Introduction to deep learning 1/?

Esten Høyland Leonardsen

02.11.22

UiO:Life Science, University of Oslo

Introduction

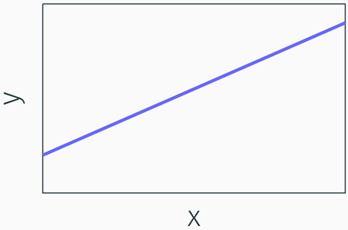
1. Building an artificial neural network (ANN)
2. Training the ANN
3. Transformation to a Convolutional Neural Network (CNN)

Building a neural network: Linear regression

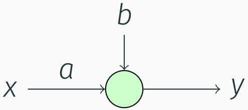
$$y = ax + b$$

Building a neural network: Linear regression

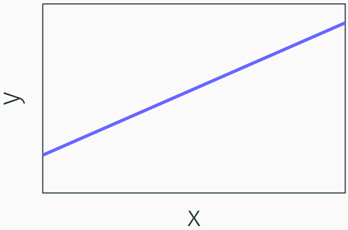
$$y = ax + b$$



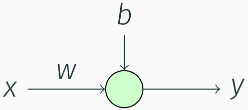
Building a neural network: Linear regression



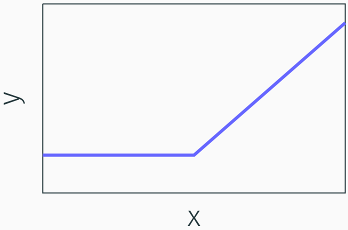
$$y = ax + b$$



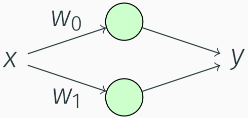
Building a neural network: Artificial neuron



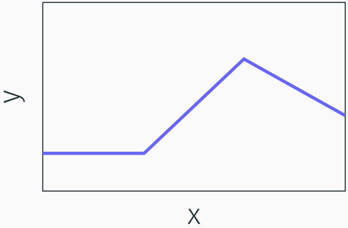
$$y = max(0, wx + b)$$



Building a neural network: Artificial neural network (ANN)



$$y = max(0, w_0 x + b_0) + max(0, w_1 x + b_1)$$



Building a neural network: Universal approximation theorem

”Any relationship that can be described with a polynomial function can be approximated by a neural network with a single hidden layer.”

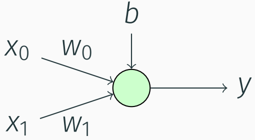
- Some guy in the 80s, probably

Building a neural network: Universal approximation theorem

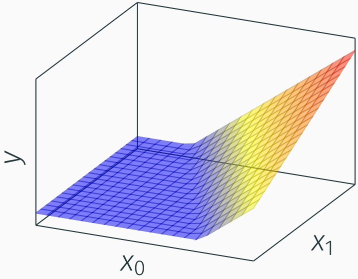
"Any relationship that can be described with a polynomial function can be approximated by a neural network with a single hidden layer."

- Some guy in the 80s, probably

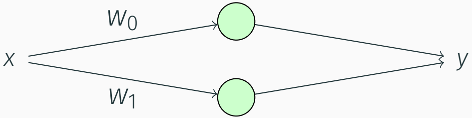
Building a neural network: Increasing dimensionality



$$y = max(0, w_0 x_0 + w_1 x_1 + b)$$

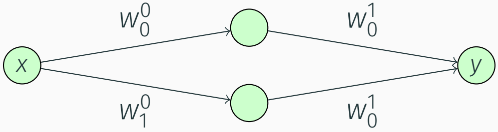


Building a neural network



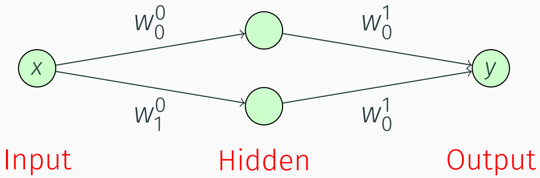
$$y = max(0, w_0 x + b_0) + max(0, w_1 x + b_1)$$

Building a neural network



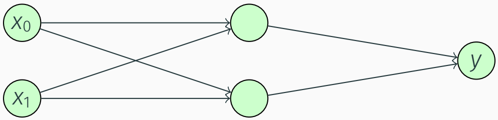
$$y = max(0, w^1_{0,0} * max(0, w^0_{0,0} * x + b_{0,0}) + w^1_{1,0} * max(0, w^0_{0,1} * x + b_{1,0}) + b_1)$$

Building a neural network



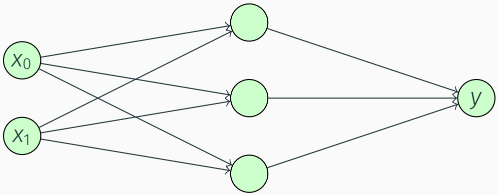
$y = max(0, w^1_{0,0} * max(0, w^0_{0,0} * x + b_{0,0}) + w^1_{1,0} * max(0, w^0_{0,1} * x + b_{1,0}) + b_1)$

Building a neural network



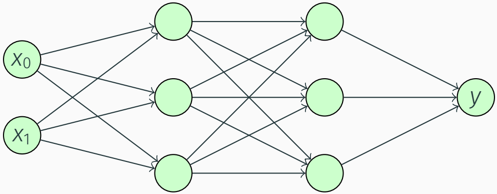
$$
\begin{aligned}
y = max(0, & w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\
& w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\
& b_1)
\end{aligned}
$$

Building a neural network



$$
\begin{aligned}
y = max(0, & w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0}) + \\
& w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^0_{1,1} * x_1 + b_{0,1}) + \\
& w^1_{2,0} * max(0, w^0_{0,2} * x_0 + w^0_{1,2} * x_1 + b_{0,2}) + \\
& b_1)
\end{aligned}
$$

Building a neural network

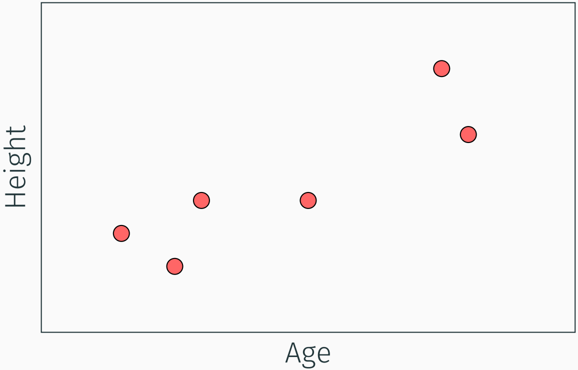


$$
\begin{aligned}
y = max(0, & w^2_{0,0} * max(0, w^1_{0,0} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0})+ \\
& \qquad w^1_{1,0} * max(0, w^0_{0,1} * x_0 + w^+_{1,1} * w_1 + b_{0,1})+ \\
& \qquad w^1_{2,0} * max(0, w^0_{0,2} * x_0 + w^+_{1,2} * w_1 + b_{0,2})+ \\
& \qquad b_{1,0})+ \\
& w^2_{1,0} * max(0, w^1_{0,1} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0})+ \\
& \qquad w^1_{1,1} * max(0, w^0_{0,1} * x_0 + w^+_{1,1} * w_1 + b_{0,1})+ \\
& \qquad w^1_{2,1} * max(0, w^0_{0,2} * x_0 + w^+_{1,2} * w_1 + b_{0,2})+ \\
& \qquad b_{1,1})+ \\
& w^2_{2,0} * max(0, w^1_{0,2} * max(0, w^0_{0,0} * x_0 + w^0_{1,0} * x_1 + b_{0,0})+ \\
& \qquad w^1_{1,2} * max(0, w^0_{0,1} * x_0 + w^+_{1,1} * w_1 + b_{0,1})+ \\
& \qquad w^1_{2,2} * max(0, w^0_{0,2} * x_0 + w^+_{1,2} * w_1 + b_{0,2})+ \\
& \qquad b_{1,2})+ \\
& b_2)
\end{aligned}
$$

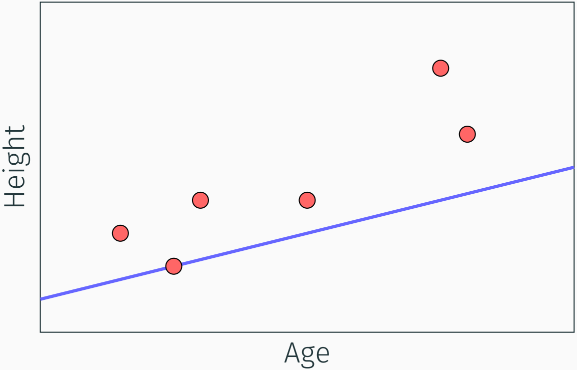
Building a neural network: Summary

- Artificial neurons are (linear) weighted sums wrapped in non-linear activation functions
- Multiple artificial neurons stacked together in a layerwise fashion comprise an artificial neural network
- Artificial neural networks allow us to model arbitrarily complex relationships between inputs and outputs

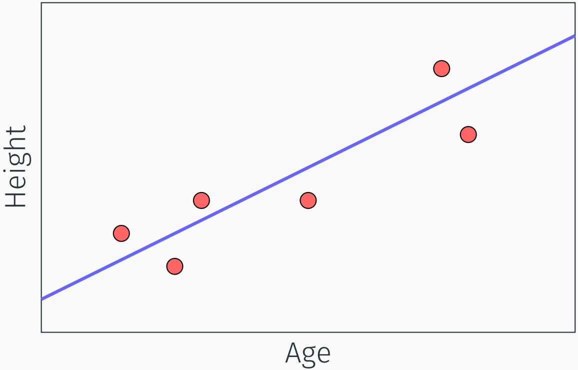
Training a neural network: Loss functions



Training a neural network: Loss functions

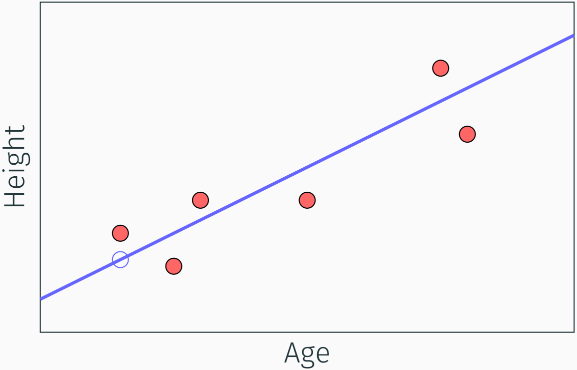


$$\hat{y} = 0.4x + 0.1$$



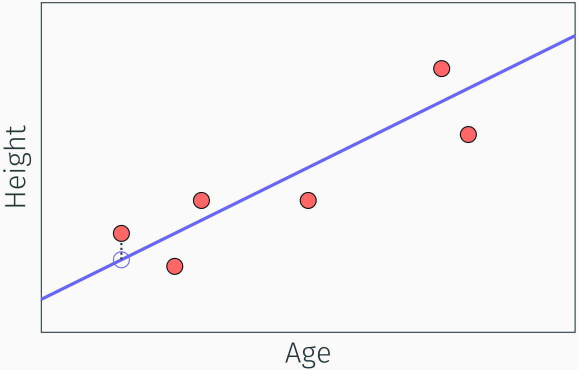
$$\hat{y} = 0.9x + 0.1$$

Training a neural network: Loss functions



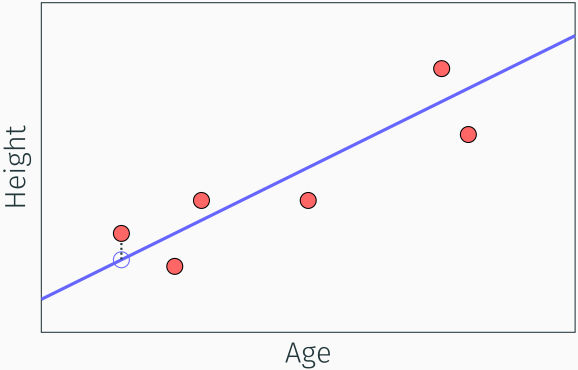
$$\hat{y} = 0.9x + 0.1$$

Training a neural network: Loss functions



$$\hat{y} = 0.9x + 0.1$$

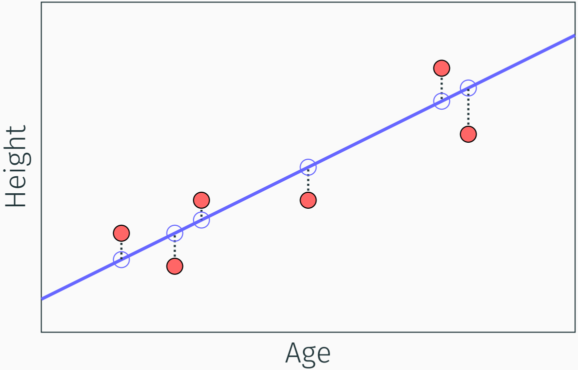
Training a neural network: Loss functions



$$\hat{y} = 0.9x + 0.1$$

$$error_i = |\hat{y} - y|$$

Training a neural network: Loss functions

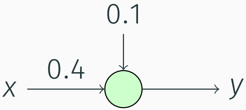


$$\hat{y} = 0.9x + 0.1$$

$$error_i = |\hat{y} - y|$$

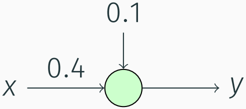
$$mse = \frac{1}{n} \sum_{i<n} (\hat{y}_i - y_i)^2$$

Training a neural network: Gradient descent



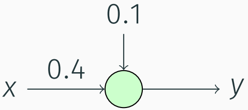
$$y = 0.4x + 0.1 \qquad loss = (y - \hat{y})^2$$

Training a neural network: Gradient descent



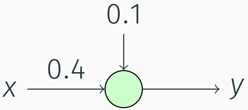
$$0.16 = 0.4 * 0.15 + 0.1 \qquad 0.019 = (0.3 - 0.16)^2$$

Training a neural network: Gradient descent



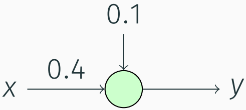
$$(0.3 - (0.4 * 0.15 + 0.1))^2 = 0.019$$

Training a neural network: Gradient descent

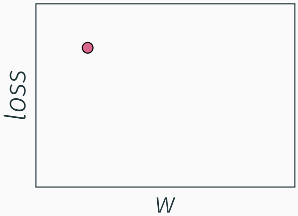


$$(\underset{y}{0.3} - (\underbrace{\overset{w}{0.4} * \overset{x}{0.15} + \overset{b}{0.1}}_{\hat{y}}))^2 = \overset{loss}{0.019}$$

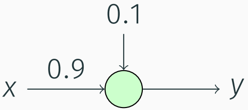
Training a neural network: Gradient descent



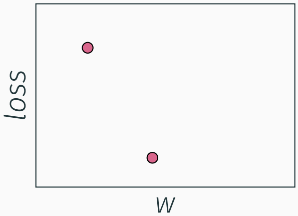
$$(0.3 - (0.4 * 0.15 + 0.1))^2 = 0.019$$



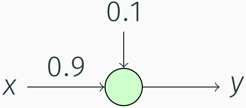
Training a neural network: Gradient descent



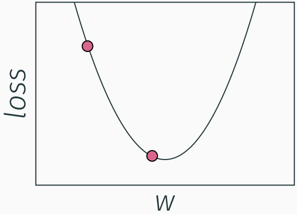
$$(0.3 - (0.9 * 0.15 + 0.1))^2 = 0.004$$



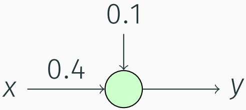
Training a neural network: Gradient descent



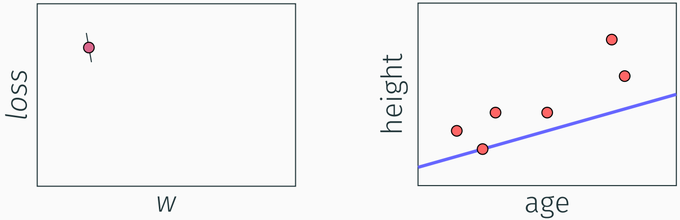
$$(0.3 - (0.9 * 0.15 + 0.1))^2 = 0.004$$



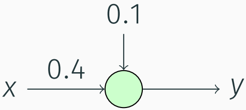
Training a neural network: Gradient descent



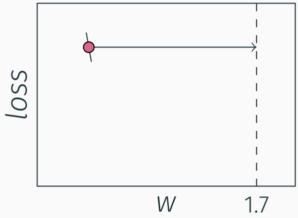
$$(0.3 - (0.4 * 0.15 + 0.1))^2 = 0.019$$

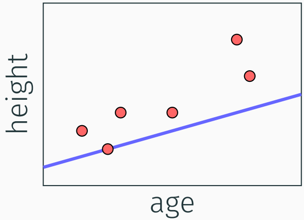


Training a neural network: Gradient descent

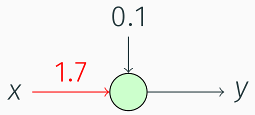


$$(0.3 - (0.4 * 0.15 + 0.1))^2 = 0.019$$

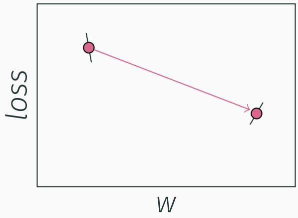


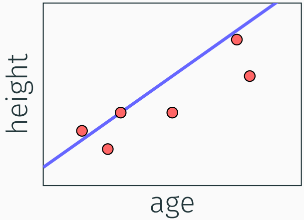


Training a neural network: Gradient descent

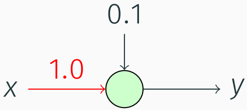


$$(0.3 - ({\color{red}1.7} * 0.15 + 0.1))^2 = 0.003$$

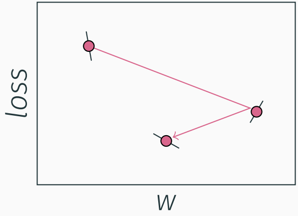


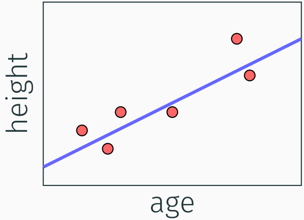


Training a neural network: Gradient descent

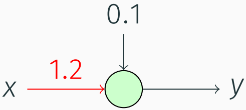


$$(0.3 - ({\color{red}1.0}*0.15 + 0.1))^2 = 0.002$$

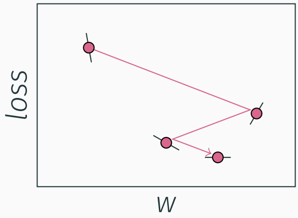


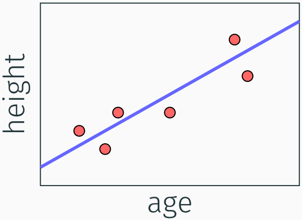


Training a neural network: Gradient descent



$$(0.3 - ({\color{red}1.2} * 0.15 + 0.1))^2 = 0.000$$

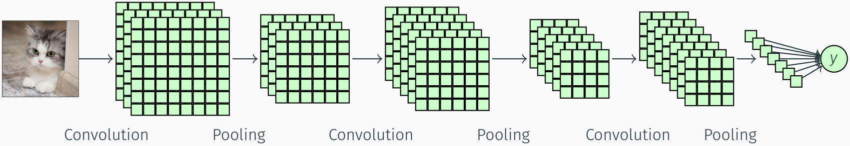




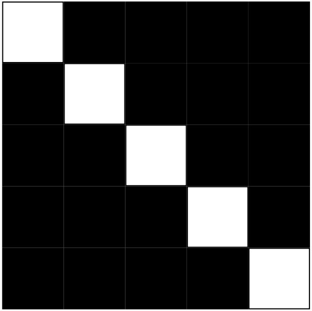
Training a neural network: Summary

- The loss function is a precise formalization of what we want the model to learn
- Calculating the gradient allows us to see how we can update the parameters of the model to decrease the loss
- Using gradient descent we can update the parameters sequentially until we have the perfect model with zero loss

Convolutional neural networks: Architecture

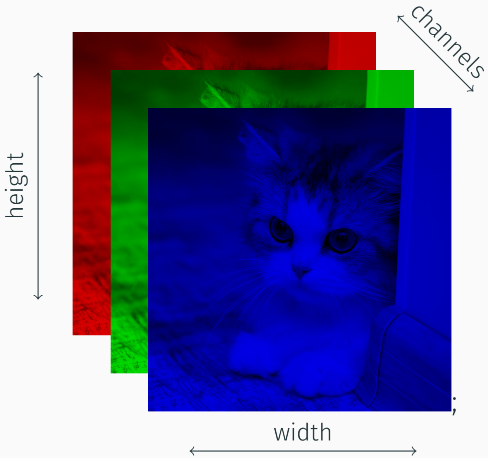


Convolutional neural networks: Images



1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Convolutional neural networks: Images



;

Convolutional neural networks: Convolution

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 0 \\ \hline \end{array} \ast \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$$

Convolutional neural networks: Convolution

$$\begin{array}{|c|c|c|}\hline \boxed{0} & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 0 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|}\hline \boxed{1} & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$$

0*1

Convolutional neural networks: Convolution

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

0*1+0*0

Convolutional neural networks: Convolution

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

0*1+0*0+1*0

Convolutional neural networks: Convolution

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

0*1+0*0+1*0+0*0

Convolutional neural networks: Convolution

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

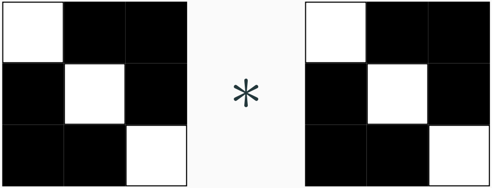
0*1+0*0+1*0+0*0+1*1+0*0+1*0+0*0+0*1=1

Convolutional neural networks: Convolution

$$\begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array} \ast \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$$

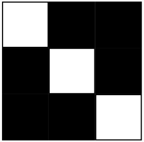
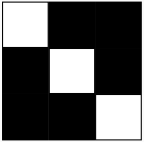
1*1+0*0+0*0+0*0+1*1+0*0+1*0+0*0+1*1=3

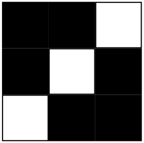
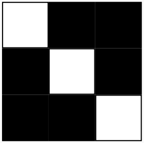
Convolutional neural networks: Convolution

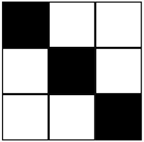
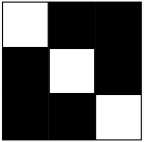


1*1+0*0+0*0+0*0+1*1+0*0+1*0+0*0+1*1=3

Convolutional neural networks: Convolutional layers

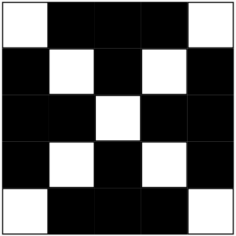
 $*$  $= 3$

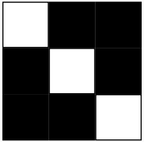
 $*$  $= 1$

 $*$  $= 0$

Convolutional neural networks: Convolutional layers

Image

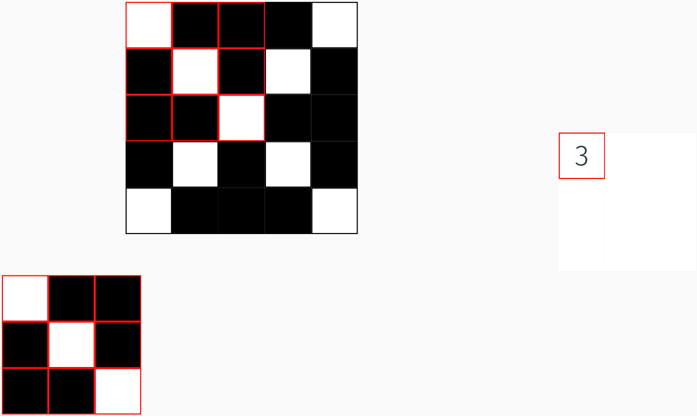




Pattern 1

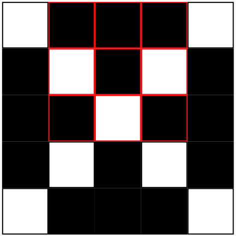
Convolutional neural networks: Convolutional layers

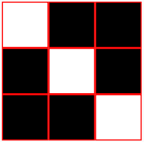
Image



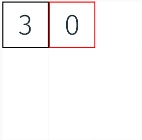
Pattern 1

Image

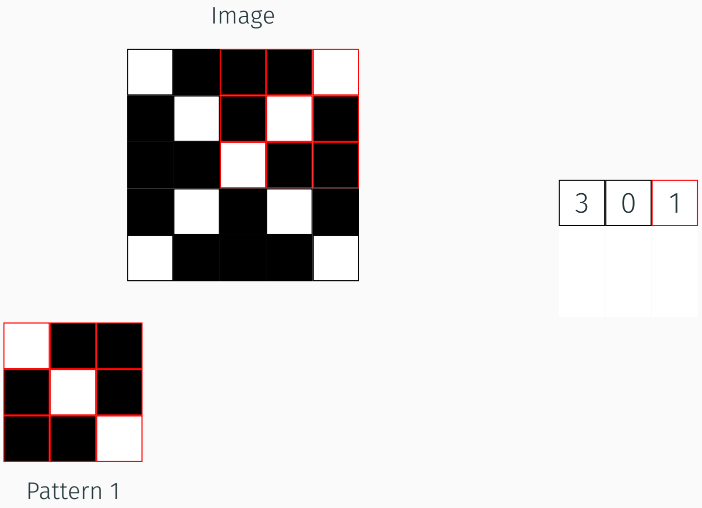




Pattern 1

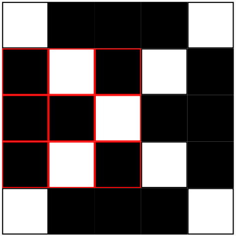


Convolutional neural networks: Convolutional layers

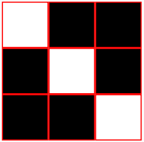


Convolutional neural networks: Convolutional layers

Image



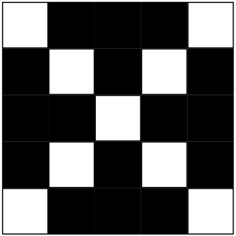
3	0	1
0		



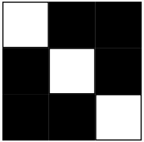
Pattern 1

Convolutional neural networks: Convolutional layers

Image



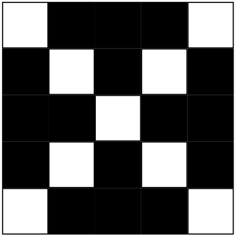
3	0	1
0	3	0
1	0	3

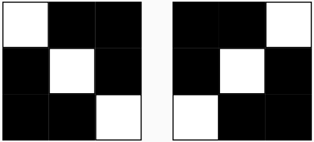


Pattern 1

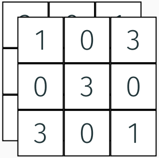
Convolutional neural networks: Convolutional layers

Image



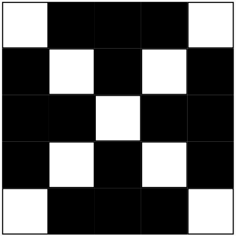


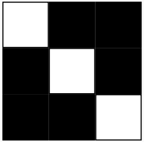
Pattern 1 Pattern 2



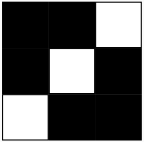
Convolutional neural networks: Convolutional layers

Image

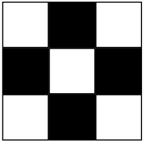




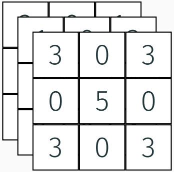
Pattern 1



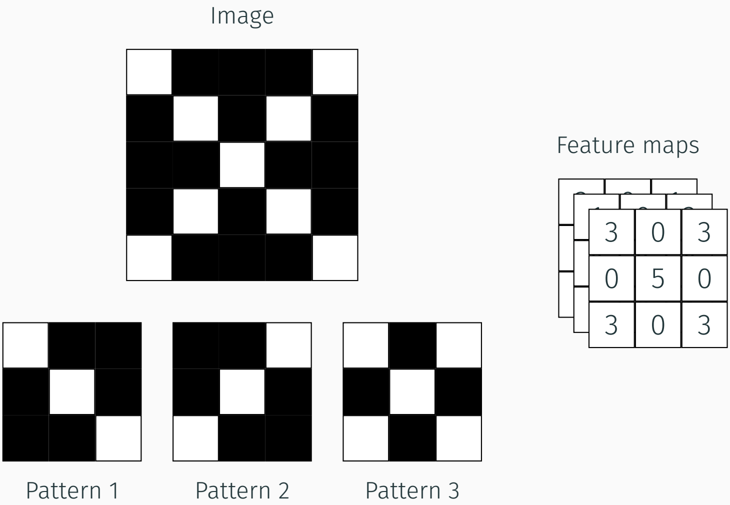
Pattern 2



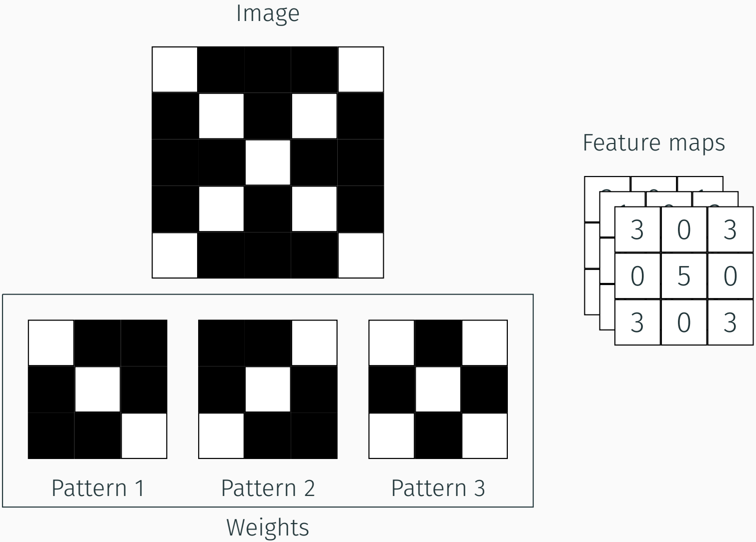
Pattern 3



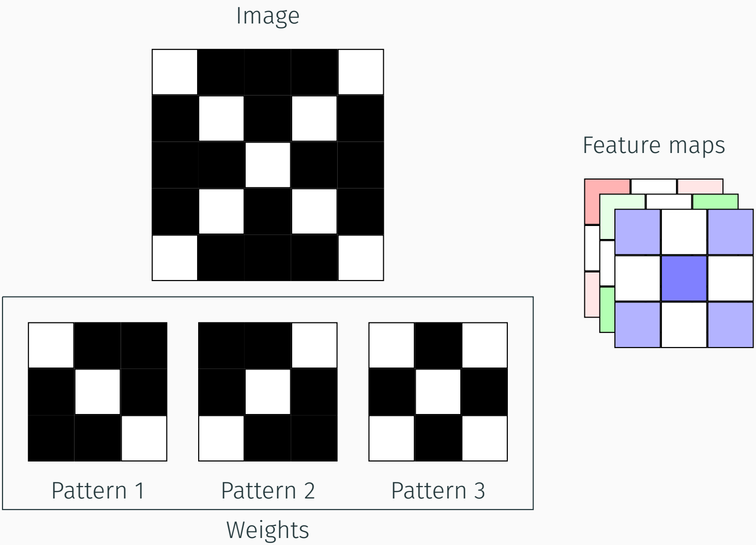
Convolutional neural networks: Convolutional layers



Convolutional neural networks: Convolutional layers



Convolutional neural networks: Convolutional layers



Convolutional neural networks: Pooling

Feature map

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

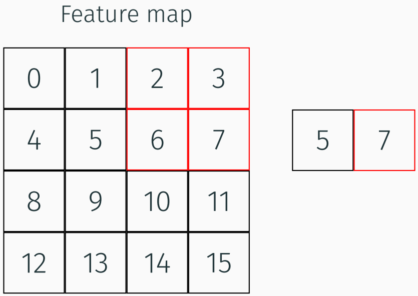
Convolutional neural networks: Pooling

Feature map

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

5

Convolutional neural networks: Pooling



Convolutional neural networks: Pooling

Feature map

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

5	7
13	

Convolutional neural networks: Pooling

Feature map

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

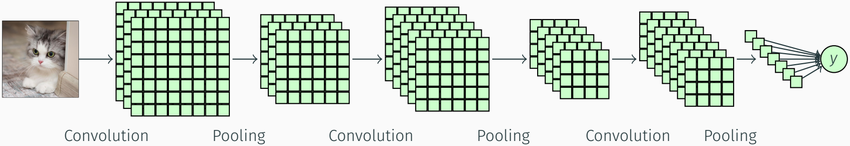
5	7
13	15

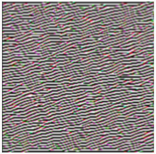
Feature map

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

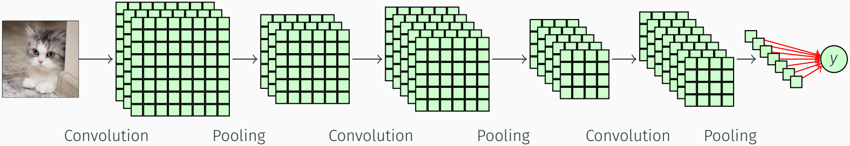
5	7
13	15

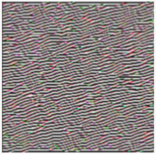
Convolutional neural networks: Architecture





Convolutional neural networks: Architecture





Convolutional neural networks: Summary

- Images in computers are stored as matrices of numbers
- The convolution operation is a pattern matcher
- A convolutional layer is a battery of pattern matchers where the patterns are learned during training
- The pooling operation reduces spatial dimensions and distills information
- A convolutional neural network consists of alternating convolution and pooling
- Alternating convolution and pooling in practice means looking for more and more abstract patterns spanning larger and larger region of the images
- The final layer of the network makes a prediction based on the patterns it has found