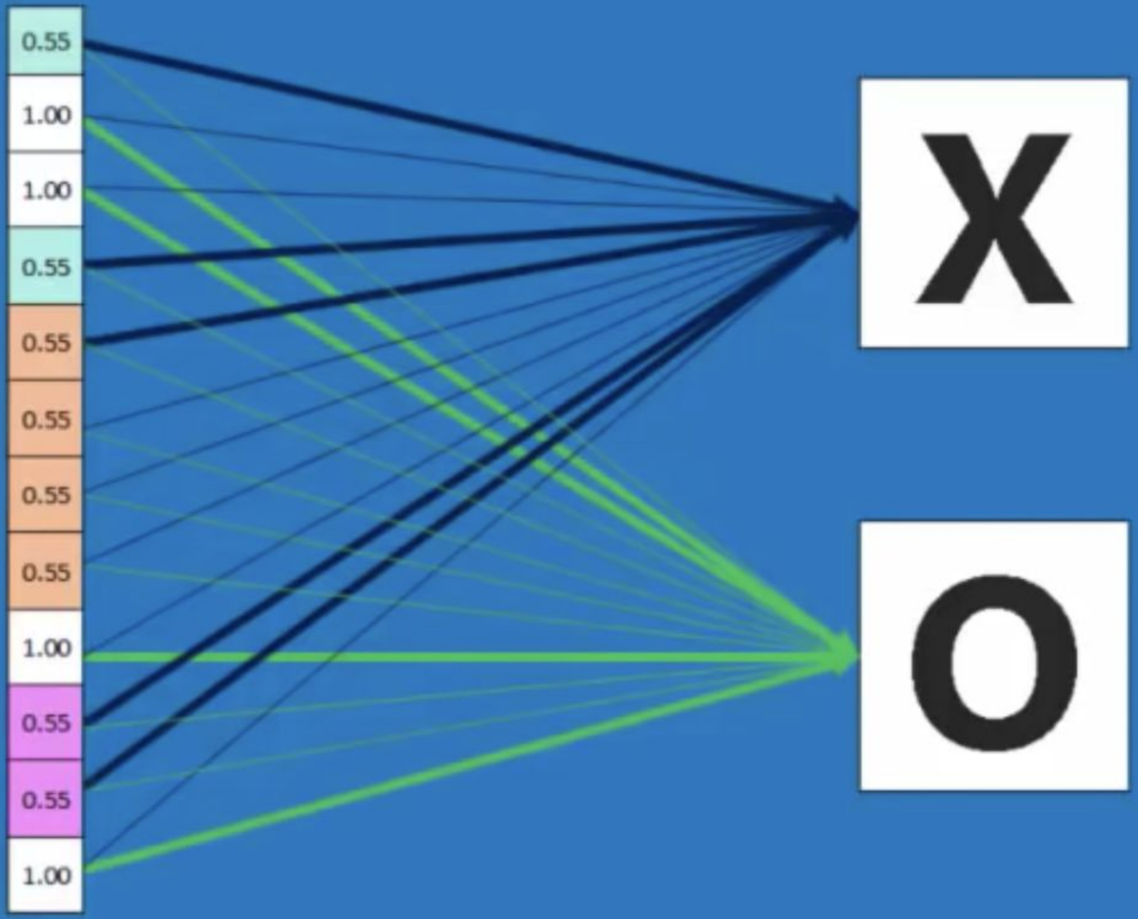
How to transition to fully connected



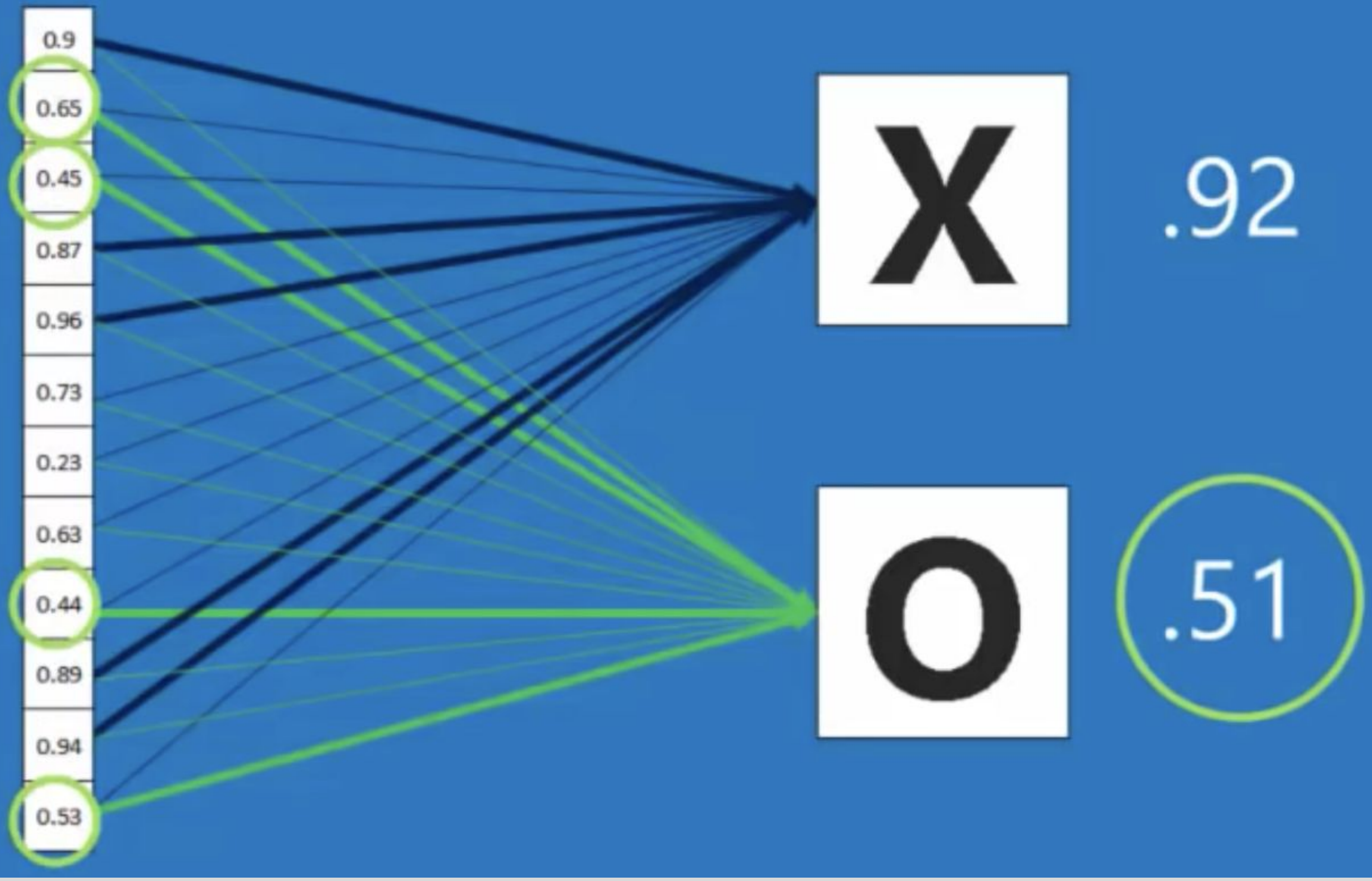
Fully connected read out



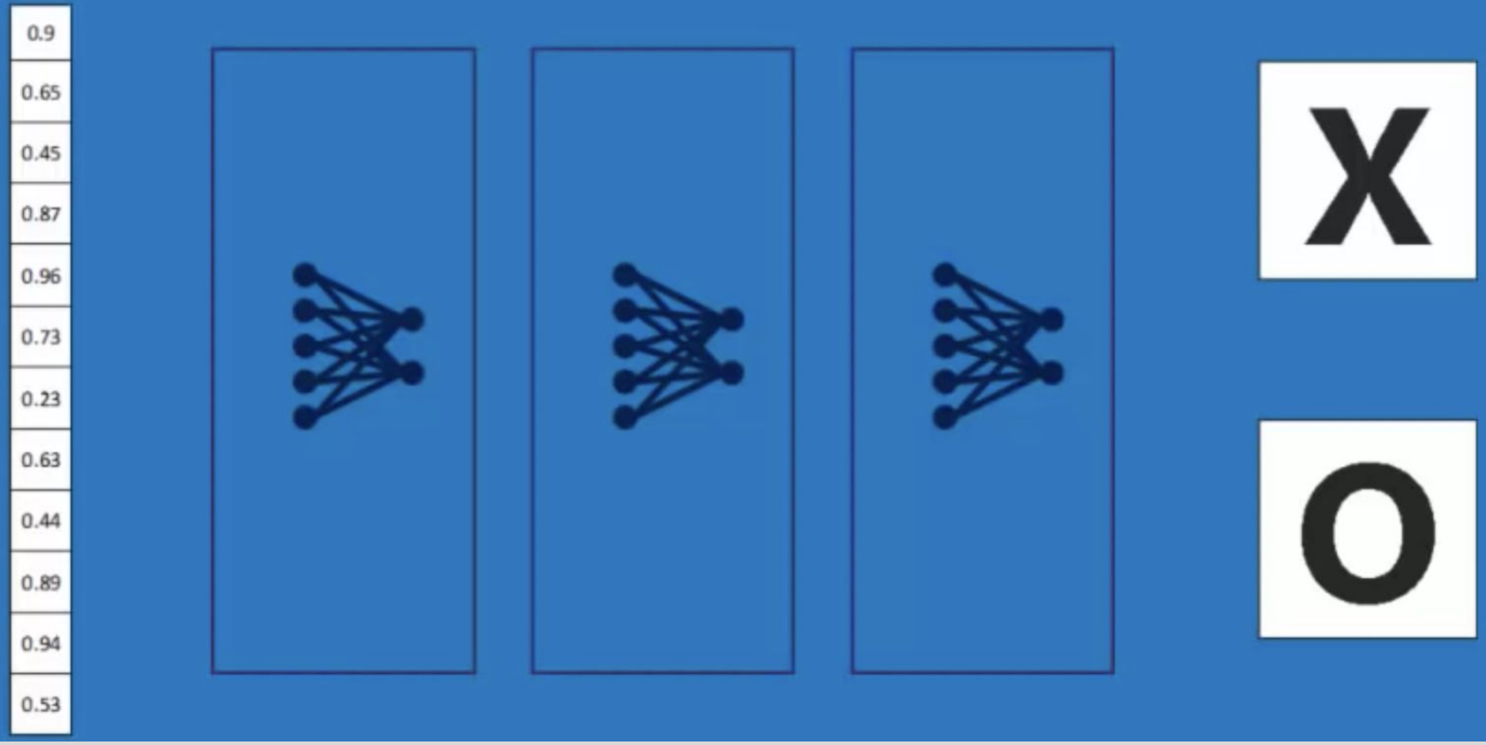
Predict both



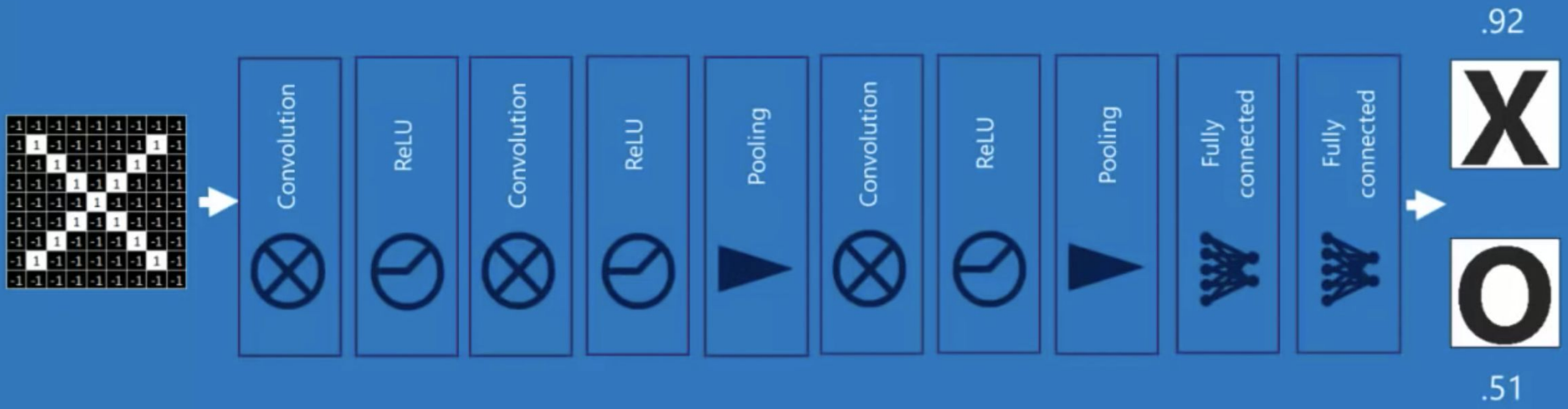
Get activations



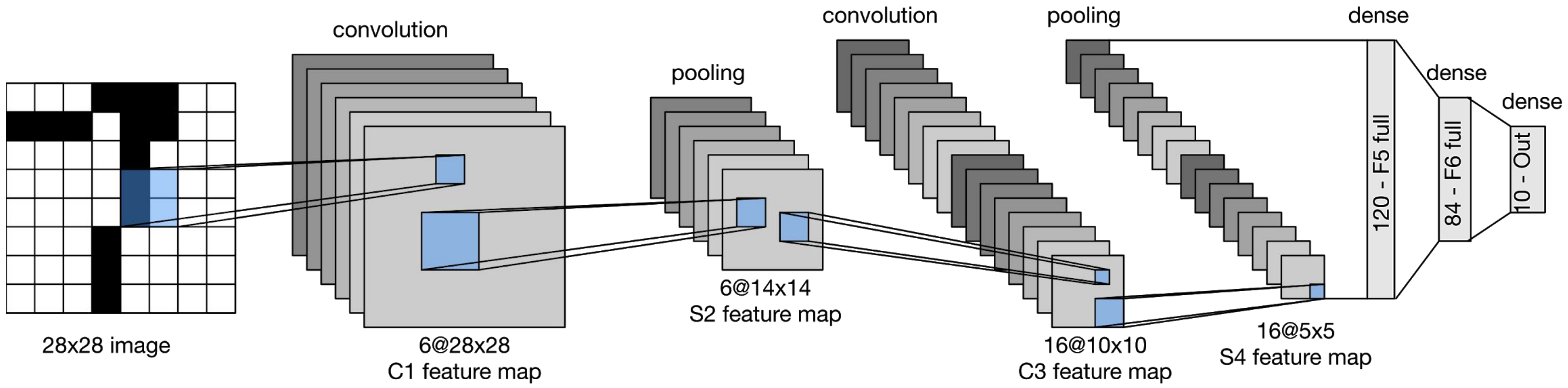
Multiple layers of fully connected



Now stack it all together



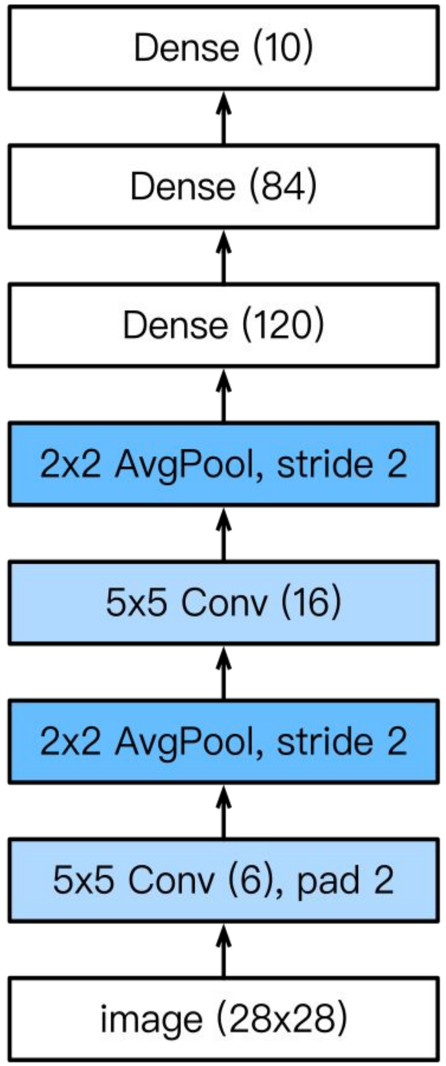
Putting things together



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

From ‘Dive into deep learning’

Another way of visualizing the same network

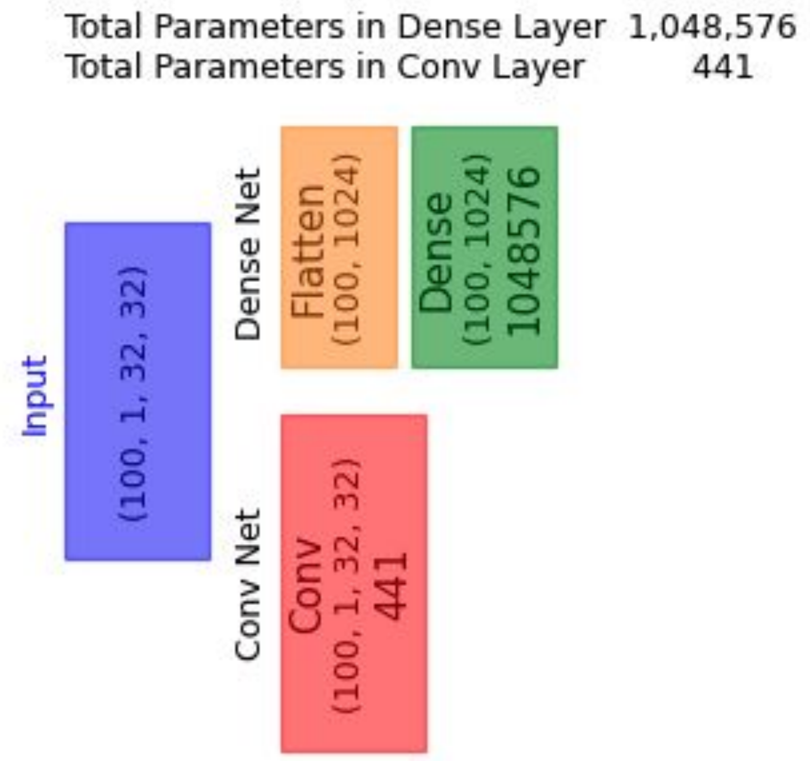


Let us calculate that: Convolution and pooling

- 1*1 image with 2 padding is 5*5
- 5*5 image, 5*5 filter => 1 output
- 300*400 image with 2 padding is 304*404
- 304*404 image, 5*5 filter =>300*400
- Maxpool 2x2 pooling window 300*400, stride 2 =150*200

Dense vs conv layers

- Now you'll compare the number of parameters in networks that use dense (fully connected) vs convolutional layers



Understand how hyperparameters affect the number of parameters

E 4.1 Explore how NN layers fit together, and how the architecture changes the # of params

Let’s implement a CNN!

neuromatch academy

Make a network

Now we're going to chain together the components of CNNs that we've been talking about.

- Convolution
- Convolution
- Pool Layer
    - Flatten the output here!
- Fully Connected Layer
- Fully Connected layer

Let’s train!

- Now we are going to put everything together and train

E 4.2 Train a CNN!

Section 5: Writing our own training loop

neuromatch academy

Dive into the data

neuromatch academy

The Data

- Fashion mnist

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

Before training, let us really understand the dataset

Visualize the dataset

Understand its properties

How hard do you think the task will be?

Which classes are likely the most confusable?

E 5.1 Get a feeling for the dataset

The training loop

neuromatch
academy

Now, we want you to design the training loop

In the last section you chained together the steps of a network.

Now, let's construct our own training loop, to really understand what's happening

What goes into a training loop

- Choose the correct criterion
    - We will use cross entropy

$$H(p,q) = -\sum_{x\in X} p(x)\log\left(q(x)\right)$$

What goes into a training loop

- Code up the training part
    - loss
    - calculating gradients
    - stepping forward
- Keep a track of the running loss i.e for each epoch we want to to know the average loss of the batches.
- We have already done the same for accuracy for you.

What goes into a training loop

- This exercise requires you to know the names of some important functions
- This is one exercise where it would be **ok to look at the solution if you get stuck**! As you're starting out, you might not know the names of particular functions. Don't worry about that! Just make sure you understand what is going on in the solution!

E 5.2 Build your training loop. Don't rush, take your time!

Section 6: Overfitting

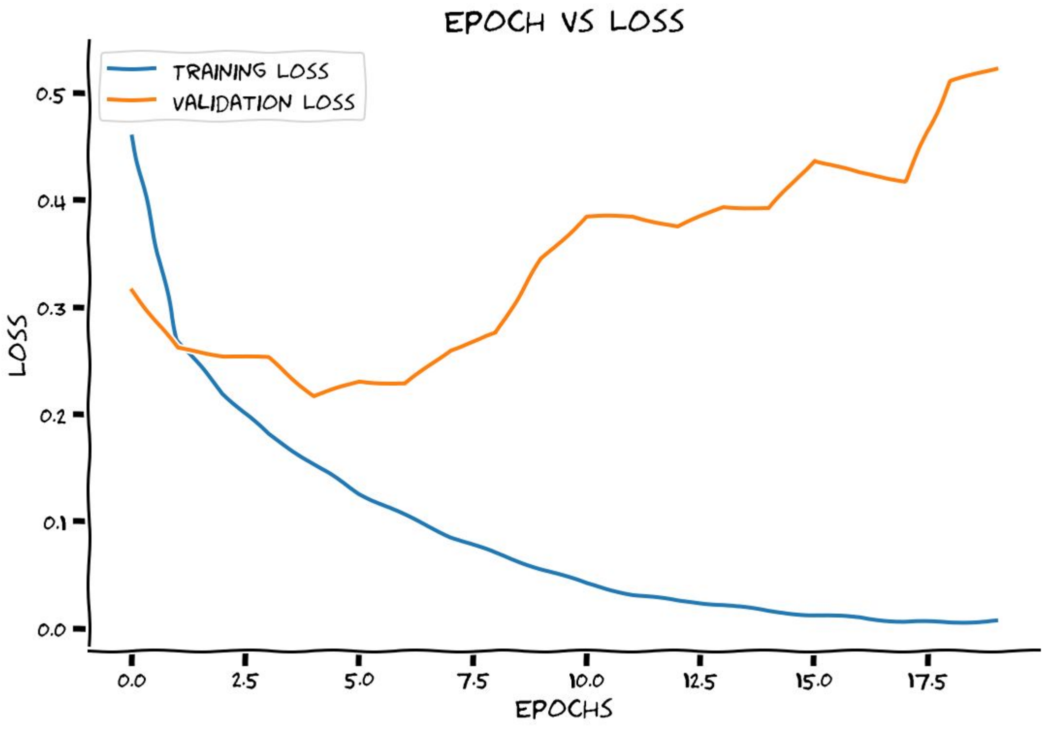
neuromatch academy

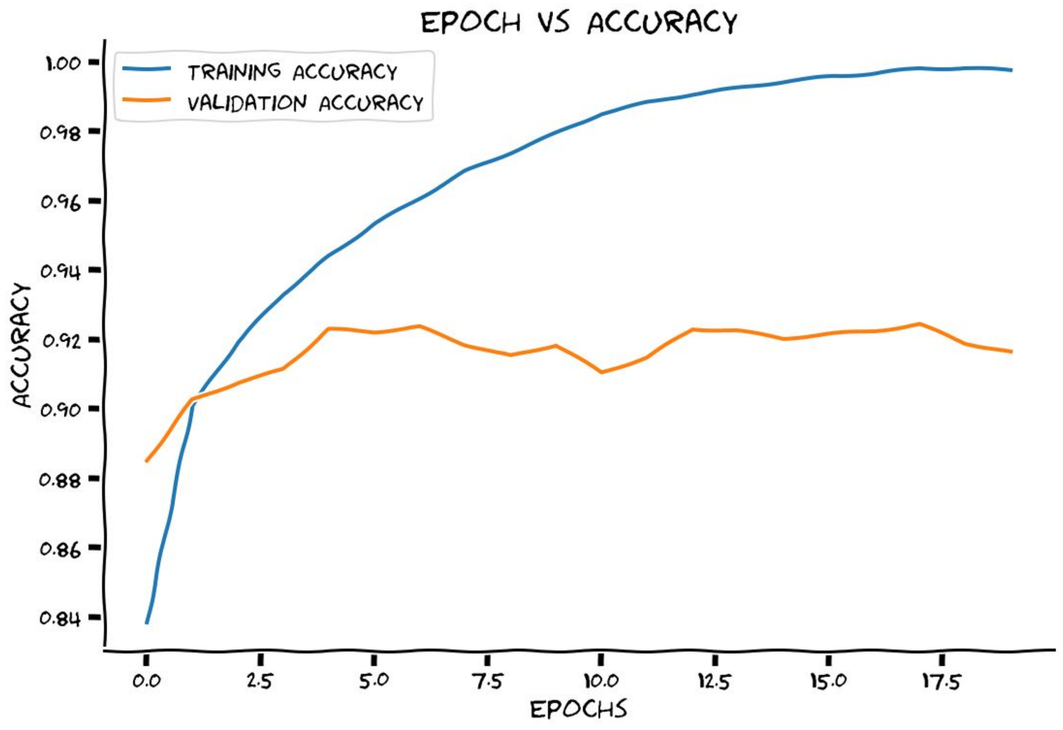
Dropout

neuromatch academy

Overfitting

In the model you just build there were strong signs of overfitting.





What can we do? Dropout

Intuition: no one neuron should be *too* important

Just like the brain, there should be redundancy in connections

Dropout: randomly set some of the hidden activations to 0

- Parameter: % of neurons to set to 0

What can we do? Dropout

Add drop-out

```
self.dropout1 = nn.Dropout(0.25)
```

```
x = self.dropout1(x)
```

Where do you have to do this?

Check how much dropout helps

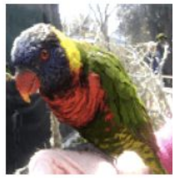
E 6.1 Add dropout. Retrain.

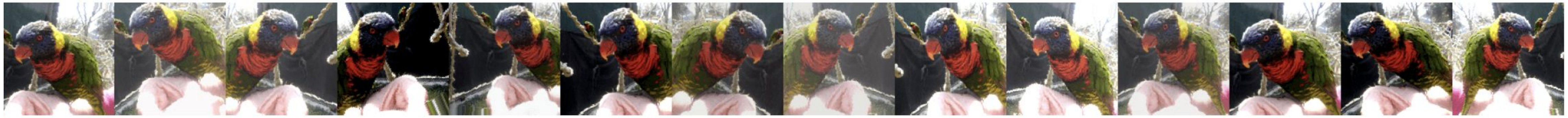
What happens to train and test performance?

Data augmentation

neuromatch academy

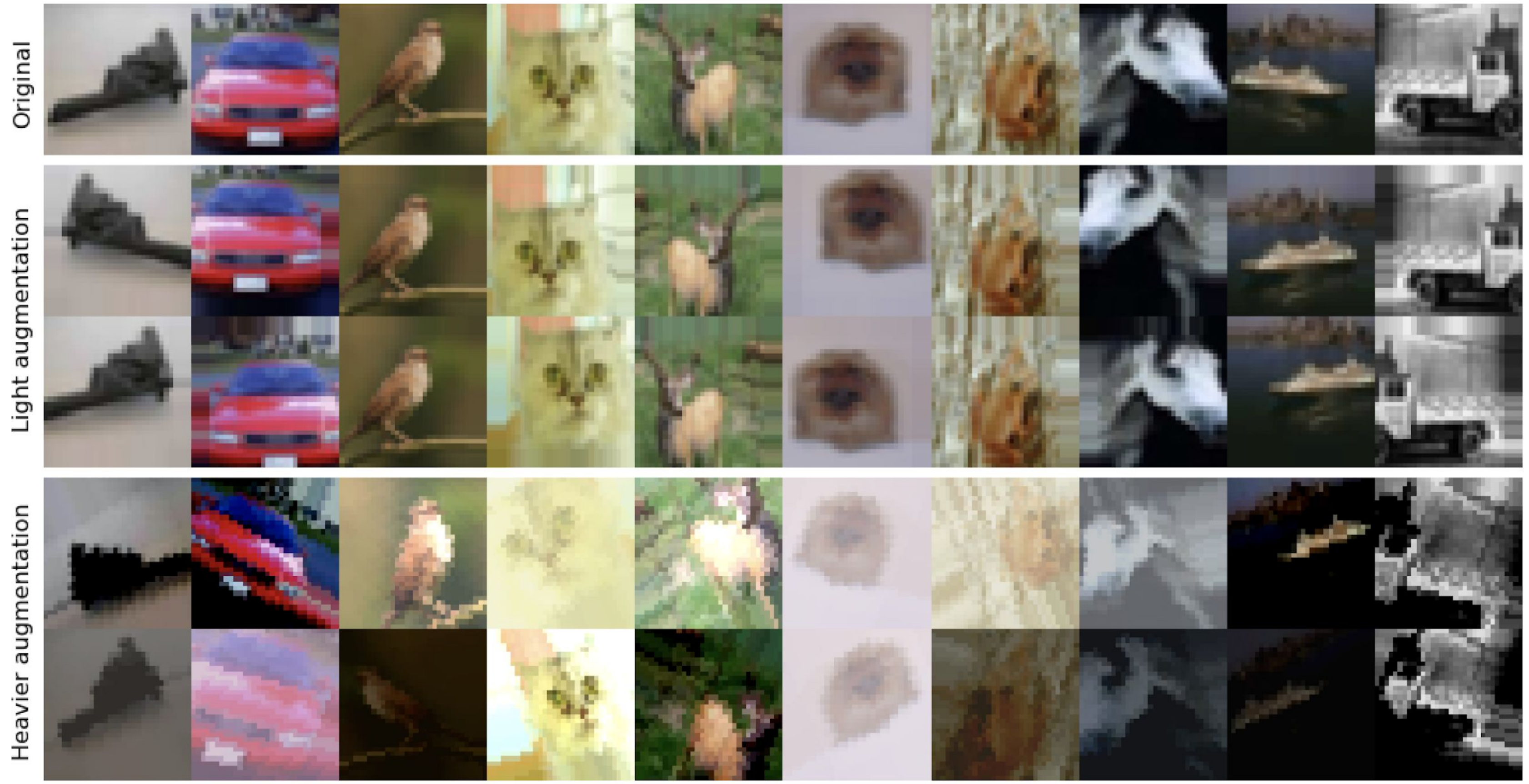
What can we do? Data augmentation





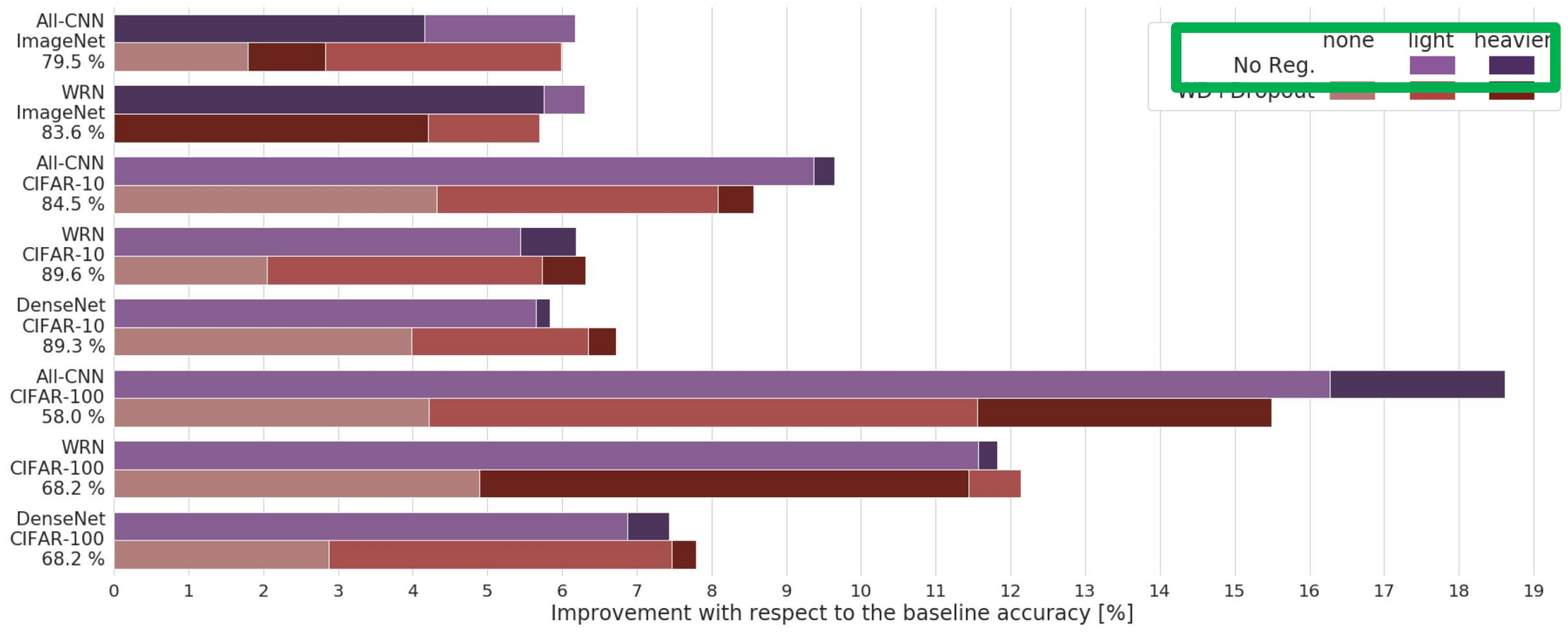
source: Hernandez Garcia Thesis

Different strengths of DA



source: Hernandez Garcia, arxiv

Data augmentation is often more important than direct regularization



source: Hernandez Garcia, arxiv

Data augmentation

Now edit the training loop.

We want:

- Resize
- Flip
- Rotations (<20 degrees)
- Color jitter

E 6.2 How much does data augmentation help?