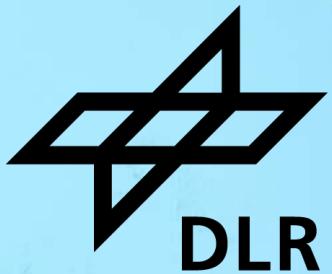


INTRODUCTION TO DEEP LEARNING

PART II – ADVANCED CONCEPT AND CNN

Auliya Fitri, Sai Vemuri, Sreerag Naveenachandran

**Machine Learning Group
Institute of Data Science**



Schedule



Date	Time	Activity
13.11.2025 Day 1	09:00 - 10:00	Introduction and basics
	10:00 - 10:30	Hands-on I
	10:30 - 10:45	Coffee break
	10:45 - 11:45	Advanced concept and Convolutional Neural Network
	11:45 - 12:15	Hands-on II
	12:15 - 12:30	Recap Day 1
14.11.2025 Day 2	09:00 - 10:00	Deep Generative Models
	10:00 - 10:30	Hands-on III
	10:30 - 10:45	Coffee break
	10:45 - 11:45	Transformers, LLM, and other interesting architectures
	11:45 - 12:15	Hands-on IV
	12:15 - 12:30	Code and knowledge sources + closing

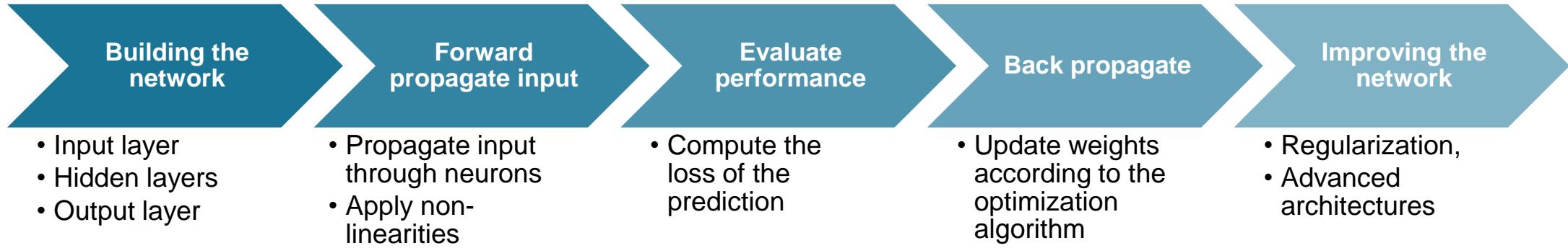


II. Advanced Concept and Convolutional Neural Network

- Regularization
- Variants of Neural Networks
- Convolutional Neural Networks (CNN)

Inspired by lectures from Paris Saclay and MIT; images taken from these, if not noted otherwise

Neural network concepts



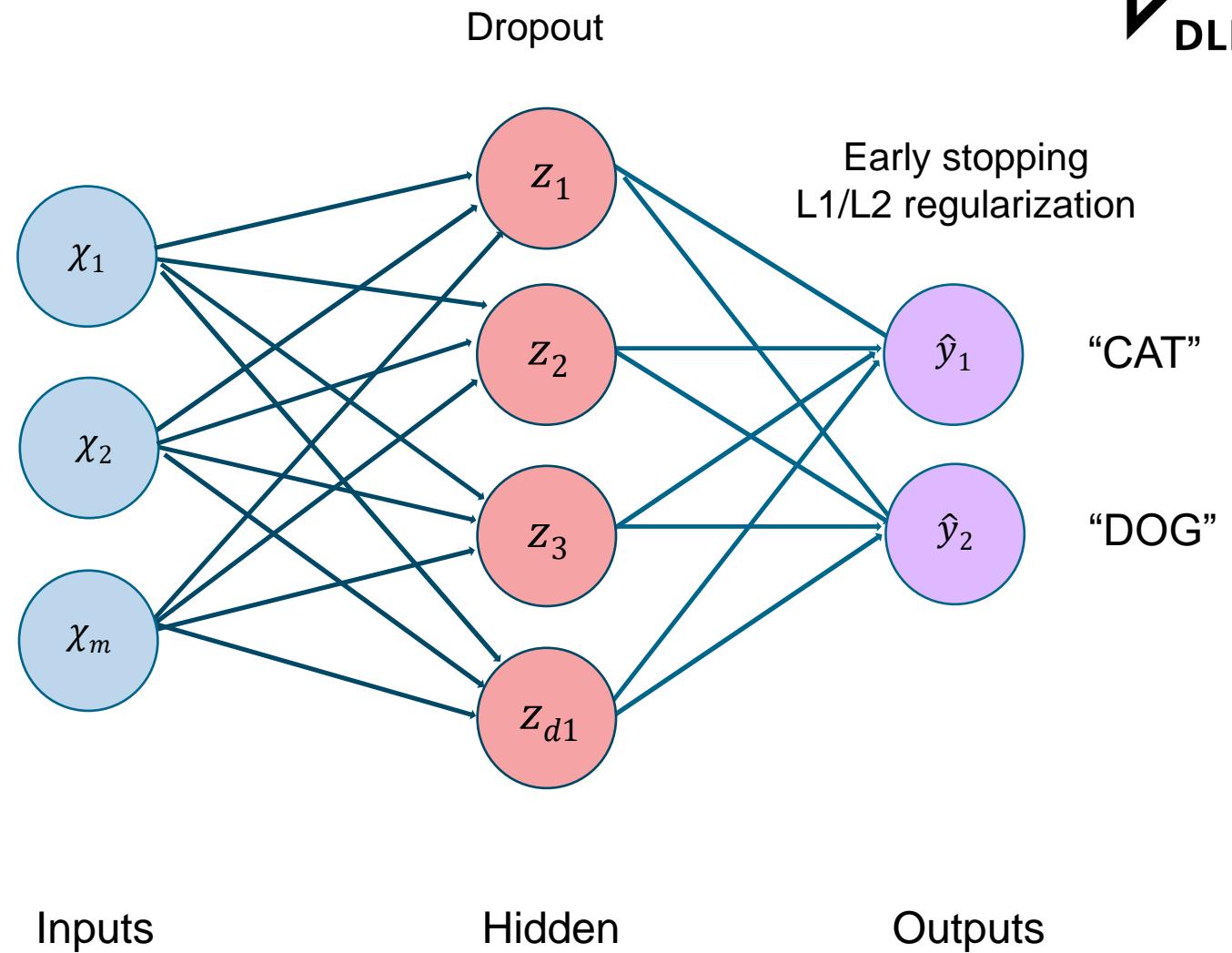
Improve the Network: Regularization

Data Augmentation



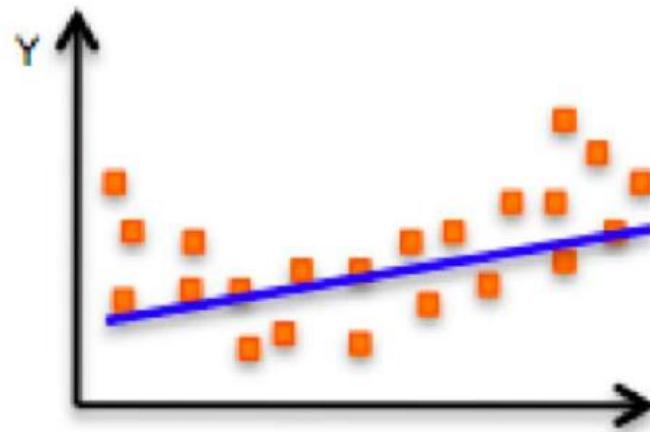
CAT

$$\begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_m \end{bmatrix}$$



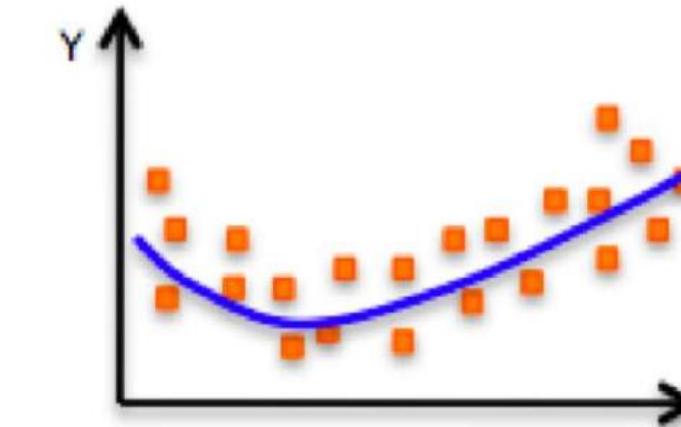
Regularization

The overfitting problem

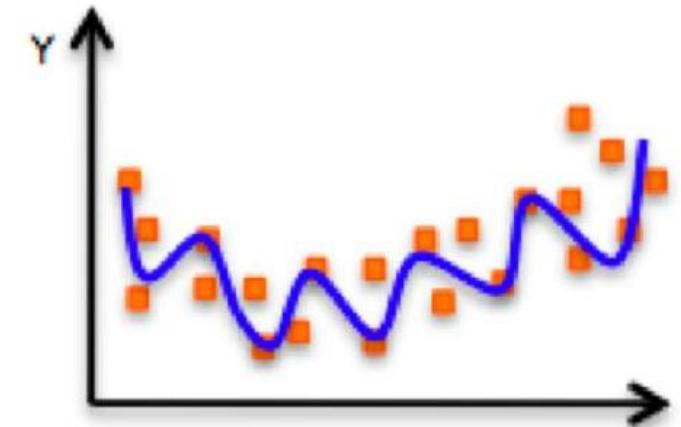


Underfitting
Model does not have capacity
to fully learn the data

= High bias



Ideal fit ← → **Tradeoff**

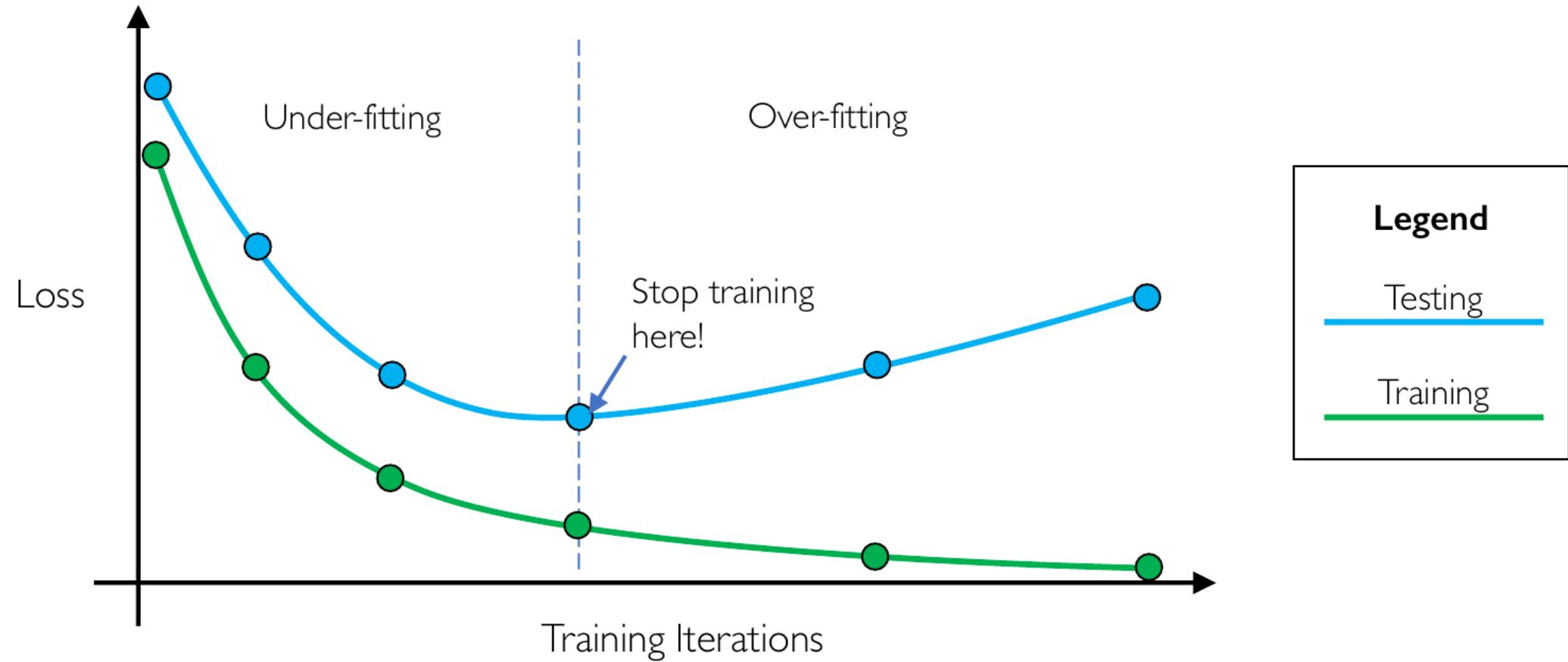


Overfitting
Too complex, extra parameters,
does not generalize well

= High variance

Regularization

Preventing overfitting: Early stopping



Regularization

Preventing overfitting: l_1 regularization



We want to find the network weights that **achieve the lowest loss**

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(\mathbf{x}^{(i)}; \mathbf{W}), y^{(i)}) + \frac{\lambda}{2} \sum_l |\mathbf{w}_l|$$

→ Leads to sparser weights (more zeroes in weights) that are not too adapted to the data at hand

Regularization

Preventing overfitting: Data augmentation



Original



Flip



Random crop



Contrast



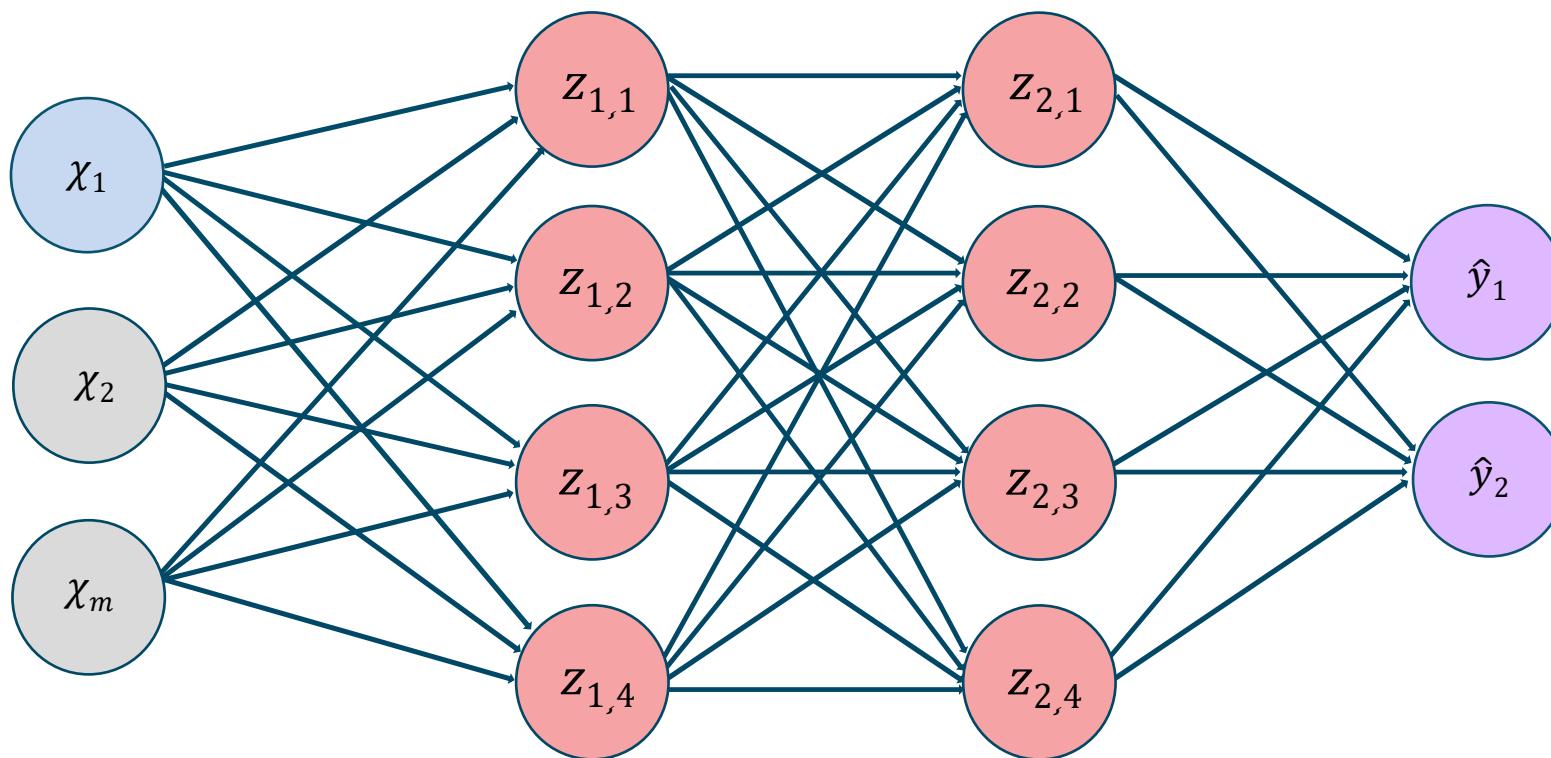
Tint



Regularization

Preventing overfitting: Dropout

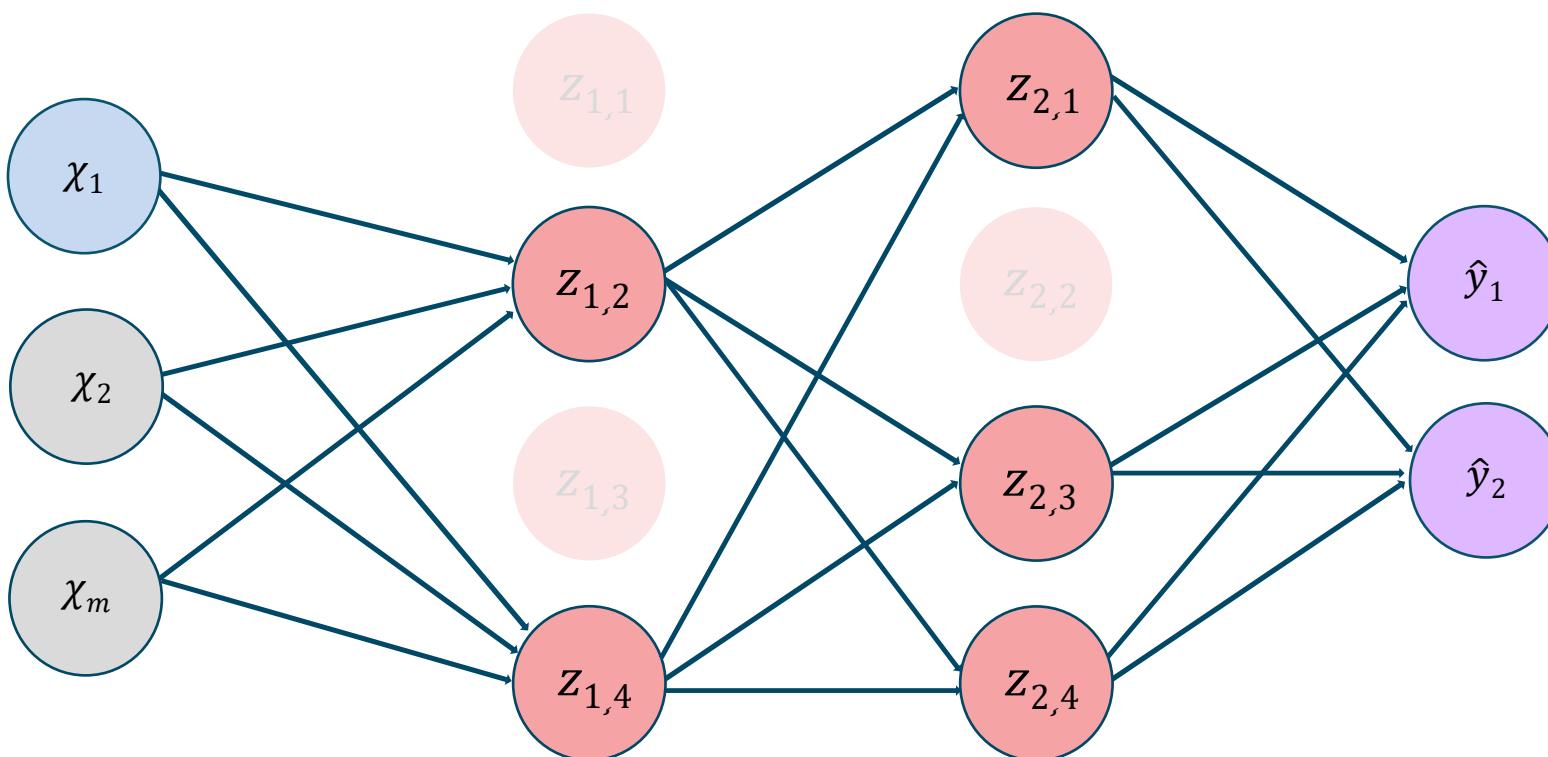
During training, randomly set some activations to 0



Regularization

Preventing overfitting: Dropout

During training, randomly set some activations to 0



Other ways to improve the network

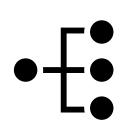


Data preprocessing

Data Cleaning (lack/noise)

Data Transformation (normalization, ...)

Data Reduction (aggregation, ...)



Network initialization

Random Initializations (e.g. Glorot)



Batch normalization

standardizes layer inputs to stabilize learning process & reducing training epochs



Optimizers

(Stochastic methods like Adam)



Training Schedules



Hyperparameters tuning



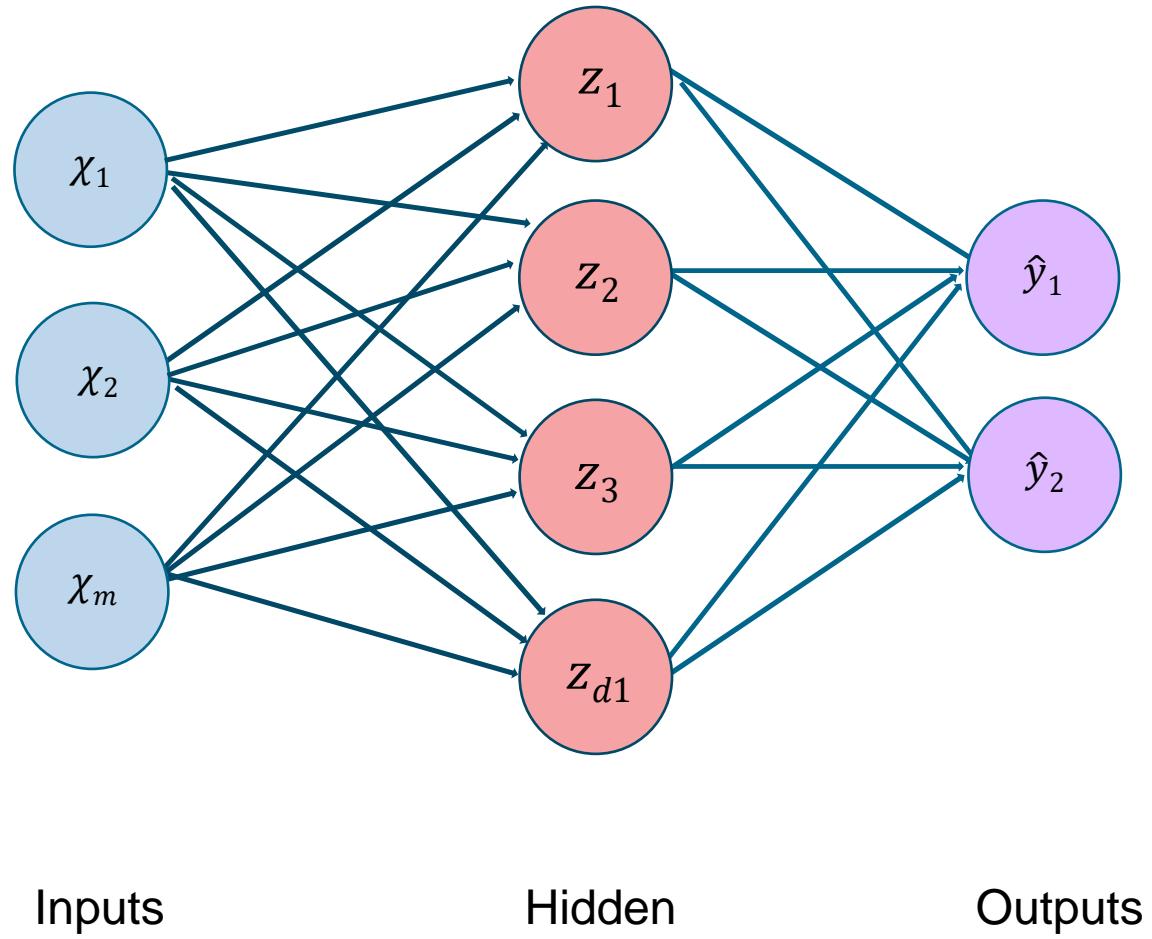
Minibatch composition

(shuffle data, take only a subset, ...)

Variants of Neural Networks



$$\begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_m \end{bmatrix}$$



“CAT”

“DOG”

Types of Neural Networks Layer

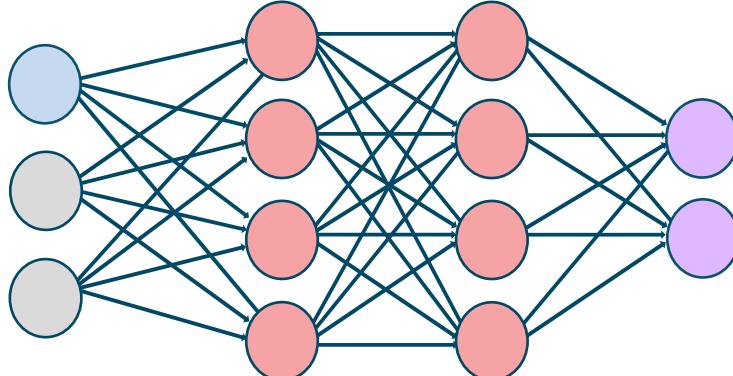


Application	Layer Type			
	Fully Connected (dense)	Convolution	Deconvolution	Recurrent
Image Classification	✓	✓		
Image segmentation		✓	✓	
Text processing	✓			✓
Speech recognition	✓			✓
Time series	✓			✓
Image-to-image translation (GAN)	✓	✓	✓	
Autoencoders	✓	✓		
Deepfake	✓	✓		✓

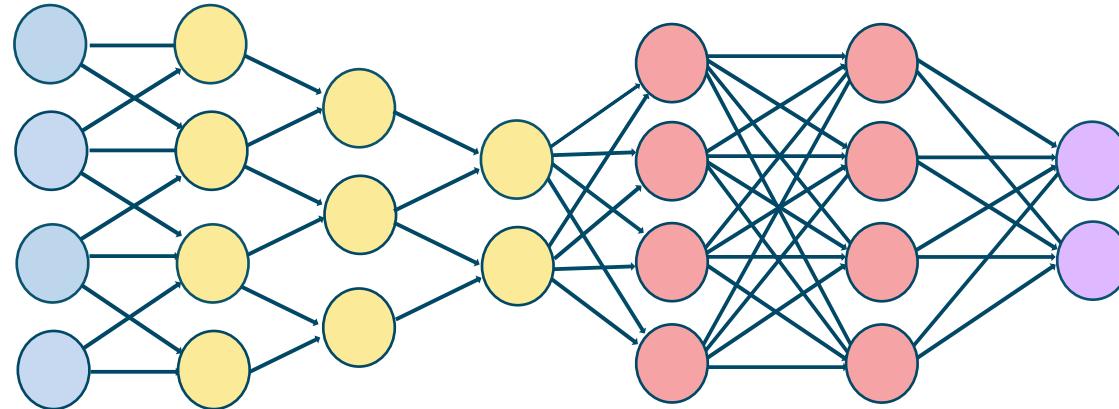
Variants of Neural Network - Examples



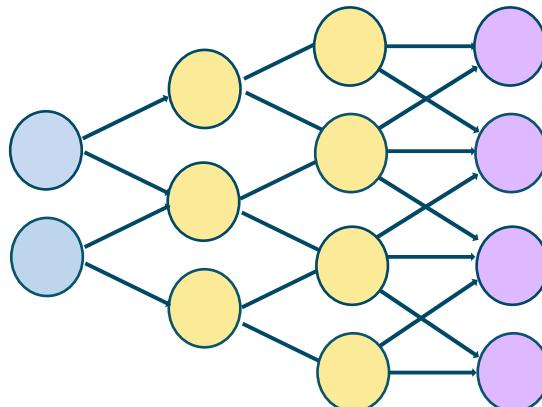
Deep Feed Forward (DFF) (fully connected)



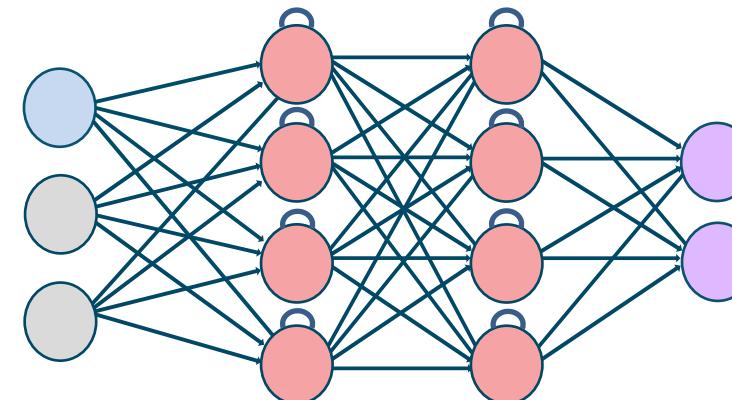
Deep convolution network (DCN)



Deconvolution network (DN)

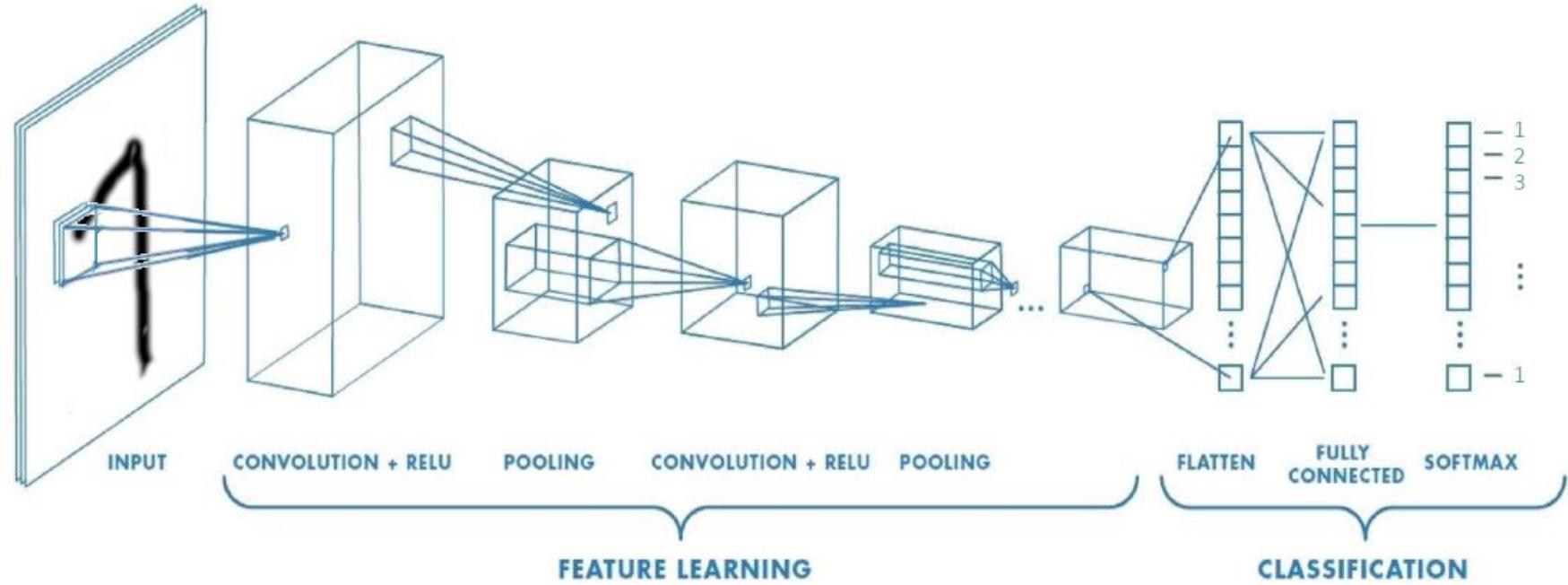


Recurrent neural network (RNN)

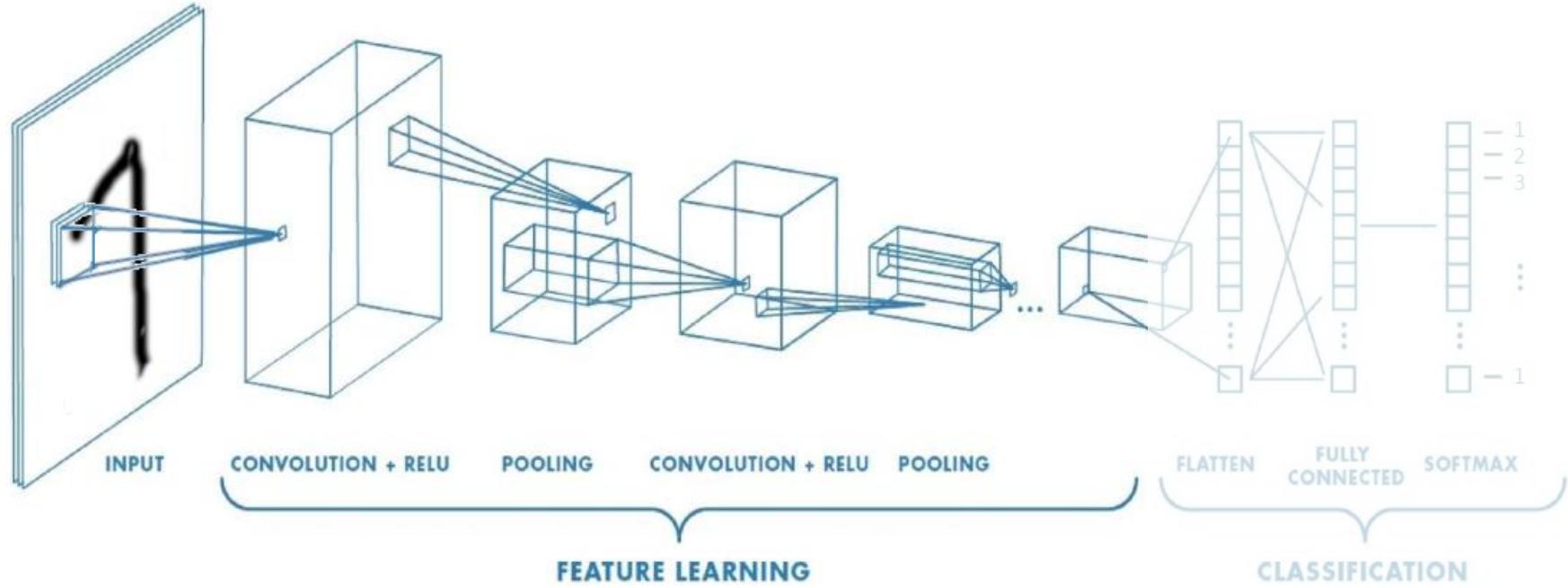


● = Input
● = Output

Convolutional Neural Networks



Feature Learning



1. Learn features in input image through **convolution**
2. Introduce **non-linearity** through activation function (real-world data is non-linear!)
3. Reduce dimensionality and preserve spatial invariance with **pooling**

Convolutional Neural Networks

Feature extraction with convolutions



Original



Sharpen



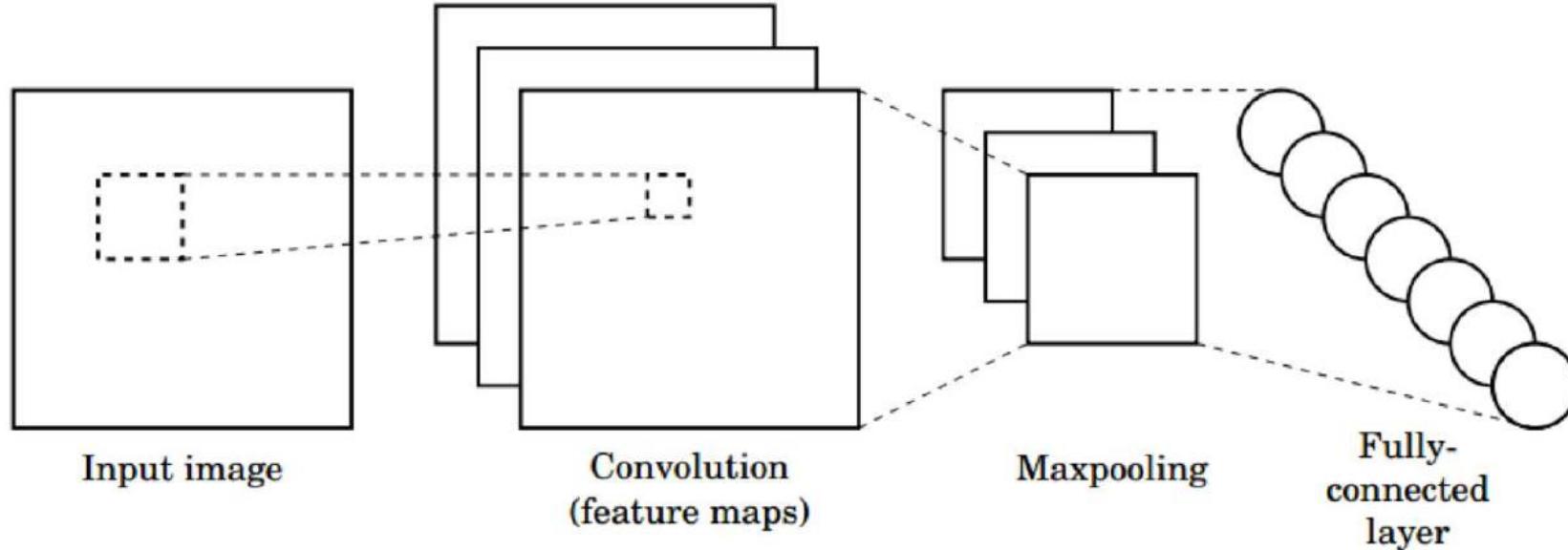
Edge Detect



“Strong” Edge
Detect

Convolutional Neural Networks

Building a CNN



- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

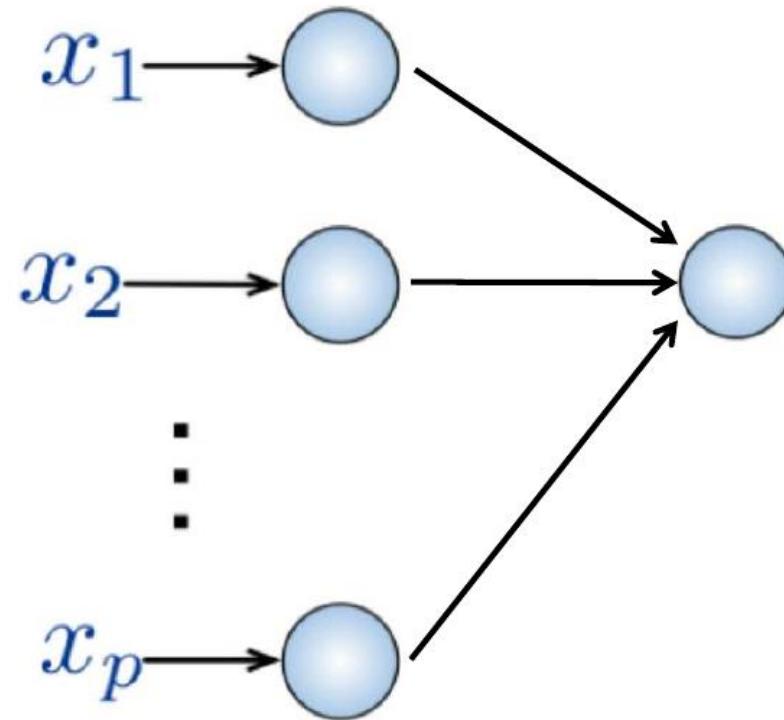
Train model with image data.
Learn weights of filters in convolutional layers.

Convolutional Neural Networks

Learning on image data

Input:

- 2D image
- Vector of pixel values



Fully Connected:

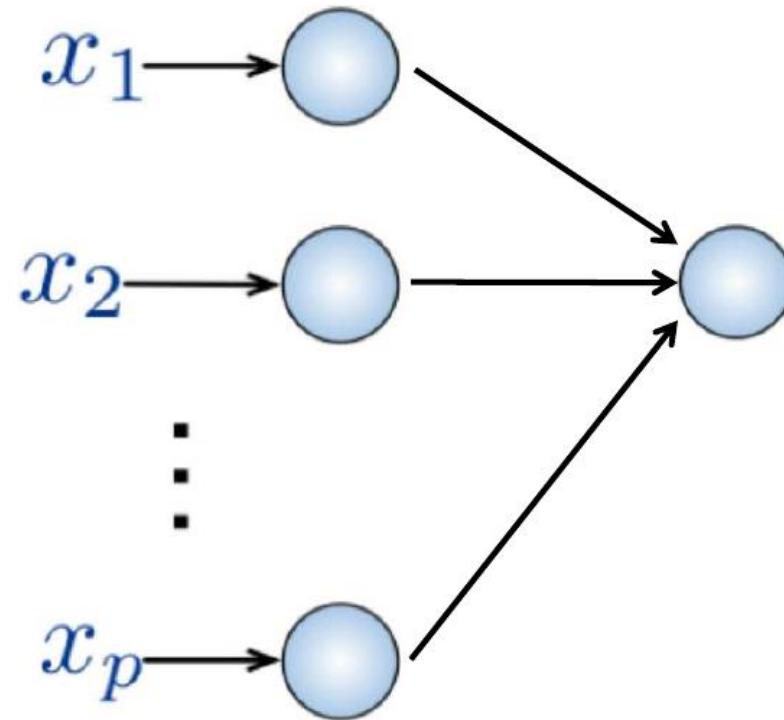
- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

Convolutional Neural Networks

Learning on image data

Input:

- 2D image
- Vector of pixel values



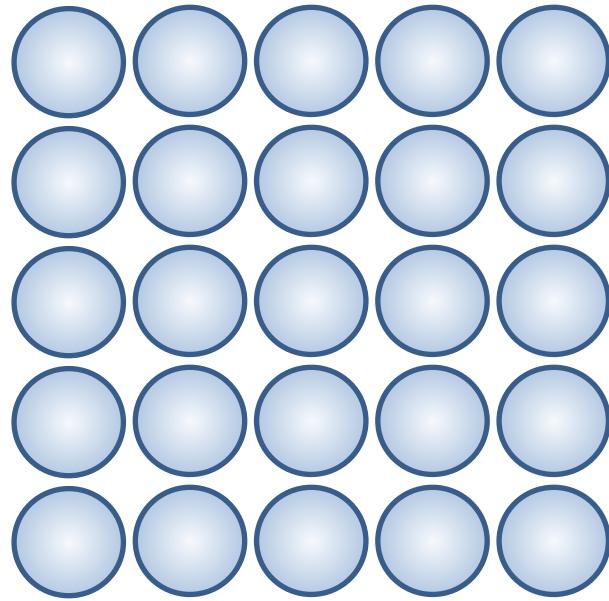
Fully Connected:

- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

How can we use **spatial structure** in the input to inform the architecture of the network?

Convolutional Neural Networks

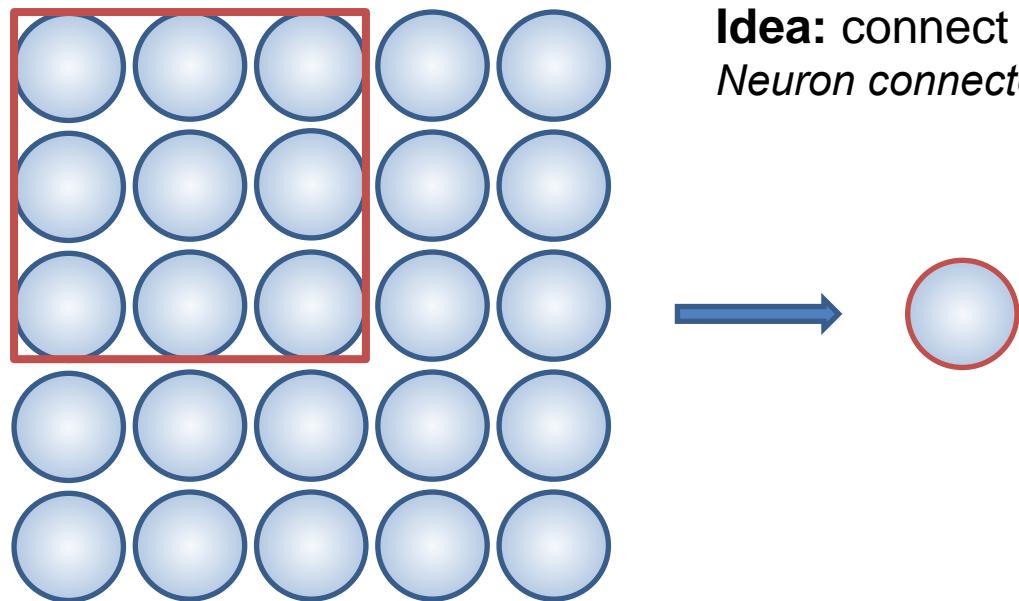
Learning on image data



Input: 2D image (5x5 or 256x256 or ...)
Array of pixel values

Convolutional Neural Networks

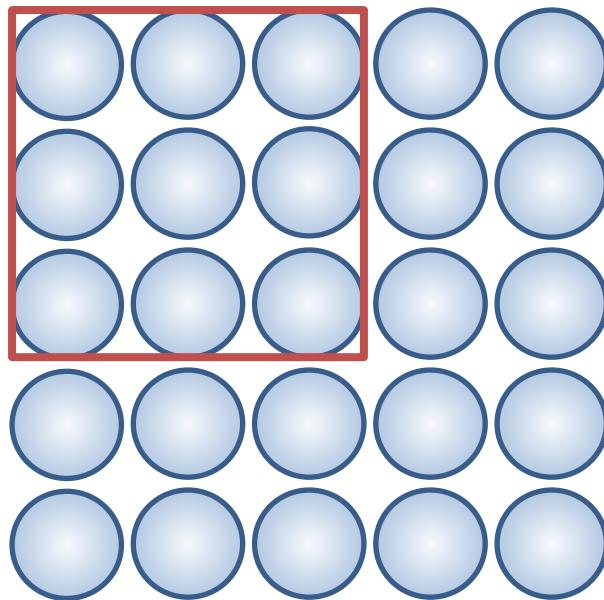
Learning on image data



Idea: connect patches of input to neurons in hidden layer
Neuron connected to the region of input only „sees“ these values.

Convolutional Neural Networks

Learning on image data



3x3 filter: matrix
of weights W_{ij}

For a neuron in hidden layer:

- Take inputs from patch
- Compute weighted sum
- Apply bias



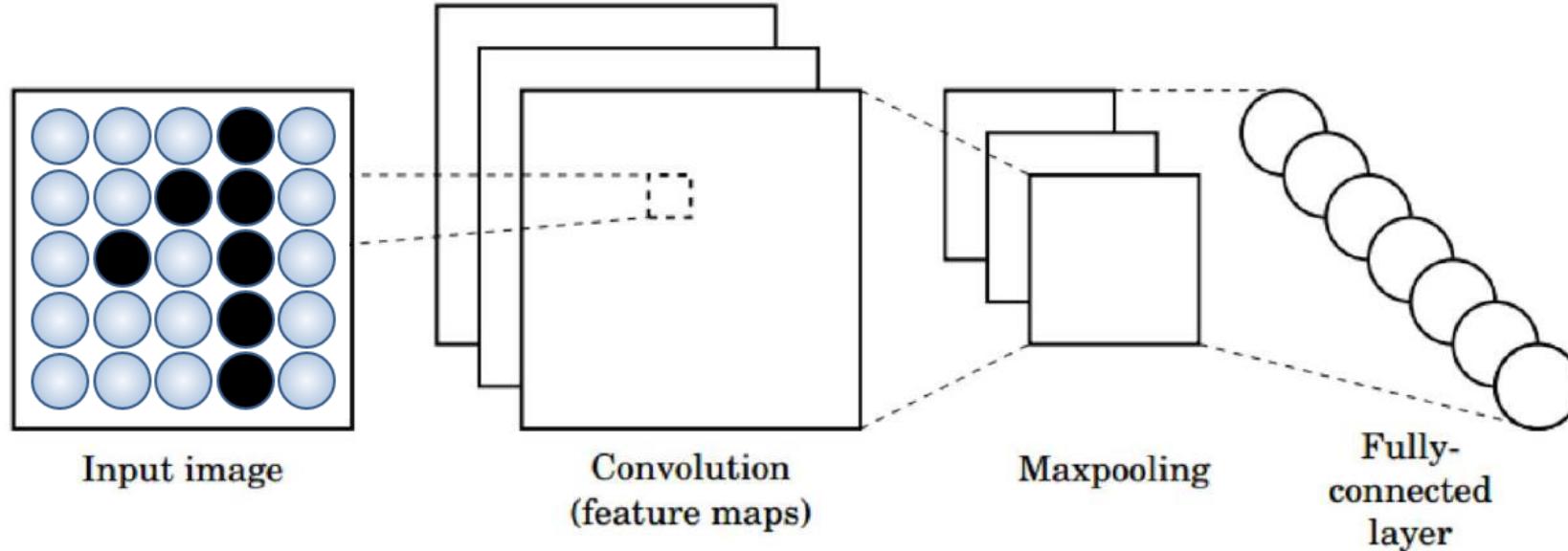
$$\sum_{i=1}^3 \sum_{j=1}^3 W_{ij} X_{i+p,j+q} + b$$

For neuron (p,q) in hidden layer

- 1) Applying a window of weights
- 2) Computing linear combinations
- 3) Activating with non-linear function

Convolutional Neural Networks

Building a CNN - Example

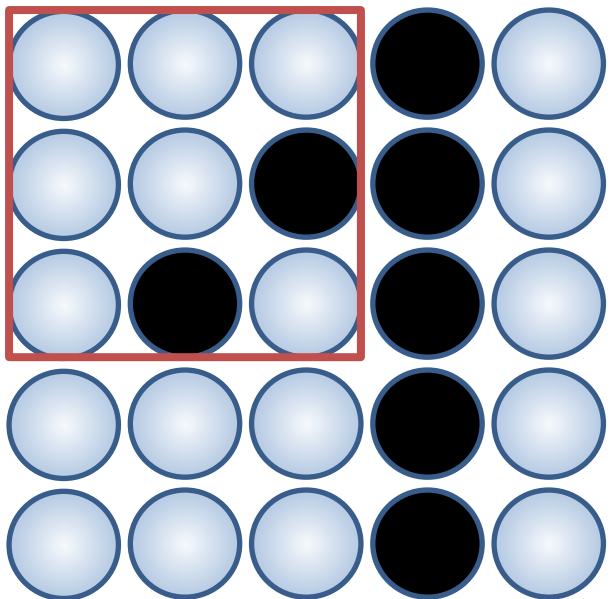
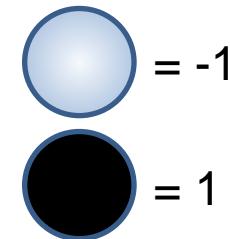


- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

Train model with image data.
Learn weights of filters in convolutional layers.

Convolutional Neural Networks

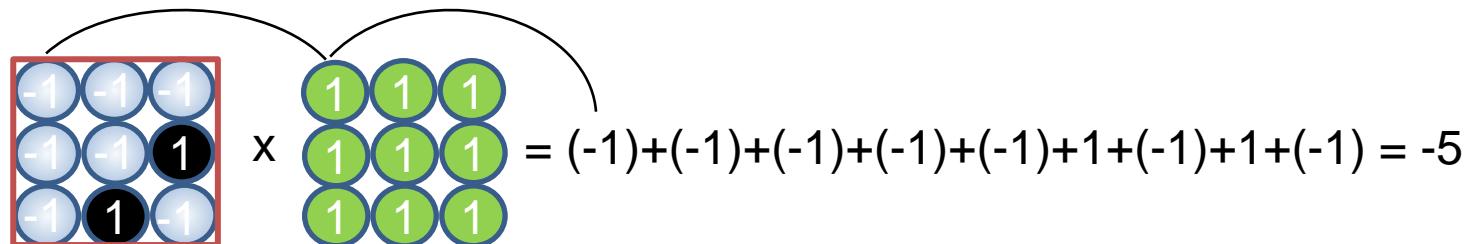
Learning on image data - Example



- Connect patch in input layer to a single neuron in subsequent layer
- Use a **sliding window** to define connections
- Ex: extracting feature with **filter** f_{i_0}



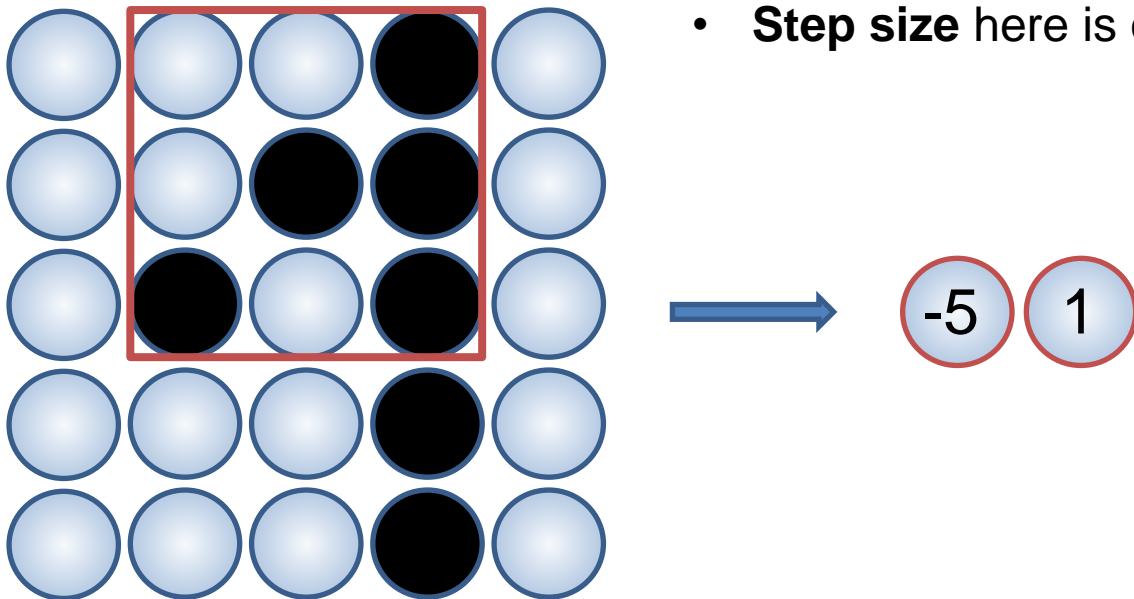
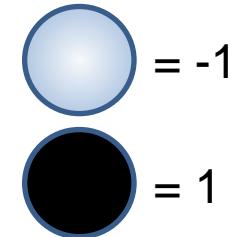
Multiply same indices of clipping and filter and sum it up.



The diagram shows the convolution operation between the input patch and the filter. The input patch is a 3x3 grid with values [-1, -1, -1; -1, -1, 1; -1, 1, -1]. The filter is a 3x3 grid of all ones. Arrows point from the input values to the filter values. The result of the multiplication and summation is shown as the equation: $= (-1)+(-1)+(-1)+(-1)+(-1)+1+(-1)+1+(-1) = -5$.

Convolutional Neural Networks

Learning on image data - Example

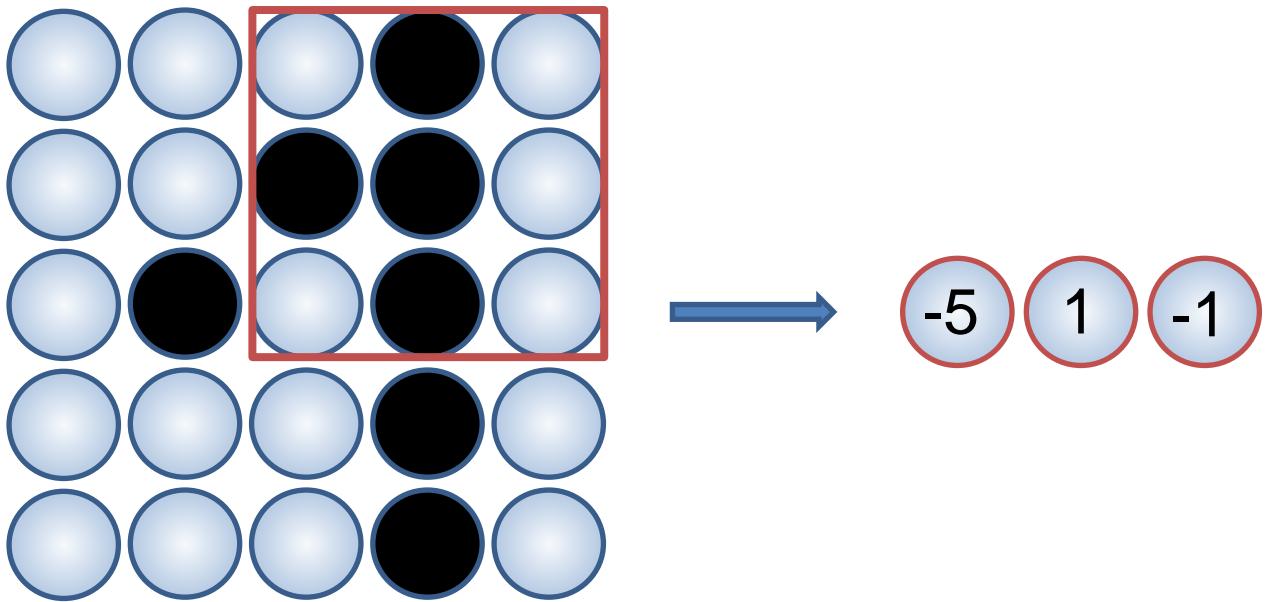


$$\begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = 1$$

Convolutional Neural Networks

Learning on image data - Example

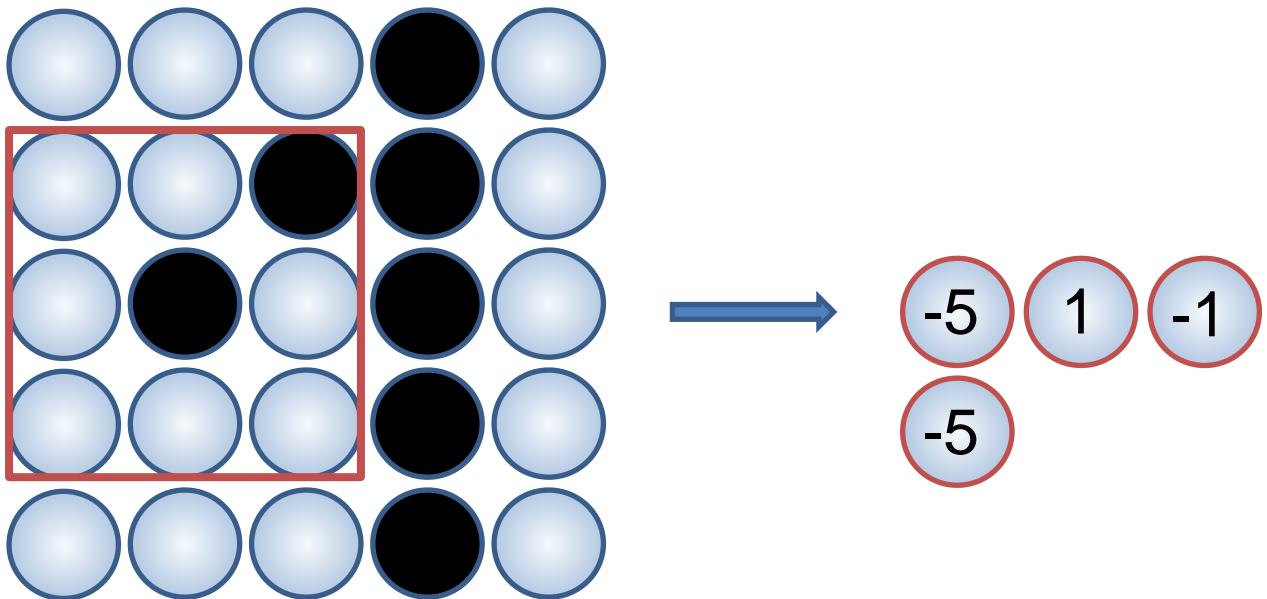
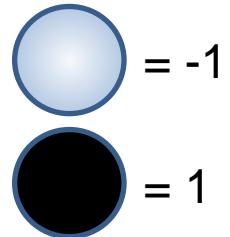
Light blue circle = -1
Black circle = 1



$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = -1$$

Convolutional Neural Networks

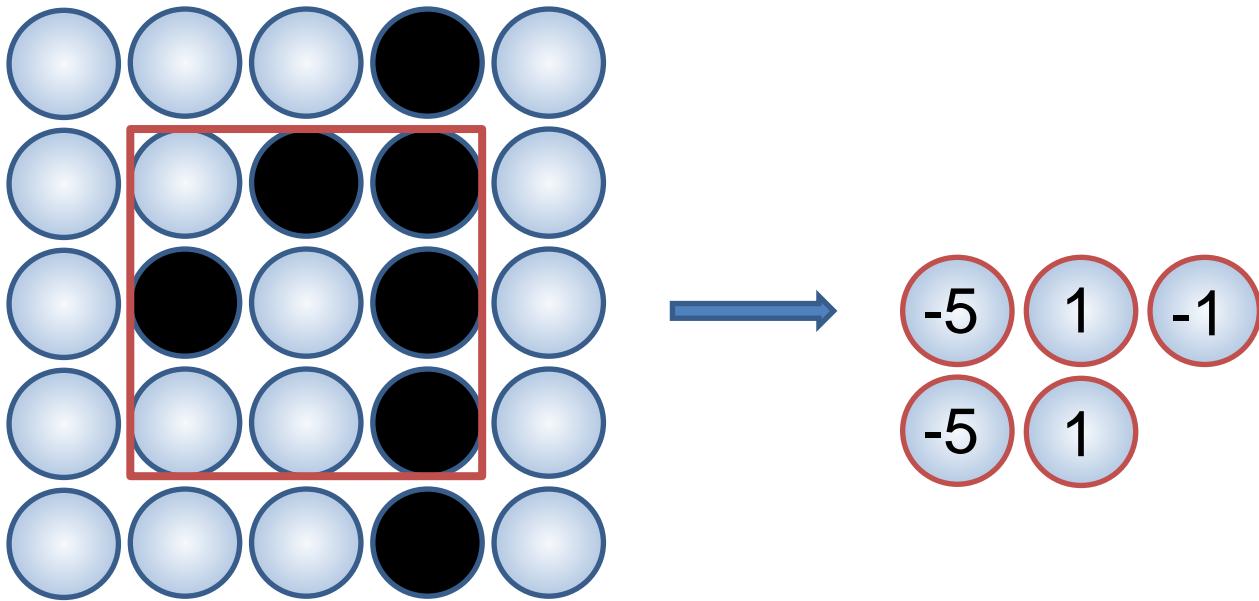
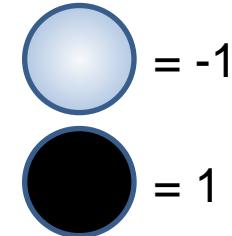
Learning on image data - Example



$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = -5$$

Convolutional Neural Networks

Learning on image data - Example

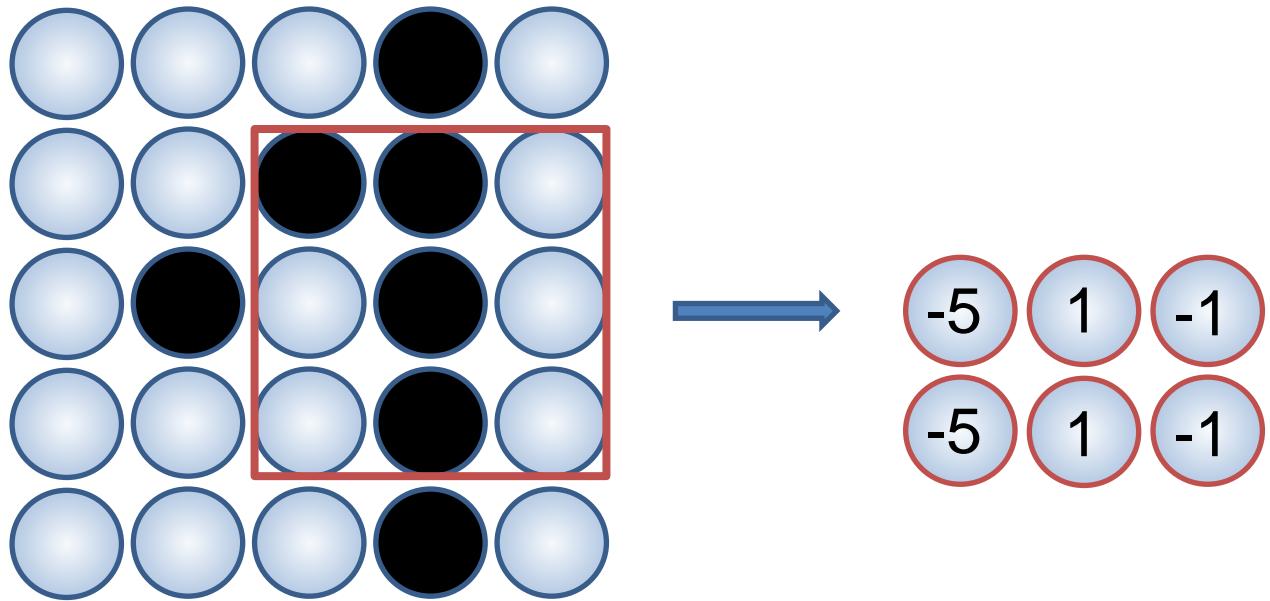


$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = 1$$

Convolutional Neural Networks

Learning on image data - Example

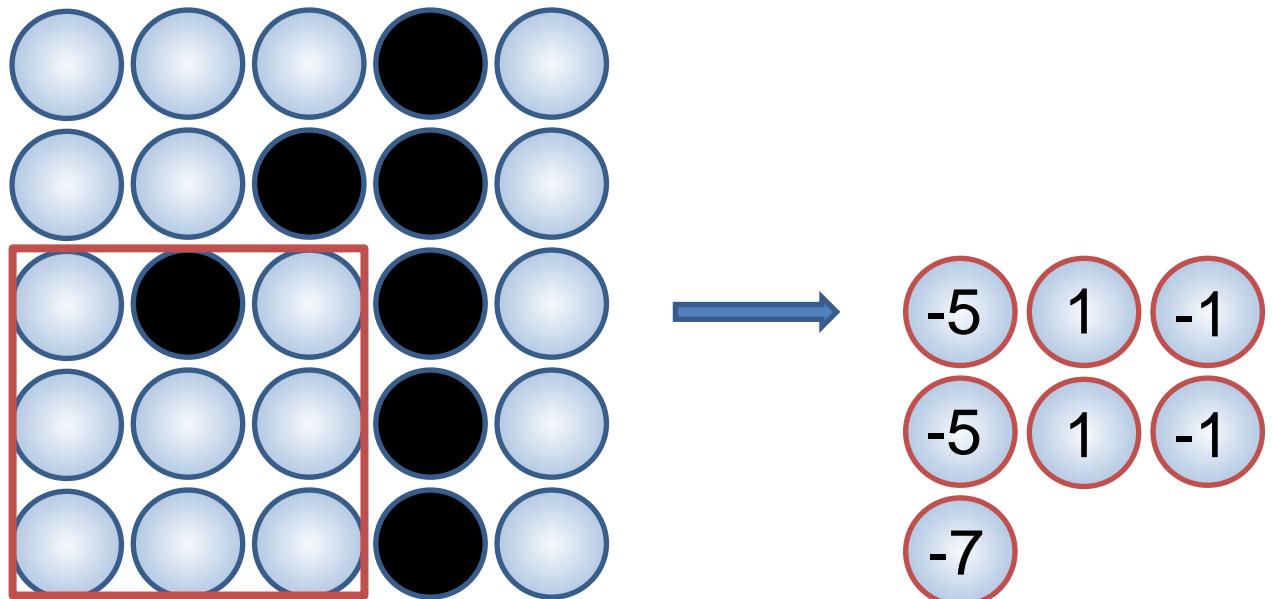
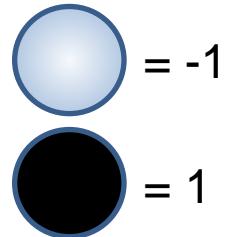
Light blue circle = -1
Black circle = 1

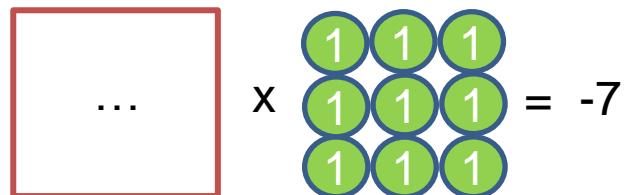


$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = -1$$

Convolutional Neural Networks

Learning on image data - Example



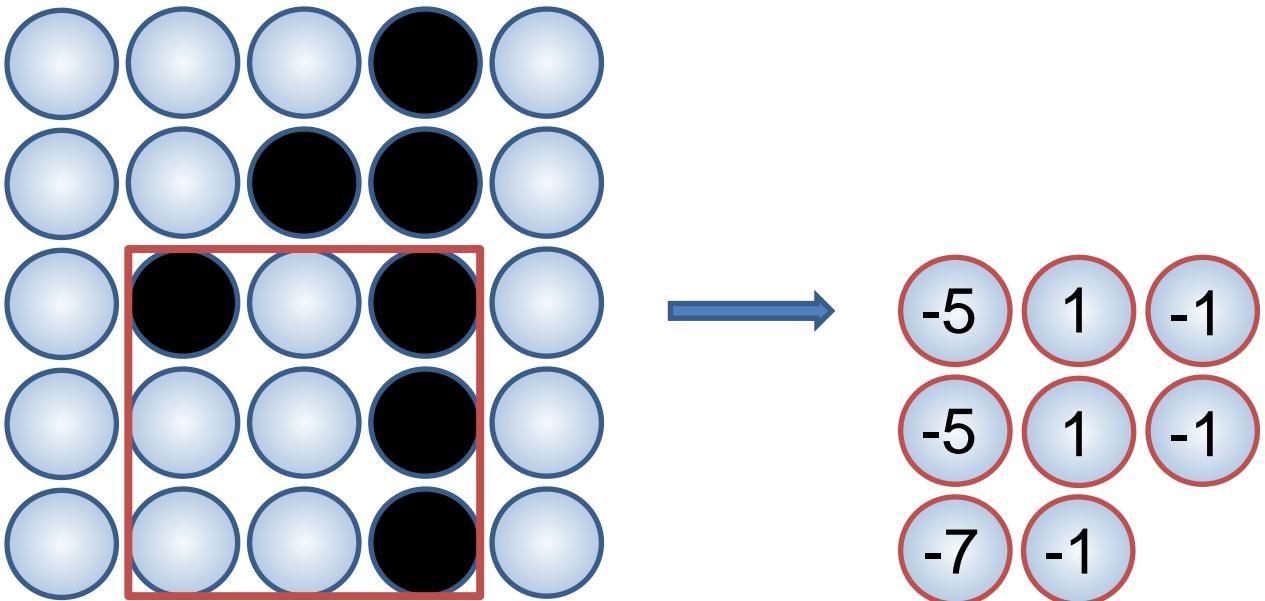
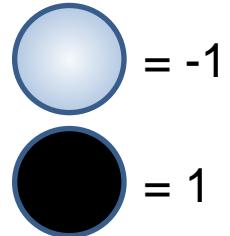


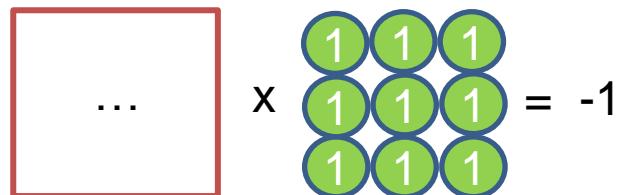
A diagram showing the convolution operation. A red box contains three dots (...), representing the input image. To its right is a multiplication sign (x). Next is a 3x3 kernel represented by green circles with the value 1. Finally, an equals sign (=) followed by the result -7.

$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = -7$$

Convolutional Neural Networks

Learning on image data - Example



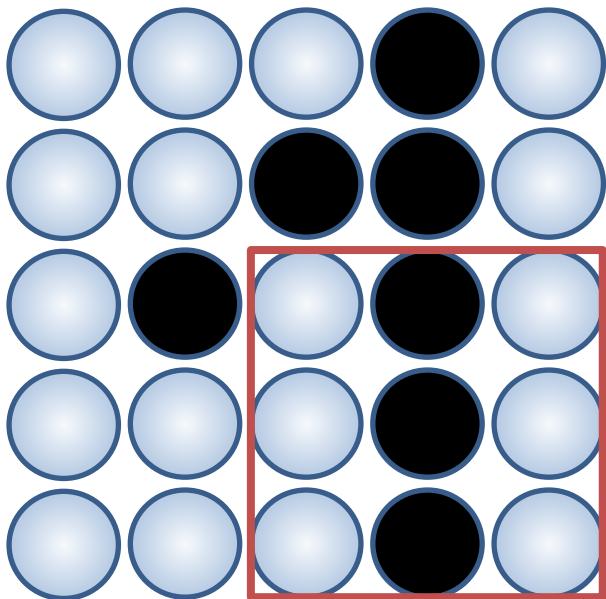


The diagram shows a red box containing three dots (...), representing the bias term, multiplied by a 3x3 kernel of all ones (1, 1, 1; 1, 1, 1; 1, 1, 1). The result is labeled = -1.

$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = -1$$

Convolutional Neural Networks

Learning on image data - Example



- This results in a “feature map” → here : 9 neurons for the 5x5 image



$$\begin{matrix} -5 & 1 & -1 \\ -5 & 1 & -1 \\ -7 & -1 & -3 \end{matrix}$$

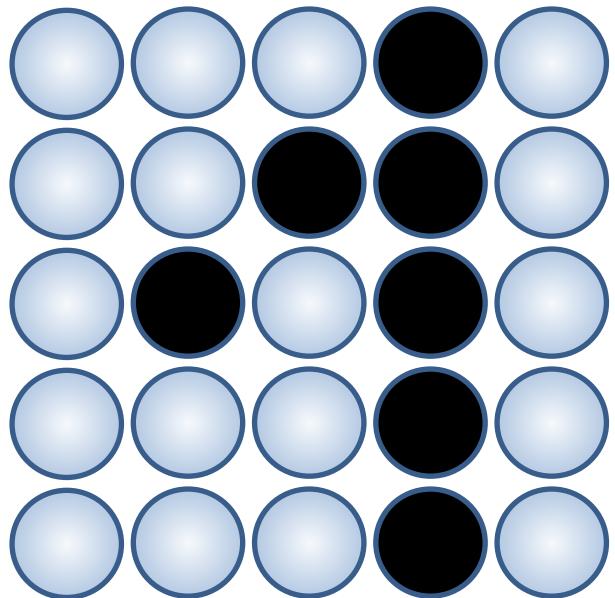
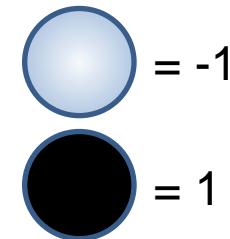
- Filter of size $3 \times 3 \rightarrow 9$ separate weights
- Scan step wise the image with the filter
- That's **Convolution**

$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = -3$$

$$\begin{matrix} \text{light blue circle} & = -1 \\ \text{black circle} & = 1 \end{matrix}$$

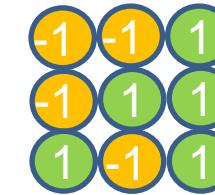
Convolutional Neural Networks

Learning – Feature Extraction



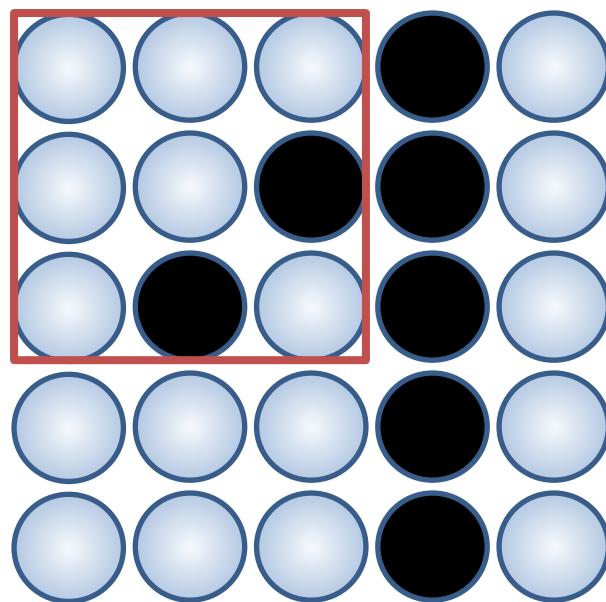
- Repeat the scan process with **multiple filters**

Next: extracting feature with filter $f_{i_1} =$



Convolutional Neural Networks

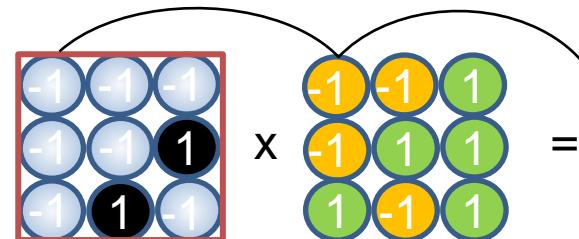
Learning – Feature Extraction



Low score !
barely matches

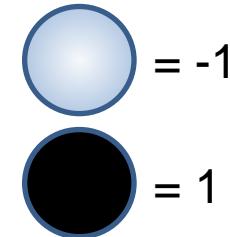


-1

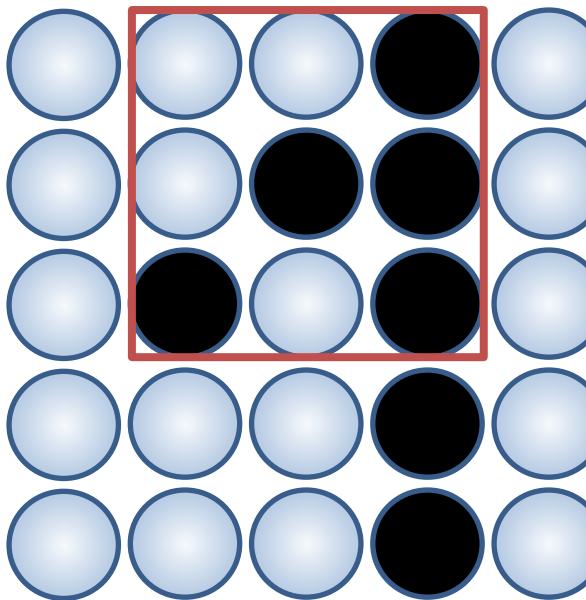

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{matrix} \times \begin{matrix} 1 & 1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{matrix} = 1+1+(-1)+1+(-1)+1+(-1)+(-1) = -1$$

Convolutional Neural Networks

Learning – Feature Extraction



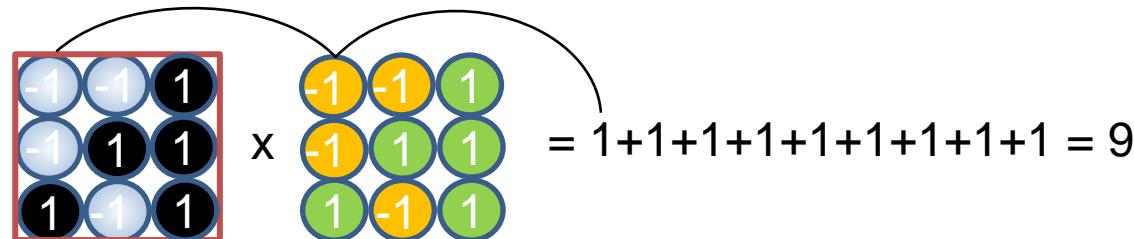
A legend showing two circles: a light blue circle labeled = -1 and a black circle labeled = 1.



Maximum score !
Filter matches full
↓



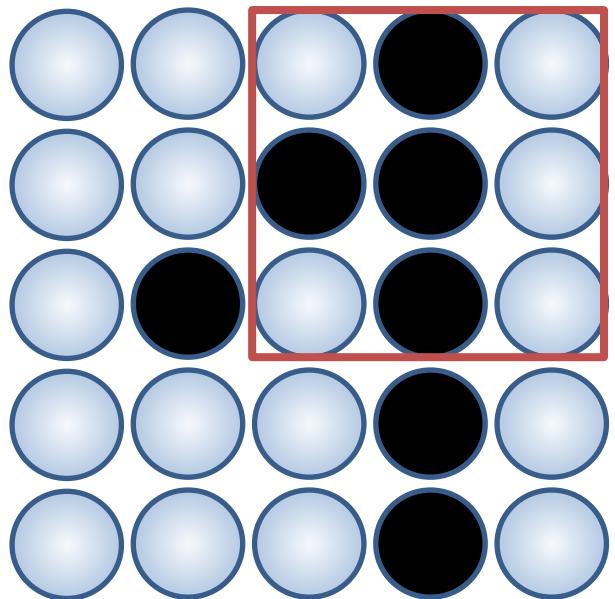
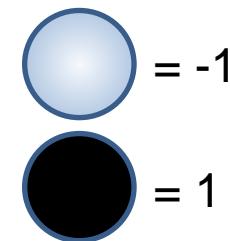
The result of the max pooling operation is shown as two adjacent circles, one red-bordered containing -1 and one blue-bordered containing 9.



A diagram illustrating the convolution operation. It shows a 3x3 input kernel with values [-1, -1, 1; -1, 1, 1; 1, -1, 1] (the bottom-right value is 1) being multiplied by a 3x3 weight kernel with values [-1, -1, 1; -1, 1, 1; 1, -1, 1] (the bottom-right value is 1). The result of the multiplication is given as the sum of the products: $= 1+1+1+1+1+1+1+1 = 9$.

Convolutional Neural Networks

Learning – Feature Extraction



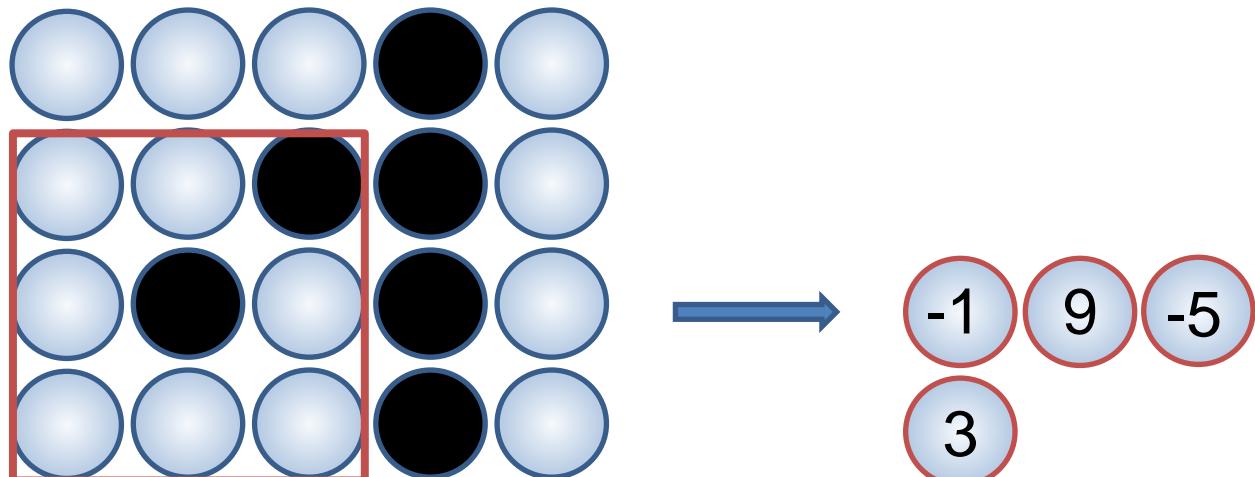
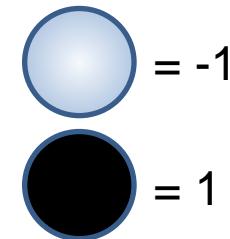
Very low score !
Many mismatches

A horizontal row of three circles with red borders and black outlines. The first circle contains '-1', the second contains '9', and the third contains '-5'. An arrow points from the input grid to this row.

A diagram illustrating a convolution operation. On the left, a 3x3 input kernel with values [-1, 1, -1; 1, 1, -1; -1, 1, -1] is shown. To its right is a 3x3 weight kernel with values [-1, -1, 1; -1, 1, 1; 1, 1, 1]. A multiplication symbol 'x' is between them. To the right of the multiplication is the result of the dot product: $= 1 + (-1) + (-1) + (-1) + 1 + (-1) + (-1) + (-1) = -5$.

Convolutional Neural Networks

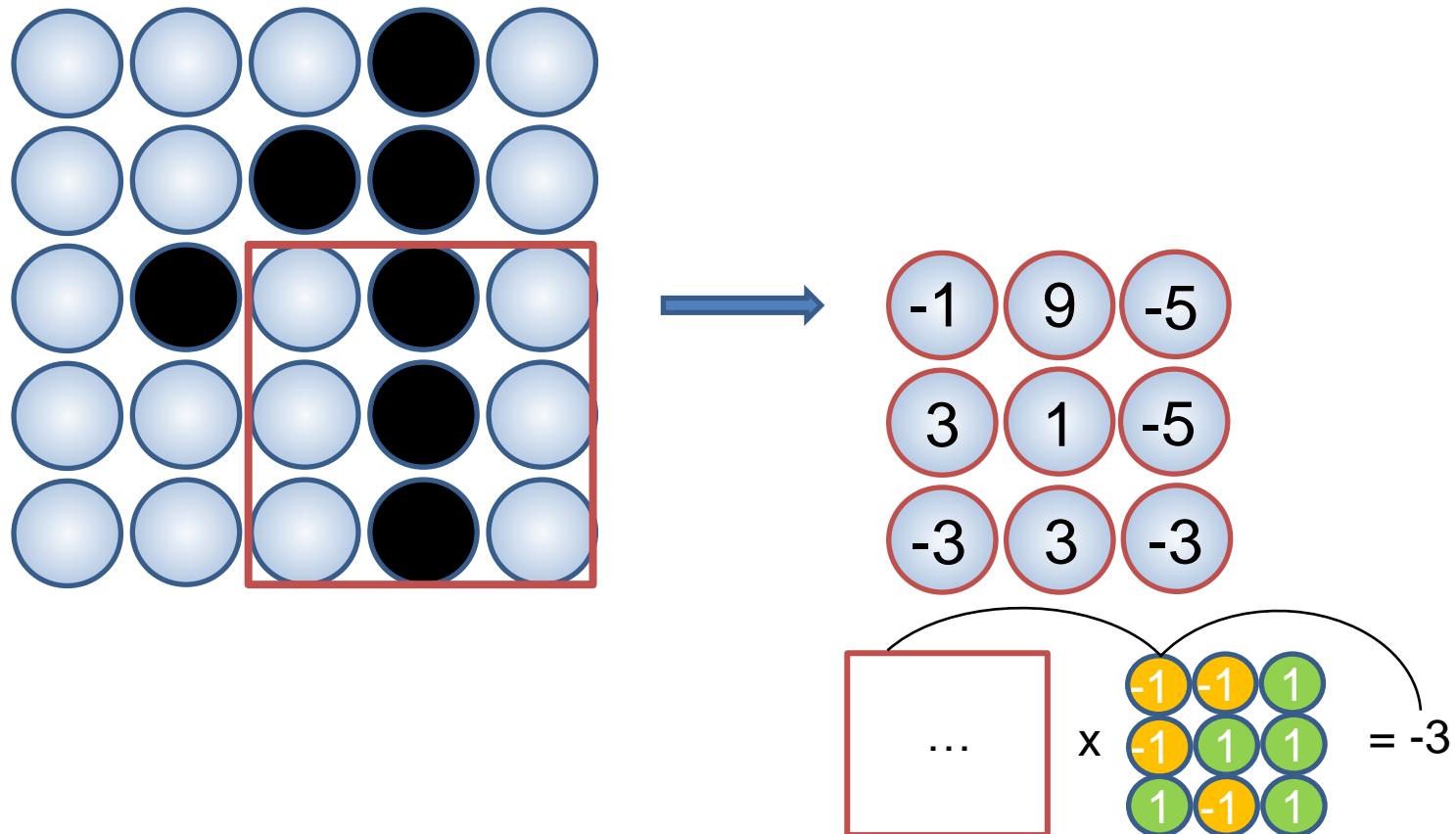
Learning – Feature Extraction



A diagram showing a matrix multiplication operation. A red-bordered box containing three dots "...", representing a larger input matrix, is multiplied by a 3x3 kernel matrix. The kernel matrix has alternating yellow and green values: -1, -1, 1; -1, 1, 1; 1, -1, 1. The result of the multiplication is 3.

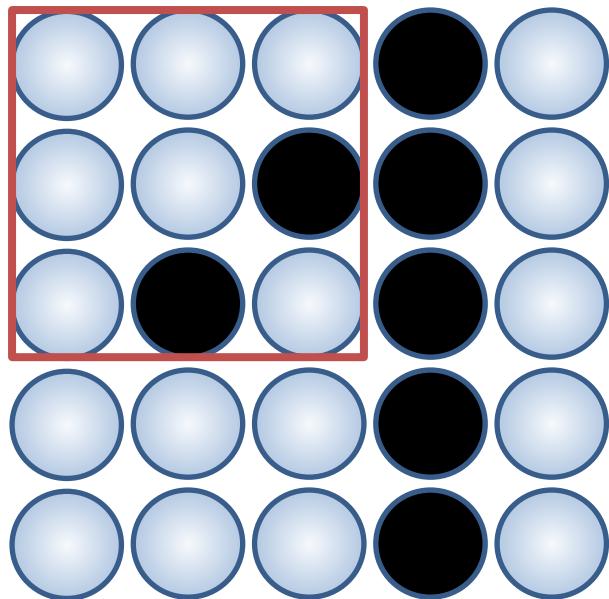
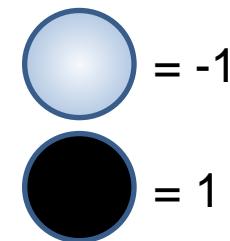
Convolutional Neural Networks

Learning – Feature Extraction



Convolutional Neural Networks

Learning – Feature Extraction

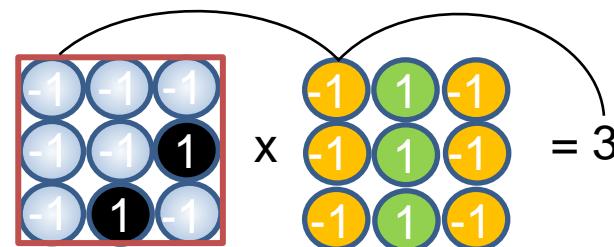


- Different feature will extract different features
- **Spatially share** parameters of each filter



Next filter $\mathbf{f}_i_2 =$



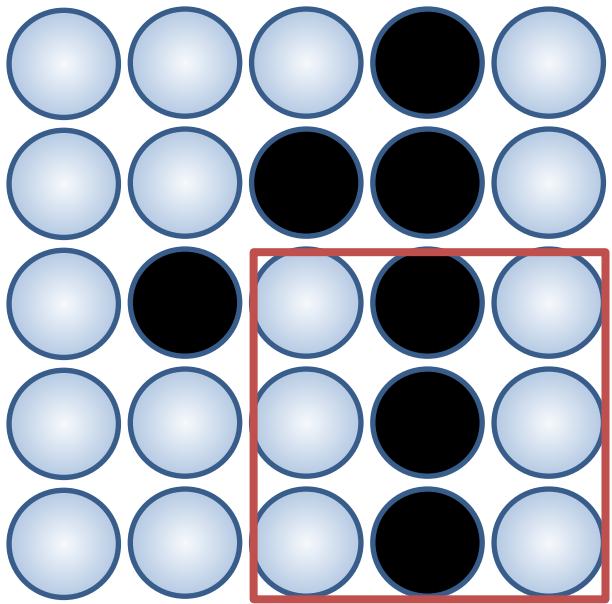
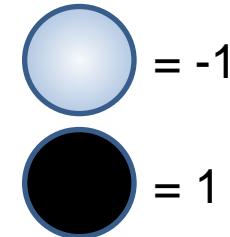


A diagram illustrating the convolution operation. It shows the 3x3 input subgrid with a red border being multiplied (x) by the 3x3 filter kernel. The result is a single output value of 3.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix} = 3$$

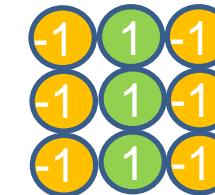
Convolutional Neural Networks

Learning – Feature Extraction



- Different feature will extract different features
- **Spatially share** parameters of each filter

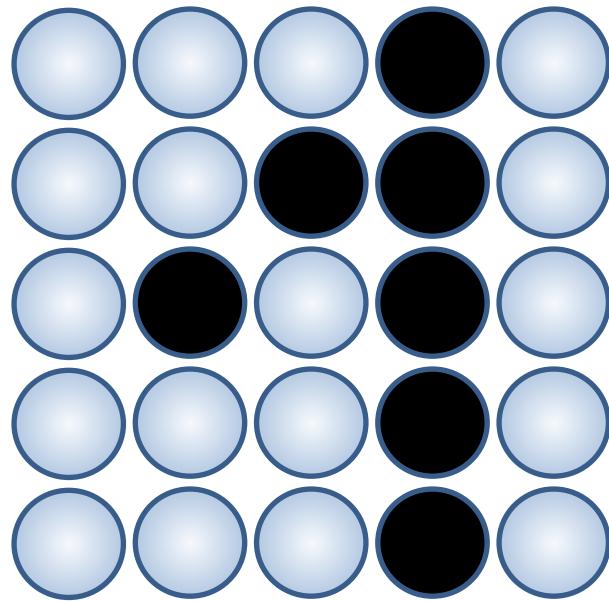
Next filter $\mathbf{f}_2 =$



A diagram illustrating the convolution operation. An input layer (3x3 grid) is multiplied by a filter \mathbf{f}_2 (3x3 kernel). The result is 9, indicated by a curved arrow pointing to the product. The input values are 3, -3, 7; 3, -3, 7; 5, -5, 9. The filter values are -1, 1, -1; -1, 1, -1; -1, 1, -1. The central value of the filter is highlighted in black.

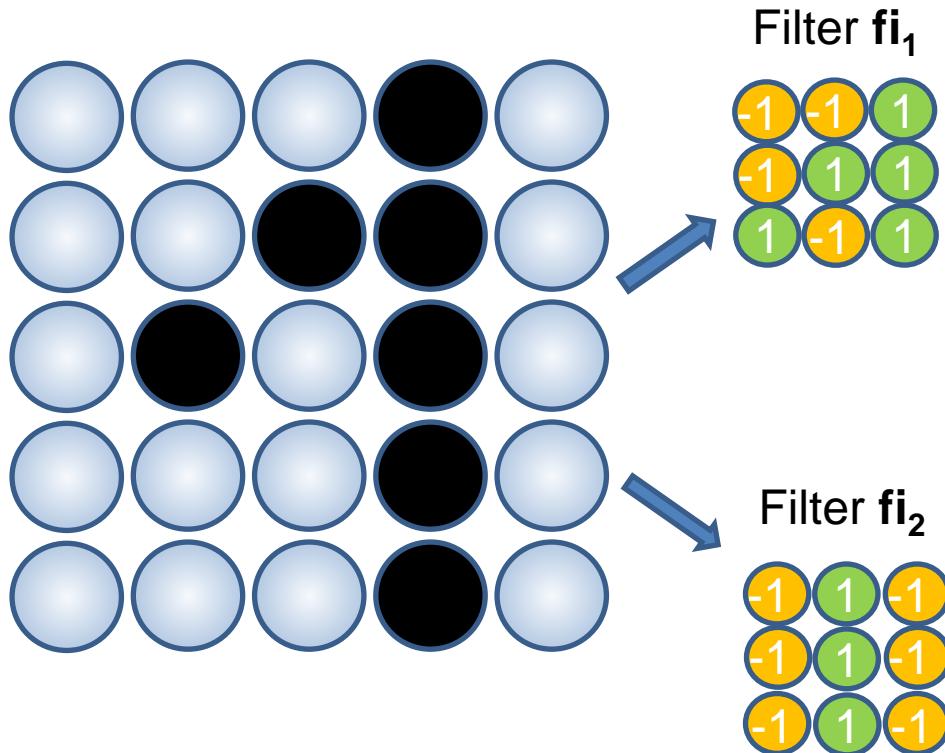
Convolutional Neural Networks

Learning – Feature Extraction: filter comparison



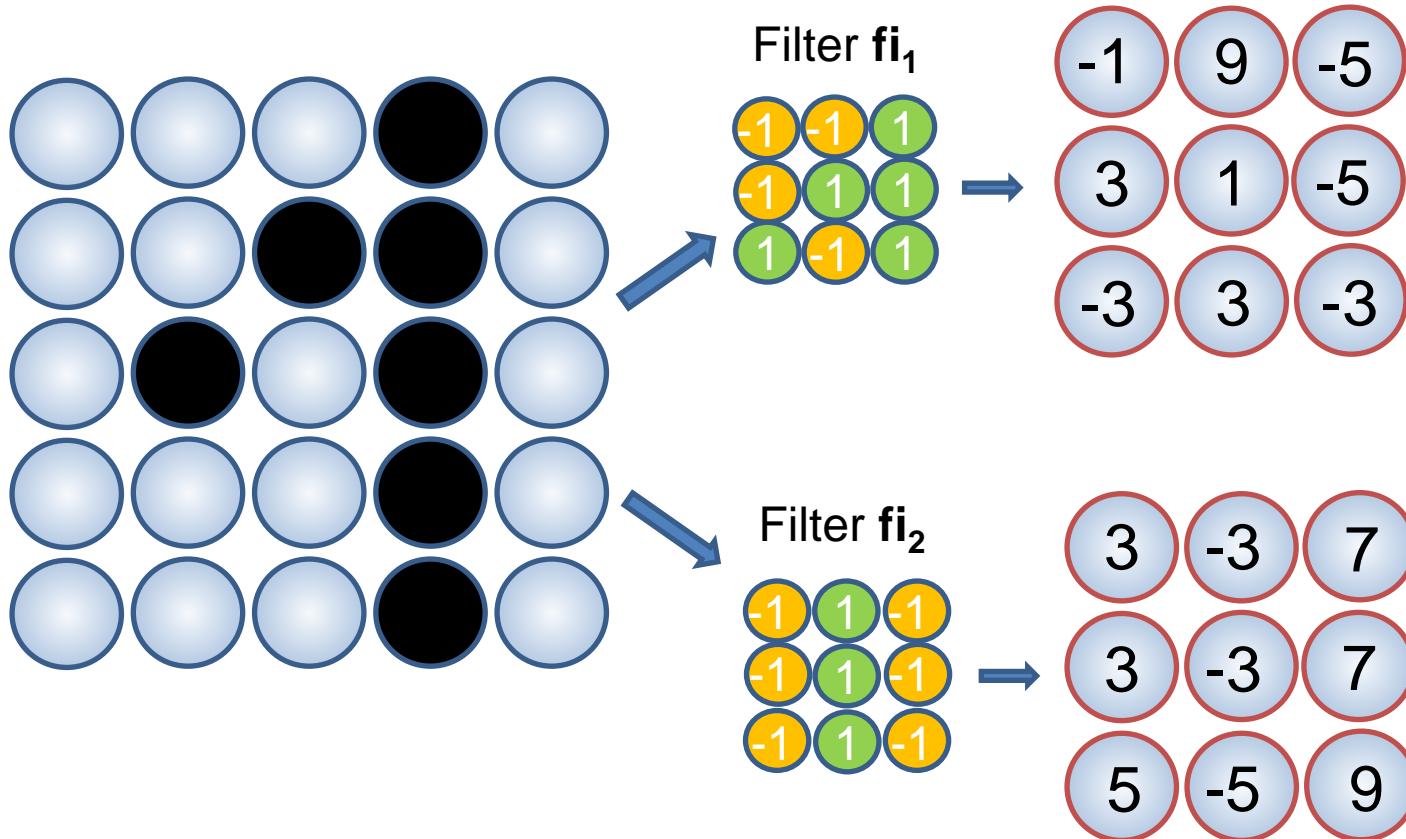
Convolutional Neural Networks

Learning – Feature Extraction: filter comparison



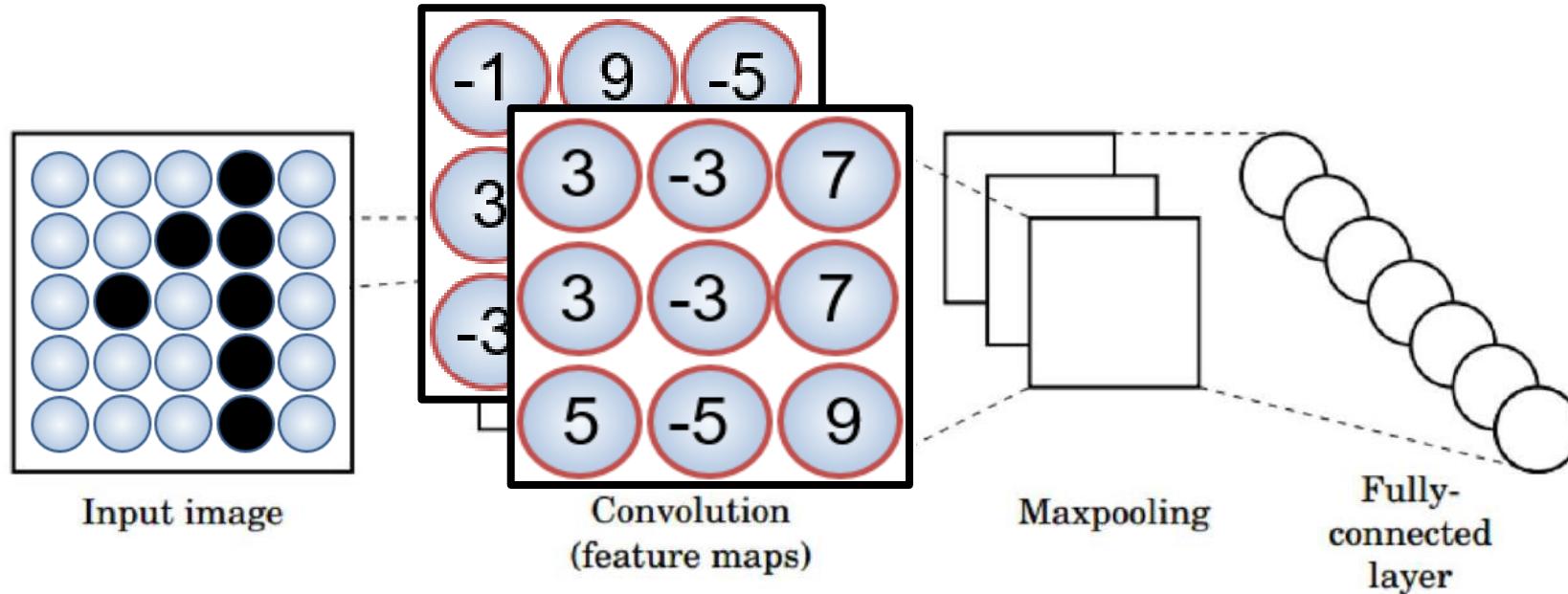
Convolutional Neural Networks

Learning – Feature Extraction: filter comparison



Convolutional Neural Networks

Building a CNN - Example

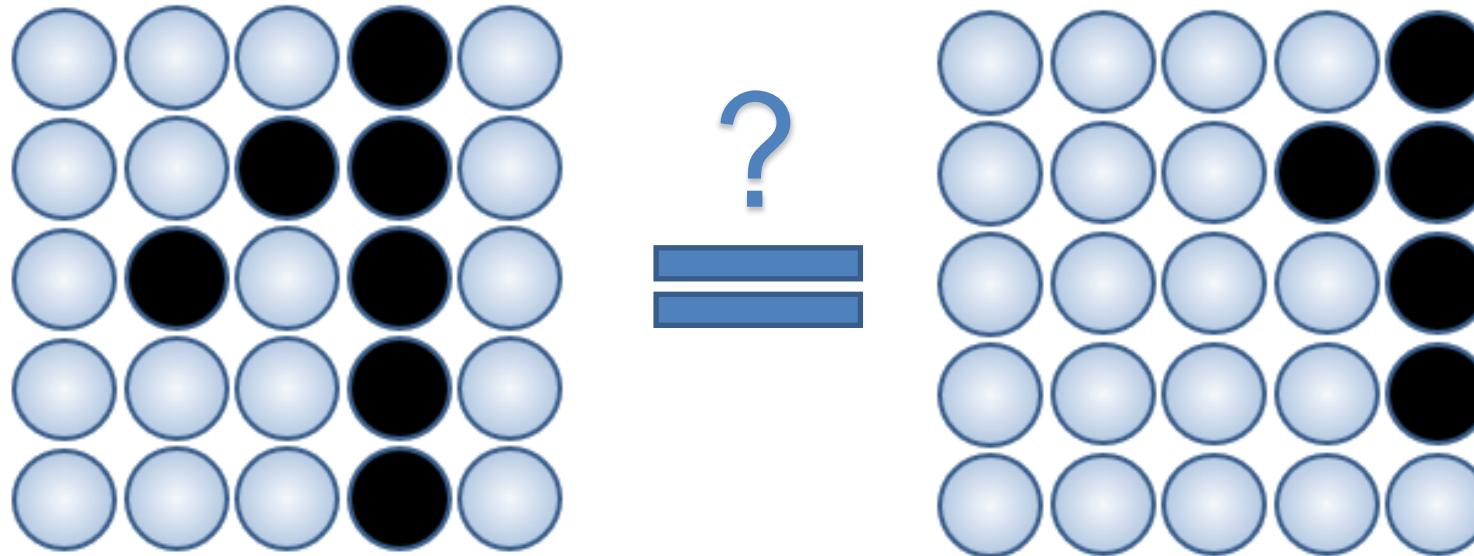


- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

Train model with image data.
Learn weights of filters in convolutional layers.

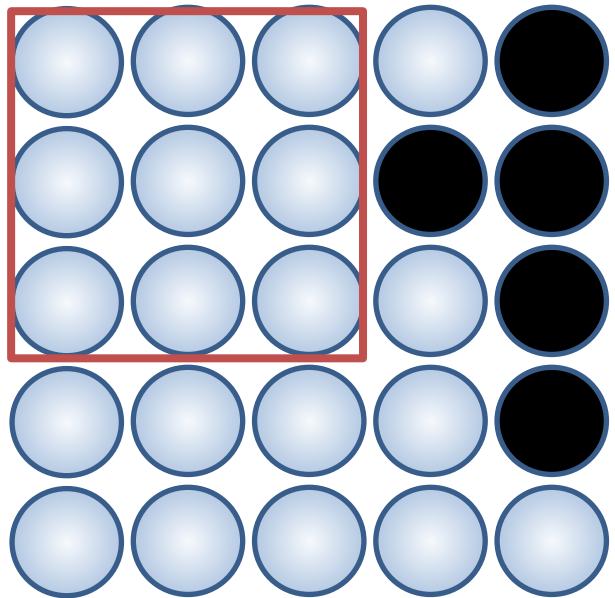
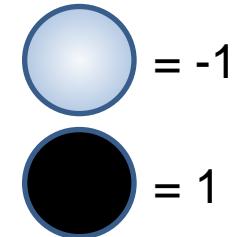
Convolutional Neural Networks

Filters to detect features



Convolutional Neural Networks

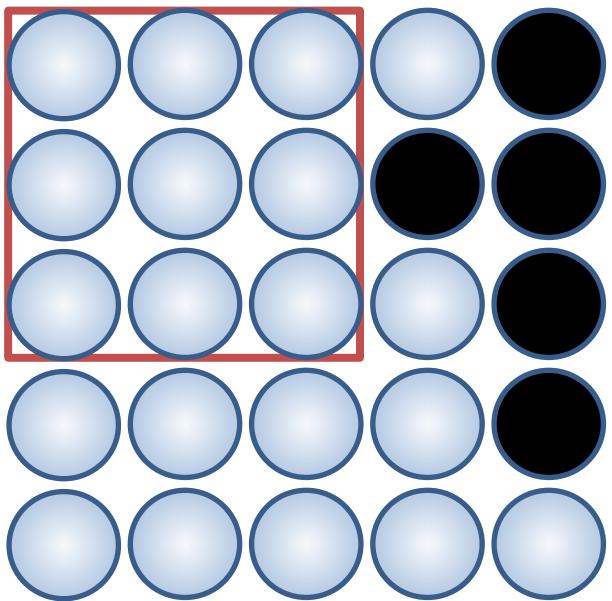
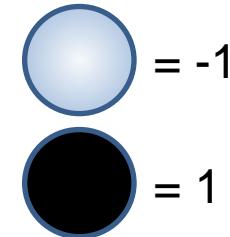
Filters to detect features



The usage of filters in detecting features allow good identification even in the event of **displacement, shrinkage, rotation or deformation**.

Convolutional Neural Networks

Filters to detect features



The usage of filters in detecting features allow good identification even in the event of **displacement, shrinkage, rotation or deformation**.

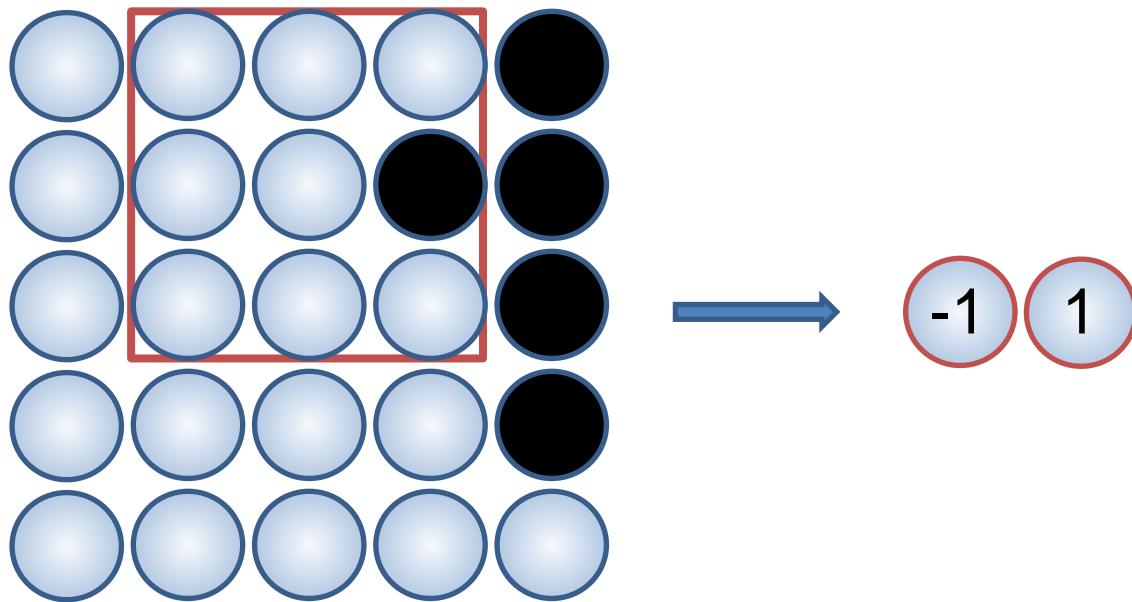
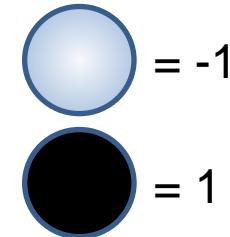
e.g. Filter $f_{i_1} =$

$$\begin{matrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{matrix}$$

$$\begin{matrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{matrix} \times \begin{matrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{matrix} = \dots$$

Convolutional Neural Networks

Filters to detect features



e.g. Filter $\mathbf{f}_1 =$

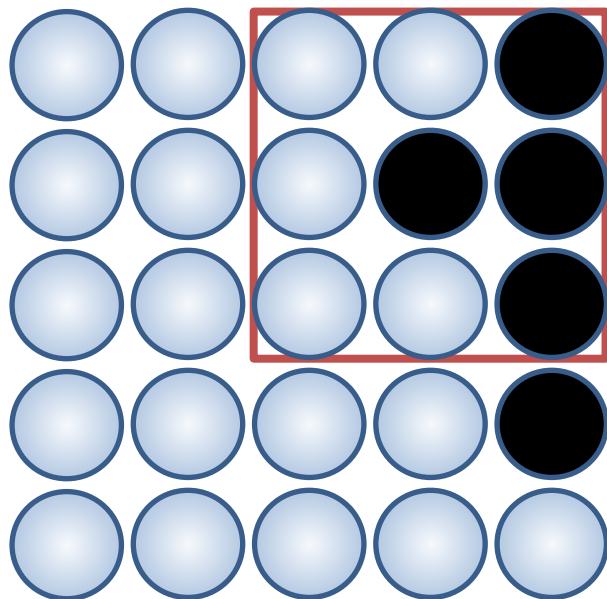


Feature/filter \mathbf{f}_1 still matches the deformed image well

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & \mathbf{1} \\ -1 & -1 & -1 \end{bmatrix} \times \begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix} = \dots$$

Convolutional Neural Networks

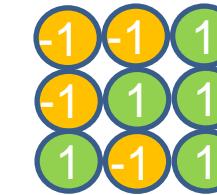
Filters to detect features



Not a full match
but still a high score



e.g. Filter $f_{i_1} =$



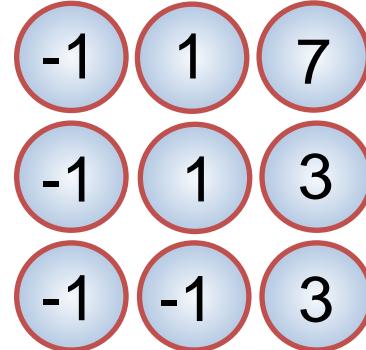
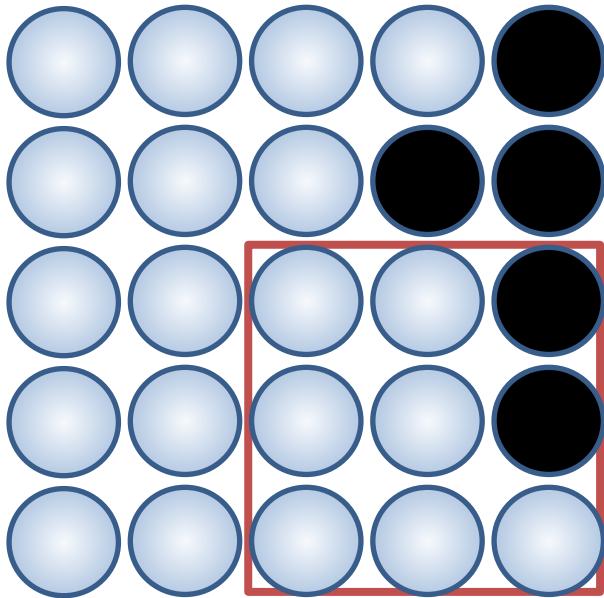
Feature/filter f_{i_1} still matches the
deformed image well
Remember! **A full match would be 9**

The diagram shows the convolution operation. On the left is the 3x3 input subgrid with values: -1, -1, 1; -1, 1, 1; -1, -1, 1. On the right is the 3x3 filter with values: -1, -1, 1; -1, 1, 1; 1, -1, 1. A multiplication symbol "x" is between them, and an equals sign "=" follows the sum of the products. The calculation is: $= 1+1+1+1+1+(-1)+1+1 = 7$.

Convolutional Neural Networks

Filters to detect features

$$\begin{array}{c} \text{Light Blue Circle} \\ = -1 \\ \text{Black Circle} \\ = 1 \end{array}$$



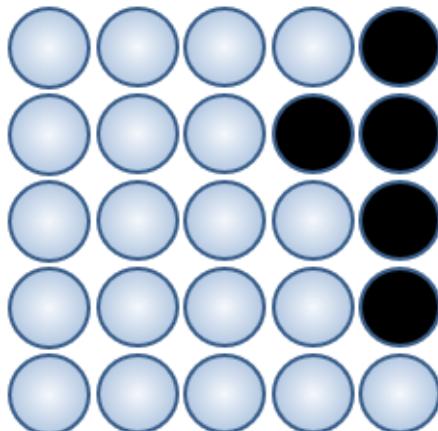
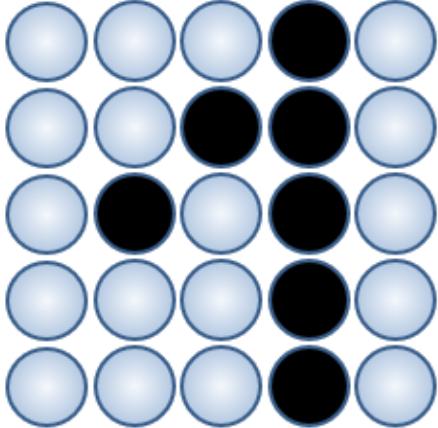
e.g. Filter $f_{i_1} =$



Finished feature map
using filter f_{i_1}

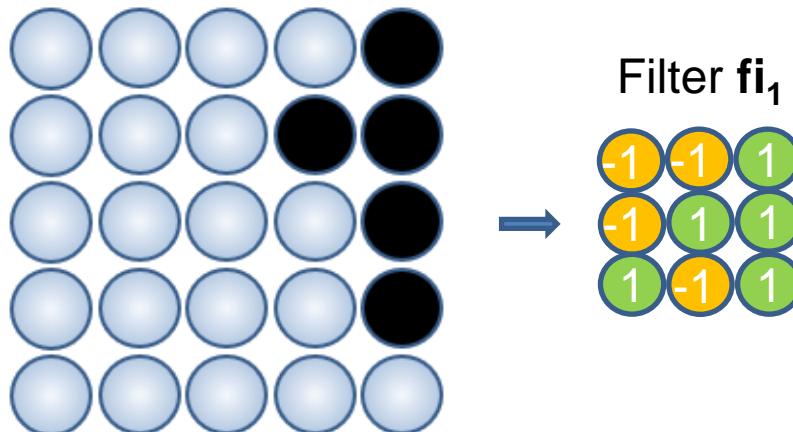
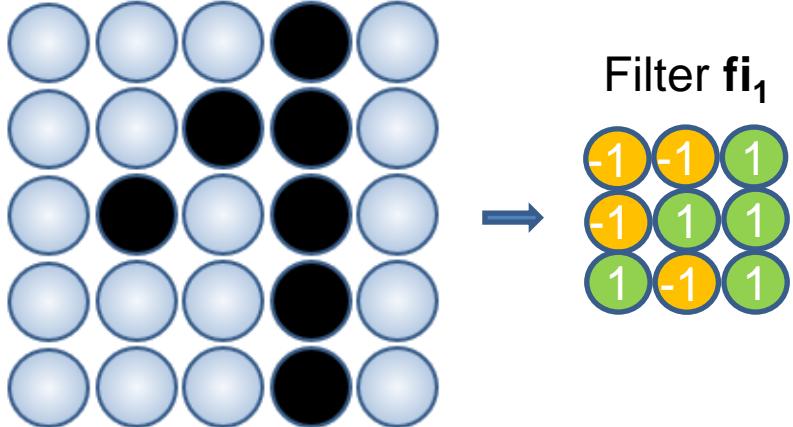
Convolutional Neural Networks

Filters to detect features: comparison



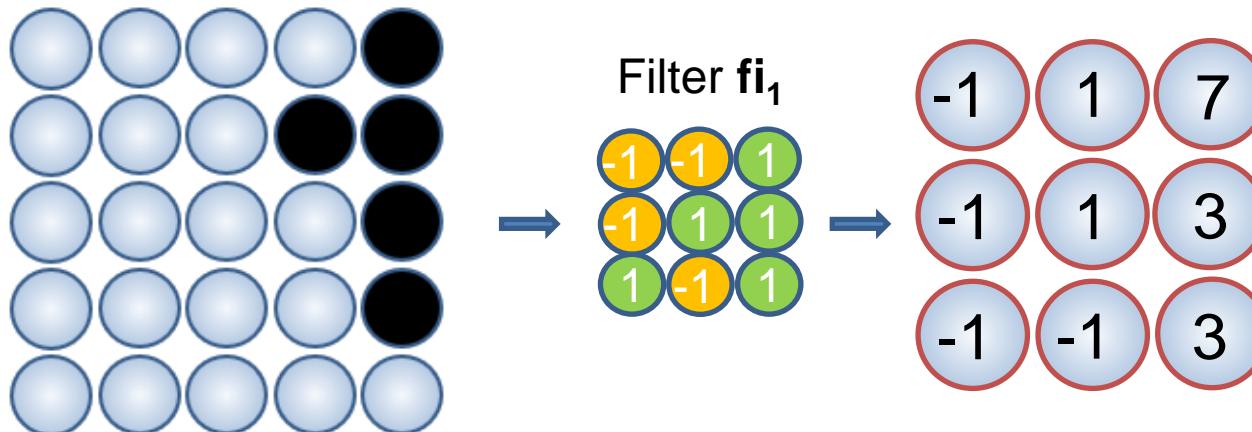
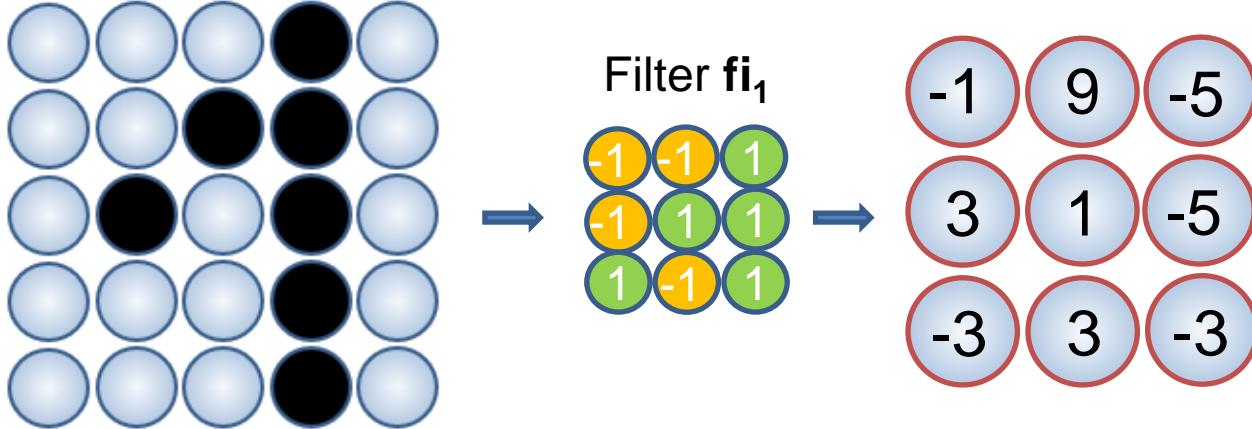
Convolutional Neural Networks

Filters to detect features: comparison



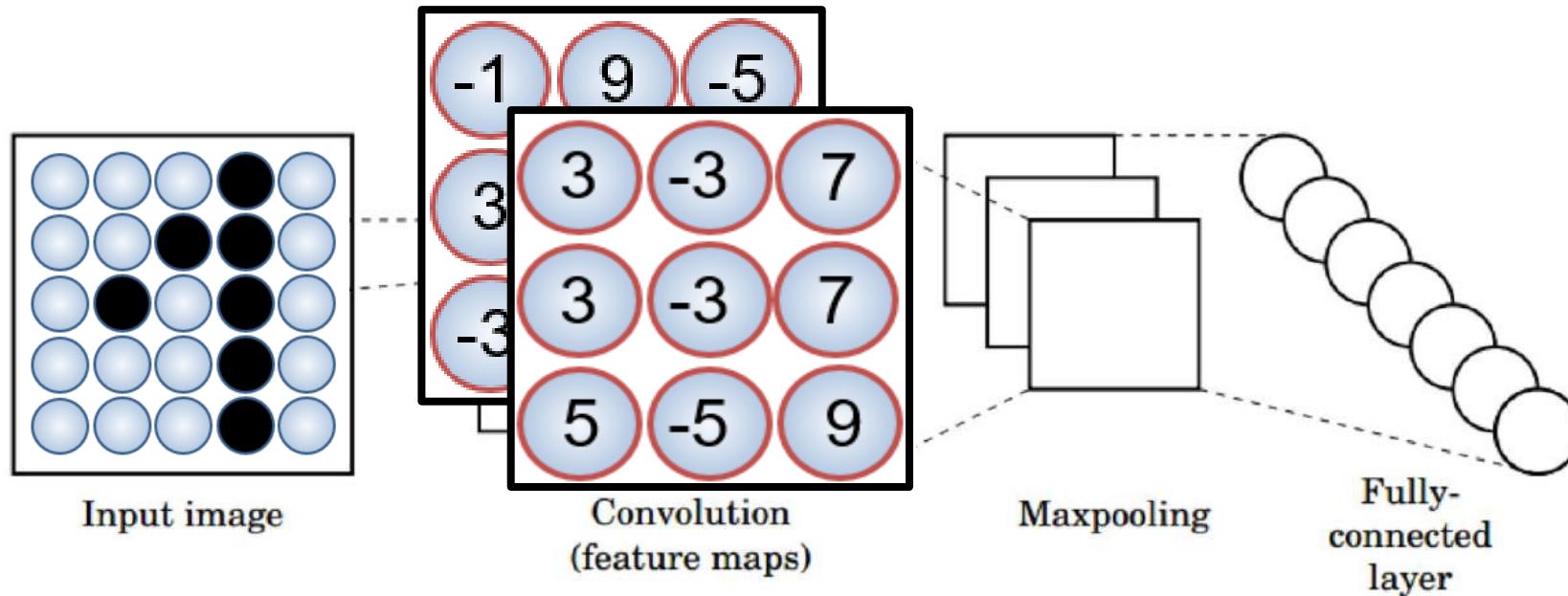
Convolutional Neural Networks

Filters to detect features: comparison



Convolutional Neural Networks

Pooling



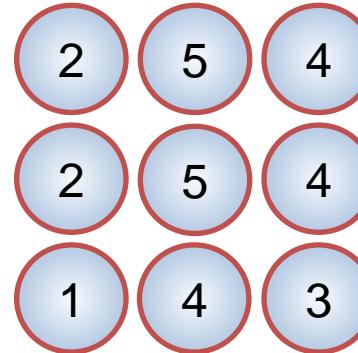
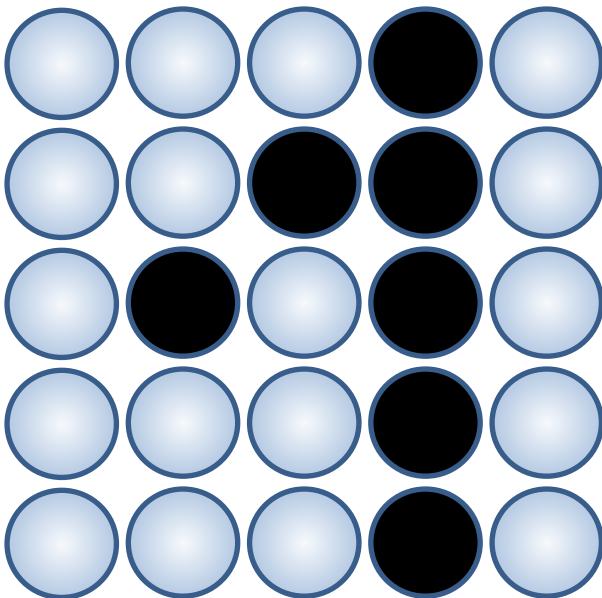
- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

**Train model with image data.
Learn weights of filters in convolutional layers.**

Convolutional Neural Networks

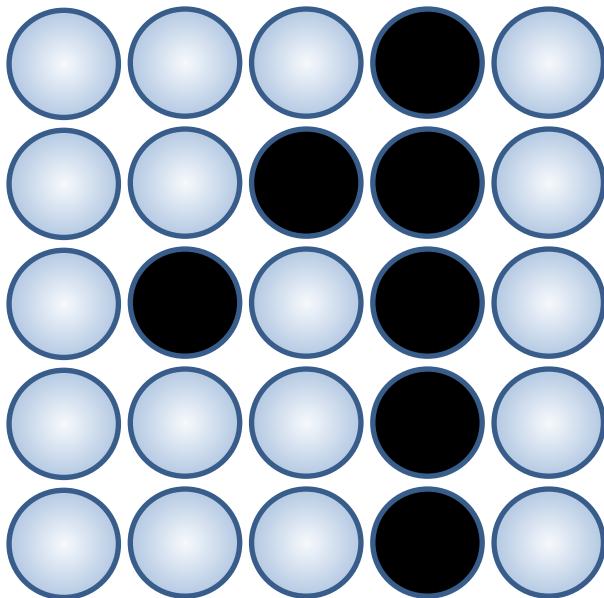
Max Pool – Example

- We already have the layer with the 9 neurons for the 5×5 image

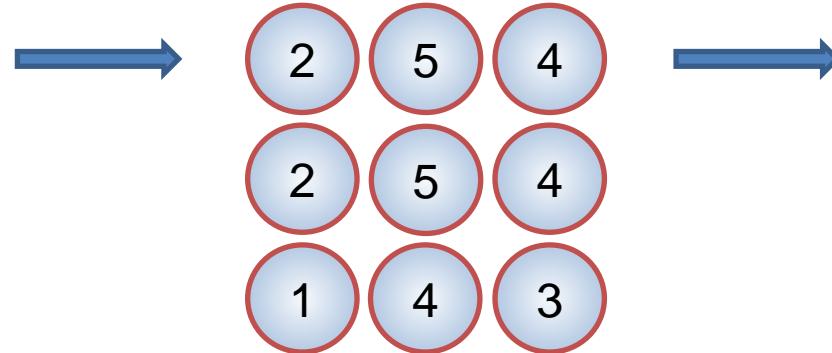


Convolutional Neural Networks

Max Pool – Example

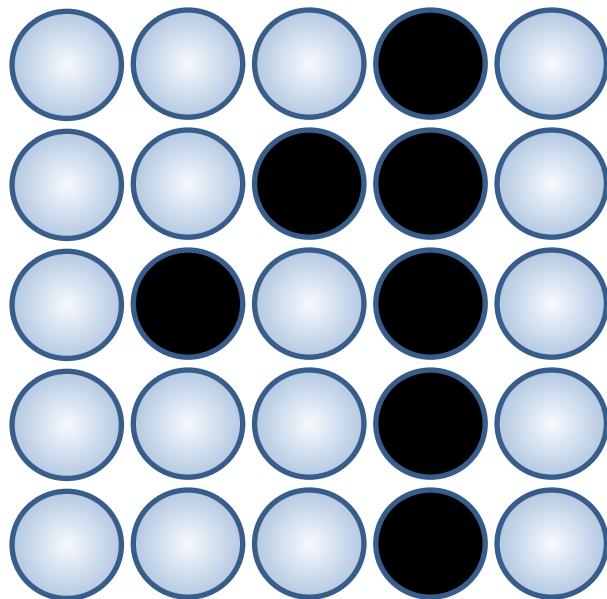


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance

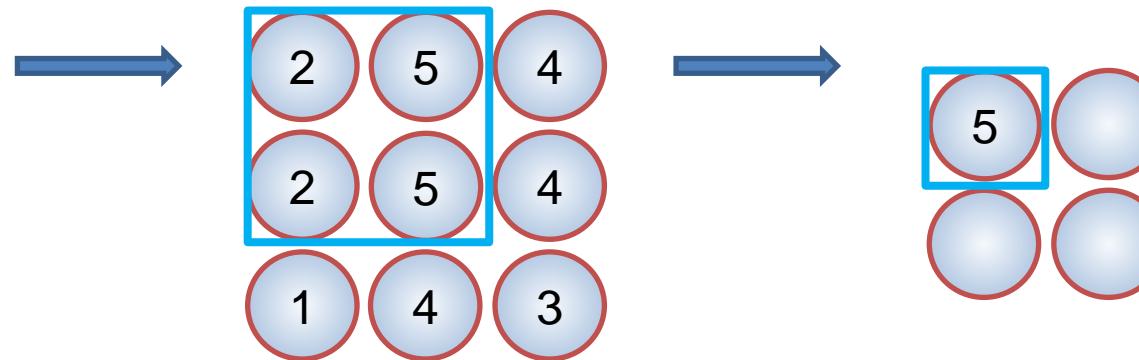


Convolutional Neural Networks

Max Pool – Example

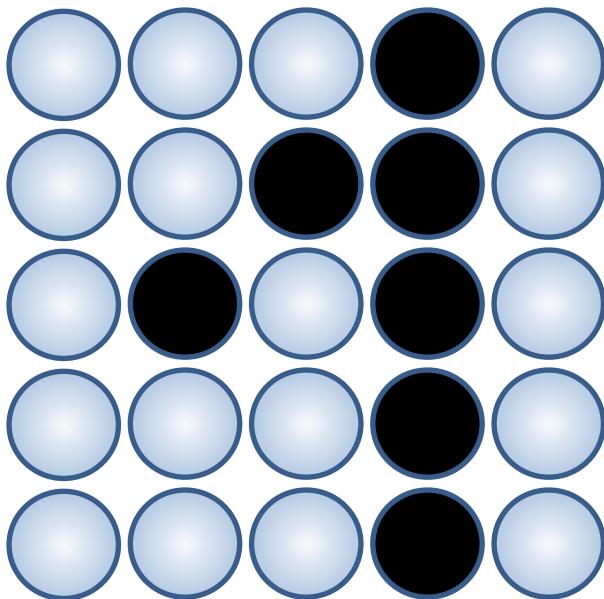


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1)

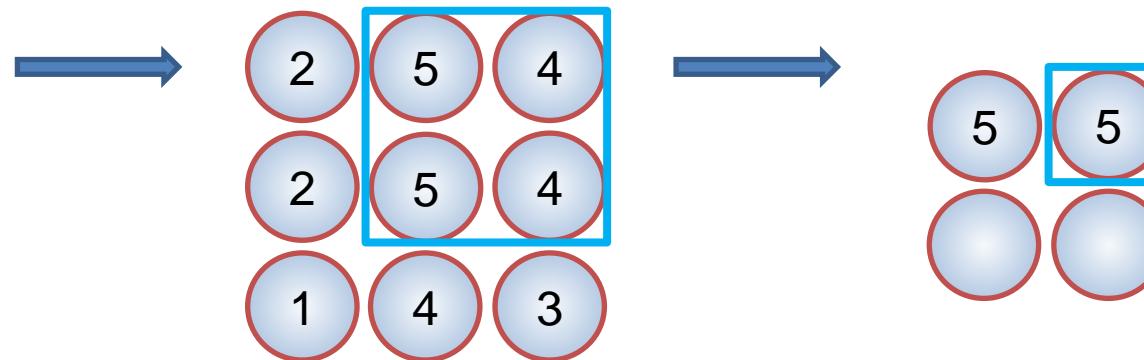


Convolutional Neural Networks

Max Pool – Example

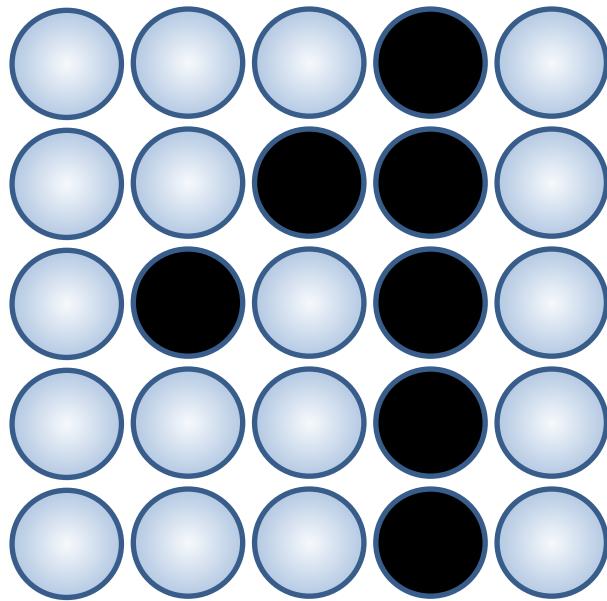


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1)

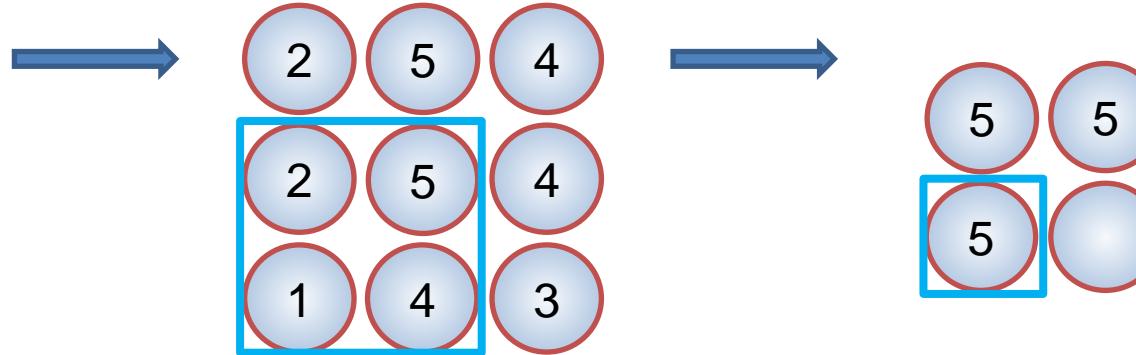


Convolutional Neural Networks

Max Pool – Example

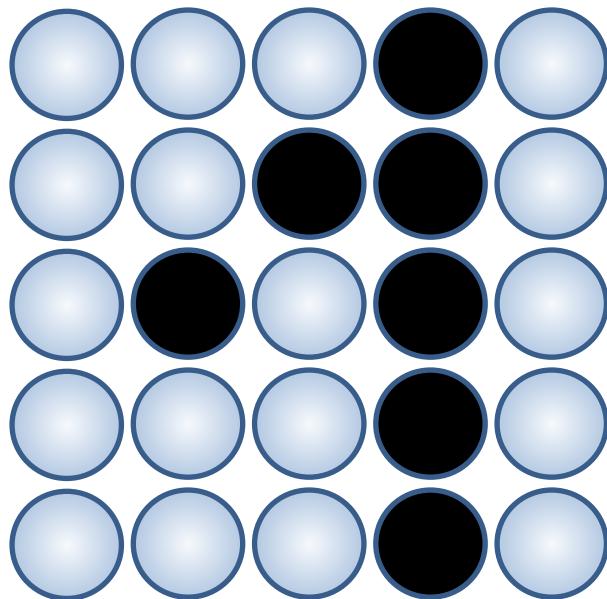


- We already have the layer with the 9 neurons for the 5×5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2×2 filters and stride 1)

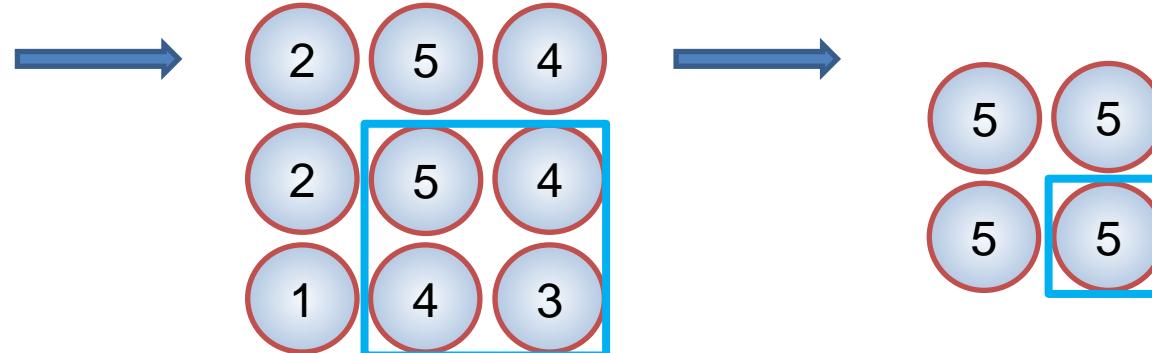


Convolutional Neural Networks

Max Pool – Example

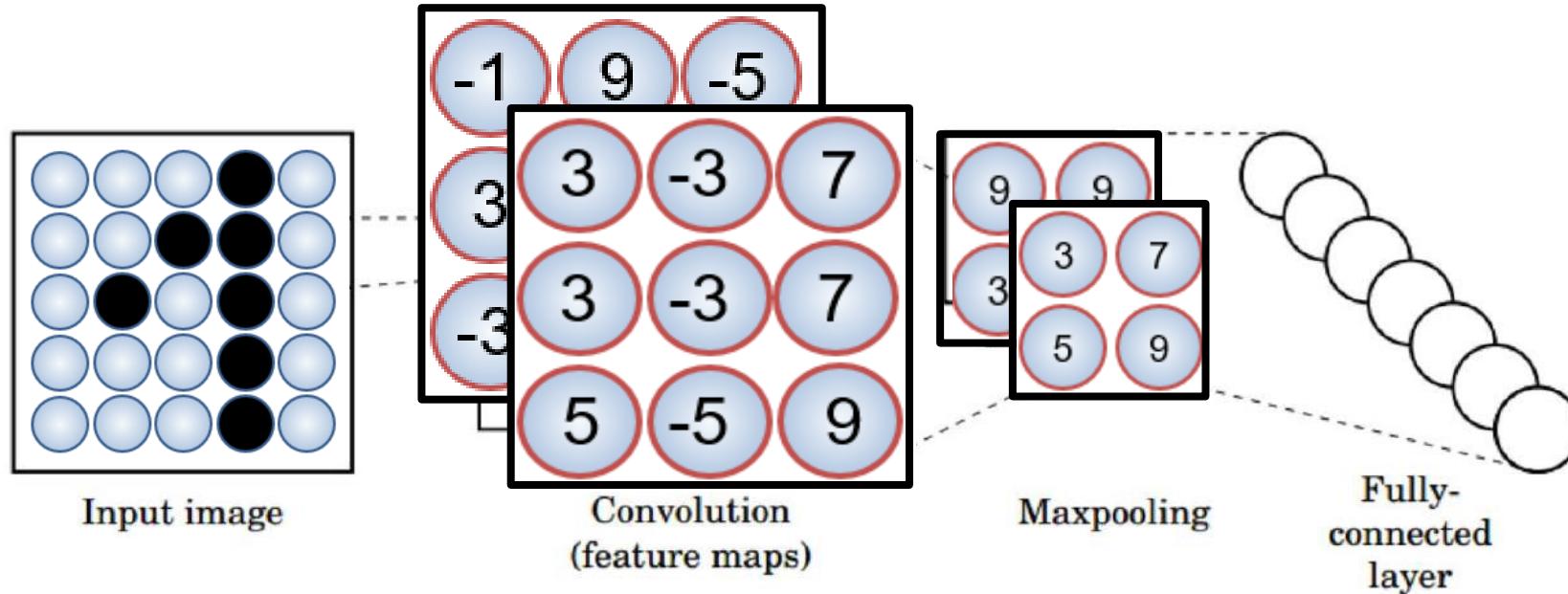


- We already have the layer with the 9 neurons for the 5×5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2×2 filters and stride 1)



Convolutional Neural Networks

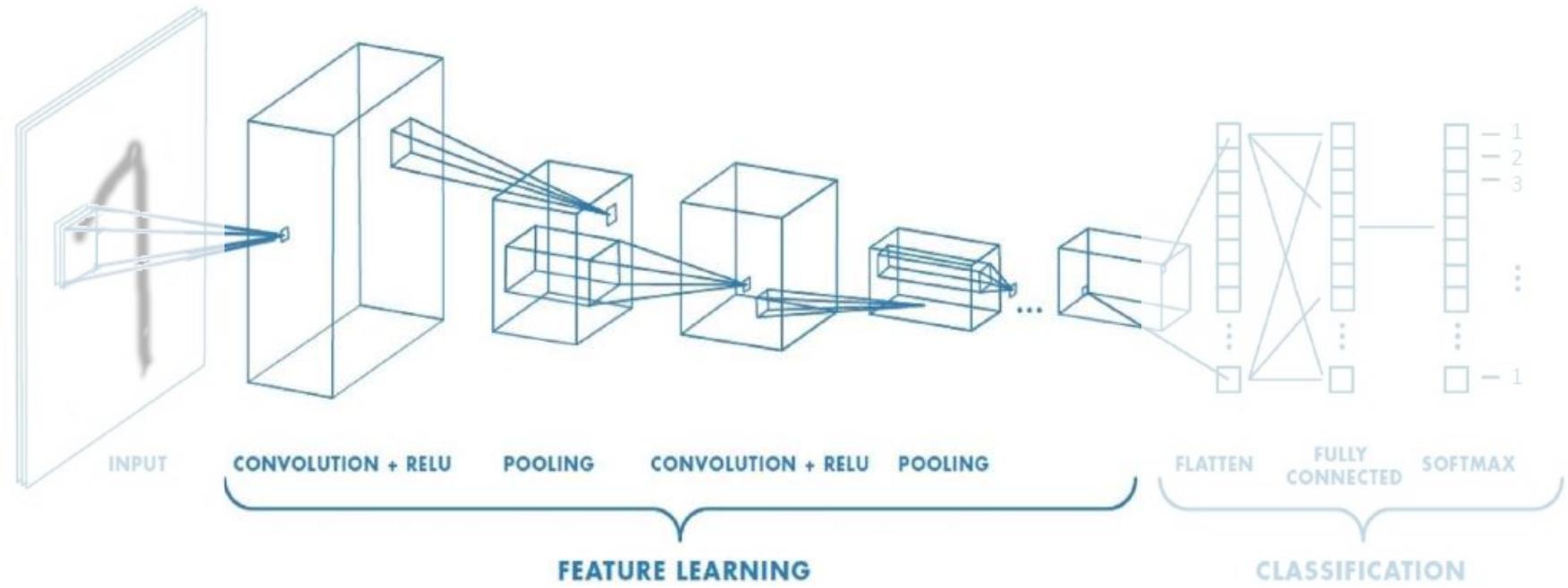
Building a CNN – Example



- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

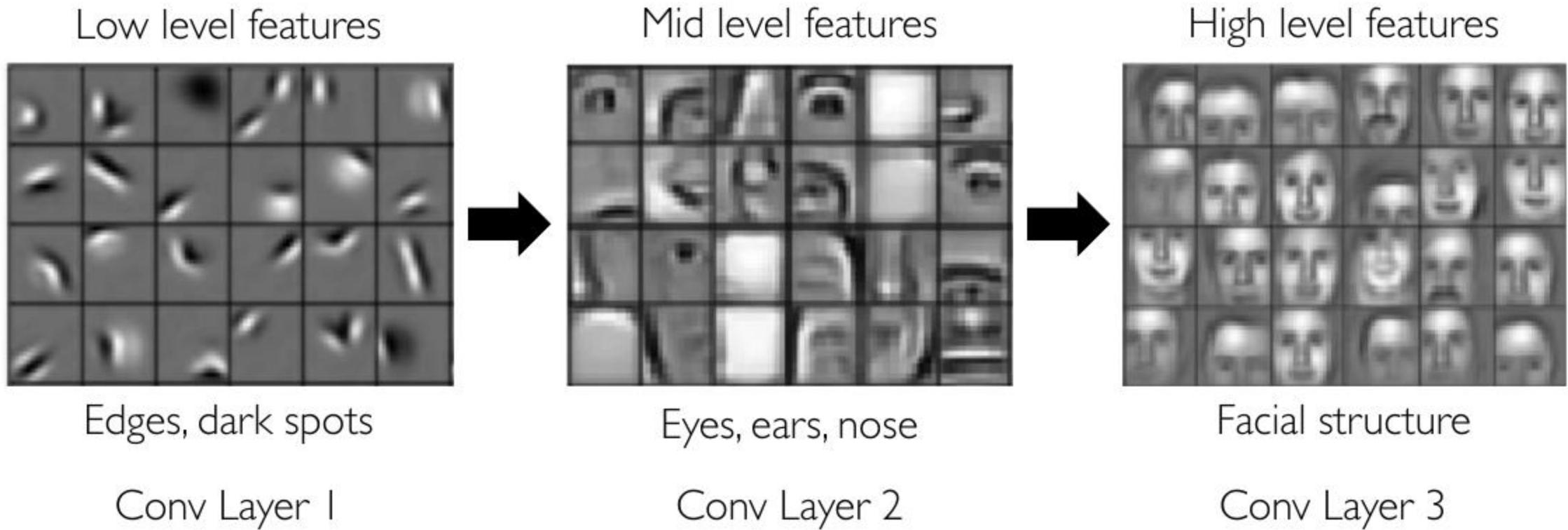
Train model with image data.
Learn weights of filters in convolutional layers.

Representation Learning

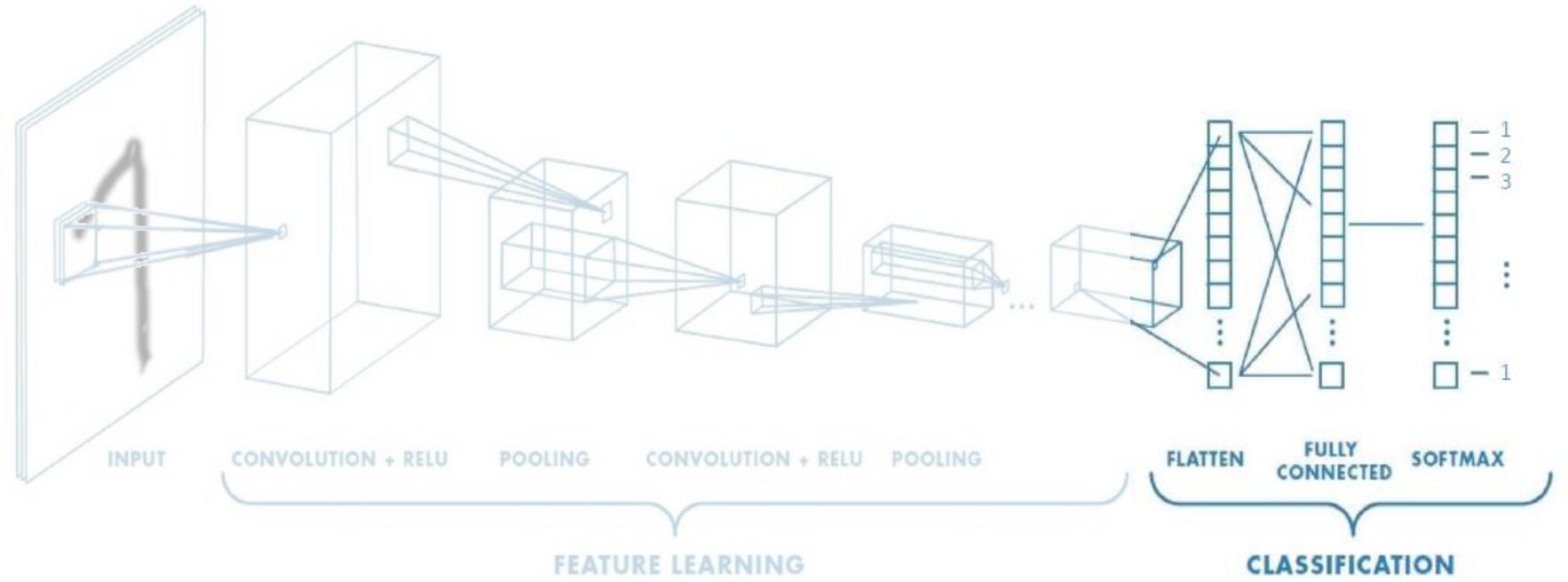


Convolutional Neural Networks

Representation Learning



Classification

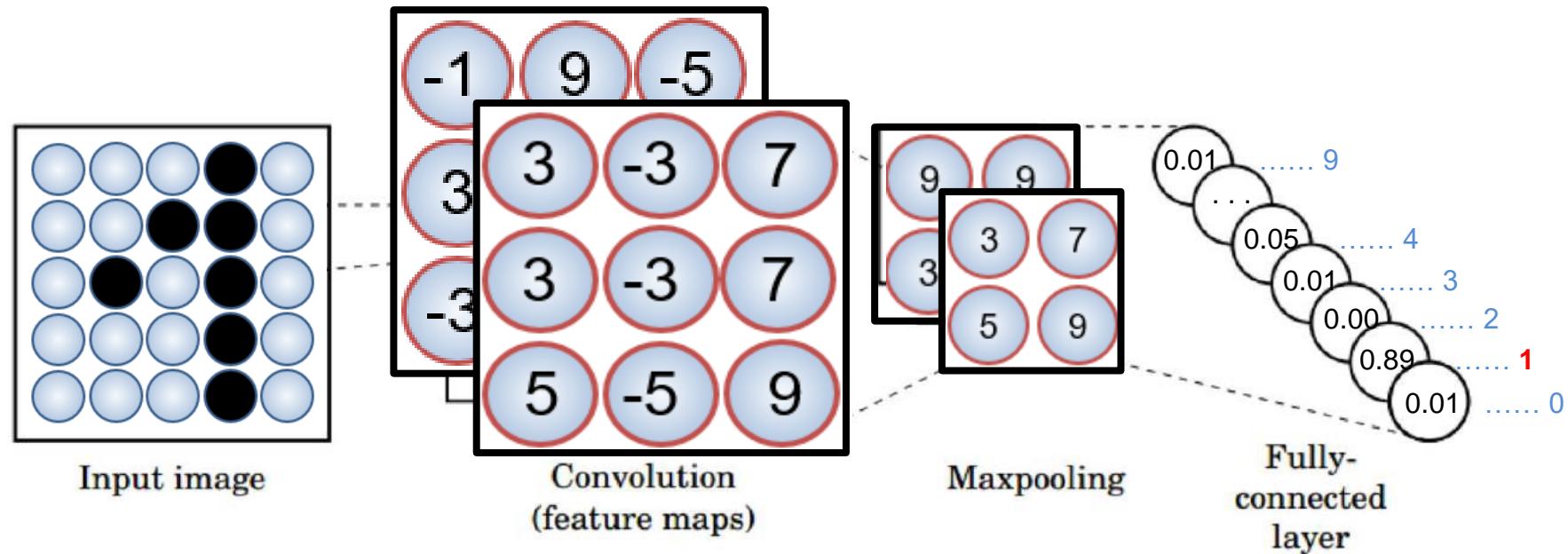


- CONV and POOL layers output high-level features of input
- Fully connected layer uses these features for classifying input image
- Express output as **probability** of image belonging to a particular class

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Convolutional Neural Networks

Building a CNN – Example



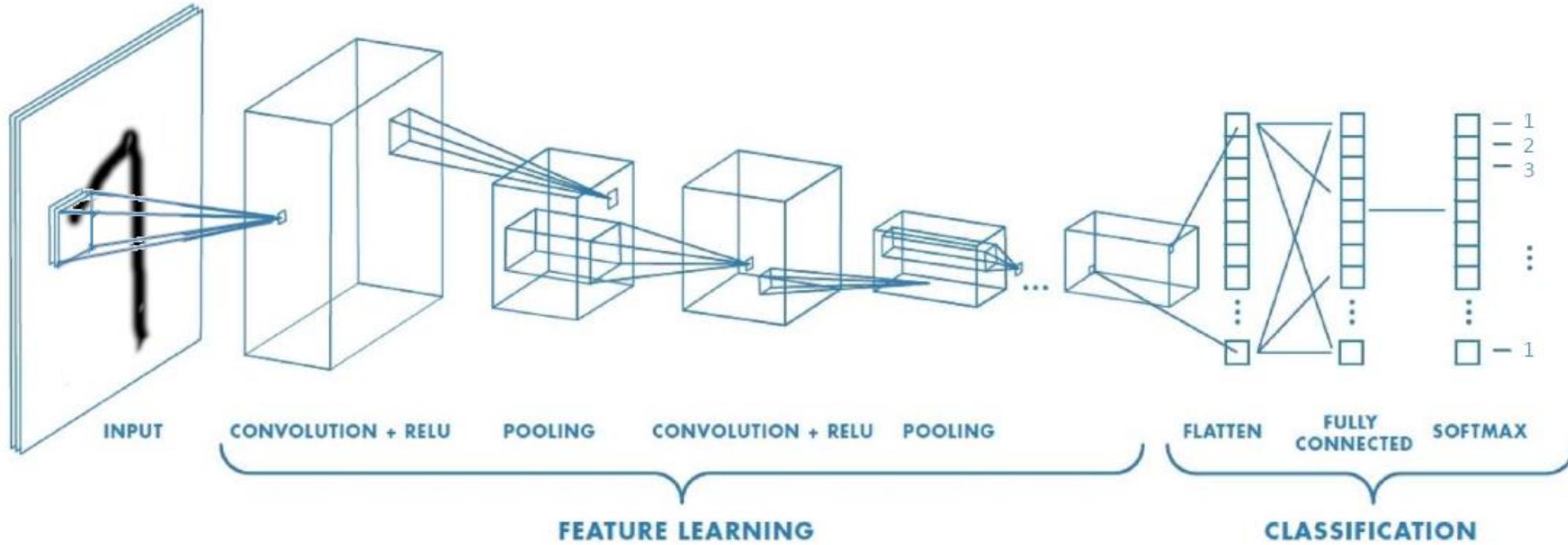
- 1. Convolution:** Apply filters with learned weights to generate feature maps.
 - 2. Non-linearity:** Often ReLU.
 - 3. Pooling:** Downsampling operation on each feature map.

Train model with image data.

Learn weights of filters in convolutional layers.

Convolutional Neural Networks

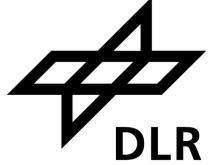
Training with backpropagation



Learn weights for convolutional filters and fully connected layers
Backpropagation: cross-entropy loss

$$J(\theta) = \sum_i y^{(i)} \log(\hat{y}^{(i)})$$

Curriculum



Next:

- Hands-on II

Imprint



Topic: **Introduction to Deep Learning**
Part II – Advanced Concept and Convolutional Neural Network

Date: 2025-11-13

Author: Auliya Fitri, Sai Vemuri, Sreerag Naveenachandran

Institute: Data Science

Image sources: All images “DLR (CC BY-NC-ND 3.0)” unless otherwise stated