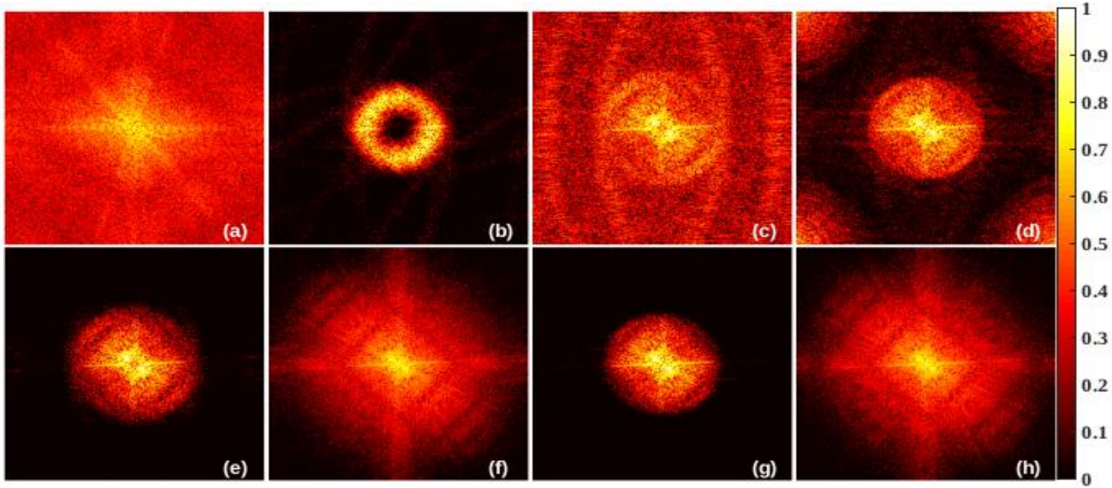



# Convolutional Neural Networks (8DC00)

Navchetan Awasthi

# Navchetan Awasthi Phd.

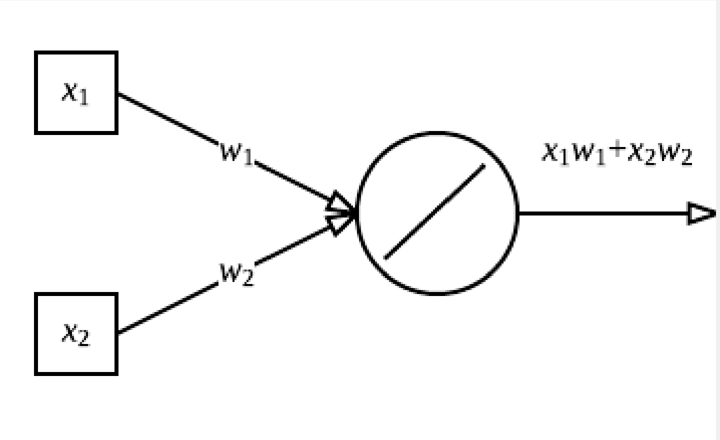
**Background:** B.Tech. Electronics and Communication Engineering, M.Tech. in Computational Science, PhD in Medical Imaging

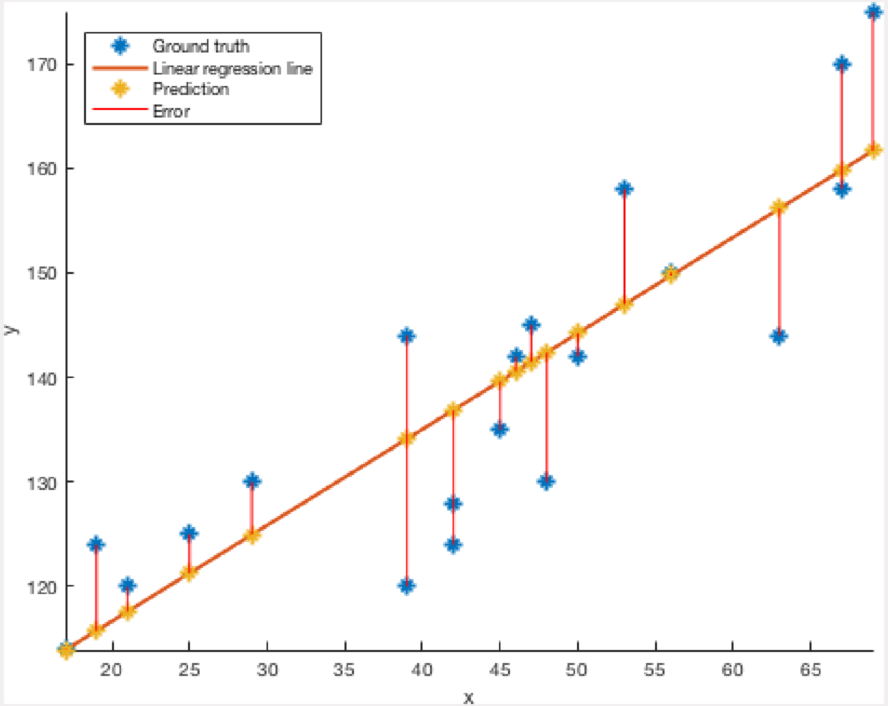
**Work/internship experience**: Harvard Medical School, Massachusetts General Hospital, Indian Institute of Science

**PhD Research:** Deep learning, Medical image analysis, Medical Image reconstruction, Ultrasound Imaging, Photoacoustics, Inverse Problems

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

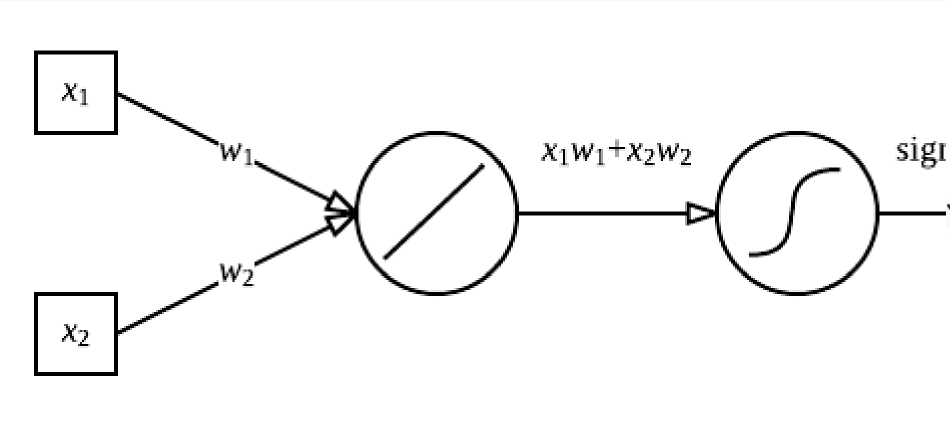
# Previously – Linear regression

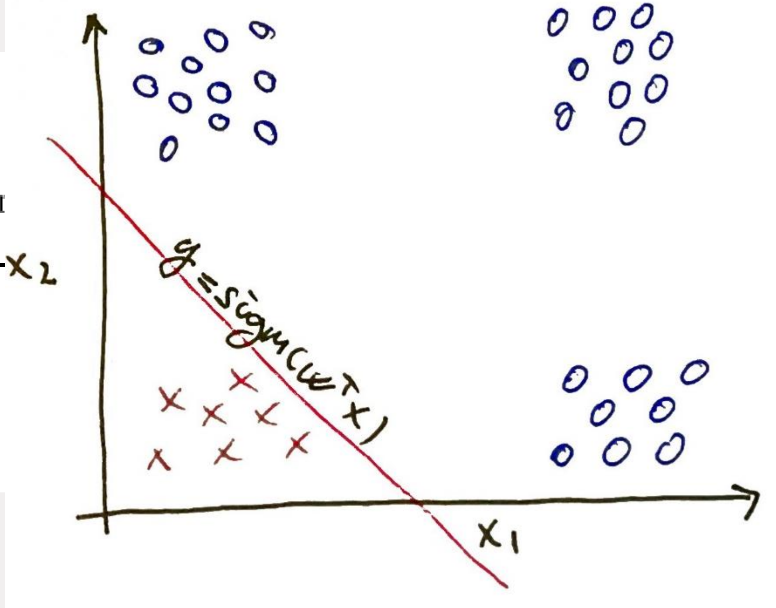


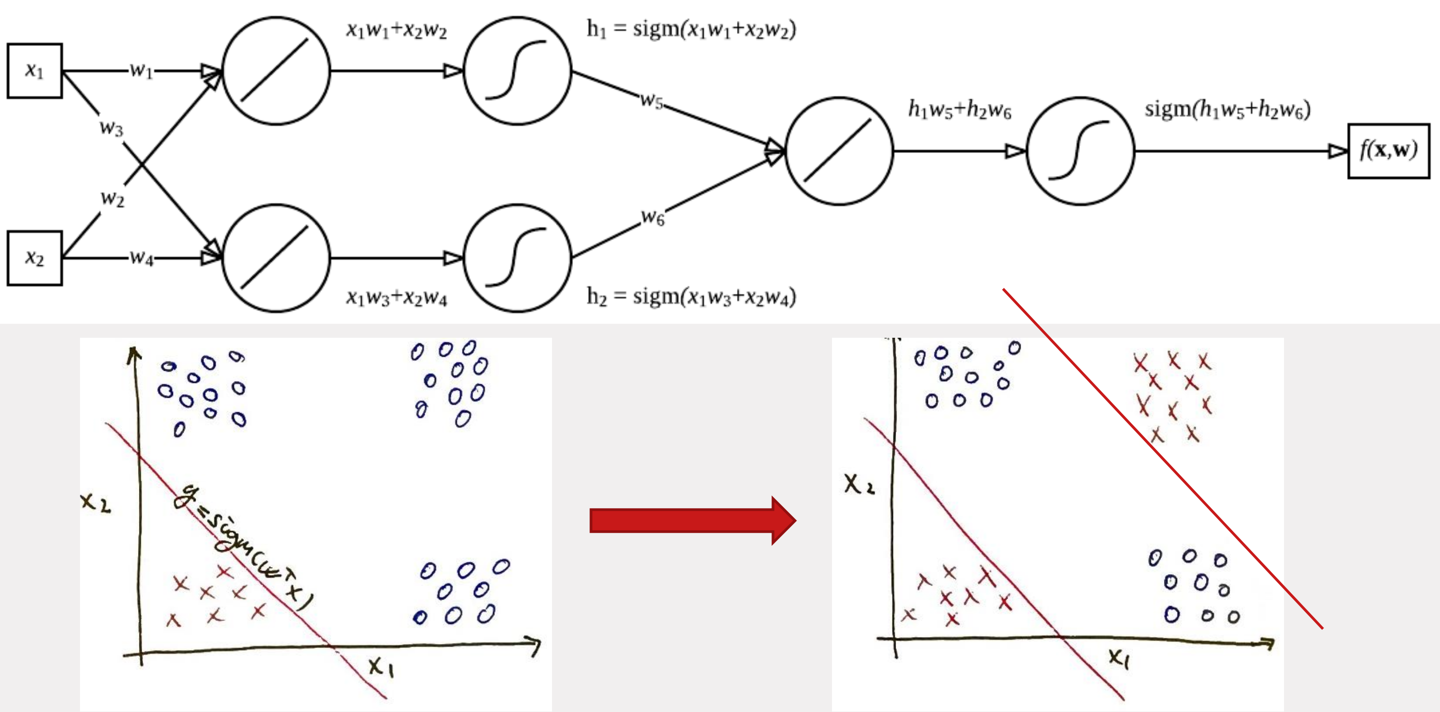
$$p(y = 1 \mid \mathbf{x}) = \sigma(\boldsymbol{\theta}^\intercal \mathbf{x})$$

# Previously – Logistic regression

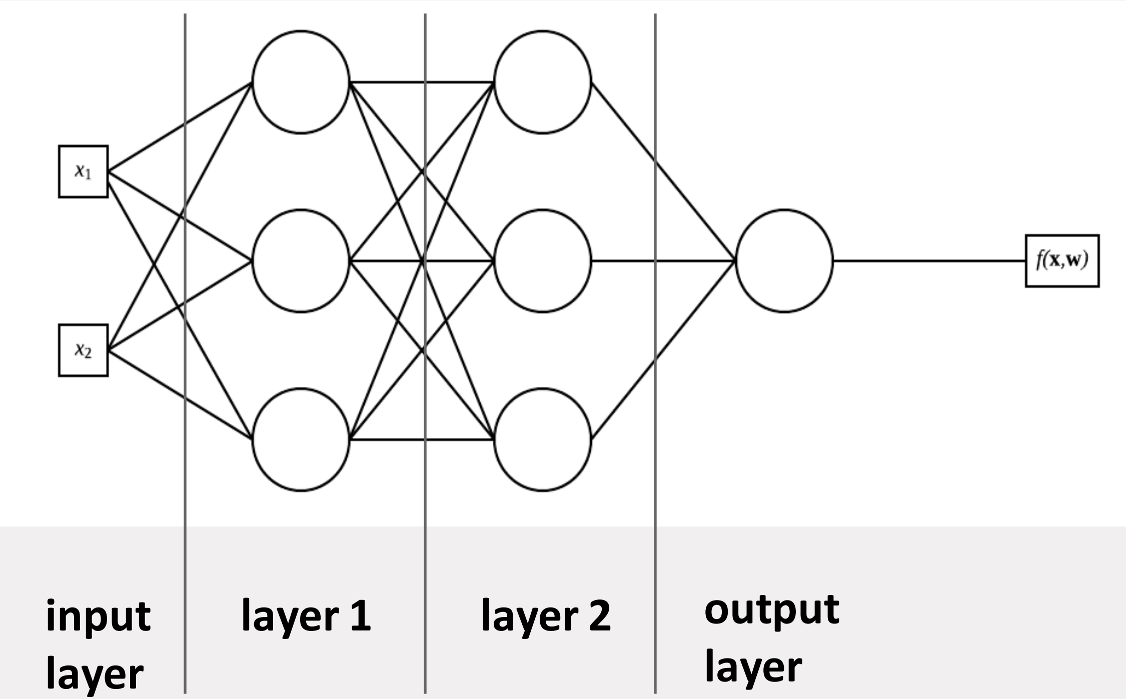


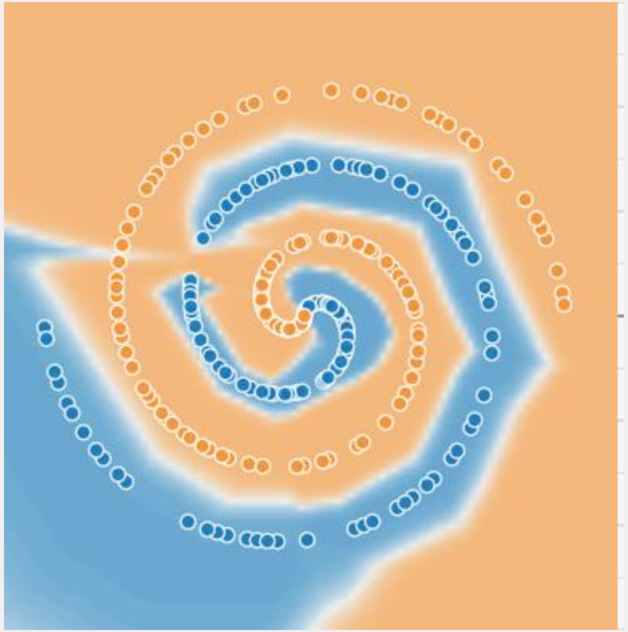



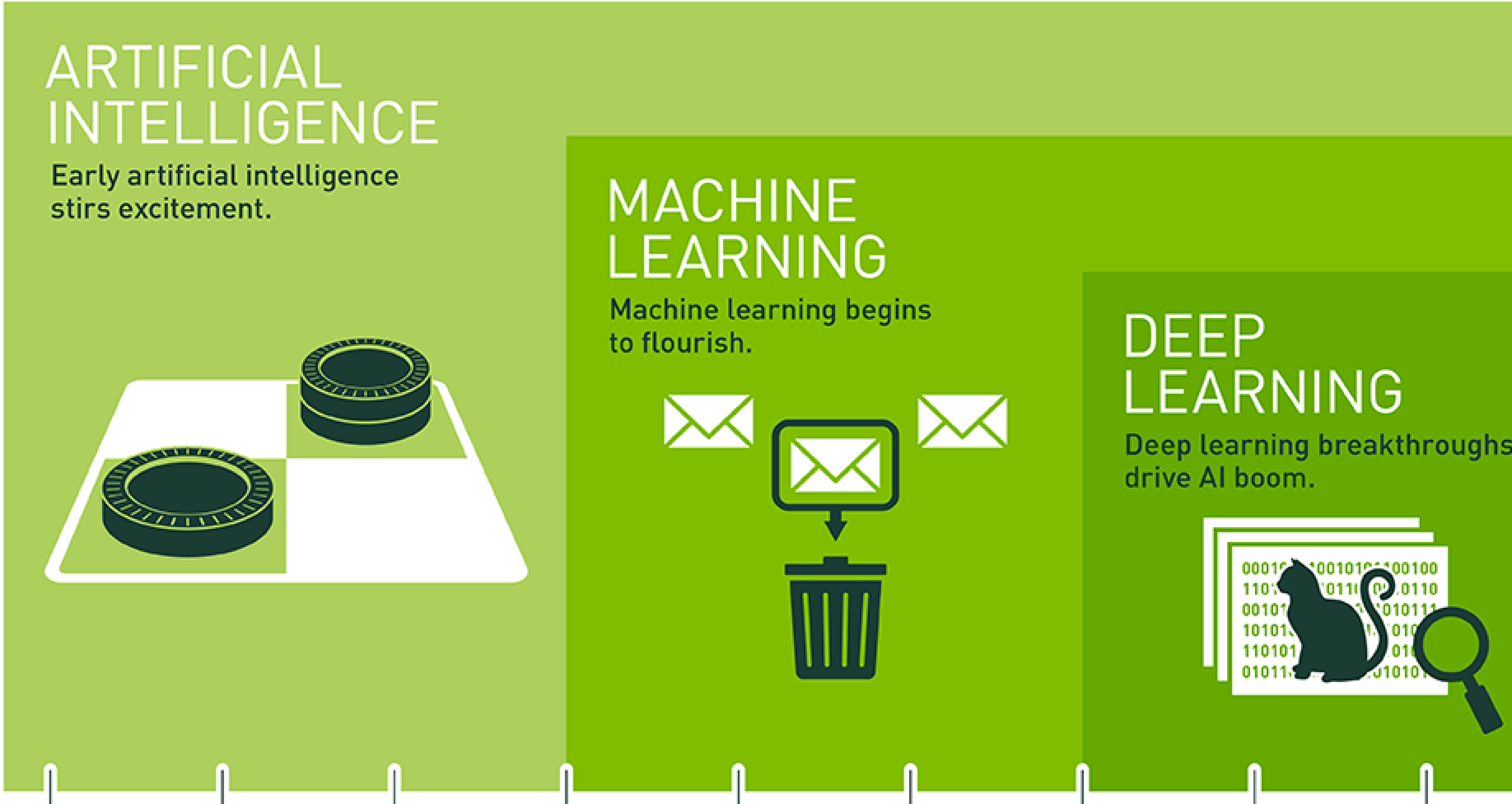
# Previously – Neural networks



# Previously – Neural networks




ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.

# 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)

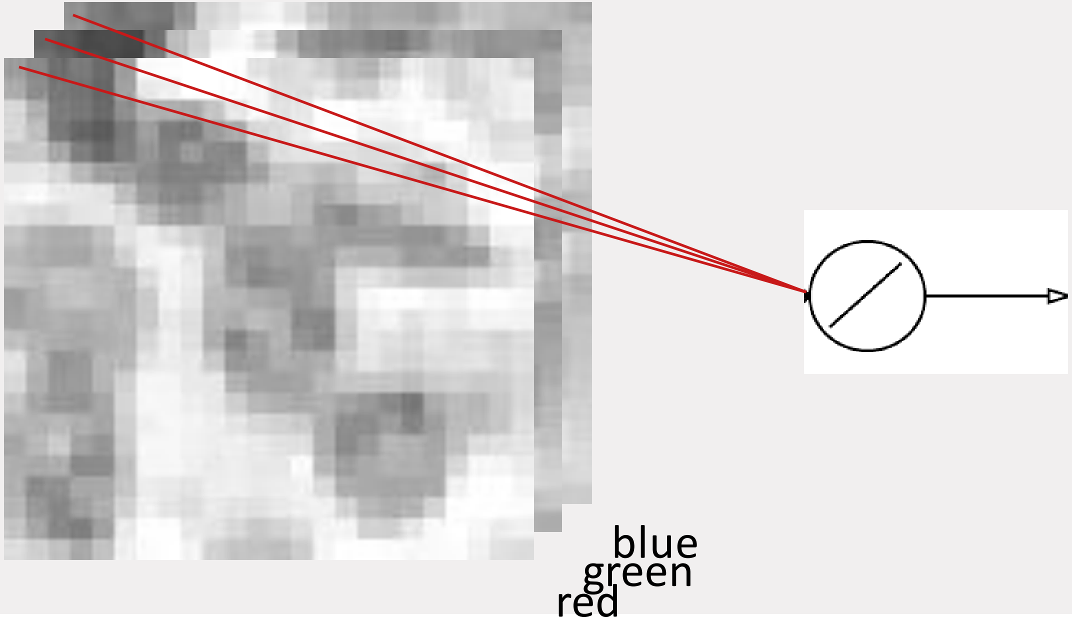
= 1728 features ($\mathbf{x}_i$ are 1728 dimensional vectors)

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

$\theta^\top x$

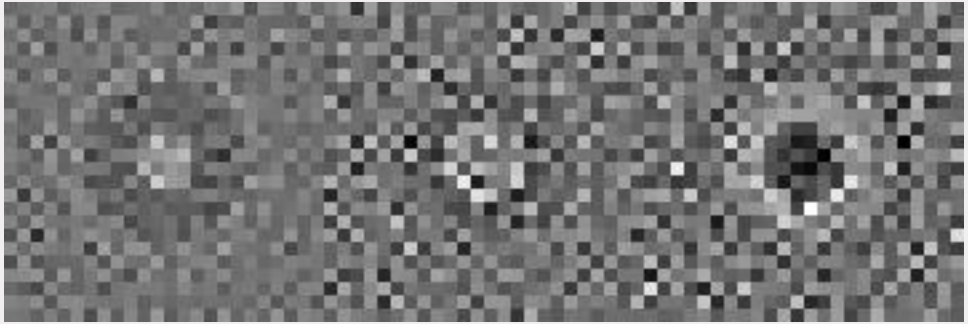
# Every pixel is an input

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

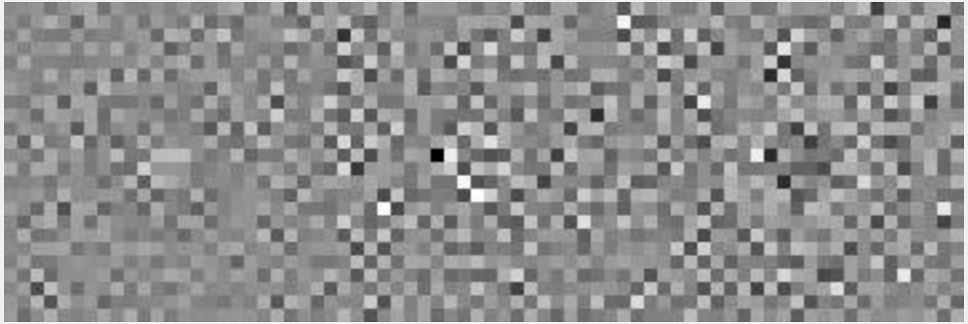
Only 25% of training samples.. Looks noisy!

24 pixels

24 pixels

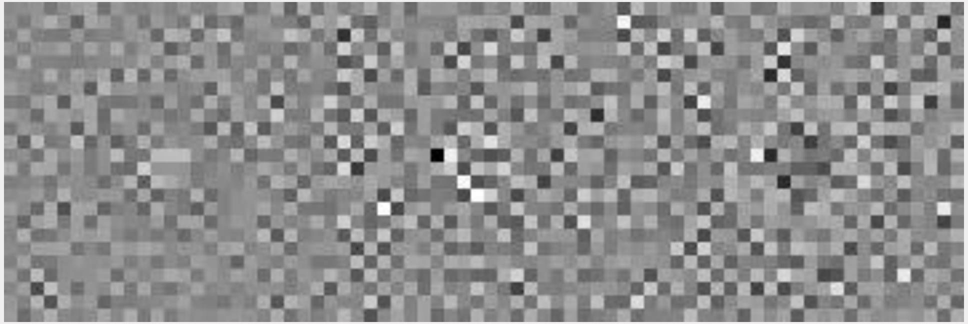
3 color channels
(red, green, blue)

24 pixels 24 pixels 24 pixels

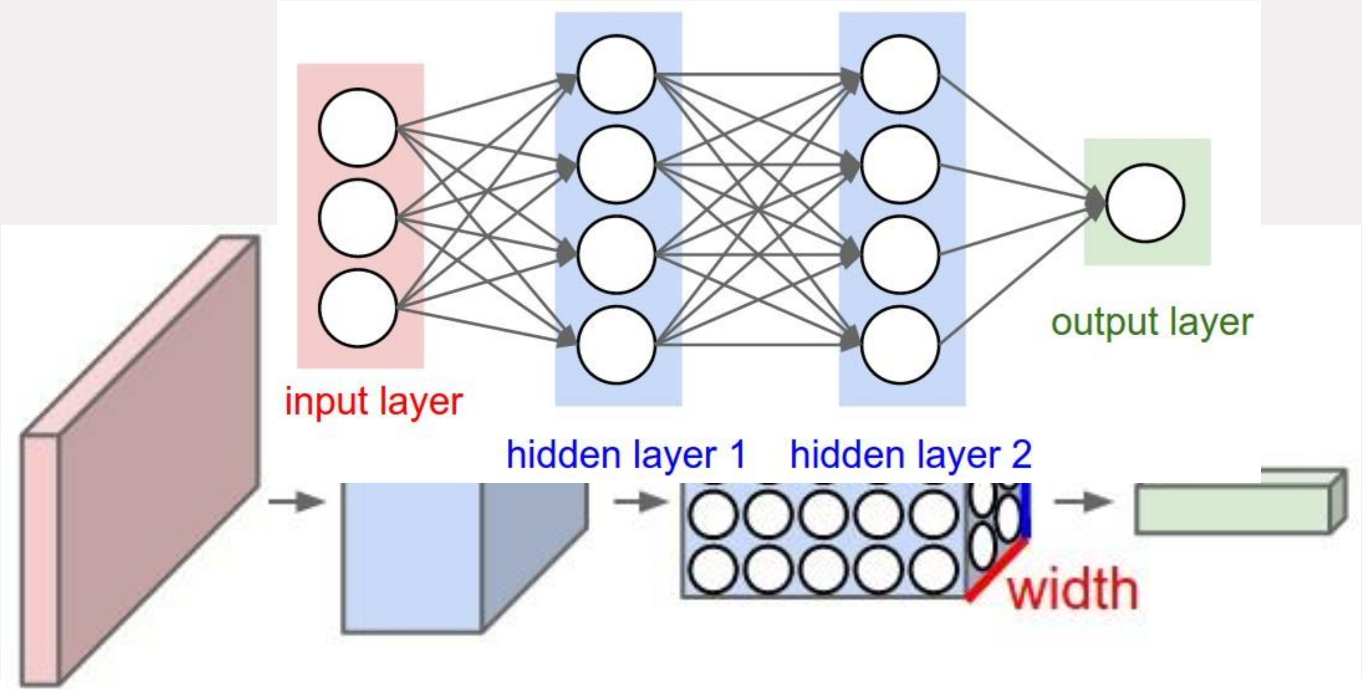


# Post-training: visualizing what model has learned

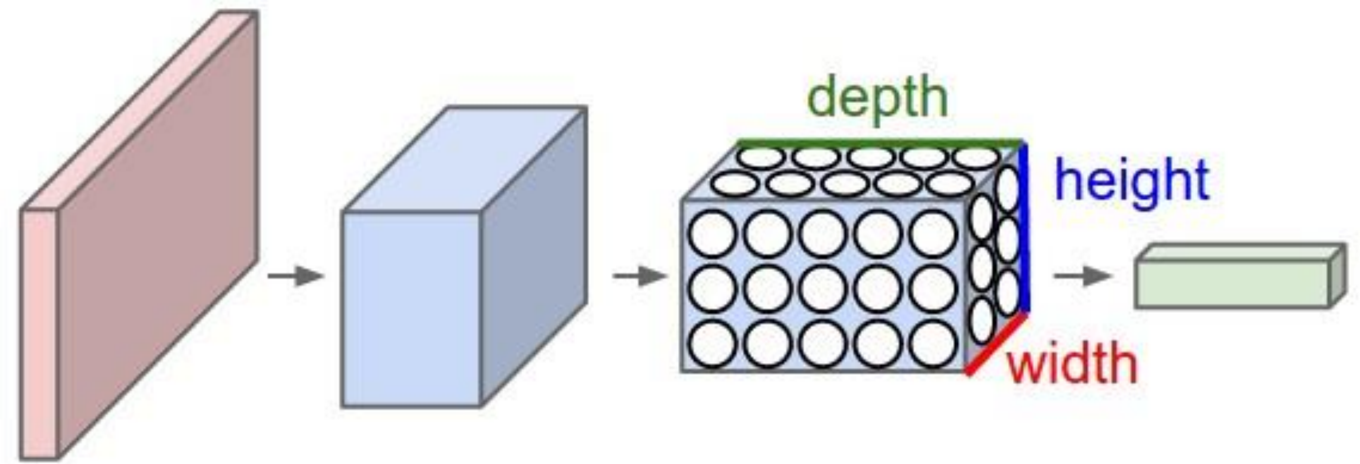
You can think of it as “there is not enough training data to reliably estimate all model weights”.

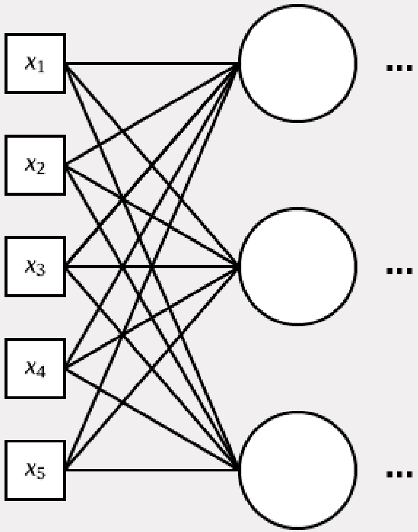
# Many weights



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.

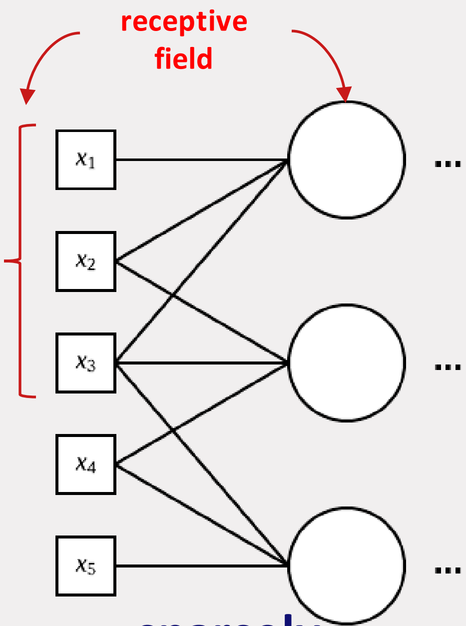


# How many weights?



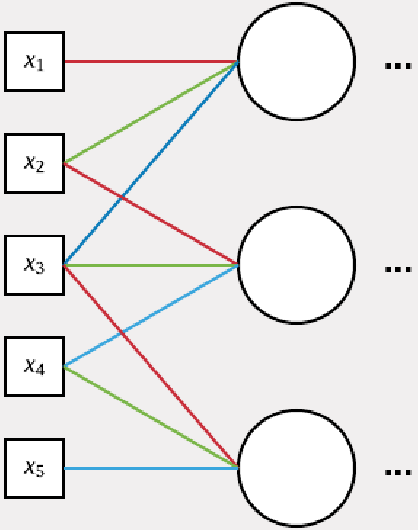

**"regular" NN**

15 weights




**sparsely connected NN**

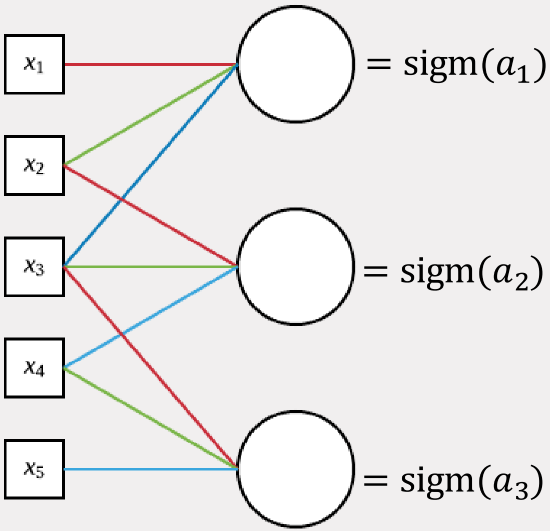
9 weights




**shared weights**

3 weights

# Shared weights




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

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

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

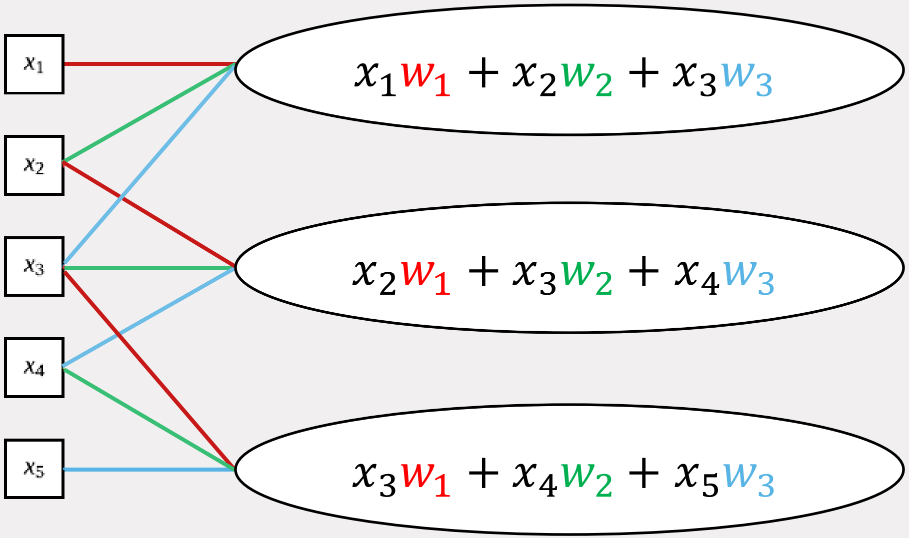
$$[a_1 \quad a_2 \quad a_3] =$$
$$[x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] * [w_1 \quad w_2 \quad w_3]$$

**shared weights**

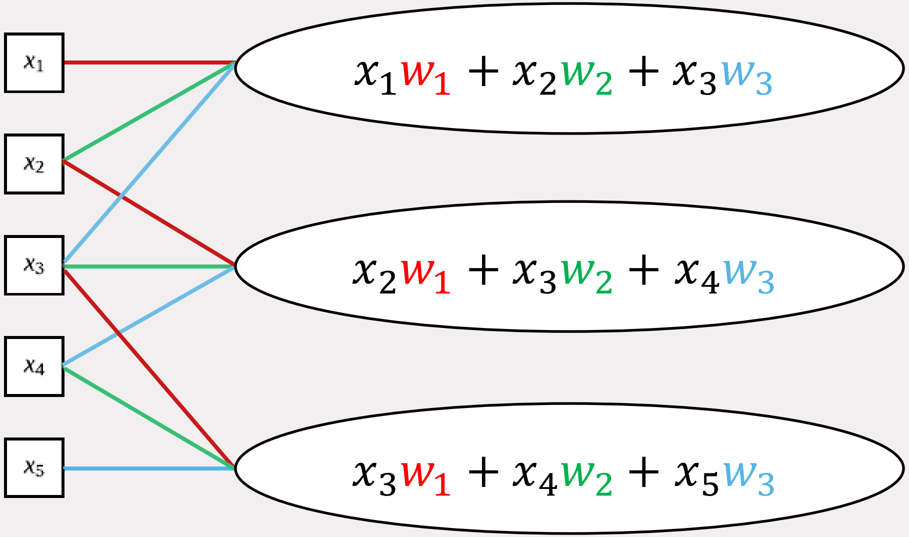
3 weights

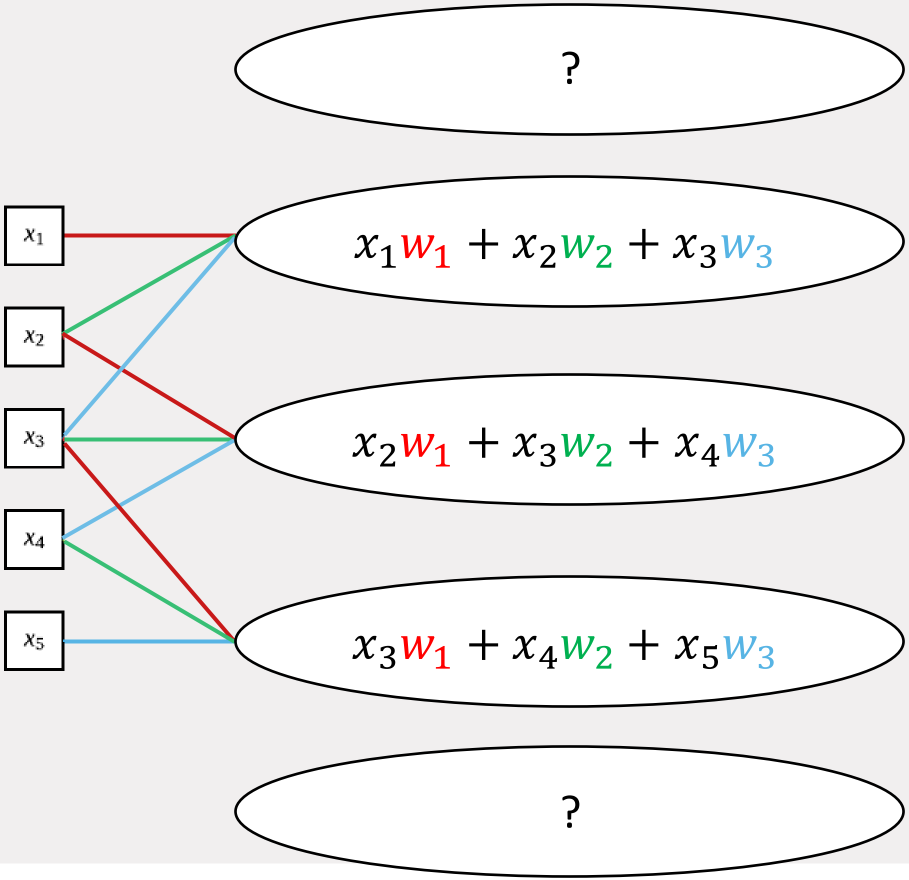
**convolution, thus convolutional NN**

# 1-D Convolution



# 1-D Convolution



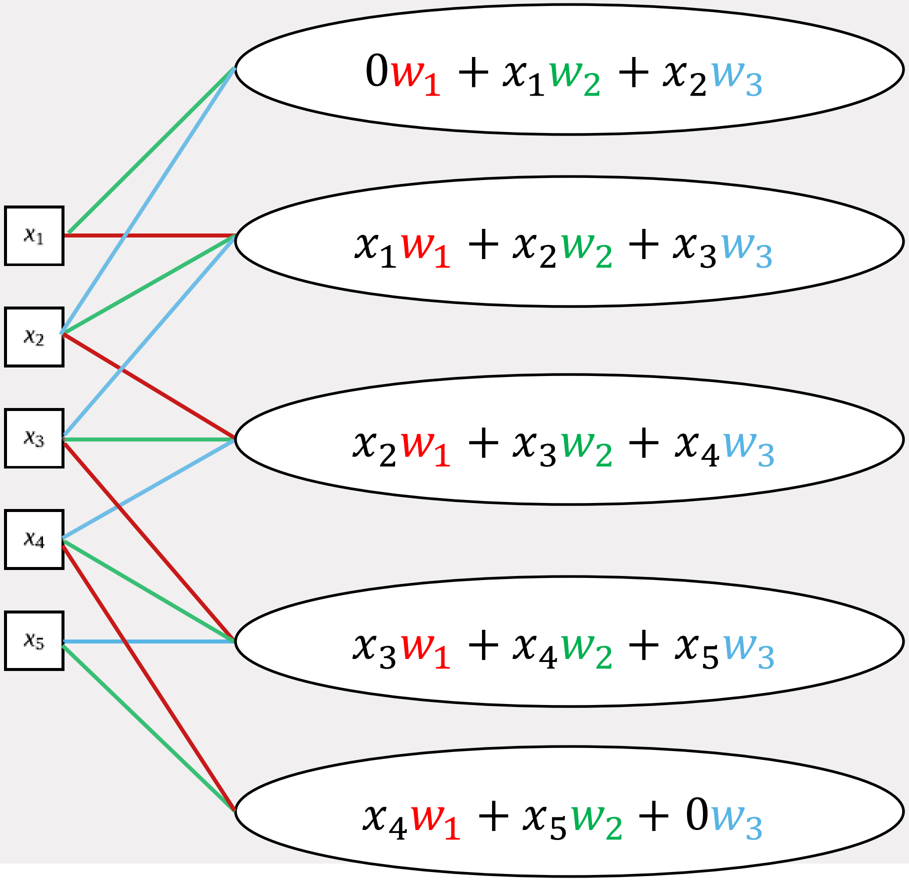



# 1-D Convolution

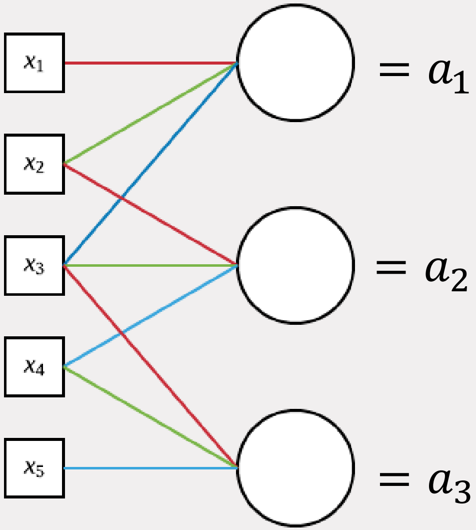
How can we keep the same number of features in hidden layer 1?

# 1-D Convolution

Zero-padding!



# 1-D Convolution




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

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

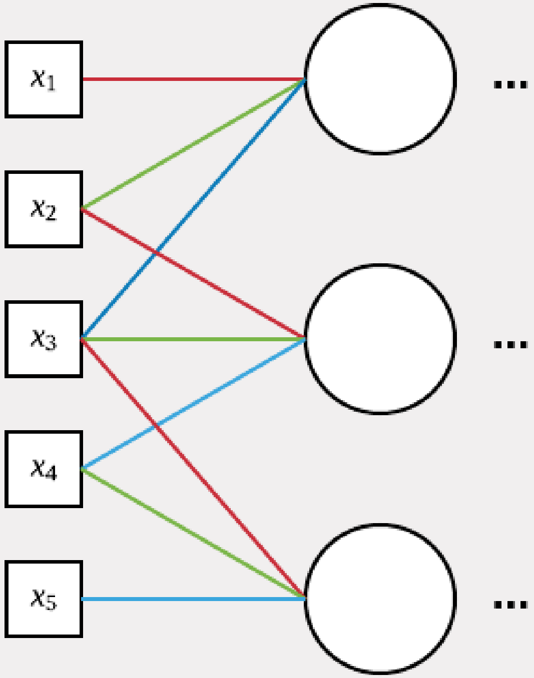
$$a_3 = x_3 w_1 + x_4 w_2 + x_5 w_3$$

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


TU/e

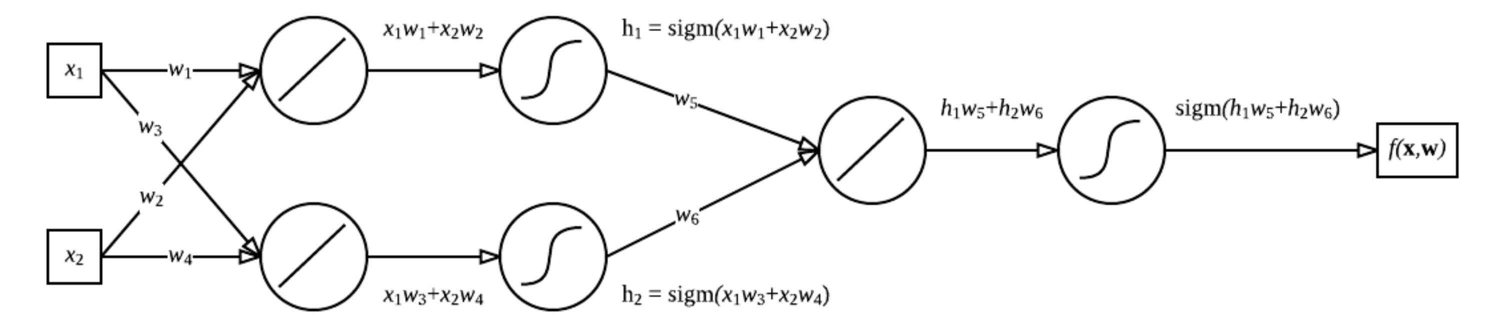
# Properties of convolutional neural networks

- Sparse connectivity
- Weight sharing
- Parallel computations



# Why does this work?

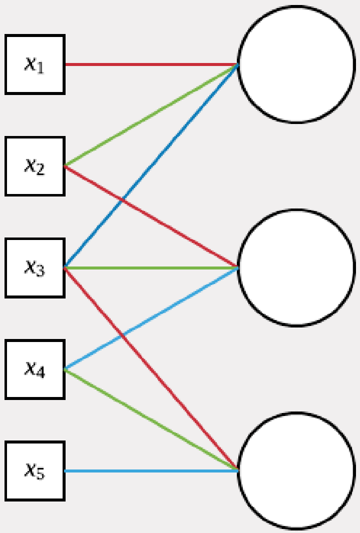
- Multiple layers
- Hidden layers contain features calculated from previous layers



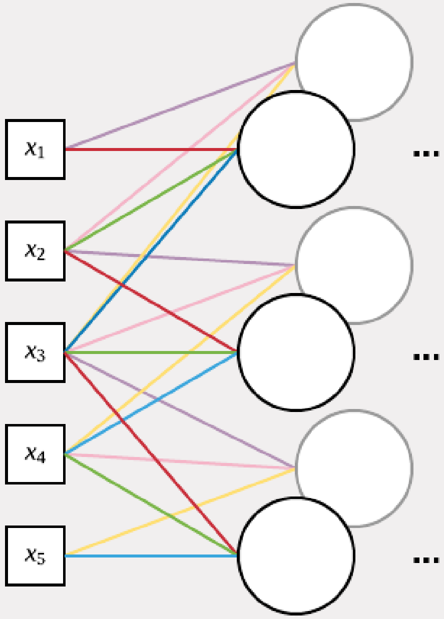
# Why does this work?

- 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).



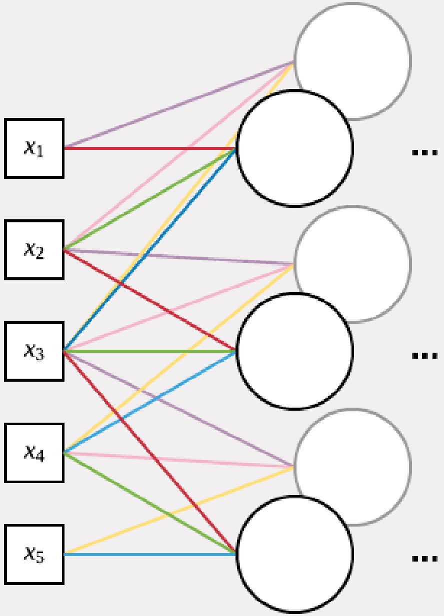

**shared weights**
3 weights




**<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.




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

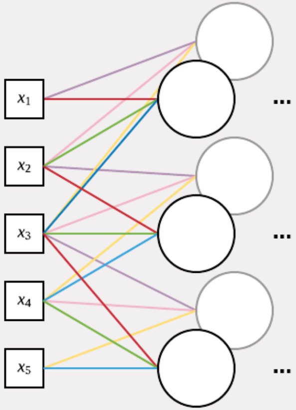
6 weights

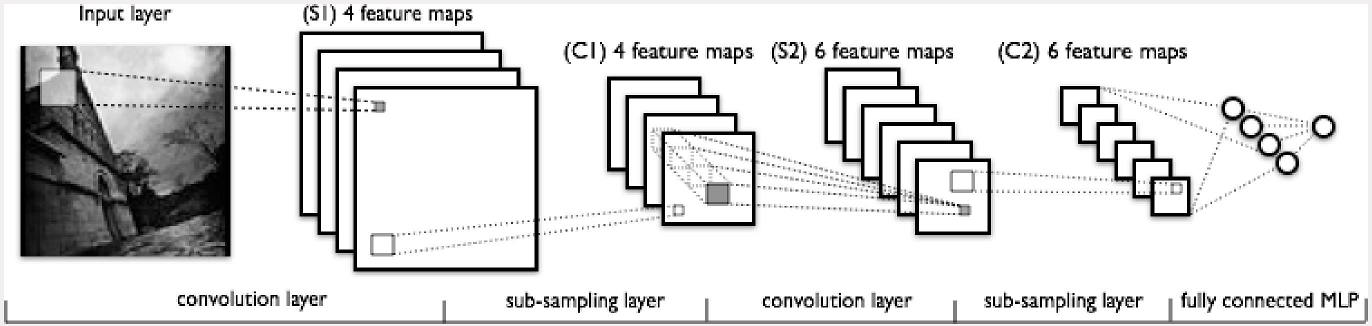
$$[a_{1,1} \quad a_{1,2} \quad a_{1,3}] = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] * [w_{1,1} \quad w_{1,2} \quad w_{1,3}]$$

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

$[w_{1,1} \quad w_{1,2} \quad w_{1,3}]$, 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.



TU/e

$$\begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 \\ I_6 & I_7 & I_8 & I_9 & I_{10} \\ I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\ I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} \end{bmatrix} * \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} =$$

$$\begin{bmatrix} \begin{matrix} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{matrix} & & \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} I_1 & \boxed{I_2} & \boxed{I_3} & \boxed{I_4} & I_5 \\ I_6 & \boxed{I_7} & \boxed{I_8} & \boxed{I_9} & I_{10} \\ I_{11} & \boxed{I_{12}} & \boxed{I_{13}} & \boxed{I_{14}} & I_{15} \\ I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} \end{bmatrix} * \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} =$$

$$\begin{bmatrix} \begin{array}{c} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{array} & \boxed{\begin{array}{c} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{array}} & \\ & & \\ & & \end{bmatrix}$$

$$
\begin{bmatrix}
I_1 & I_2 & I_3 & I_4 & I_5 \\
I_6 & I_7 & I_8 & I_9 & I_{10} \\
I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\
I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\
I_{21} & I_{22} & I_{23} & I_{24} & I_{25}
\end{bmatrix}
*
\begin{bmatrix}
w_1 & w_2 & w_3 \\
w_4 & w_5 & w_6 \\
w_7 & w_8 & w_9
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
\begin{matrix} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{matrix} &
\begin{matrix} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{matrix} &
\boxed{\begin{matrix} I_3w_1+I_4w_2+I_5w_3 \\ + \\ I_8w_4+I_9w_5+I_{10}w_6 \\ + \\ I_{13}w_7+I_{14}w_8+I_{15}w_9 \end{matrix}} \\
 & & \\
 & &
\end{bmatrix}
$$

$$
\begin{bmatrix}
I_1 & I_2 & I_3 & I_4 & I_5 \\
I_6 & I_7 & I_8 & I_9 & I_{10} \\
I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\
I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\
I_{21} & I_{22} & I_{23} & I_{24} & I_{25}
\end{bmatrix}
*
\begin{bmatrix}
w_1 & w_2 & w_3 \\
w_4 & w_5 & w_6 \\
w_7 & w_8 & w_9
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
\begin{array}{c} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{array} &
\begin{array}{c} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{array} &
\begin{array}{c} I_3w_1+I_4w_2+I_5w_3 \\ + \\ I_8w_4+I_9w_5+I_{10}w_6 \\ + \\ I_{13}w_7+I_{14}w_8+I_{15}w_9 \end{array} \\
\begin{array}{c} I_6w_1+I_7w_2+I_8w_3 \\ + \\ I_{11}w_4+I_{12}w_5+I_{13}w_6 \\ + \\ I_{16}w_7+I_{17}w_8+I_{18}w_9 \end{array} & & \\
 & &
\end{bmatrix}
$$

$$
\begin{bmatrix}
I_1 & I_2 & I_3 & I_4 & I_5 \\
I_6 & I_7 & I_8 & I_9 & I_{10} \\
I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\
I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\
I_{21} & I_{22} & I_{23} & I_{24} & I_{25}
\end{bmatrix}
*
\begin{bmatrix}
w_1 & w_2 & w_3 \\
w_4 & w_5 & w_6 \\
w_7 & w_8 & w_9
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
\begin{matrix} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{matrix} &
\begin{matrix} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{matrix} &
\begin{matrix} I_3w_1+I_4w_2+I_5w_3 \\ + \\ I_8w_4+I_9w_5+I_{10}w_6 \\ + \\ I_{13}w_7+I_{14}w_8+I_{15}w_9 \end{matrix} \\
\begin{matrix} I_6w_1+I_7w_2+I_8w_3 \\ + \\ I_{11}w_4+I_{12}w_5+I_{13}w_6 \\ + \\ I_{16}w_7+I_{17}w_8+I_{18}w_9 \end{matrix} &
\boxed{\begin{matrix} I_7w_1+I_8w_2+I_9w_3 \\ + \\ I_{12}w_4+I_{13}w_5+I_{14}w_6 \\ + \\ I_{17}w_7+I_{18}w_8+I_{19}w_9 \end{matrix}} &
 \\
 & &
\end{bmatrix}
$$

$$
\begin{bmatrix}
I_1 & I_2 & I_3 & I_4 & I_5 \\
I_6 & I_7 & I_8 & I_9 & I_{10} \\
I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\
I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\
I_{21} & I_{22} & I_{23} & I_{24} & I_{25}
\end{bmatrix}
*
\begin{bmatrix}
w_1 & w_2 & w_3 \\
w_4 & w_5 & w_6 \\
w_7 & w_8 & w_9
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
\begin{matrix} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{matrix} &
\begin{matrix} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{matrix} &
\begin{matrix} I_3w_1+I_4w_2+I_5w_3 \\ + \\ I_8w_4+I_9w_5+I_{10}w_6 \\ + \\ I_{13}w_7+I_{14}w_8+I_{15}w_9 \end{matrix} \\
\begin{matrix} I_6w_1+I_7w_2+I_8w_3 \\ + \\ I_{11}w_4+I_{12}w_5+I_{13}w_6 \\ + \\ I_{16}w_7+I_{17}w_8+I_{18}w_9 \end{matrix} &
\begin{matrix} I_7w_1+I_8w_2+I_9w_3 \\ + \\ I_{12}w_4+I_{13}w_5+I_{14}w_6 \\ + \\ I_{17}w_7+I_{18}w_8+I_{19}w_9 \end{matrix} &
\begin{matrix} I_8w_1+I_9w_2+I_{10}w_3 \\ + \\ I_{13}w_4+I_{14}w_5+I_{15}w_6 \\ + \\ I_{18}w_7+I_{19}w_8+I_{20}w_9 \end{matrix} \\
 & &
\end{bmatrix}
$$

$$\begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 \\ I_6 & I_7 & I_8 & I_9 & I_{10} \\ I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\ I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} \end{bmatrix} * \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} =$$

$$\begin{bmatrix}
\begin{matrix} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{matrix} &
\begin{matrix} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{matrix} &
\begin{matrix} I_3w_1+I_4w_2+I_5w_3 \\ + \\ I_8w_4+I_9w_5+I_{10}w_6 \\ + \\ I_{13}w_7+I_{14}w_8+I_{15}w_9 \end{matrix} \\
\begin{matrix} I_6w_1+I_7w_2+I_8w_3 \\ + \\ I_{11}w_4+I_{12}w_5+I_{13}w_6 \\ + \\ I_{16}w_7+I_{17}w_8+I_{18}w_9 \end{matrix} &
\begin{matrix} I_7w_1+I_8w_2+I_9w_3 \\ + \\ I_{12}w_4+I_{13}w_5+I_{14}w_6 \\ + \\ I_{17}w_7+I_{18}w_8+I_{19}w_9 \end{matrix} &
\begin{matrix} I_8w_1+I_9w_2+I_{10}w_3 \\ + \\ I_{13}w_4+I_{14}w_5+I_{15}w_6 \\ + \\ I_{18}w_7+I_{19}w_8+I_{20}w_9 \end{matrix} \\
\boxed{\begin{matrix} I_{11}w_1+I_{12}w_2+I_{13}w_3 \\ + \\ I_{16}w_4+I_{17}w_5+I_{18}w_6 \\ + \\ I_{21}w_7+I_{22}w_8+I_{23}w_9 \end{matrix}} & & 
\end{bmatrix}$$

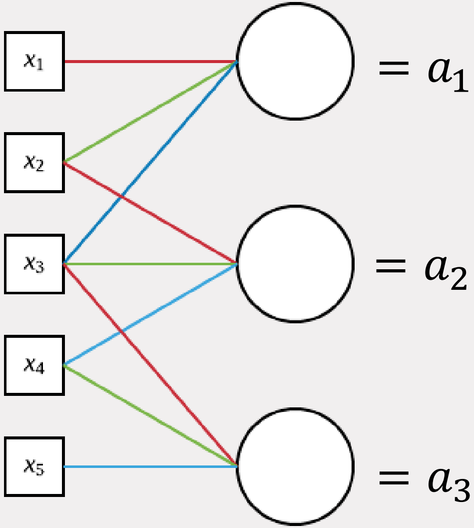
$$\begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 \\ I_6 & I_7 & I_8 & I_9 & I_{10} \\ I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\ I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} \end{bmatrix} * \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} =$$

$$\begin{bmatrix}
\begin{array}{c} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{array} &
\begin{array}{c} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{array} &
\begin{array}{c} I_3w_1+I_4w_2+I_5w_3 \\ + \\ I_8w_4+I_9w_5+I_{10}w_6 \\ + \\ I_{13}w_7+I_{14}w_8+I_{15}w_9 \end{array} \\
\begin{array}{c} I_6w_1+I_7w_2+I_8w_3 \\ + \\ I_{11}w_4+I_{12}w_5+I_{13}w_6 \\ + \\ I_{16}w_7+I_{17}w_8+I_{18}w_9 \end{array} &
\begin{array}{c} I_7w_1+I_8w_2+I_9w_3 \\ + \\ I_{12}w_4+I_{13}w_5+I_{14}w_6 \\ + \\ I_{17}w_7+I_{18}w_8+I_{19}w_9 \end{array} &
\begin{array}{c} I_8w_1+I_9w_2+I_{10}w_3 \\ + \\ I_{13}w_4+I_{14}w_5+I_{15}w_6 \\ + \\ I_{18}w_7+I_{19}w_8+I_{20}w_9 \end{array} \\
\begin{array}{c} I_{11}w_1+I_{12}w_2+I_{13}w_3 \\ + \\ I_{16}w_4+I_{17}w_5+I_{18}w_6 \\ + \\ I_{21}w_7+I_{22}w_8+I_{23}w_9 \end{array} &
\boxed{\begin{array}{c} I_{12}w_1+I_{13}w_2+I_{14}w_3 \\ + \\ I_{17}w_4+I_{18}w_5+I_{19}w_6 \\ + \\ I_{22}w_7+I_{23}w_8+I_{24}w_9 \end{array}} &
\end{bmatrix}$$

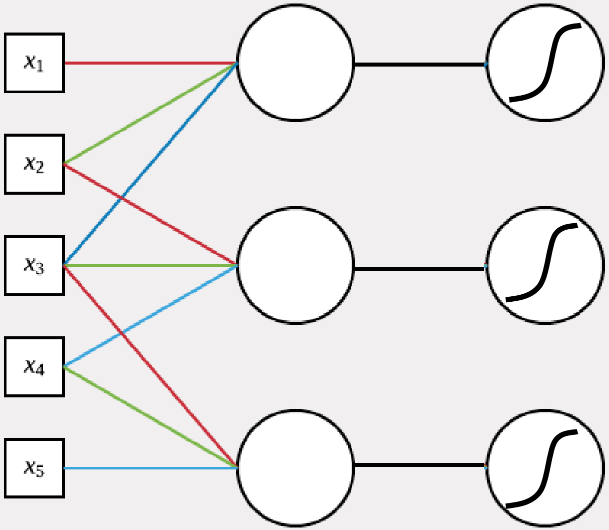
$$
\begin{bmatrix}
I_1 & I_2 & I_3 & I_4 & I_5 \\
I_6 & I_7 & I_8 & I_9 & I_{10} \\
I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\
I_{16} & I_{17} & I_{18} & I_{19} & I_{20} \\
I_{21} & I_{22} & I_{23} & I_{24} & I_{25}
\end{bmatrix}
*
\begin{bmatrix}
w_1 & w_2 & w_3 \\
w_4 & w_5 & w_6 \\
w_7 & w_8 & w_9
\end{bmatrix}
+ b =
$$

$$
\begin{bmatrix}
\begin{matrix} I_1w_1+I_2w_2+I_3w_3 \\ + \\ I_6w_4+I_7w_5+I_8w_6 \\ + \\ I_{11}w_7+I_{12}w_8+I_{13}w_9 \end{matrix} &
\begin{matrix} I_2w_1+I_3w_2+I_4w_3 \\ + \\ I_7w_4+I_8w_5+I_9w_6 \\ + \\ I_{12}w_7+I_{13}w_8+I_{14}w_9 \end{matrix} &
\begin{matrix} I_3w_1+I_4w_2+I_5w_3 \\ + \\ I_8w_4+I_9w_5+I_{10}w_6 \\ + \\ I_{13}w_7+I_{14}w_8+I_{15}w_9 \end{matrix} \\
\begin{matrix} I_6w_1+I_7w_2+I_8w_3 \\ + \\ I_{11}w_4+I_{12}w_5+I_{13}w_6 \\ + \\ I_{16}w_7+I_{17}w_8+I_{18}w_9 \end{matrix} &
\begin{matrix} I_7w_1+I_8w_2+I_9w_3 \\ + \\ I_{12}w_4+I_{13}w_5+I_{14}w_6 \\ + \\ I_{17}w_7+I_{18}w_8+I_{19}w_9 \end{matrix} &
\begin{matrix} I_8w_1+I_9w_2+I_{10}w_3 \\ + \\ I_{13}w_4+I_{14}w_5+I_{15}w_6 \\ + \\ I_{18}w_7+I_{19}w_8+I_{20}w_9 \end{matrix} \\
\begin{matrix} I_{11}w_1+I_{12}w_2+I_{13}w_3 \\ + \\ I_{16}w_4+I_{17}w_5+I_{18}w_6 \\ + \\ I_{21}w_7+I_{22}w_8+I_{23}w_9 \end{matrix} &
\begin{matrix} I_{12}w_1+I_{13}w_2+I_{14}w_3 \\ + \\ I_{17}w_4+I_{18}w_5+I_{19}w_6 \\ + \\ I_{22}w_7+I_{23}w_8+I_{24}w_9 \end{matrix} &
\boxed{\begin{matrix} I_{13}w_1+I_{14}w_2+I_{15}w_3 \\ + \\ I_{18}w_4+I_{19}w_5+I_{20}w_6 \\ + \\ I_{23}w_7+I_{24}w_8+I_{25}w_9 \end{matrix}}
\end{bmatrix}
+ b
$$

# Convolutions + non-linearity



# Convolutions + non-linearity (sigmoid)

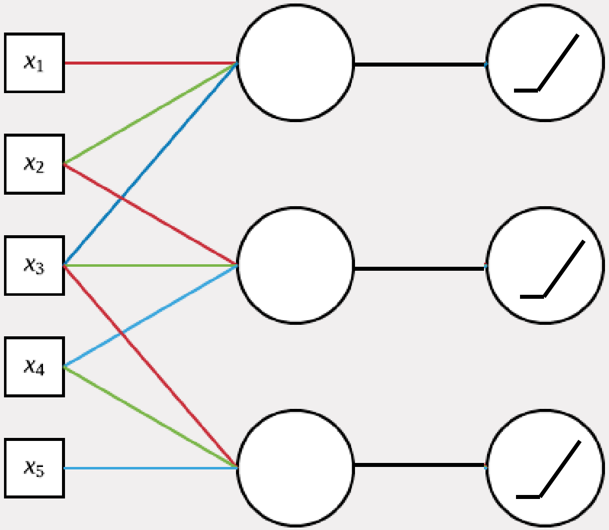



$= sigm(a_1)$

$= sigm(a_2)$

$= sigm(a_3)$

# Convolutions + non-linearity (ReLU)




$= ReLU(a_1)$

$= ReLU(a_2)$

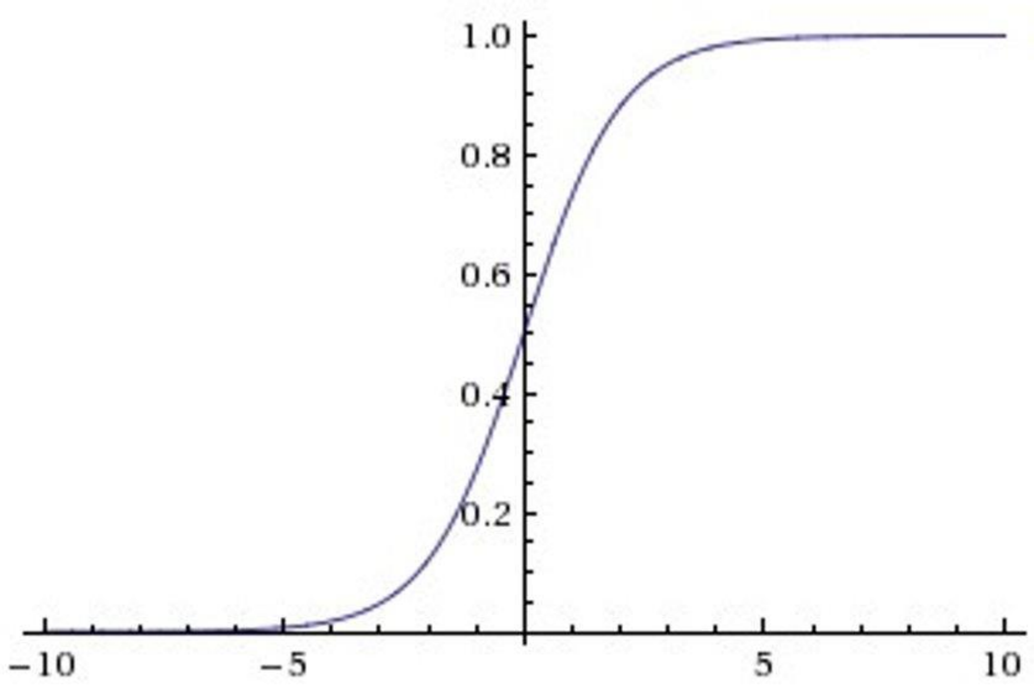
$= ReLU(a_3)$

# Commonly used activation Functions

- Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it.
- There are several activation functions you may encounter in practice.

Credits : https://cs231n.github.io/

# Sigmoid



# Sigmoid

- The sigmoid non-linearity has the mathematical form $\sigma(x)=1/(1+e-x)$.
- It takes a real-valued number and “squashes” it into range between 0 and 1.
- In particular, large negative numbers become 0 and large positive numbers become 1.
- The sigmoid function has seen frequent use historically since it has a nice interpretation as the firing rate of a neuron: from not firing at all (0) to fully-saturated firing at an assumed maximum frequency (1).
- In practice, the sigmoid non-linearity has recently fallen out of favor and it is rarely ever used.

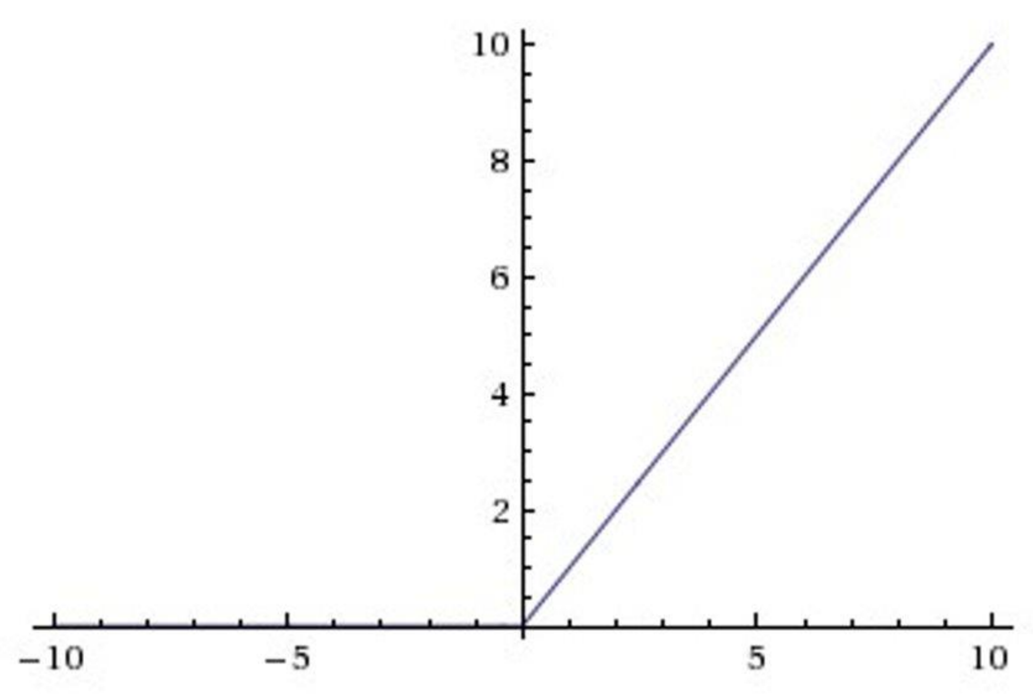
Credits : https://cs231n.github.io/

# Drawbacks of Sigmoid

- A very undesirable property of the sigmoid neuron is that when the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero.
- This (local) gradient will be multiplied to the gradient of this gate's output for the whole objective.
- Therefore, if the local gradient is very small, it will effectively "kill" the gradient and almost no signal will flow through the neuron to its weights and recursively to its data.
- Additionally, one must pay extra caution when initializing the weights of sigmoid neurons to prevent saturation.
- For example, if the initial weights are too large then most neurons would become saturated and the network will barely learn.

Credits : https://cs231n.github.io/

# ReLU



# ReLU

- The Rectified Linear Unit has become very popular in the last few years.
- It computes the function $f(x)=\max(0,x)$.
- In other words, the activation is simply thresholded at zero.

Credits : https://cs231n.github.io/

# ReLU

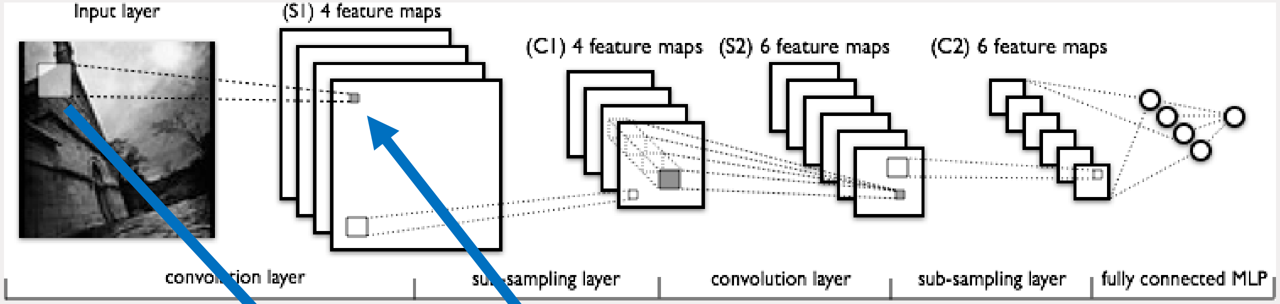
- It was found to greatly accelerate (e.g. a factor of 6 in Krizhevsky et al.) the convergence of stochastic gradient descent compared to the sigmoid/tanh functions.
- It is argued that this is due to its linear, non-saturating form.
- Compared to tanh/sigmoid neurons that involve expensive operations (exponentials, etc.), the ReLU can be implemented by simply thresholding a matrix of activations at zero.

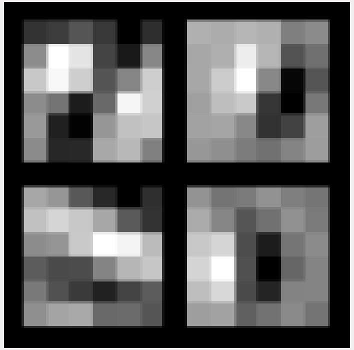
Credits : https://cs231n.github.io/

# Break!

Quokka (Australia)
https://imgur.com/r/aww/GqJi0iI

# Convolutional kernels



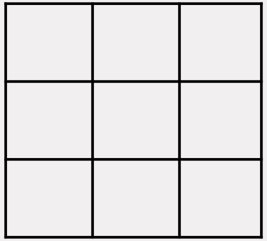



these are the weights (convolutional kernels) that produce the four feature maps

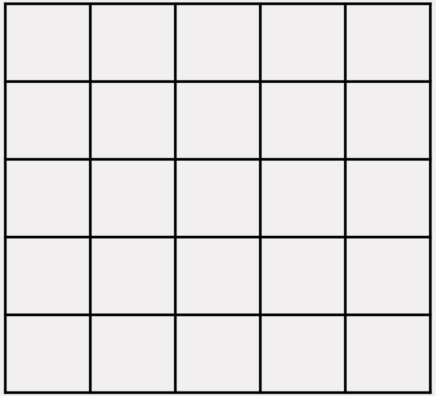
# Kernel size

1 x 1

3 x 3

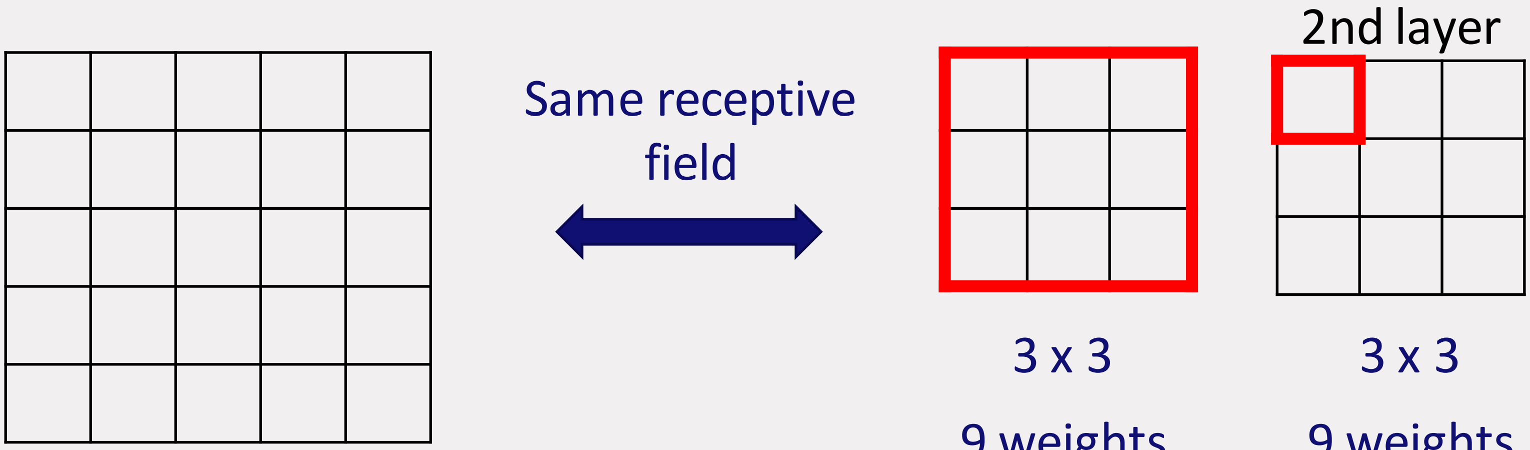


5 x 5



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

# Receptive field




5 x 5

25 weights

3 x 3

9 weights

3 x 3

9 weights

= 18 weights

# Receptive Field

https://distill.pub/2019/computing-receptive-fields/

## Computing Receptive Fields of Convolutional Neural Networks

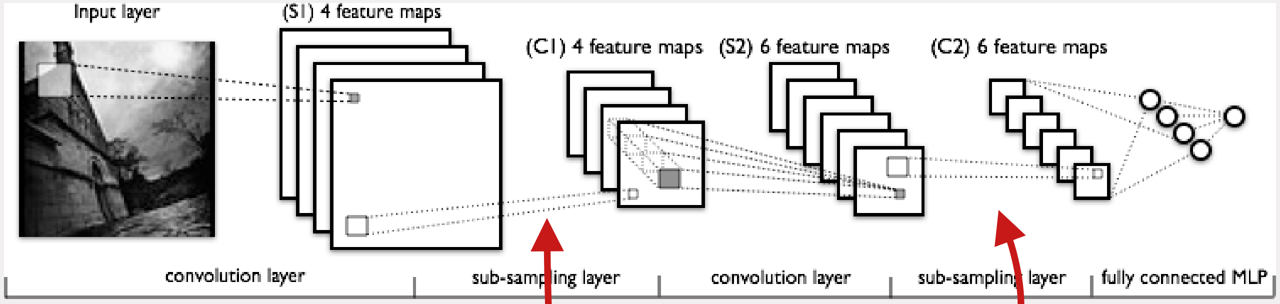
Mathematical derivations and open-source library to compute receptive fields of convnets, enabling the mapping of extracted features to input signals.

# Stride

- We must specify the **stride** with which we slide the filter.
- When the stride is 1 then we move the filters one pixel at a time.
- When the stride is 2 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around.
- This will produce smaller output volumes spatially.

Credits : https://cs231n.github.io/

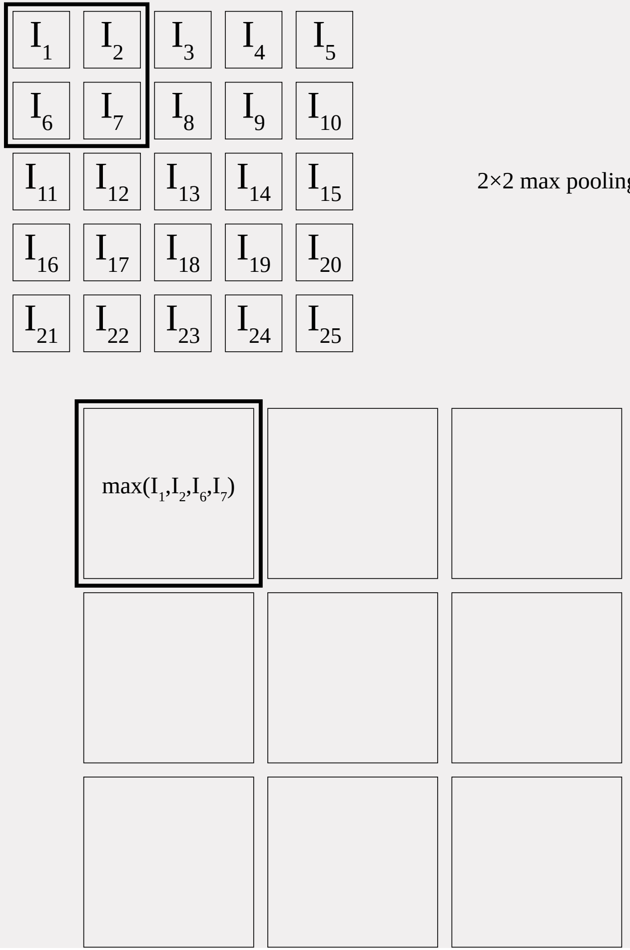
# Max-pooling



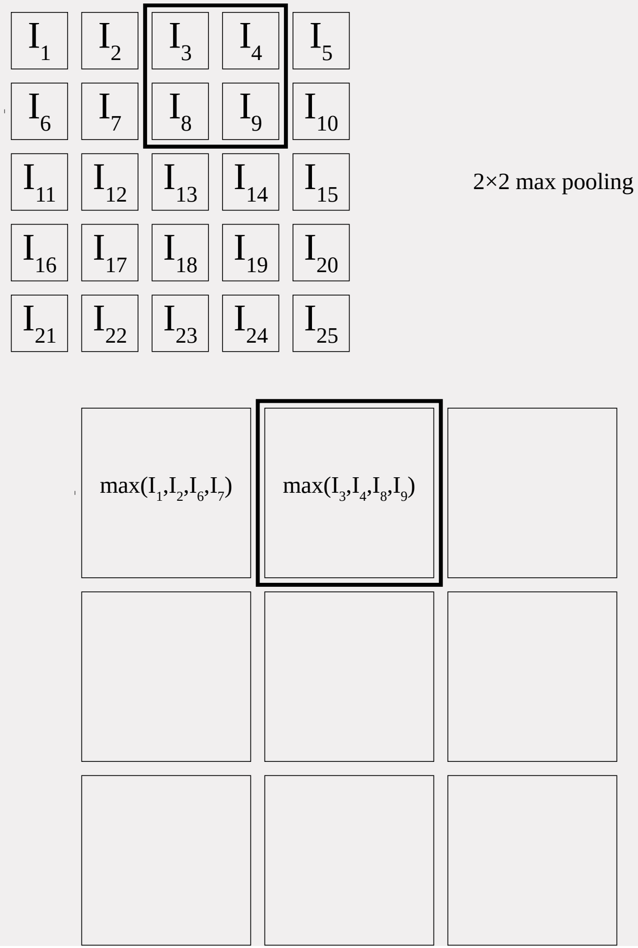

**max-pooling**

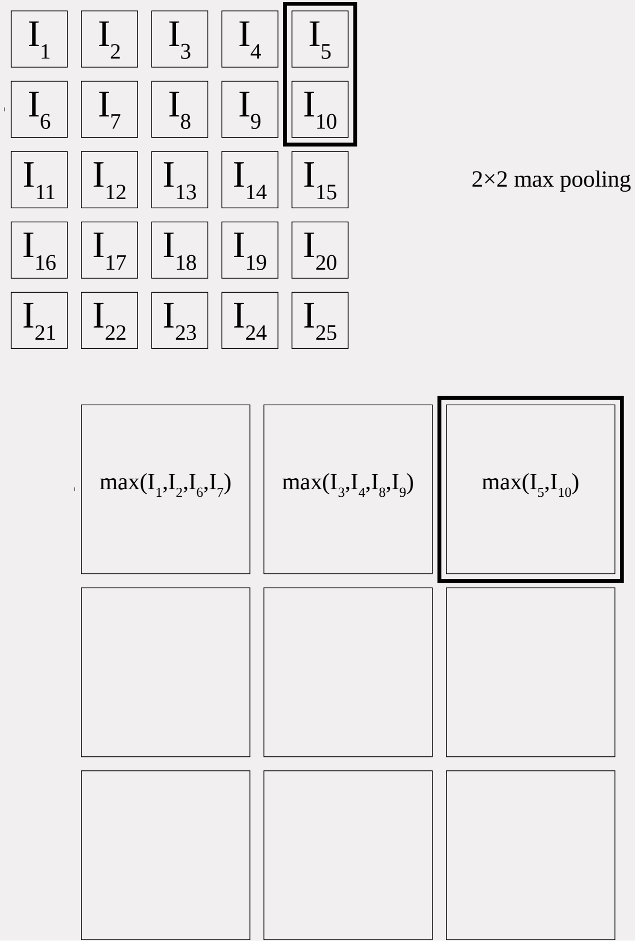
# Max-pooling

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

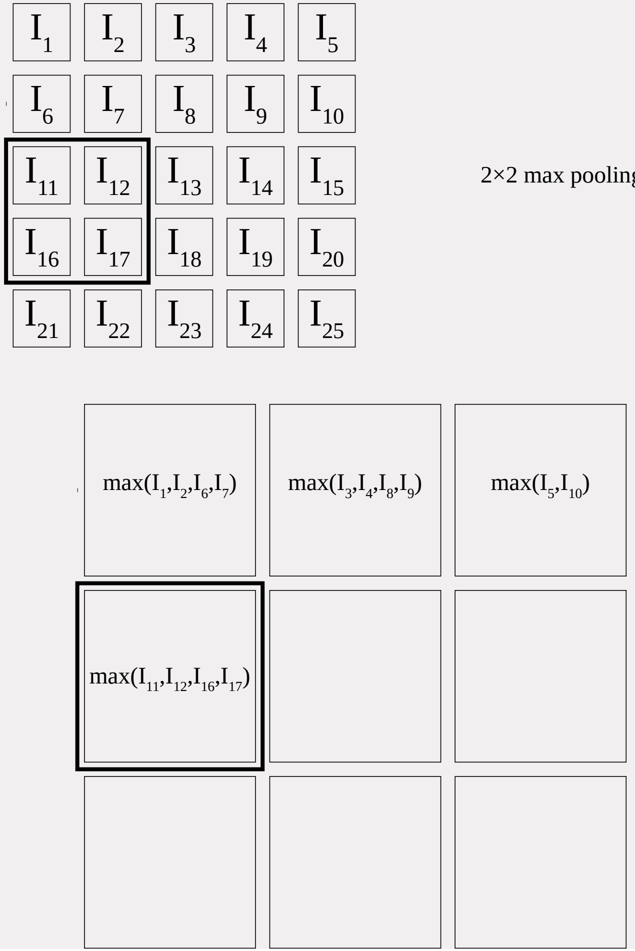



2×2 max pooling

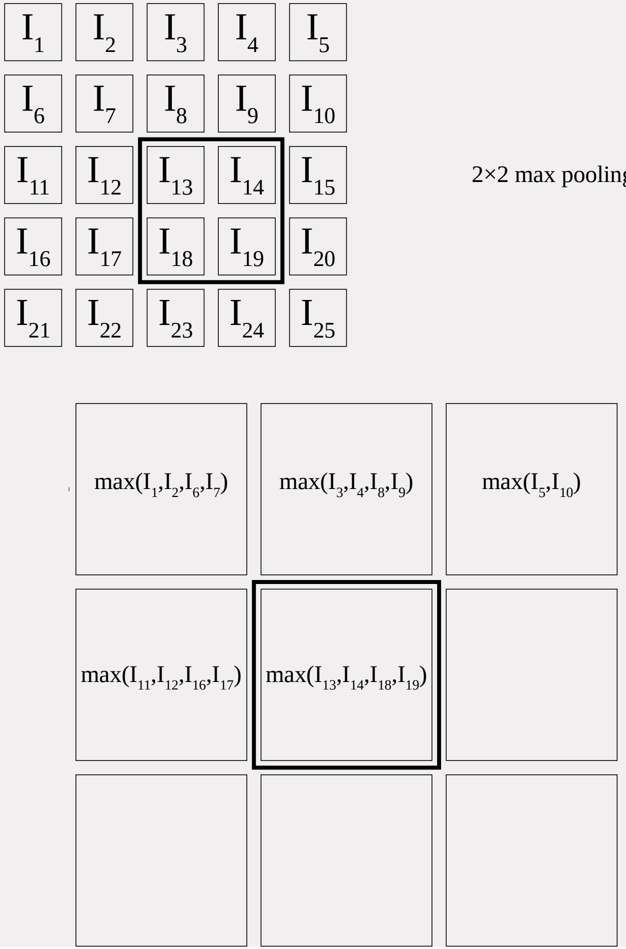

$I_1$ $I_2$ $I_3$ $I_4$ $I_5$
$I_6$ $I_7$ $I_8$ $I_9$ $I_{10}$
$I_{11}$ $I_{12}$ $I_{13}$ $I_{14}$ $I_{15}$
$I_{16}$ $I_{17}$ $I_{18}$ $I_{19}$ $I_{20}$
$I_{21}$ $I_{22}$ $I_{23}$ $I_{24}$ $I_{25}$
2×2 max pooling
$\max(I_1,I_2,I_6,I_7)$
$\max(I_3,I_4,I_8,I_9)$




In this example:
no zero padding




2×2 max pooling




2×2 max pooling

| | | | | |
|---|---|---|---|---|
| $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ |
| $I_6$ | $I_7$ | $I_8$ | $I_9$ | $I_{10}$ |
| $I_{11}$ | $I_{12}$ | $I_{13}$ | $I_{14}$ | $I_{15}$ |
| $I_{16}$ | $I_{17}$ | $I_{18}$ | $I_{19}$ | $I_{20}$ |
| $\boxed{I_{21}\ I_{22}}$ | | $I_{23}$ | $I_{24}$ | $I_{25}$ |

2×2 max pooling

| | | |
|---|---|---|
| $\max(I_1,I_2,I_6,I_7)$ | $\max(I_3,I_4,I_8,I_9)$ | $\max(I_5,I_{10})$ |
| $\max(I_{11},I_{12},I_{16},I_{17})$ | $\max(I_{13},I_{14},I_{18},I_{19})$ | $\max(I_{15},I_{20})$ |
| $\boxed{\max(I_{21},I_{22})}$ | | |

| | | | | |
|---|---|---|---|---|
| $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ |
| $I_6$ | $I_7$ | $I_8$ | $I_9$ | $I_{10}$ |
| $I_{11}$ | $I_{12}$ | $I_{13}$ | $I_{14}$ | $I_{15}$ |
| $I_{16}$ | $I_{17}$ | $I_{18}$ | $I_{19}$ | $I_{20}$ |
| $I_{21}$ | $I_{22}$ | $\boxed{I_{23}}$ | $\boxed{I_{24}}$ | $I_{25}$ |

2×2 max pooling

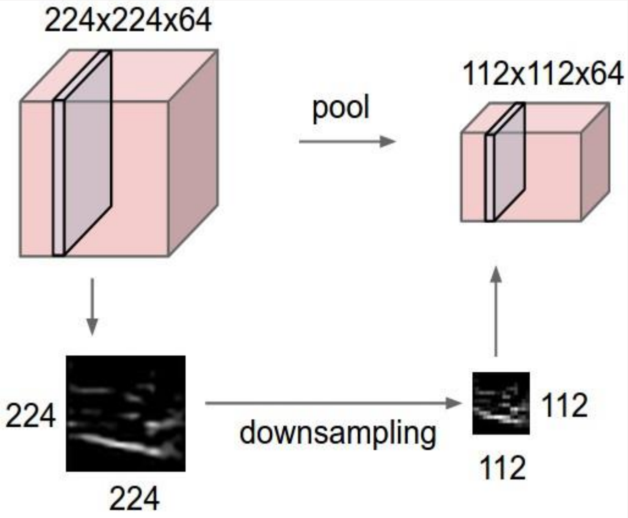
| | | |
|---|---|---|
| $\max(I_1,I_2,I_6,I_7)$ | $\max(I_3,I_4,I_8,I_9)$ | $\max(I_5,I_{10})$ |
| $\max(I_{11},I_{12},I_{16},I_{17})$ | $\max(I_{13},I_{14},I_{18},I_{19})$ | $\max(I_{15},I_{20})$ |
| $\max(I_{21},I_{22})$ | $\boxed{\max(I_{23},I_{24})}$ | |

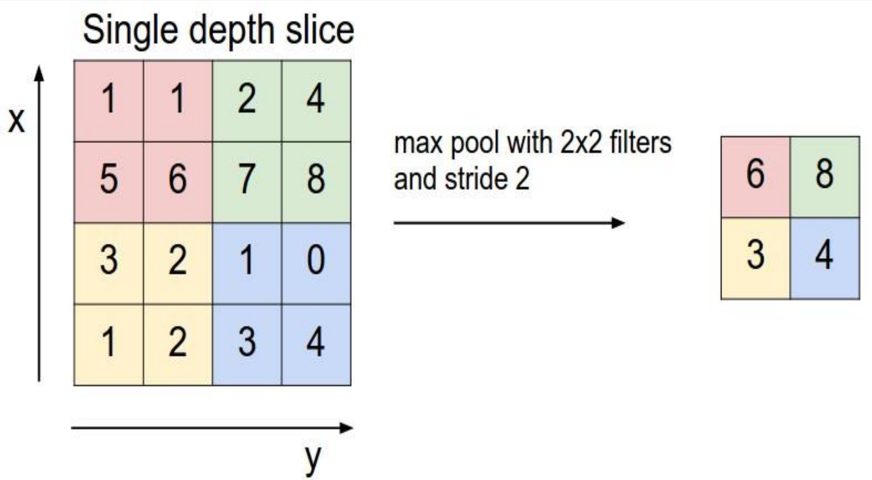
| | | | | |
|---|---|---|---|---|
| $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ |
| $I_6$ | $I_7$ | $I_8$ | $I_9$ | $I_{10}$ |
| $I_{11}$ | $I_{12}$ | $I_{13}$ | $I_{14}$ | $I_{15}$ |
| $I_{16}$ | $I_{17}$ | $I_{18}$ | $I_{19}$ | $I_{20}$ |
| $I_{21}$ | $I_{22}$ | $I_{23}$ | $I_{24}$ | $\boxed{I_{25}}$ |

2×2 max pooling

| | | |
|---|---|---|
| $\max(I_1,I_2,I_6,I_7)$ | $\max(I_3,I_4,I_8,I_9)$ | $\max(I_5,I_{10})$ |
| $\max(I_{11},I_{12},I_{16},I_{17})$ | $\max(I_{13},I_{14},I_{18},I_{19})$ | $\max(I_{15},I_{20})$ |
| $\max(I_{21},I_{22})$ | $\max(I_{23},I_{24})$ | $\boxed{I_{25}}$ |

# Example:




| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square). https://cs231n.github.io/

# 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

# Demo

https://poloclub.github.io/cnn-explainer/

# 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

Through a process called ‘backpropagation’. A good explanation can be found here: [https://www.youtube.com/watch?v=i94OvYb6noo](https://www.youtube.com/watch?v=i94OvYb6noo) (5:10 - 28:00, not exam material)

- 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**