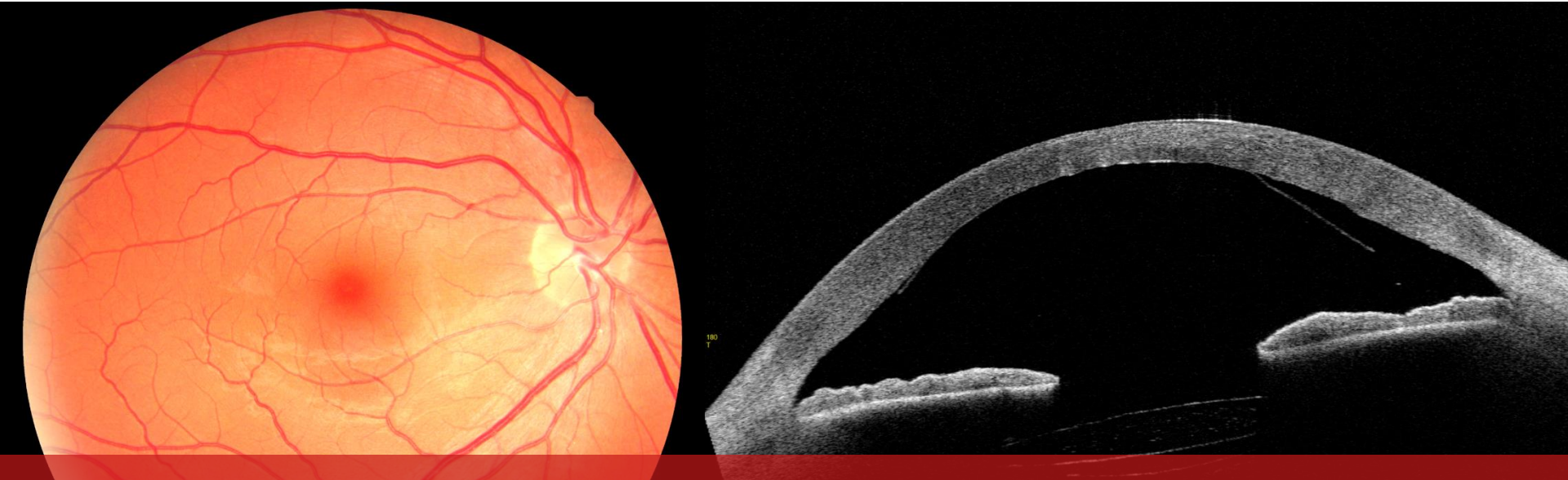


Convolutional Neural Networks (8DC00)

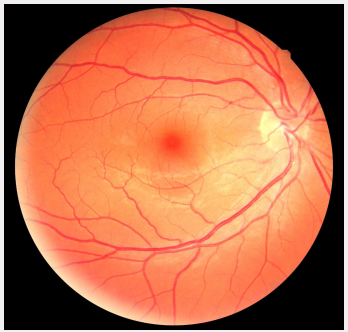
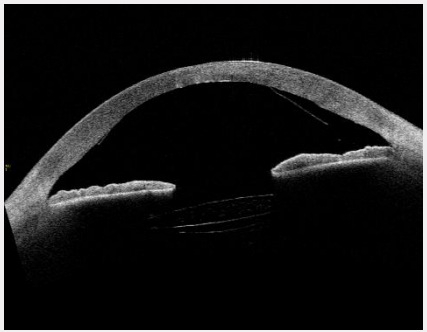
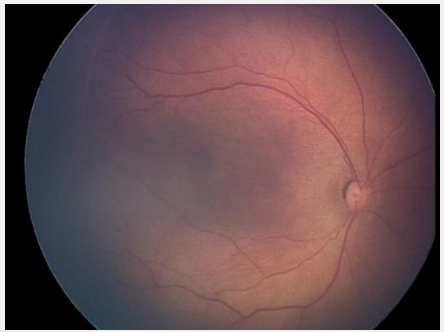
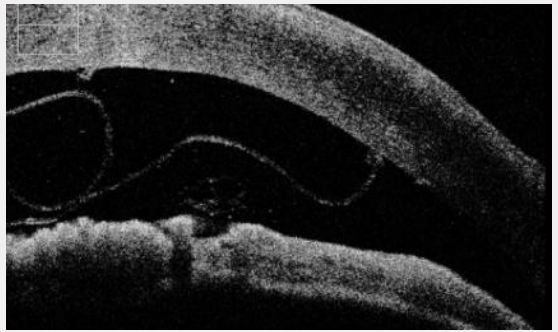
Friso G. Heslinga

Friso G. Heslinga MSc.

Background: BSc and MSc in Biomedical Engineering, MSc in Health Sciences (University of Twente).

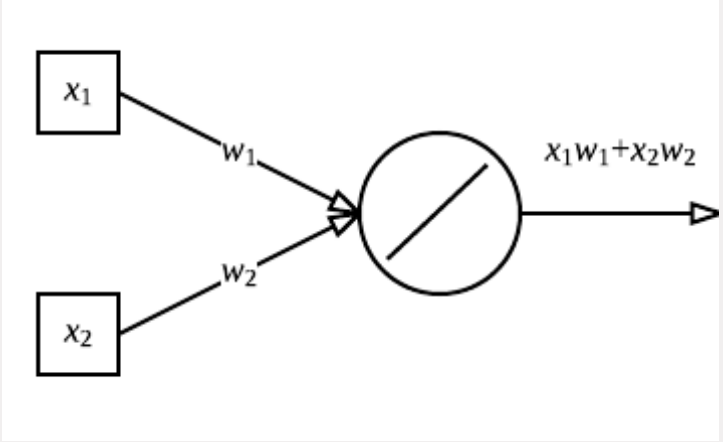
Work/internship experience: University of Western Australia, University of California - Berkeley, Harvard Medical School

PhD Research: Deep learning, Medical image analysis, Ophthalmology

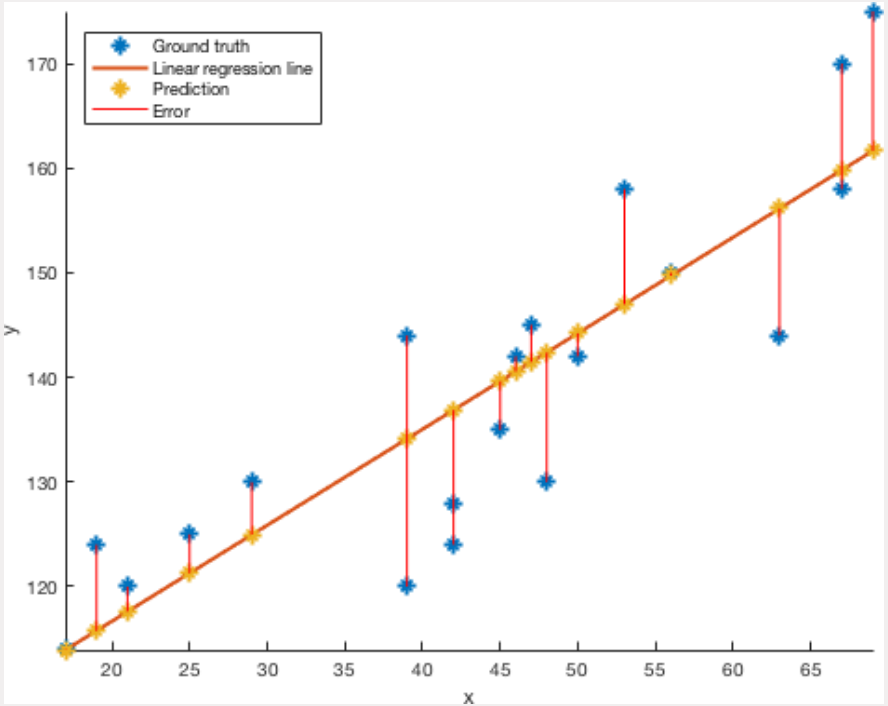





Week	Date	Lecturer	Topics
1	1 Sept.	Maureen	Course introduction; Software demo; Image registration (1)
	3 Sept.	Maureen	Image registration (2); Geometrical transformations
2	8 Sept.	Maureen	Point-based registration
	10 Sept.	Maureen	Intersity-based registration; Evaluation metrics
3	15 Sept.	*Catch-up day (no lecture)*	
	17 Sept.	Cornel Zachiu (UMCU)	Guest lecture 1: Image analysis for adaptive radiotherapy
4	22 Sept.	Mitko	Introduction to CAD; k-NN; Decision trees
	24 Sept.	Mitko	Generalization and overfitting
5	29 Sept.	Mitko	Logistic regression; Neural networks
	1 Oct.	Friso	Convolutional neural networks
6	6 Oct.	Friso	Deep learning frameworks and applications
	8 Oct.	Friso	Unsupervised machine learning
7	13 Oct.	Maureen	Deep learning for deformable image registration
	15 Oct.	Geert-Jan Rutten (ETZ)	Guest lecture 2: Image analysis in neurosurgery applications
8	20 Oct	*Self-study (no lecture)*	Active shape models
	22 Oct	*Self-study (no lecture)*	Active shape models

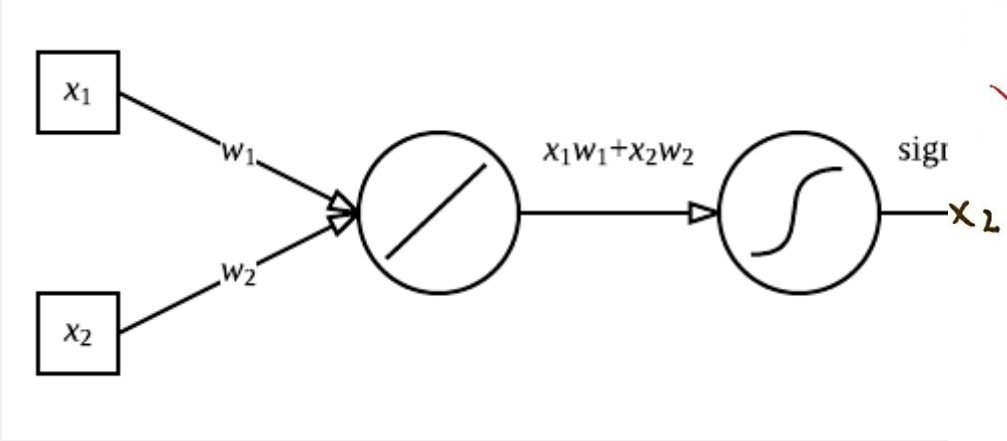
Previously – Linear regression

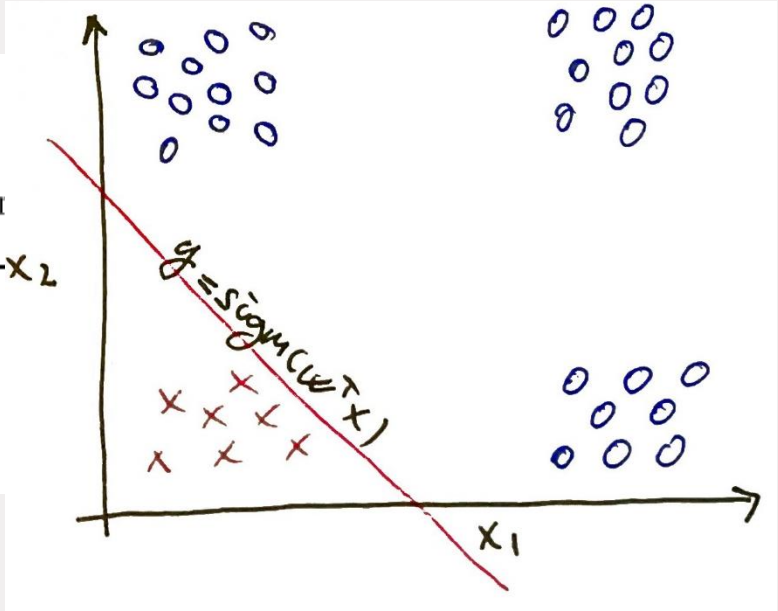


$$\hat{y} = \theta^T \mathbf{x}$$



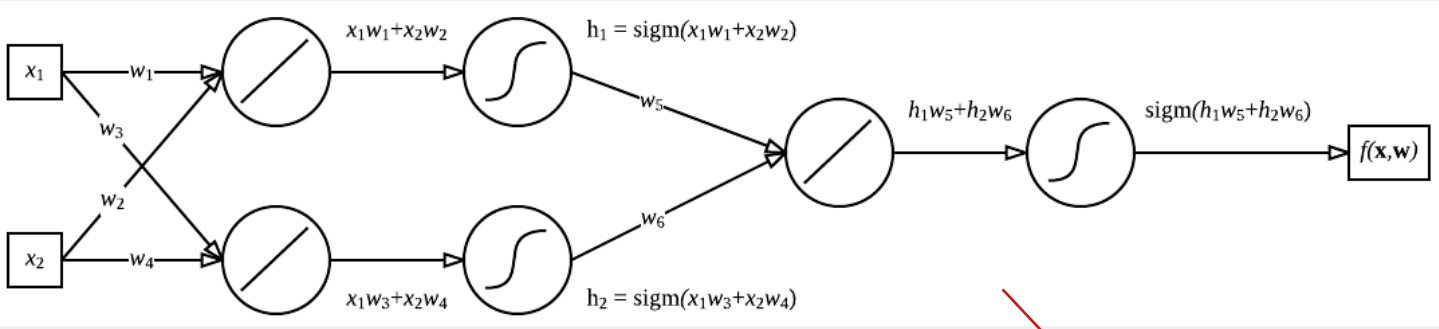
Previously – Logistic regression

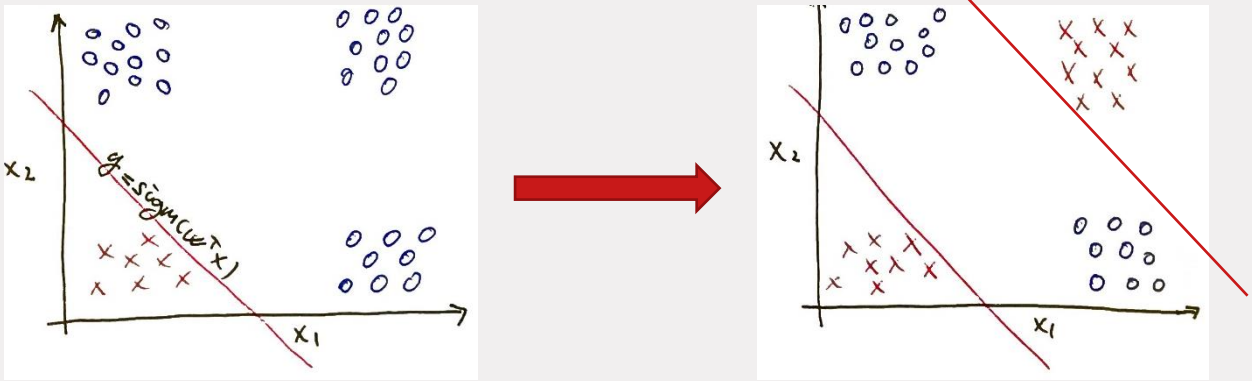




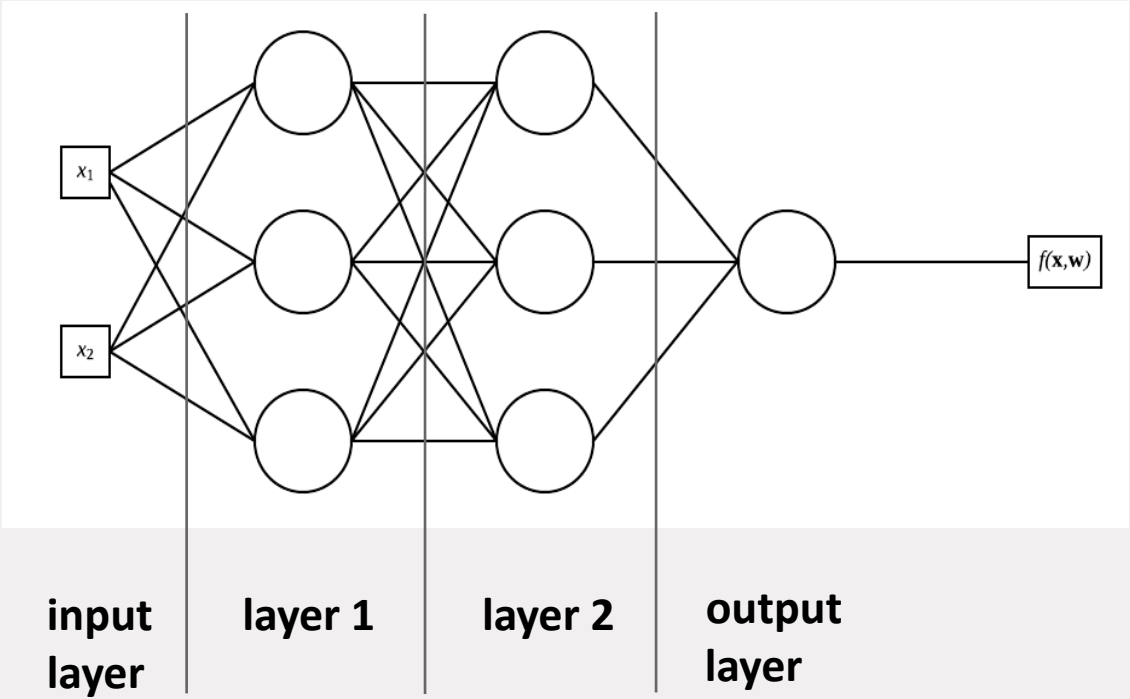
$$p(y = 1 \mid \mathbf{x}) = \sigma(\boldsymbol{\theta}^\intercal \mathbf{x})$$

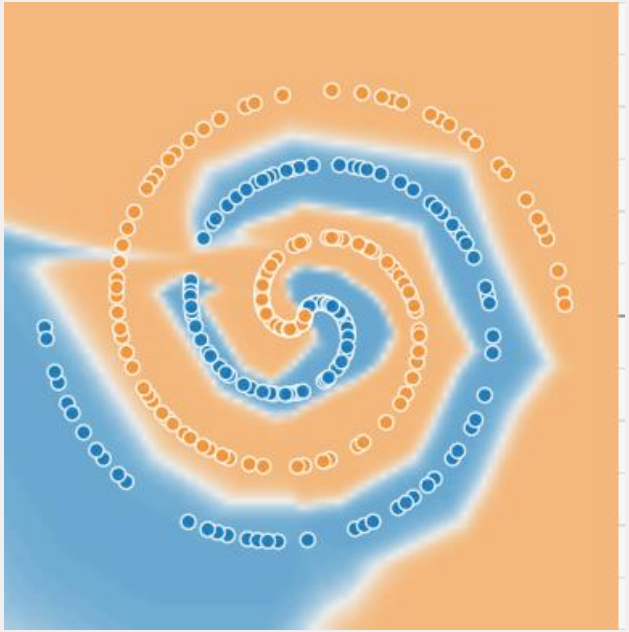
Previously – Neural networks





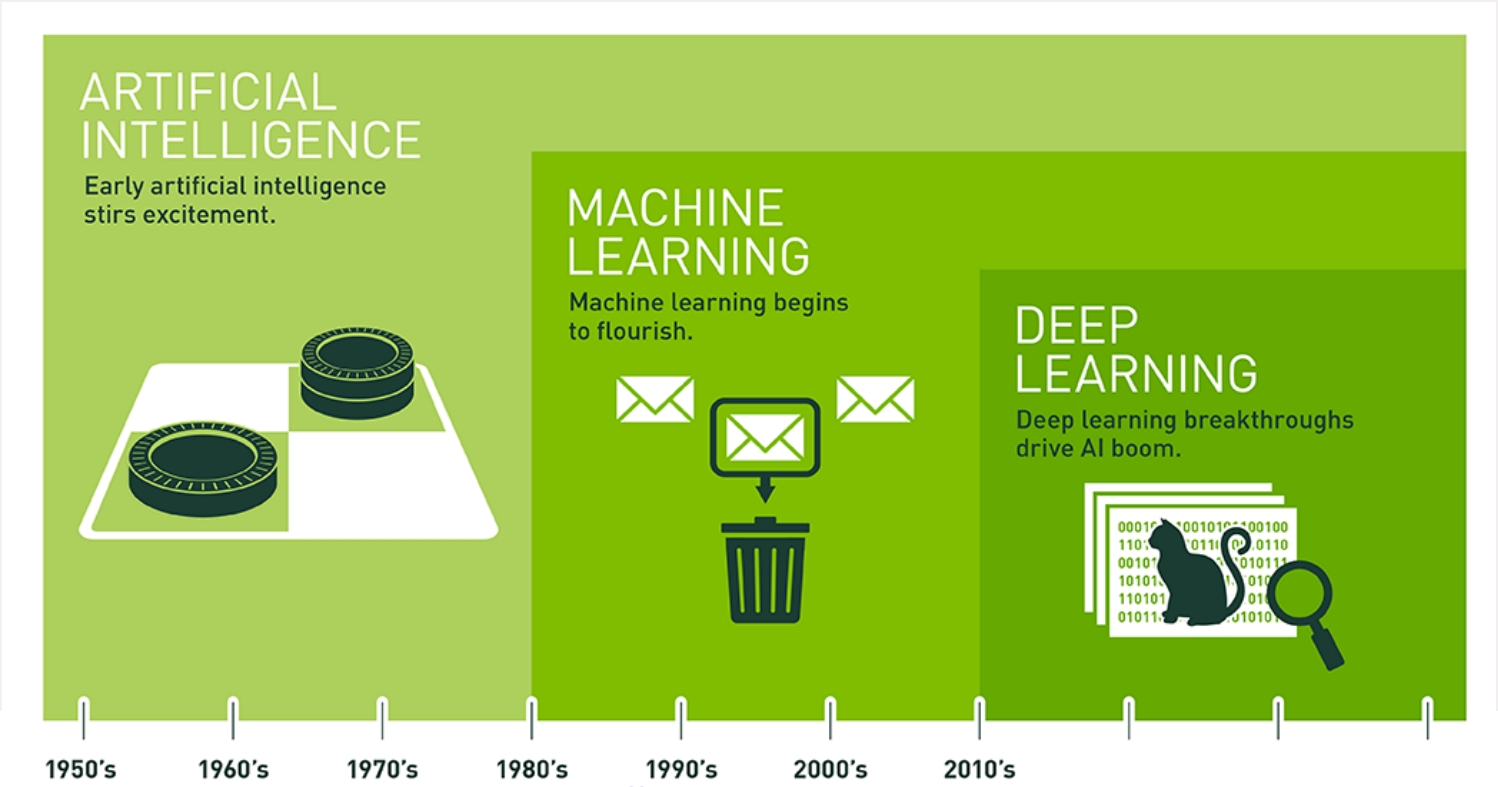
Previously – Neural networks





Previously – Machine learning

Figure source: nvidea.com



Today: Convolutional neural networks (CNN)

- Neural networks → Convolutional neural networks

Building blocks for deep learning models for image analysis:

- Convolutional layer
- Max-pooling layer

- Not needed for the project, but will be on the exam

Learning outcomes

- Student can explain the concept of **convolutions** in a neural network
- Student can describe why we can use a **convolutional approach** for (medical) **images**
- Student can explain why convolutions enable development of **deep** (and large) **neural networks**
- Students can explain and apply the **max-pooling layer** in a convolutional neural network
- Students can motivate the choice for a **kernel size**

Lecture outline

- Images as input to neural network
- Reducing # of weights
- 1D convolutions
- 2D convolutions
- *Break (15 mins)*
- Kernels
- Max-pooling
- Interactive example

Images as input to neural networks

24 pixels

24 pixels

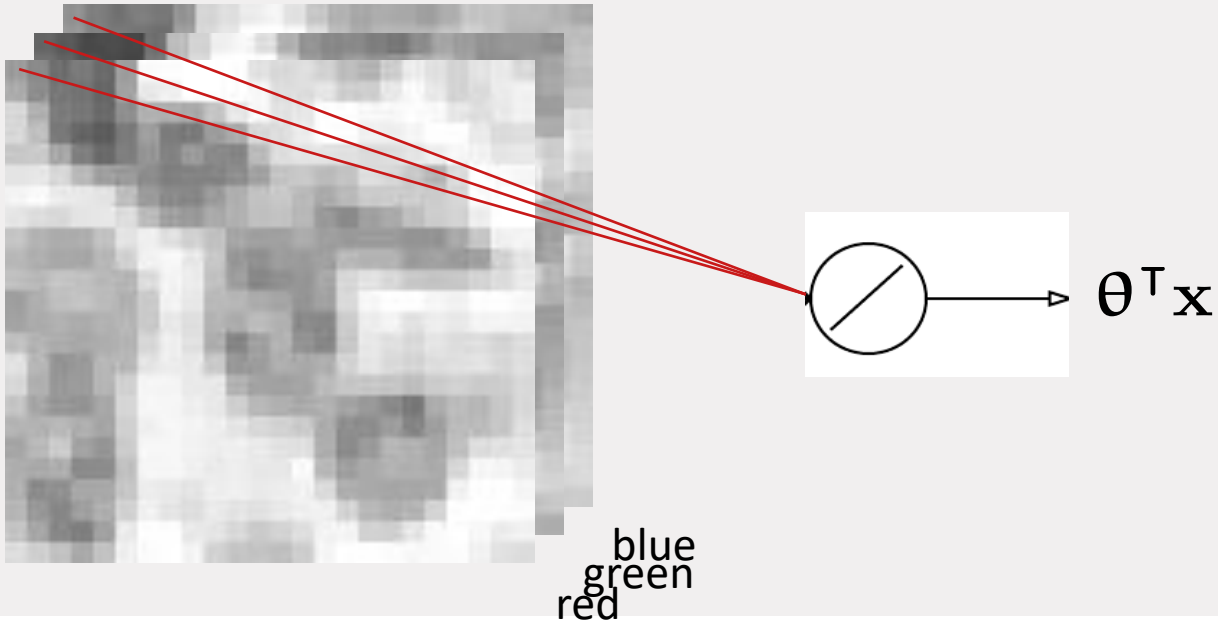
3 color channels
(red, green, blue)

= 1728 features ($\mathbf{x}_i$ are 1728 dimensional vectors)

If we train linear regression with these inputs (such as in the first practical), we will have 1728 weights w and a bias b.

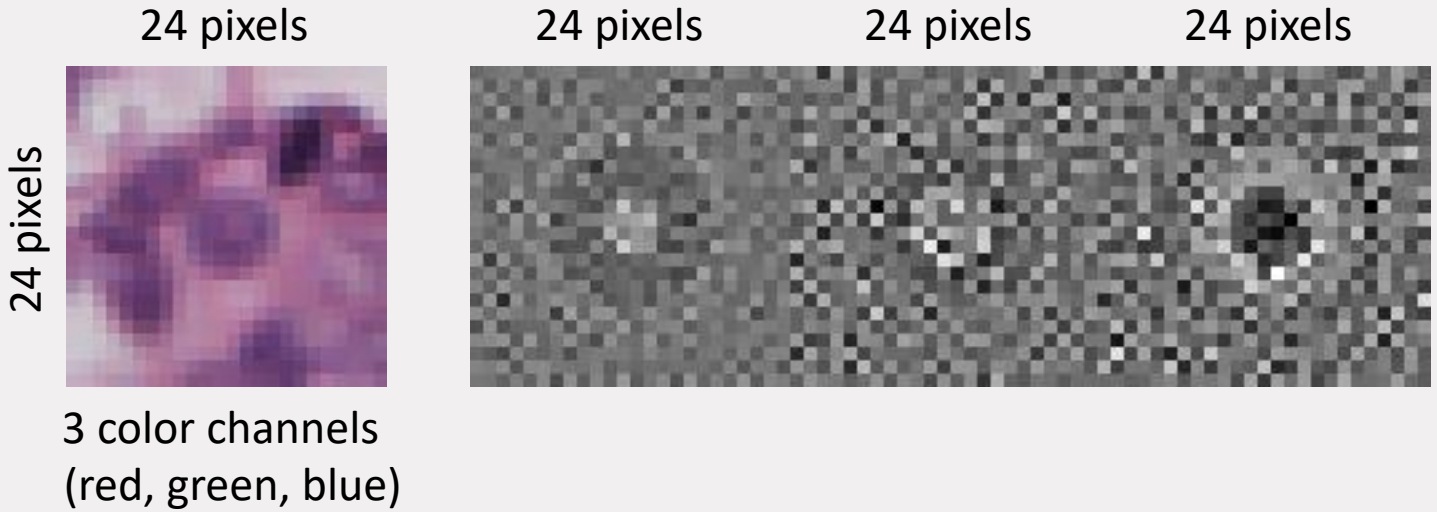
Every pixel is an input

Every pixel from every color channel is multiplied by a weight



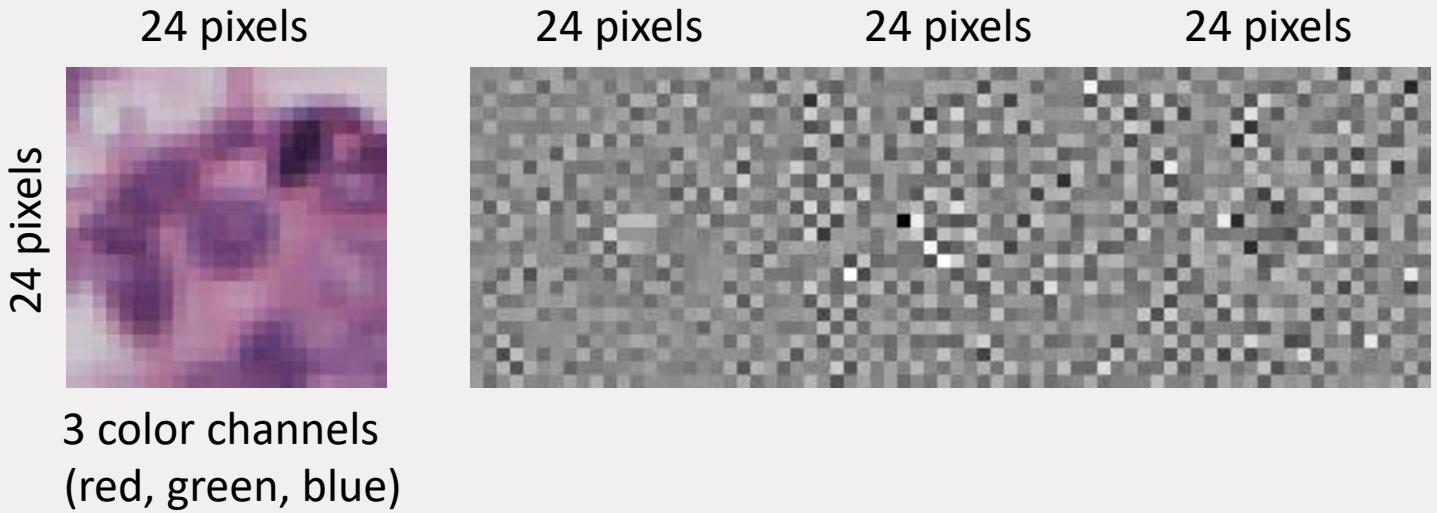
Post-training: visualizing what model has learned

Reshape vector of parameters into 24x24x3 image



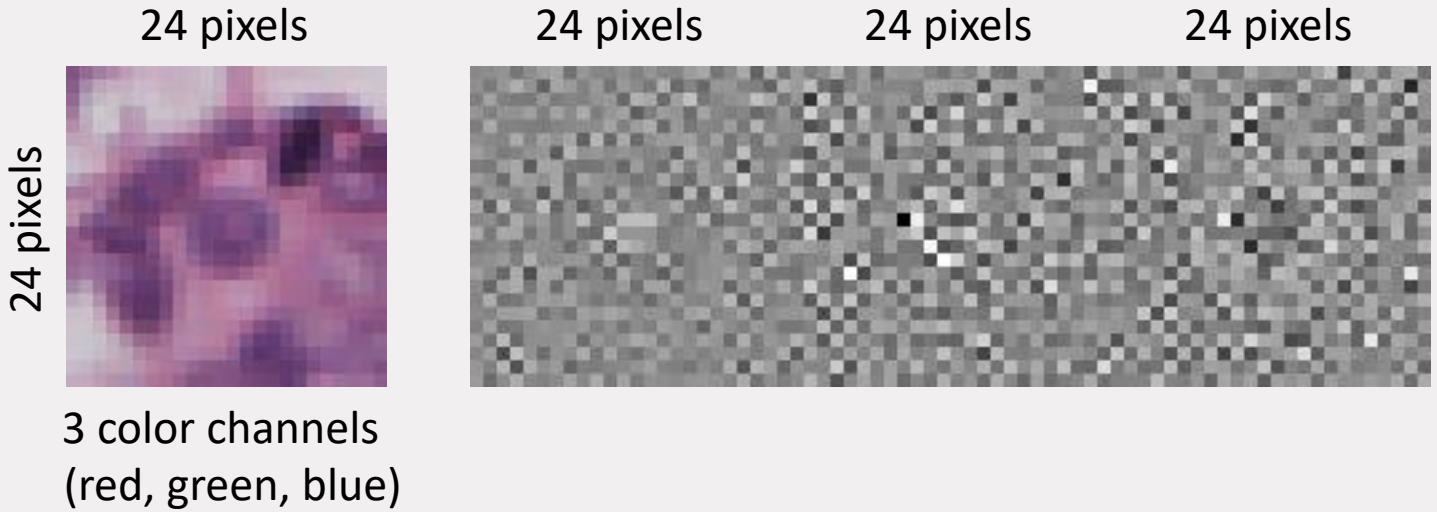
Post-training: visualizing what model has learned

Only 25% of training samples.. Looks noisy!

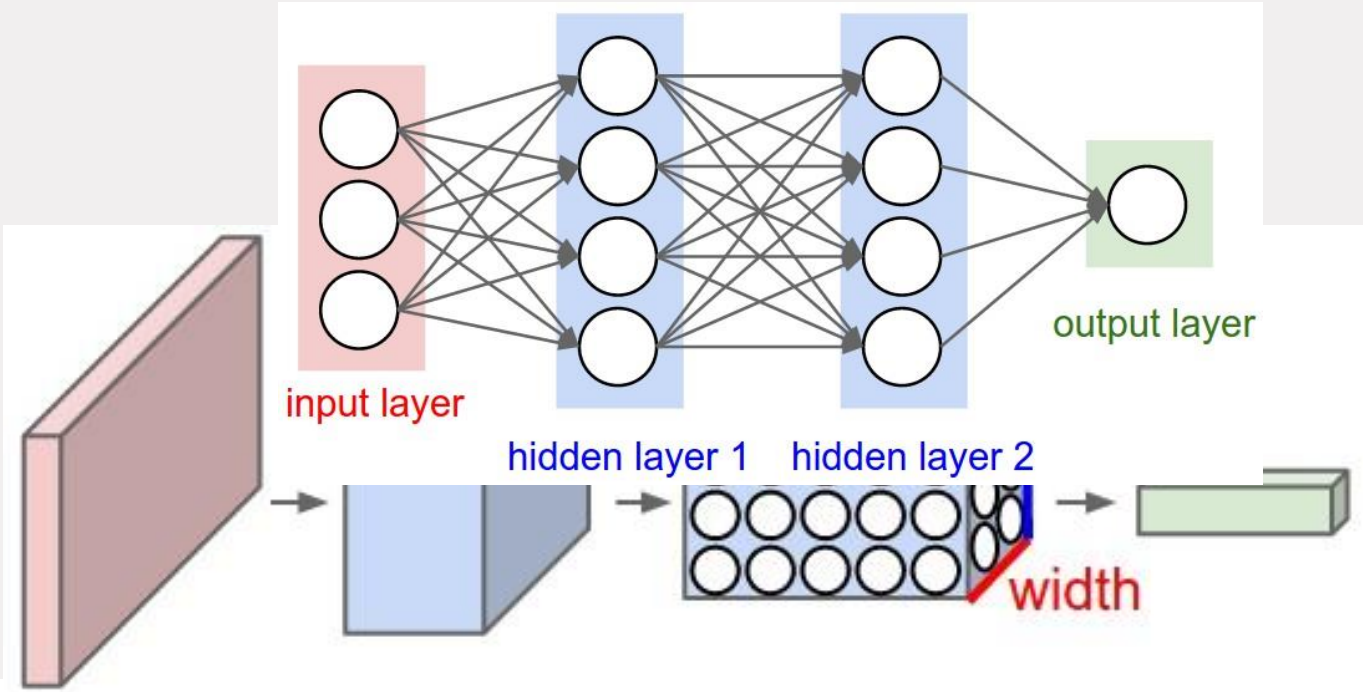


Post-training: visualizing what model has learned

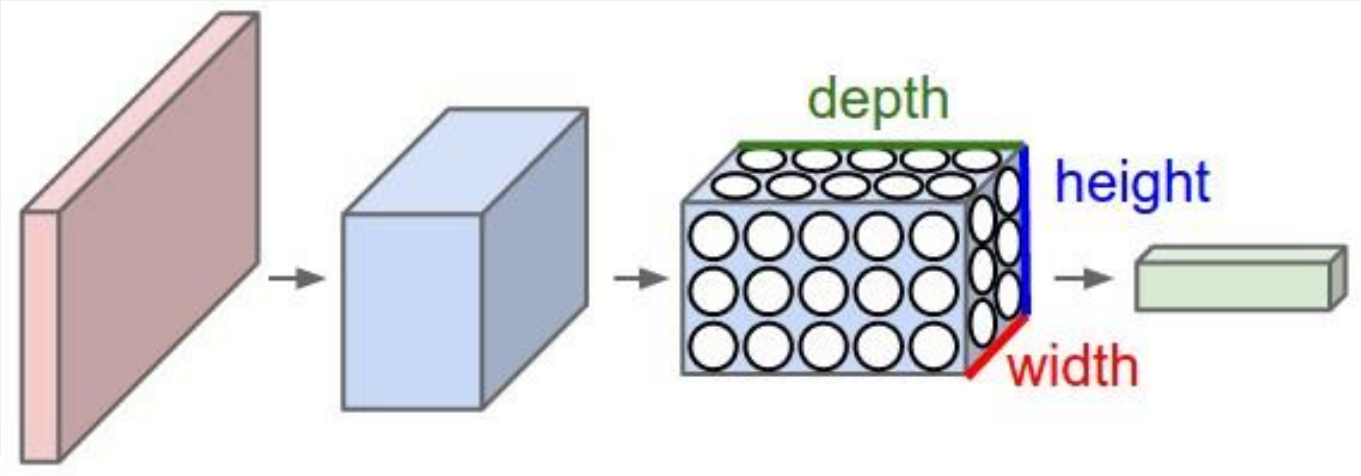
You can think of it as “there is not enough training data to reliably estimate all model weights”.



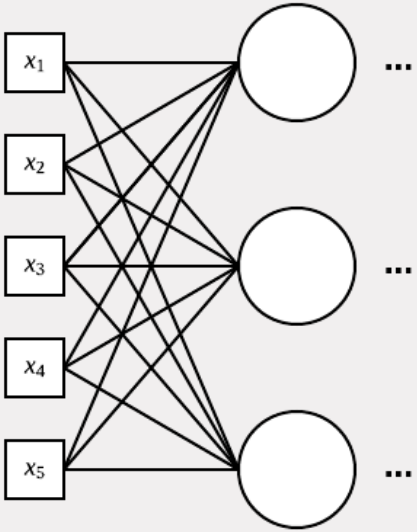
Many weights



With large inputs such as images and deep networks, the number of weights ”explodes”. We need a way to reduce the number of weights, without sacrificing performance.

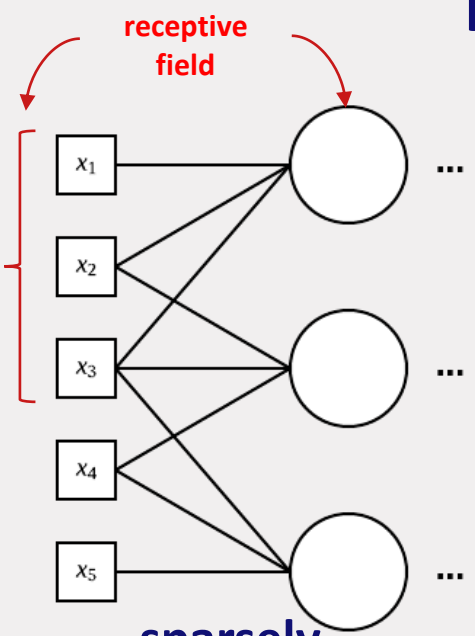


How many weights?



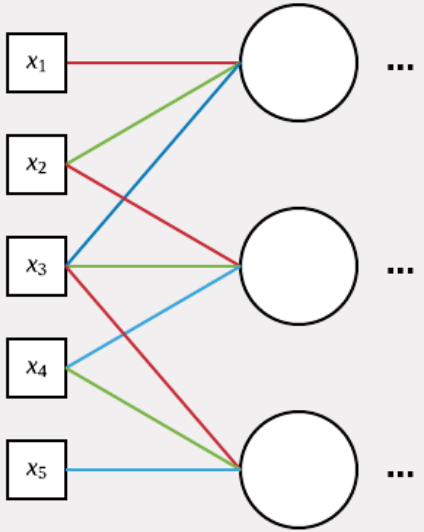
“regular” NN

15 weights



sparsely connected NN

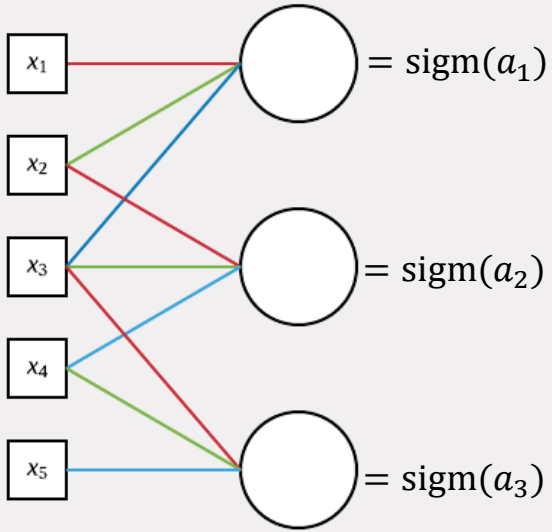
9 weights



shared weights

3 weights

Shared weights



$$a_1 = x_1 w_1 + x_2 w_2 + x_3 w_3$$
$$a_2 = x_2 w_1 + x_3 w_2 + x_4 w_3$$
$$a_3 = x_3 w_1 + x_4 w_2 + x_5 w_3$$

$$[a_1 \quad a_2 \quad a_3] = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] * [w_1 \quad w_2 \quad w_3]$$

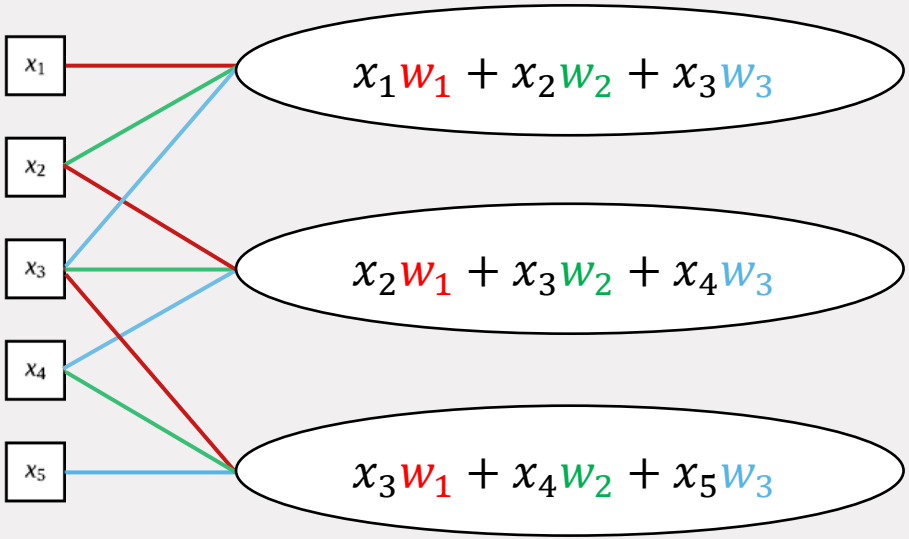
shared weights

3 weights

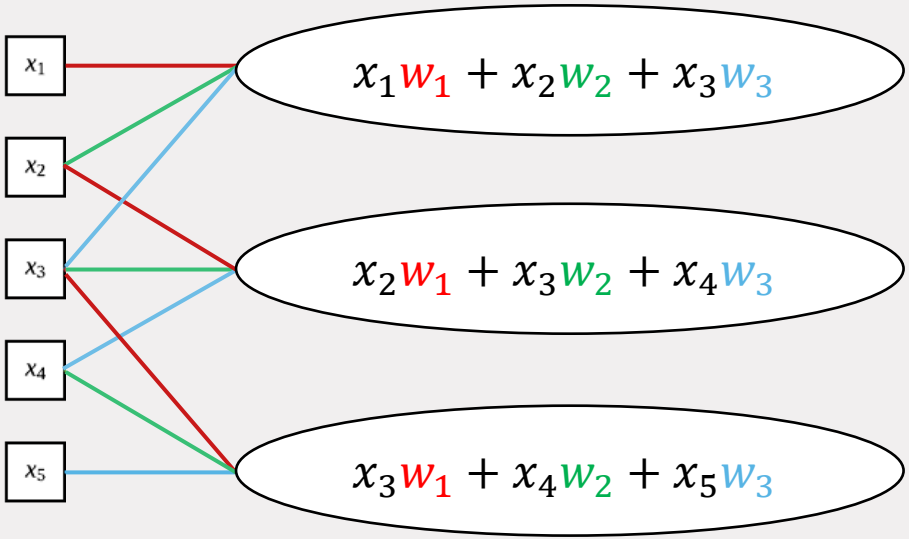
convolution, thus convolutional NN

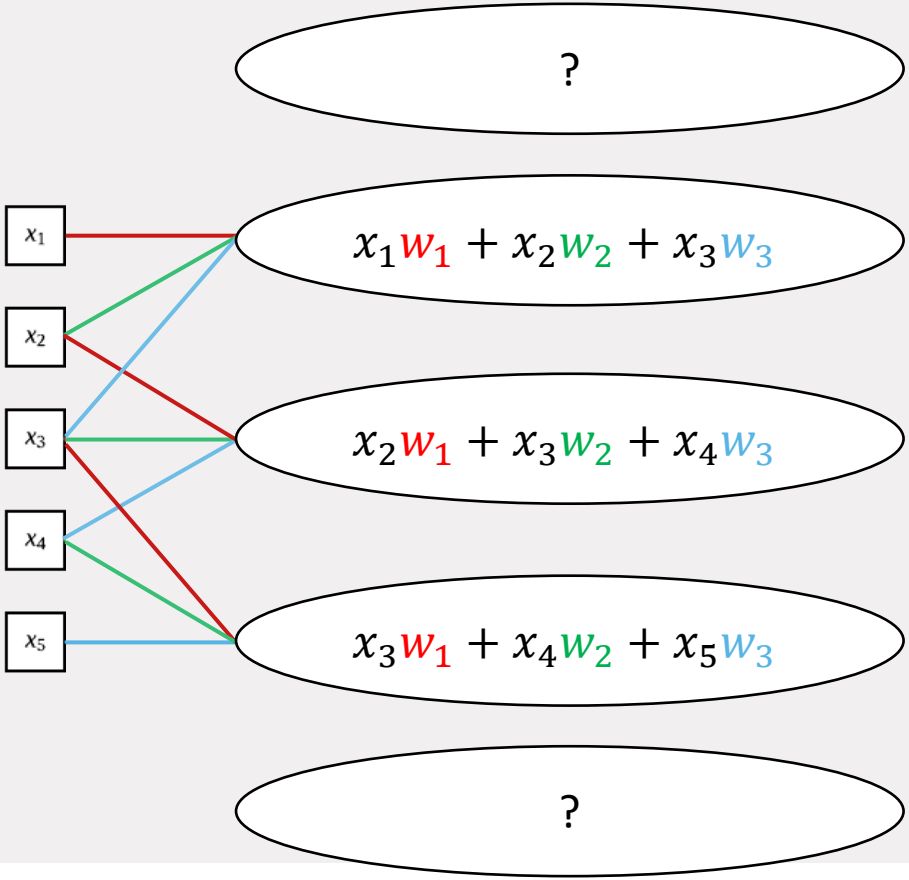
TU/e

1-D Convolution



1-D Convolution

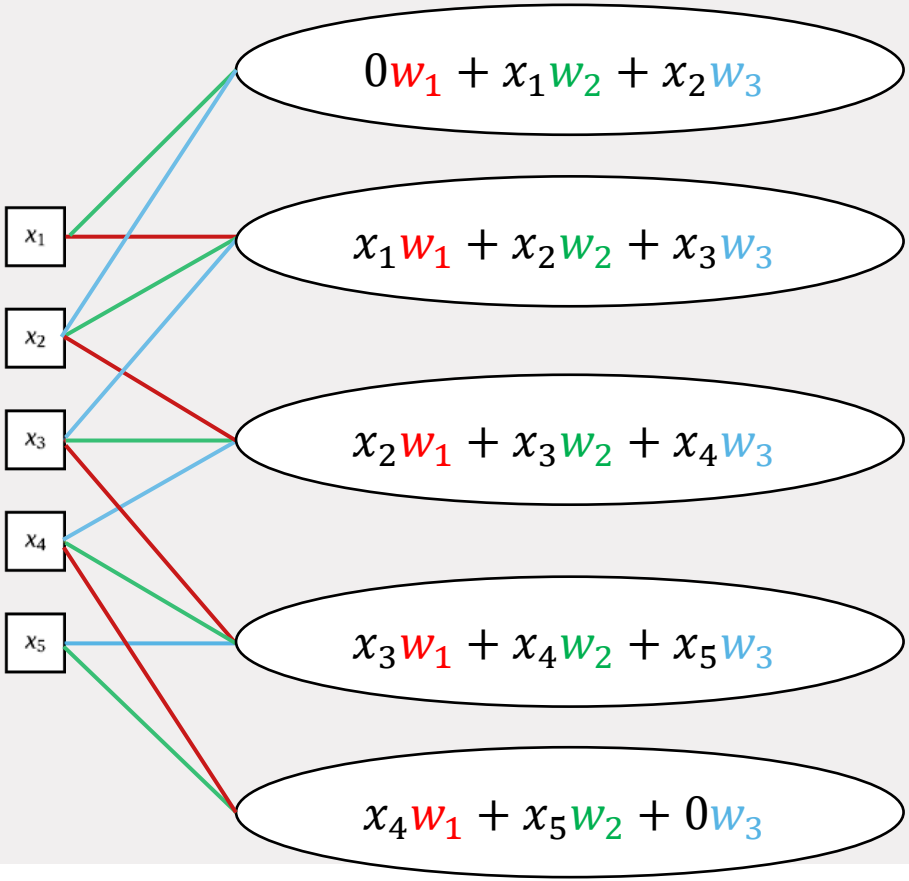




1-D Convolution

How can we keep the same number of features in hidden layer 1?

TU/e

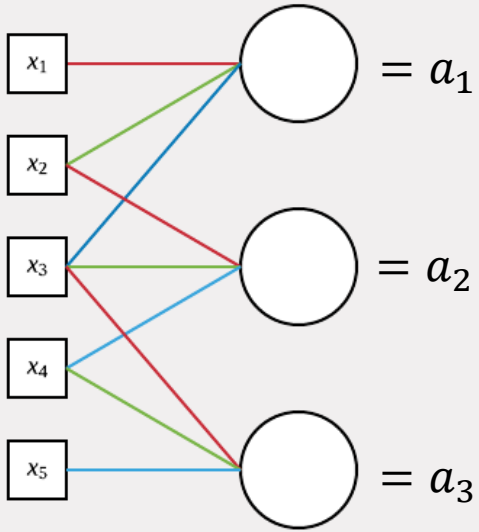


1-D Convolution

Zero-padding!

TU/e

1-D Convolution



$$a_1 = x_1 w_1 + x_2 w_2 + x_3 w_3$$

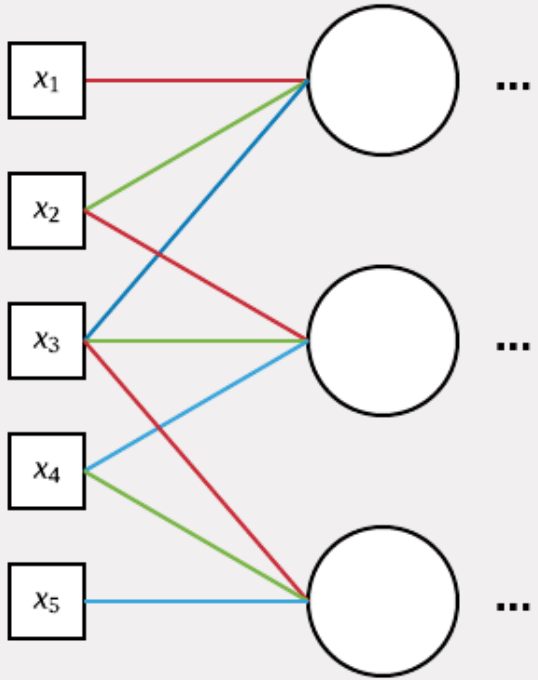
$$a_2 = x_2 w_1 + x_3 w_2 + x_4 w_3$$

$$a_3 = x_3 w_1 + x_4 w_2 + x_5 w_3$$

$$[a_1 \quad a_2 \quad a_3] = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] * [w_1 \quad w_2 \quad w_3]$$

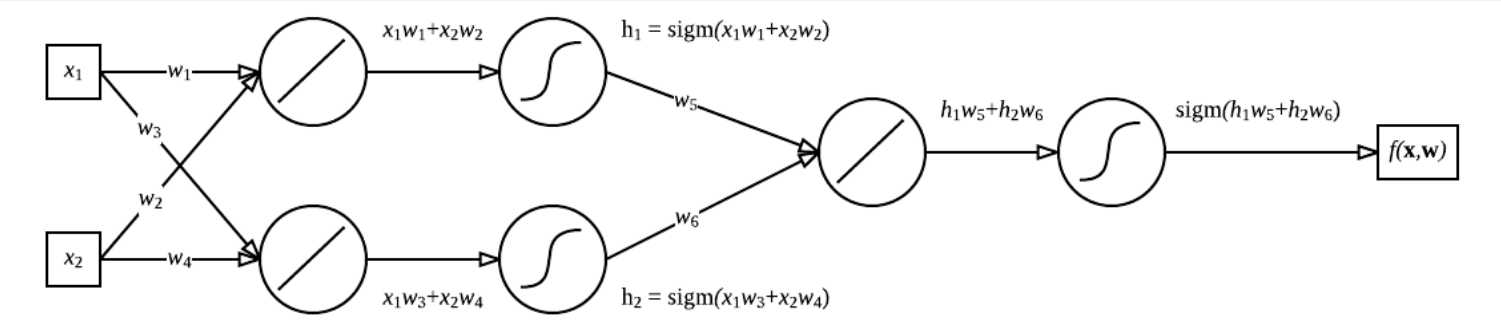
Properties of convolutional neural networks

- Sparse connectivity
- Weight sharing
- Parallel computations



Why does this work?

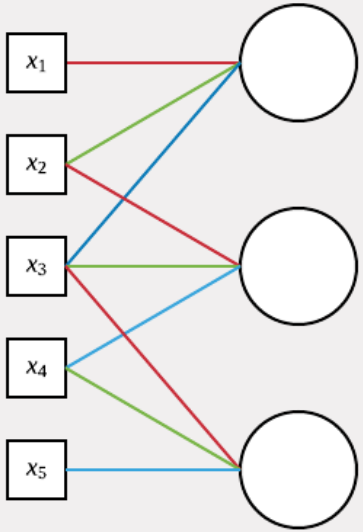
- Multiple layers
- Hidden layers contain features calculated from previous layers



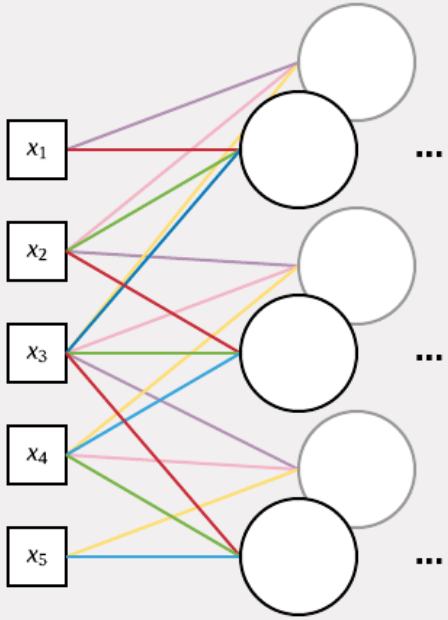
Why does this work?

- Different layers contain different transformation
  - Simple (e.g. edges, colors)
  - Complex (final layers)

One added benefit is that the learned transformations will be equivariant with translation (if the features/image is shifted up/down the features will still be detected).



shared weights
3 weights

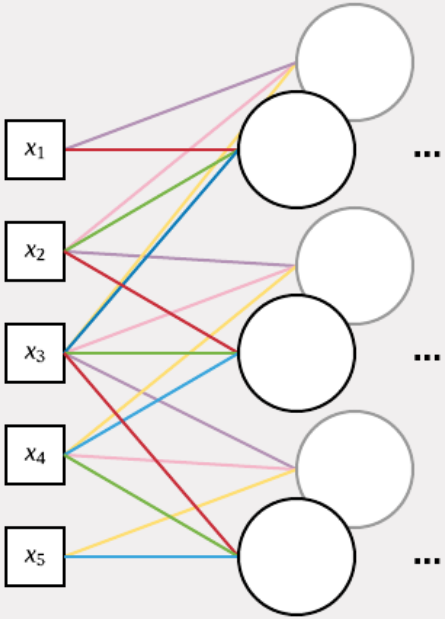


<u>two sets</u> shared weights
6 weights

We can add additional sets of weights that can learn additional interesting transformation of the input.

Note that the added neurons <u>are not</u> a new layer. They are part of layer 1.

TU/e



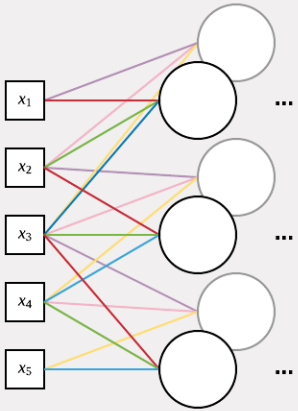
<u>two sets</u> shared weights
6 weights

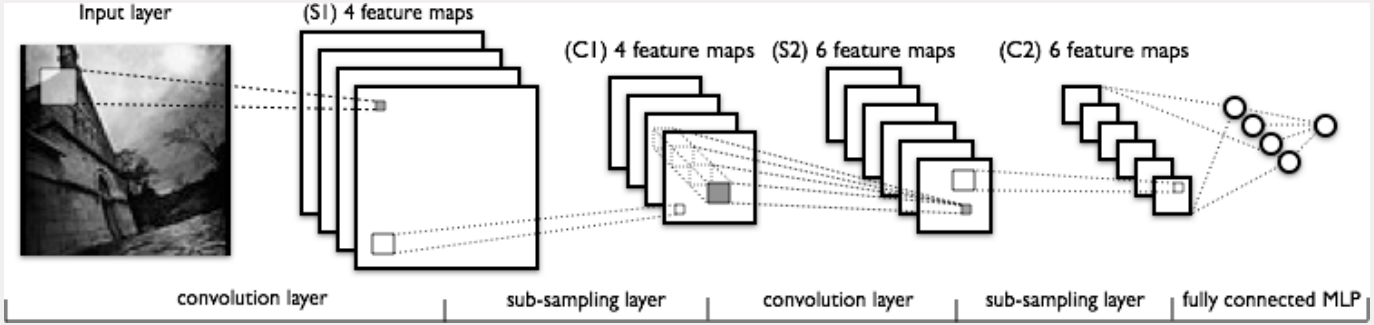
$$[a_{1,1} \quad a_{1,2} \quad a_{1,3}] = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] * [w_{1,1} \quad w_{1,2} \quad w_{1,3}]$$

$$[a_{2,1} \quad a_{2,2} \quad a_{2,3}] = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] * [w_{2,1} \quad w_{2,2} \quad w_{2,3}]$$

$[w_{1,1} \quad w_{1,2} \quad w_{1,3}]$, and $[w_{2,1} \quad w_{2,2} \quad w_{2,3}]$ are **convolution kernels**. They extract <u>features</u>. However, they are not hand-designed features – they were <u>learned</u> by the neural network.

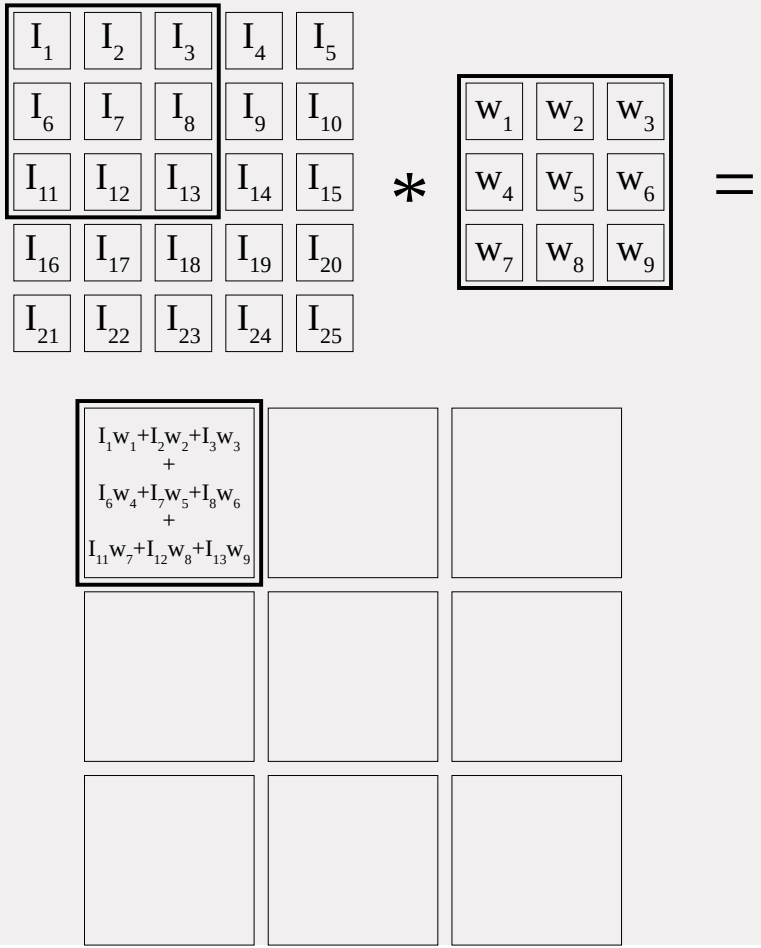
Convolutional neural networks are ideal for images

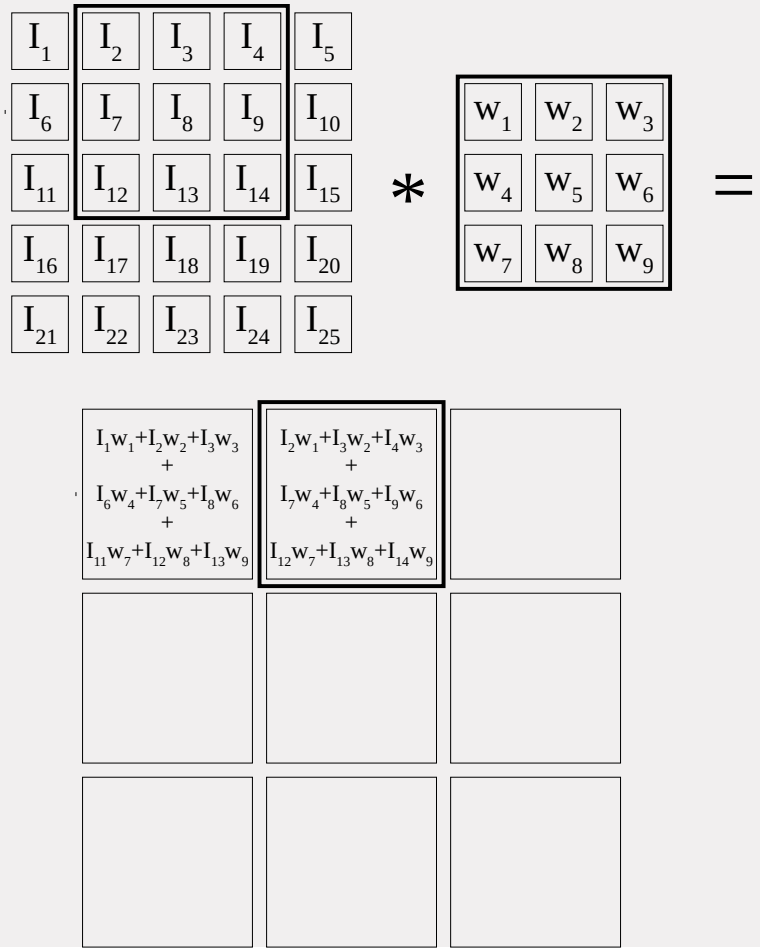


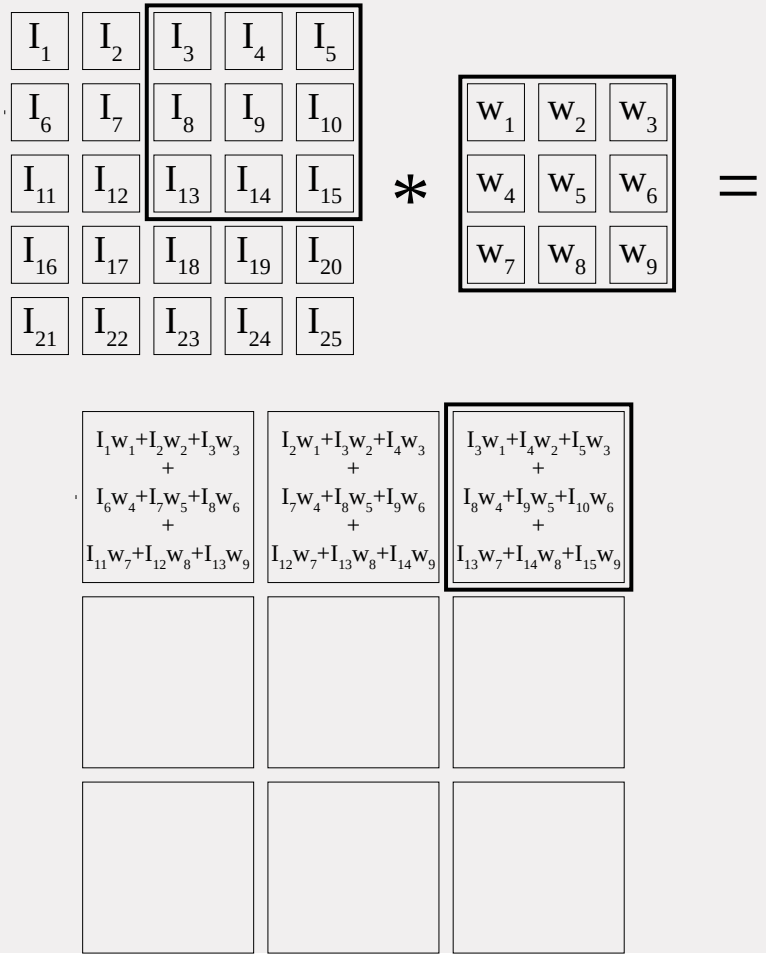


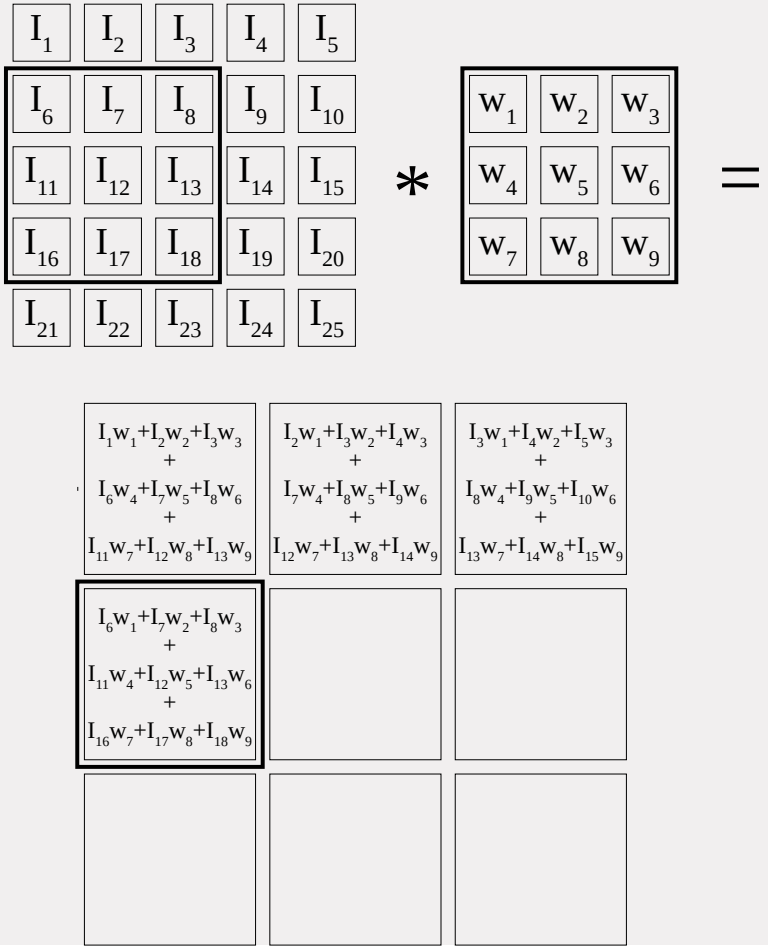
1D NN ⟷ **2D NN (images)**

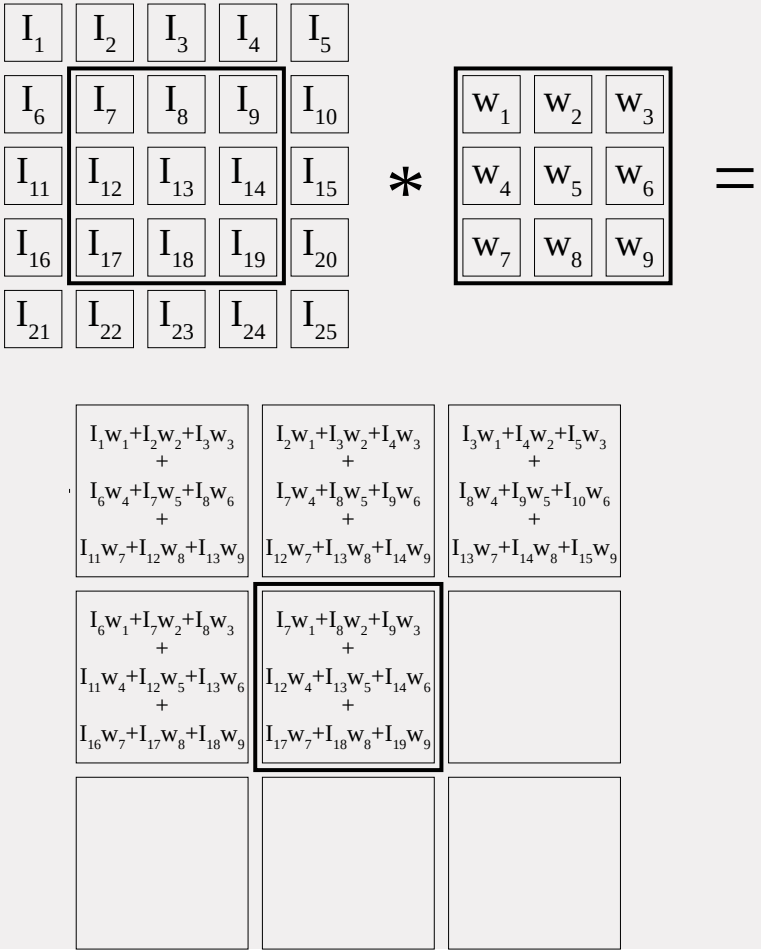
Because of the weight sharing, convolutional neural networks only work with structured data (such as images) as inputs.

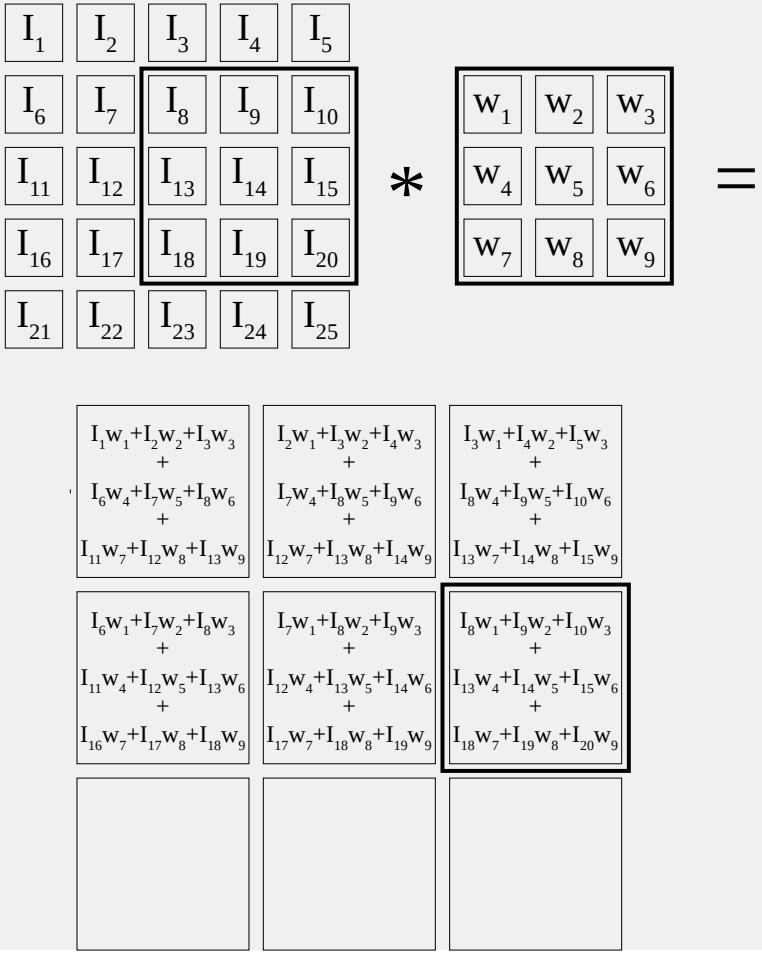

I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3
+
I6w4+I7w5+I8w6
+
I11w7+I12w8+I13w9

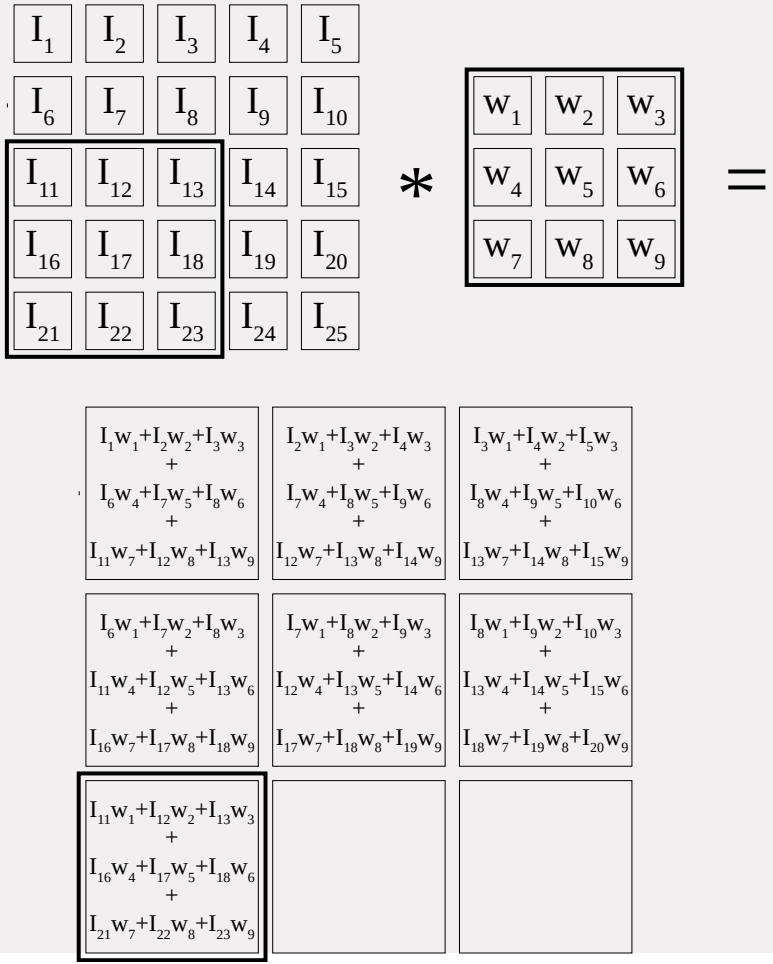

I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9

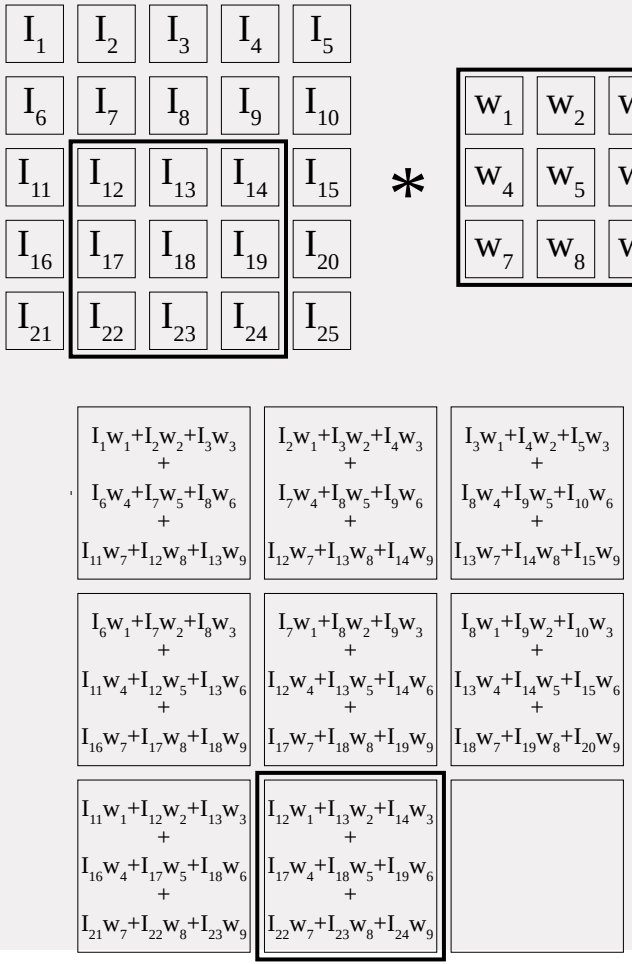

I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9
I3w1+I4w2+I5w3 + I8w4+I9w5+I10w6 + I13w7+I14w8+I15w9

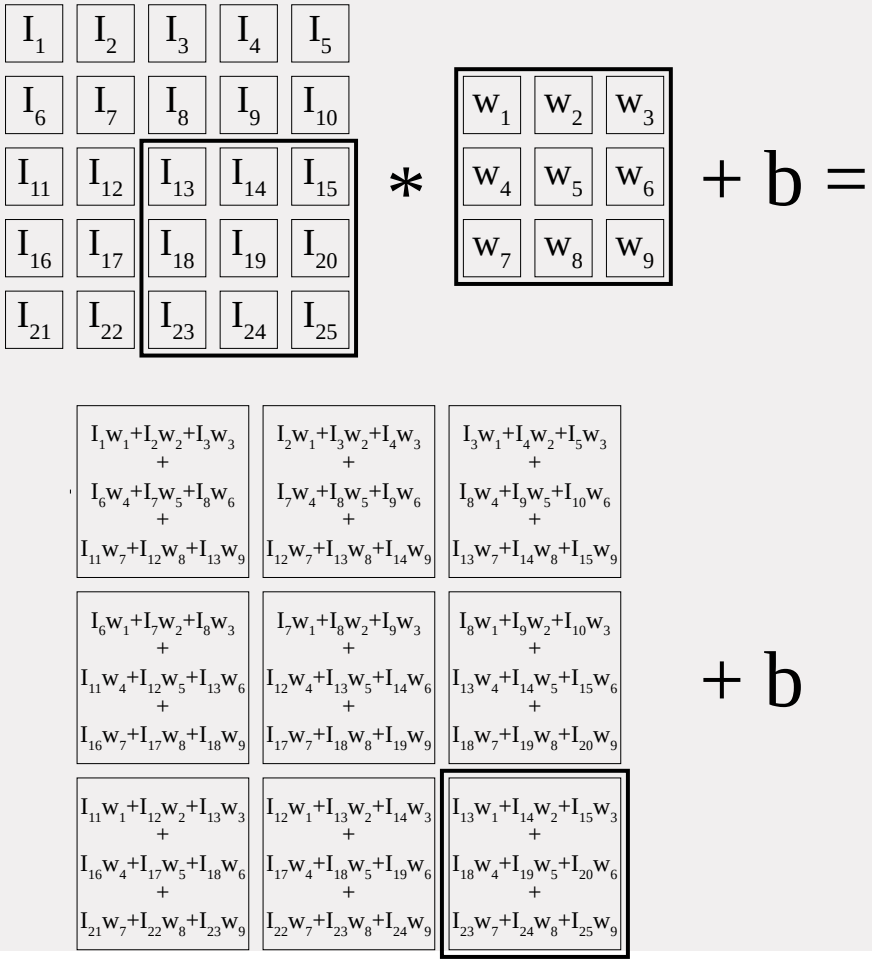

I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9
I3w1+I4w2+I5w3 + I8w4+I9w5+I10w6 + I13w7+I14w8+I15w9
I6w1+I7w2+I8w3 + I11w4+I12w5+I13w6 + I16w7+I17w8+I18w9


I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9
I3w1+I4w2+I5w3 + I8w4+I9w5+I10w6 + I13w7+I14w8+I15w9
I6w1+I7w2+I8w3 + I11w4+I12w5+I13w6 + I16w7+I17w8+I18w9
I7w1+I8w2+I9w3 + I12w4+I13w5+I14w6 + I17w7+I18w8+I19w9

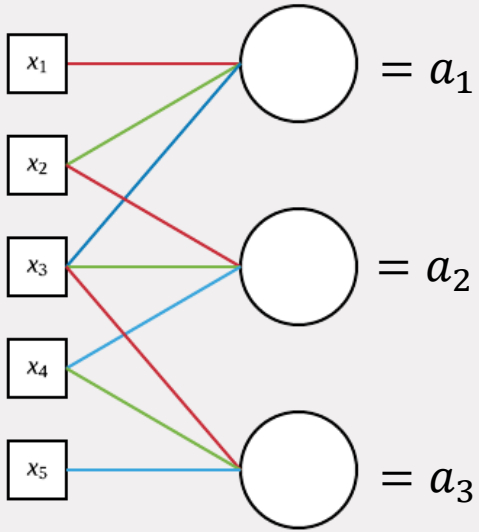

I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9
I3w1+I4w2+I5w3 + I8w4+I9w5+I10w6 + I13w7+I14w8+I15w9
I6w1+I7w2+I8w3 + I11w4+I12w5+I13w6 + I16w7+I17w8+I18w9
I7w1+I8w2+I9w3 + I12w4+I13w5+I14w6 + I17w7+I18w8+I19w9
I8w1+I9w2+I10w3 + I13w4+I14w5+I15w6 + I18w7+I19w8+I20w9


I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9
I3w1+I4w2+I5w3 + I8w4+I9w5+I10w6 + I13w7+I14w8+I15w9
I6w1+I7w2+I8w3 + I11w4+I12w5+I13w6 + I16w7+I17w8+I18w9
I7w1+I8w2+I9w3 + I12w4+I13w5+I14w6 + I17w7+I18w8+I19w9
I8w1+I9w2+I10w3 + I13w4+I14w5+I15w6 + I18w7+I19w8+I20w9
I11w1+I12w2+I13w3 + I16w4+I17w5+I18w6 + I21w7+I22w8+I23w9

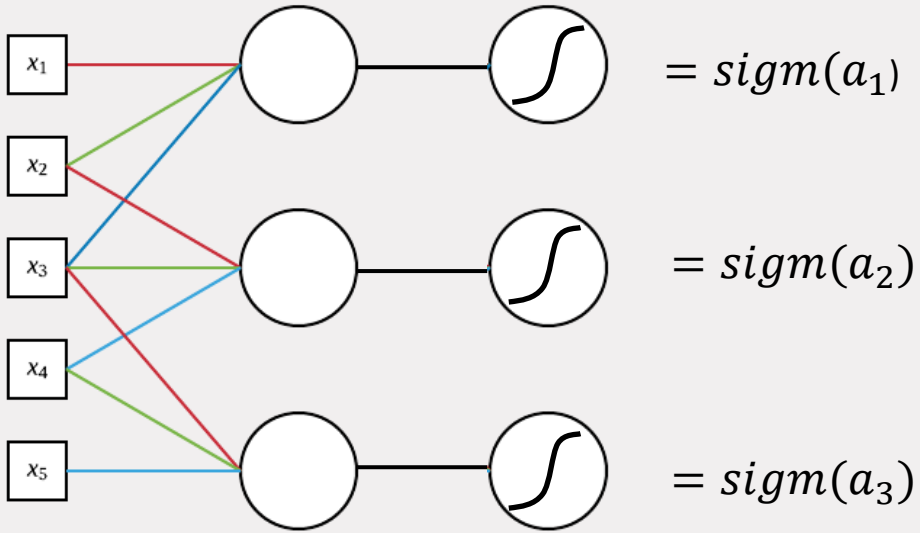

I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
=
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9
I3w1+I4w2+I5w3 + I8w4+I9w5+I10w6 + I13w7+I14w8+I15w9
I6w1+I7w2+I8w3 + I11w4+I12w5+I13w6 + I16w7+I17w8+I18w9
I7w1+I8w2+I9w3 + I12w4+I13w5+I14w6 + I17w7+I18w8+I19w9
I8w1+I9w2+I10w3 + I13w4+I14w5+I15w6 + I18w7+I19w8+I20w9
I11w1+I12w2+I13w3 + I16w4+I17w5+I18w6 + I21w7+I22w8+I23w9
I12w1+I13w2+I14w3 + I17w4+I18w5+I19w6 + I22w7+I23w8+I24w9


I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
*
w1 w2 w3
w4 w5 w6
w7 w8 w9
+ b =
I1w1+I2w2+I3w3 + I6w4+I7w5+I8w6 + I11w7+I12w8+I13w9
I2w1+I3w2+I4w3 + I7w4+I8w5+I9w6 + I12w7+I13w8+I14w9
I3w1+I4w2+I5w3 + I8w4+I9w5+I10w6 + I13w7+I14w8+I15w9
I6w1+I7w2+I8w3 + I11w4+I12w5+I13w6 + I16w7+I17w8+I18w9
I7w1+I8w2+I9w3 + I12w4+I13w5+I14w6 + I17w7+I18w8+I19w9
I8w1+I9w2+I10w3 + I13w4+I14w5+I15w6 + I18w7+I19w8+I20w9
I11w1+I12w2+I13w3 + I16w4+I17w5+I18w6 + I21w7+I22w8+I23w9
I12w1+I13w2+I14w3 + I17w4+I18w5+I19w6 + I22w7+I23w8+I24w9
I13w1+I14w2+I15w3 + I18w4+I19w5+I20w6 + I23w7+I24w8+I25w9
+ b

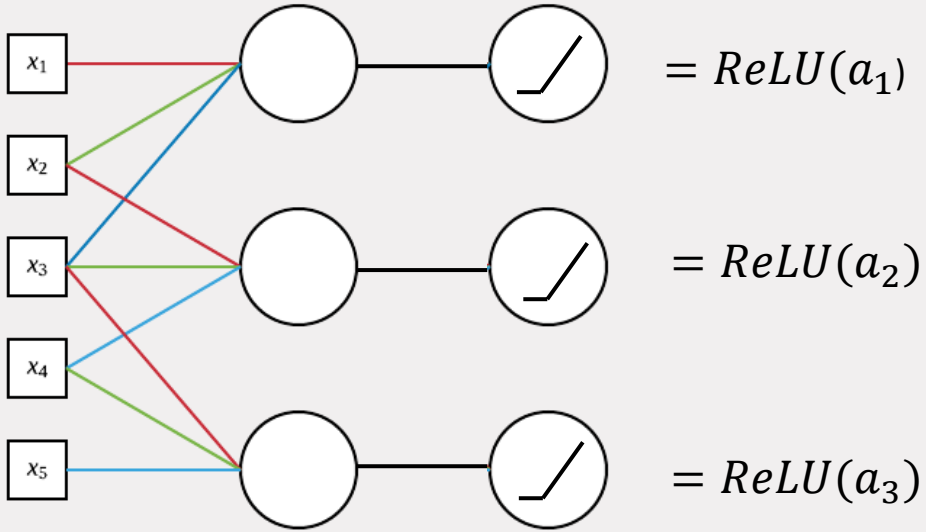
Convolutions + non-linearity



Convolutions + non-linearity (sigmoid)



Convolutions + non-linearity (ReLU)

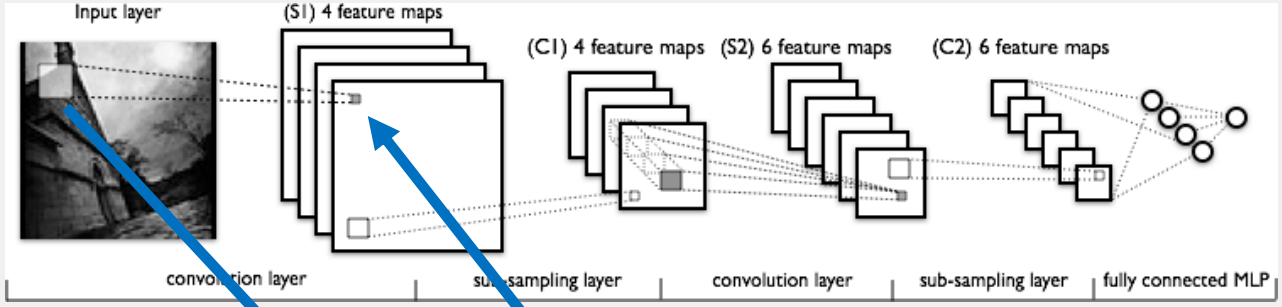


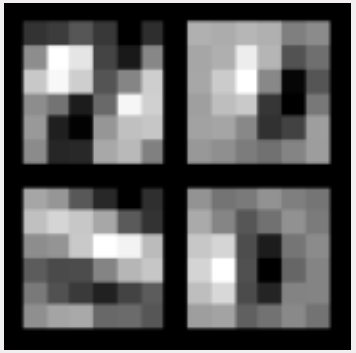
Break!

Quokka (Australia)

https://imgur.com/r/aww/GqJi0iI

Convolutional kernels



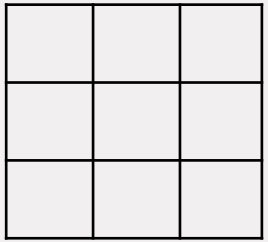


these are the weights (convolutional kernels) that produce the four feature maps

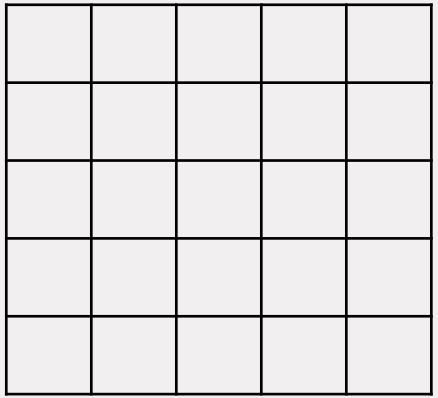
Kernel size

1 x 1

3 x 3

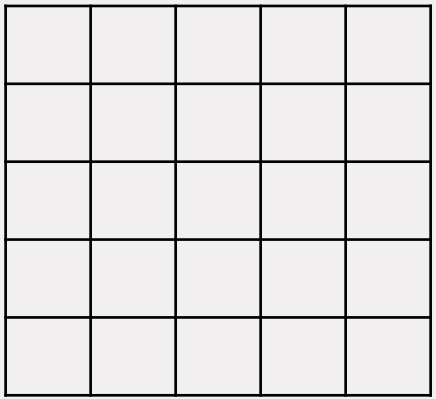


5 x 5



- More weights → more information
- More computations / memory
- Receptive field

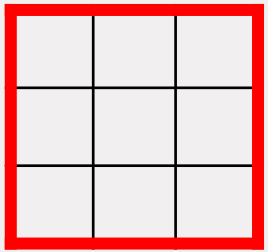
Receptive field



5 x 5

25 weights

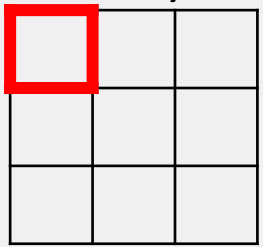
Same receptive field



3 x 3

9 weights

2nd layer

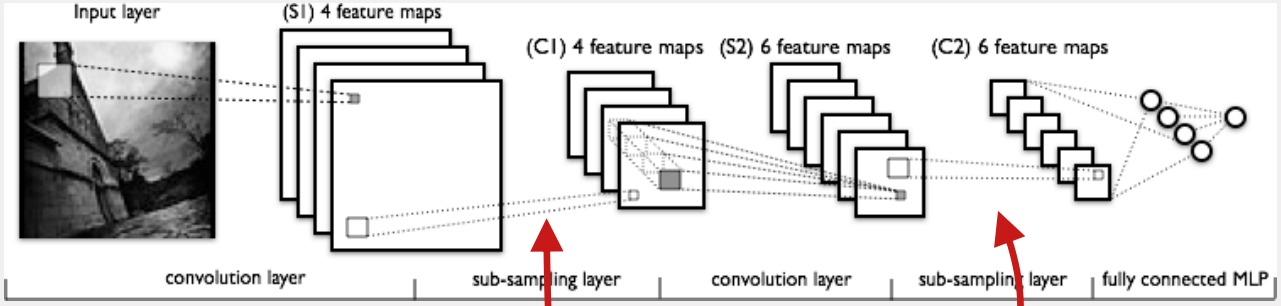


3 x 3

9 weights

= 18 weights

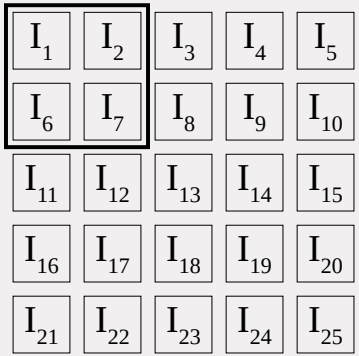
Max-pooling



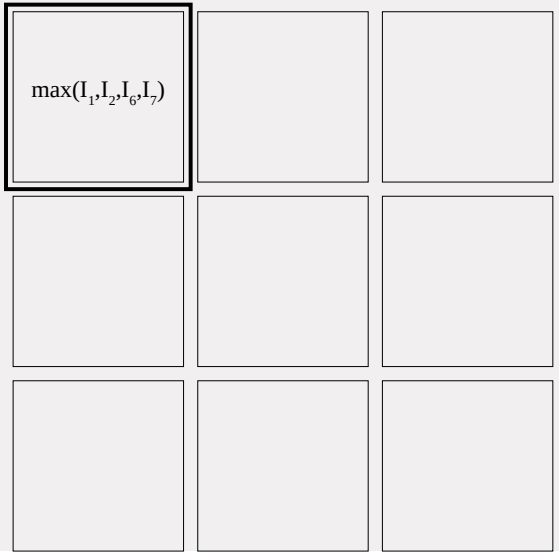
max-pooling

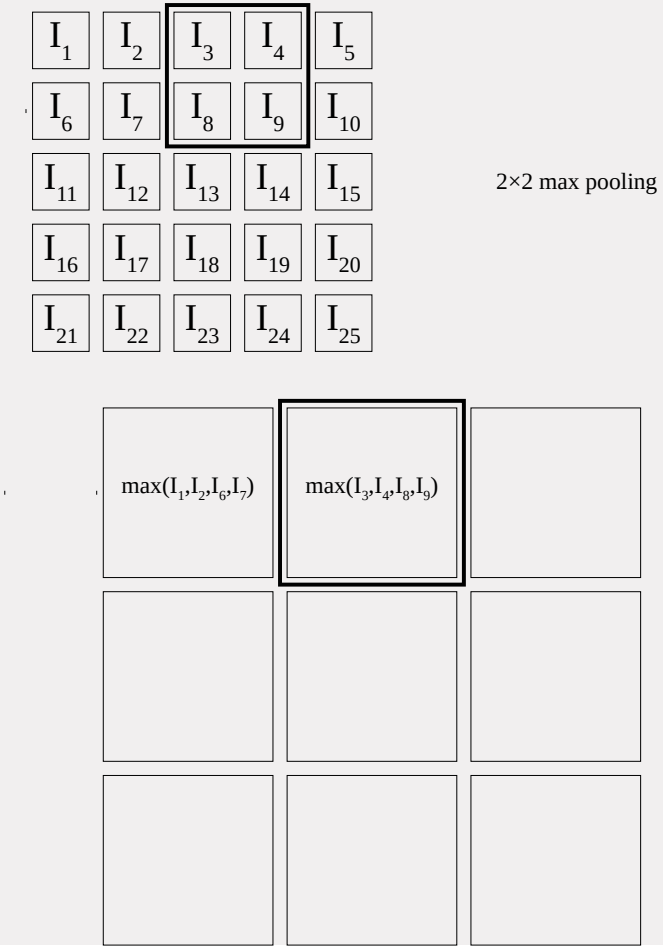
Max-pooling

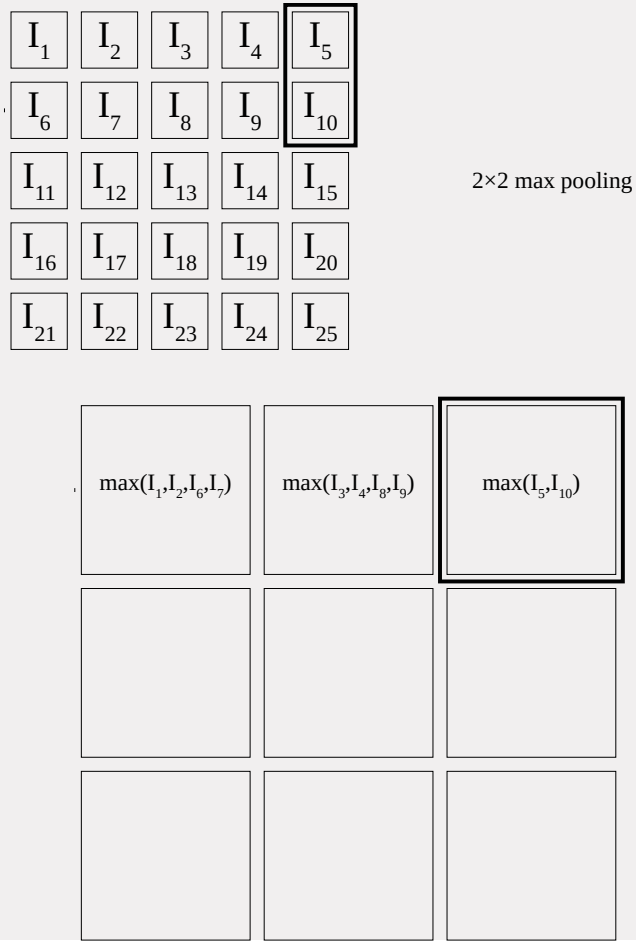
- Reduce size of feature space
- Maximum of features
- Typical kernel size = 2 x 2



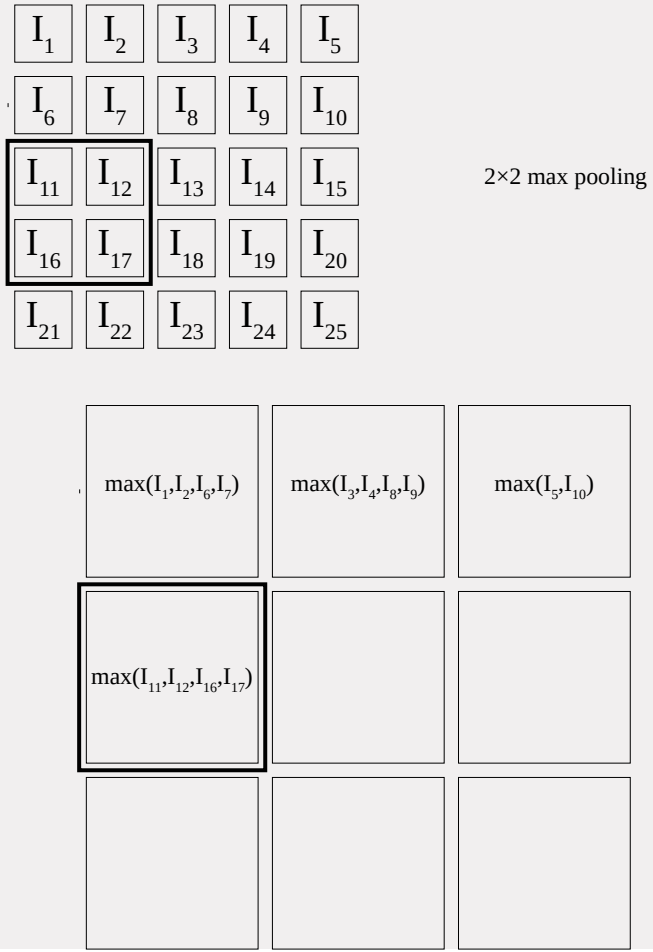
2×2 max pooling

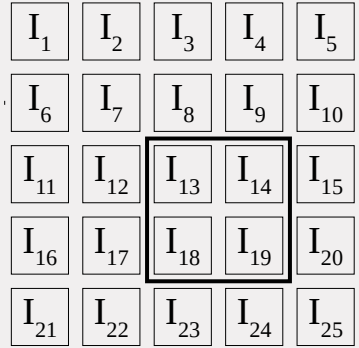



I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
2×2 max pooling
max(I1,I2,I6,I7)
max(I3,I4,I8,I9)

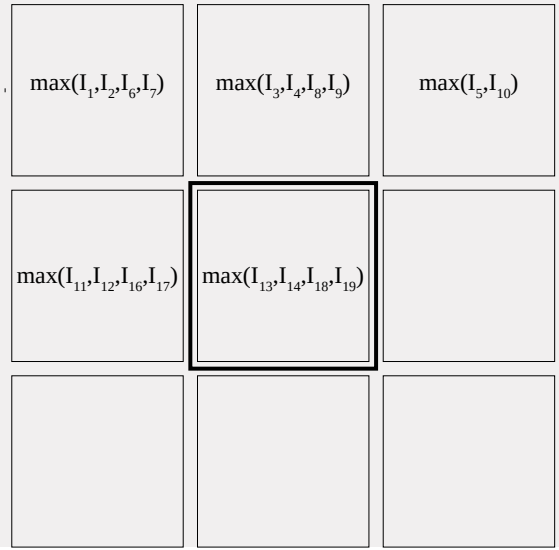


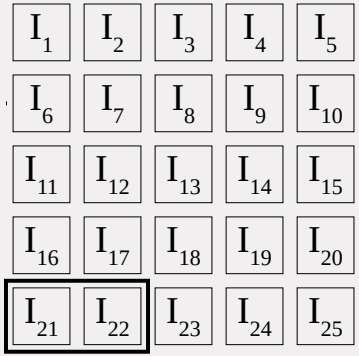
In this example:
no zero padding


I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
2×2 max pooling
max(I1,I2,I6,I7)
max(I3,I4,I8,I9)
max(I5,I10)
max(I11,I12,I16,I17)

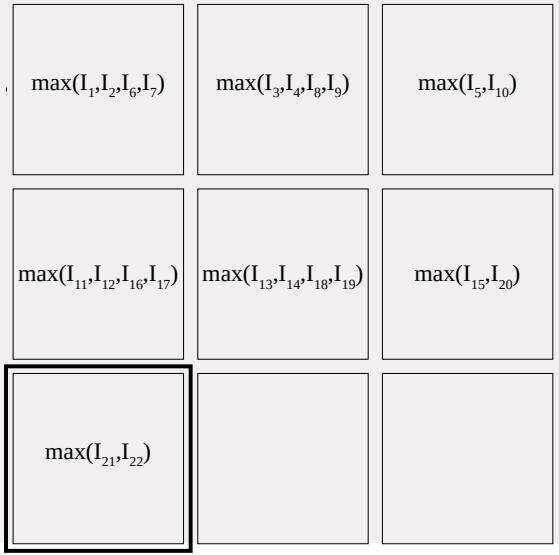


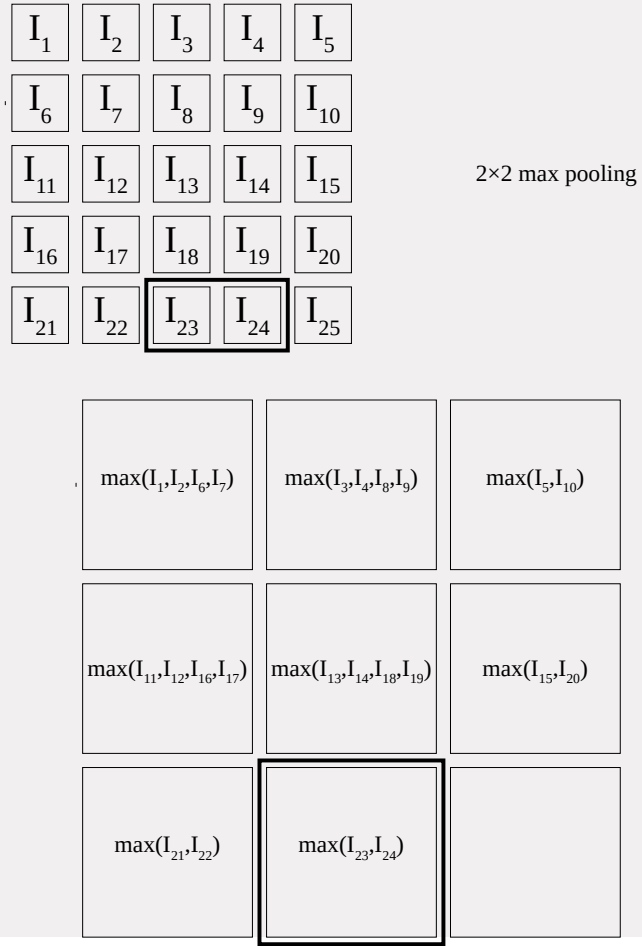
2×2 max pooling

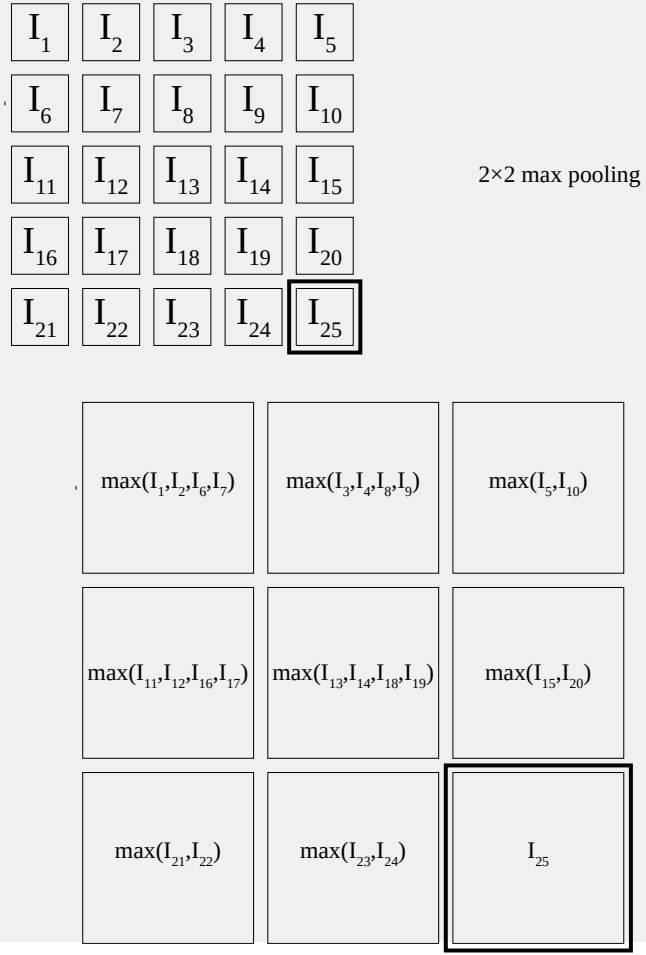




2×2 max pooling




I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
2×2 max pooling
max(I1,I2,I6,I7) max(I3,I4,I8,I9) max(I5,I10)
max(I11,I12,I16,I17) max(I13,I14,I18,I19) max(I15,I20)
max(I21,I22) max(I23,I24)


I1 I2 I3 I4 I5
I6 I7 I8 I9 I10
I11 I12 I13 I14 I15
I16 I17 I18 I19 I20
I21 I22 I23 I24 I25
2×2 max pooling
max(I1,I2,I6,I7)
max(I3,I4,I8,I9)
max(I5,I10)
max(I11,I12,I16,I17)
max(I13,I14,I18,I19)
max(I15,I20)
max(I21,I22)
max(I23,I24)
I25

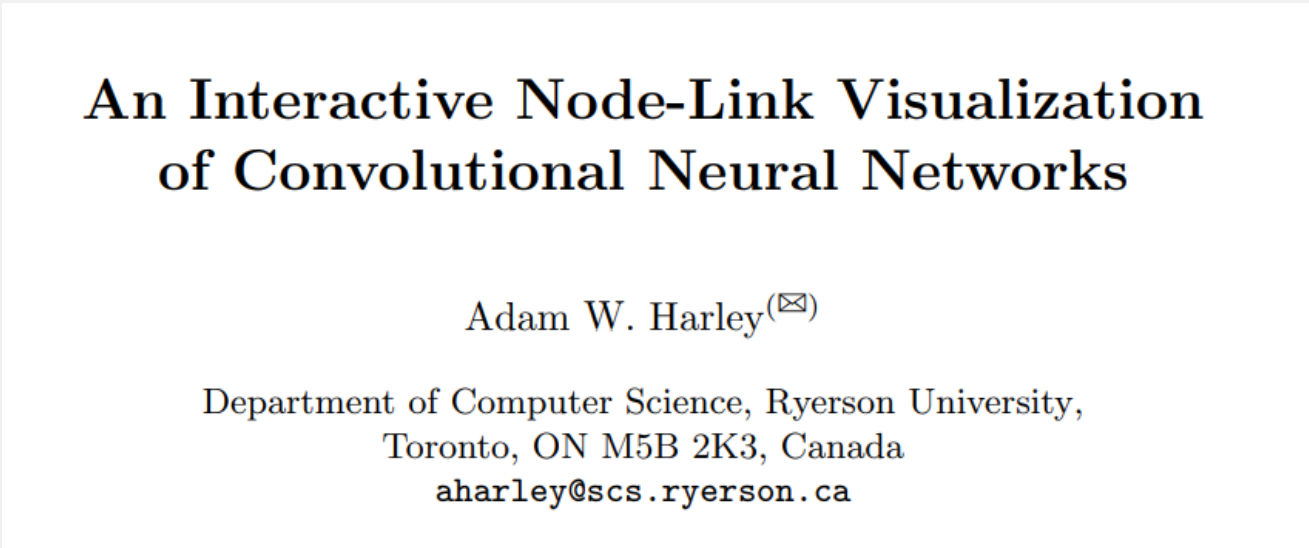
Benefits of max-pooling

- “Quickly” reduces the number size of the feature maps
- Introduces translation invariance (slightly translated version of the input image will result in the same output)

Alternative: Average Pooling

Let try it out!

http://scs.ryerson.ca/~aharley/vis/conv/flat.html



An Interactive Node-Link Visualization of Convolutional Neural Networks

Adam W. Harley(✉)

Department of Computer Science, Ryerson University,
Toronto, ON M5B 2K3, Canada
aharley@scs.ryerson.ca

Training of Convolutional Neural Networks

- Similar to training of ‘fully connected’ neural networks
- Choose some (random) initial values for network weights
- Optimize networks weights with respect to a loss function that describes difference between network output and label/annotation
- Update networks weights iteratively
- Keep track of model performance on train & validation set

Summary

- Concept of **convolutions** in a neural network
- Why can we use a **convolutional approach** for (medical) **images**
- Convolutions enable development of **deep** (and large) **neural networks**
- **Max-pooling layer** in a convolutional neural network
- **Kernel size**
- **Receptive field**