

Deep Learning Tutorial

Part II: Advanced concepts

April 13, 2022

Auliya Fitri, Jakob Gawlikowski

Machine Learning Group
Institute of Data Science (DW)
Jena



Curriculum

A: Theoretical introduction – Morning

- I. Introduction and basics
- II. Advanced concepts
- III. Practical application

B. Hands-on seminar – Afternoon

Run prepared Jupyter Notebooks online on Binder, or locally on your own laptop.



Curriculum

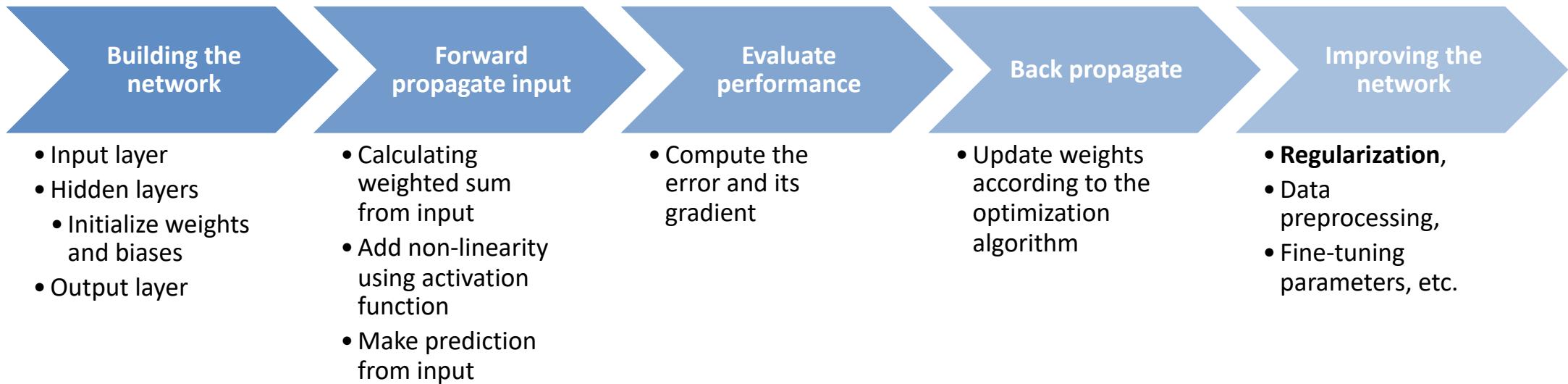
II. Advanced concepts

- Regularization
- Convolutional Neural Networks (CNNs)

Inspired by lectures from MIT and Hacettepe Üniversitesi; images taken from these, if not noted otherwise



Neural network concepts

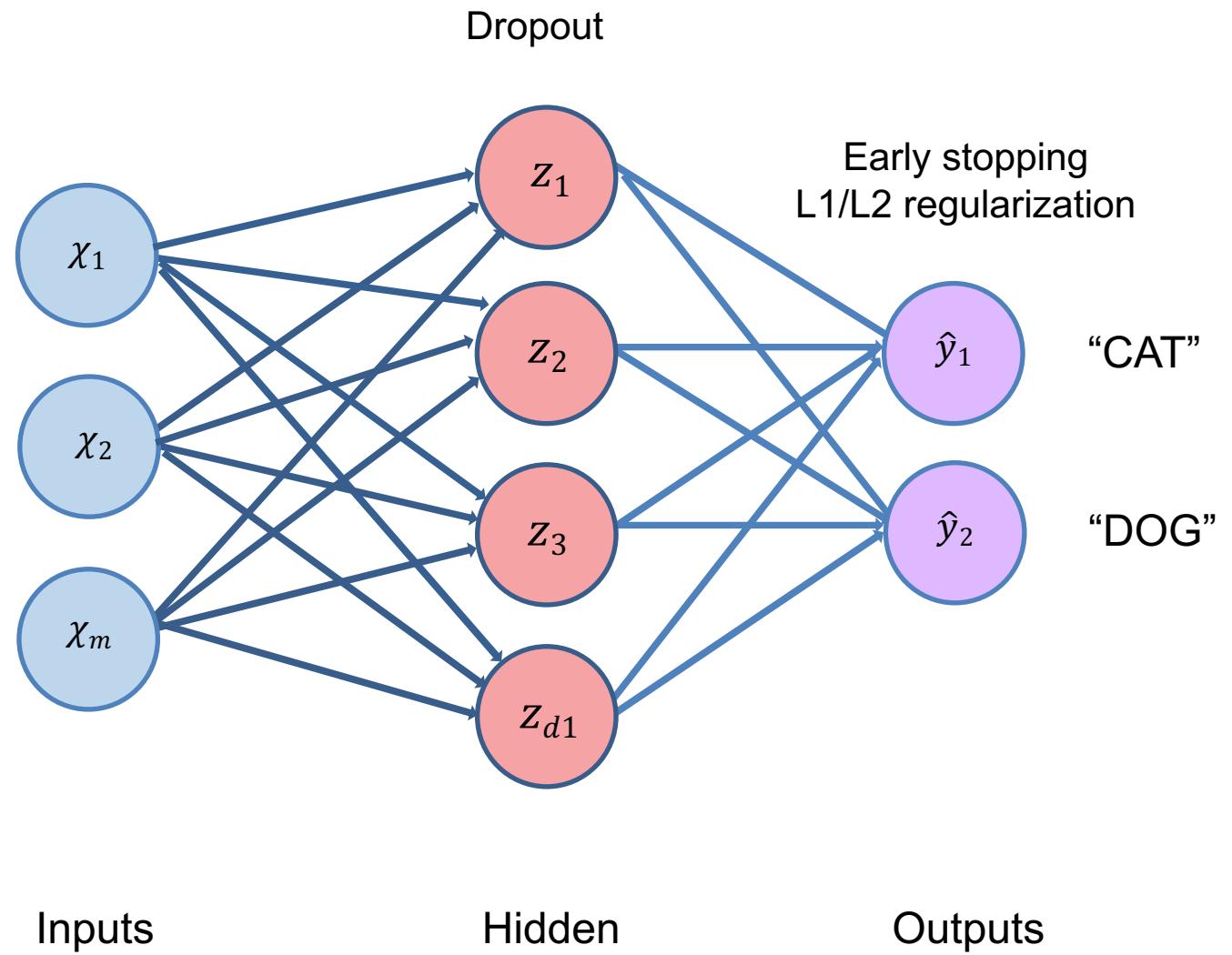


Improve the Network: Regularization

Data Augmentation

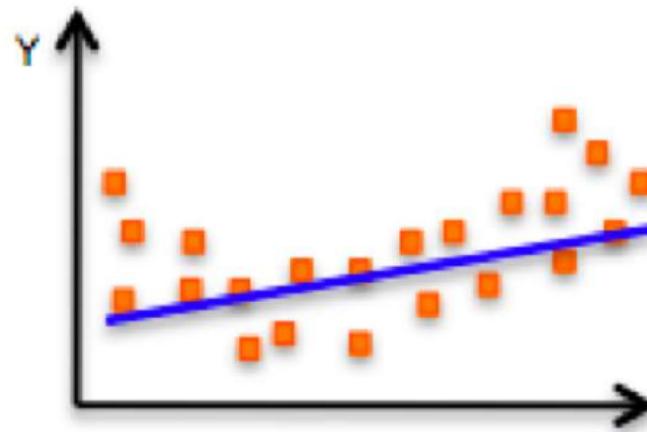


$$\begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_m \end{bmatrix}$$



Regularization

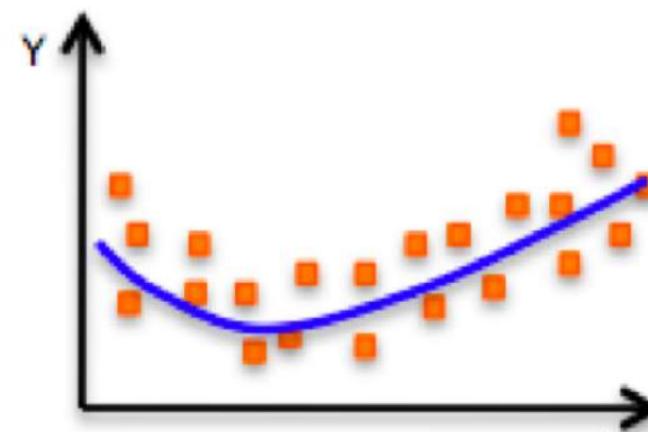
The overfitting problem



Underfitting

Model does not have capacity
to fully learn the data

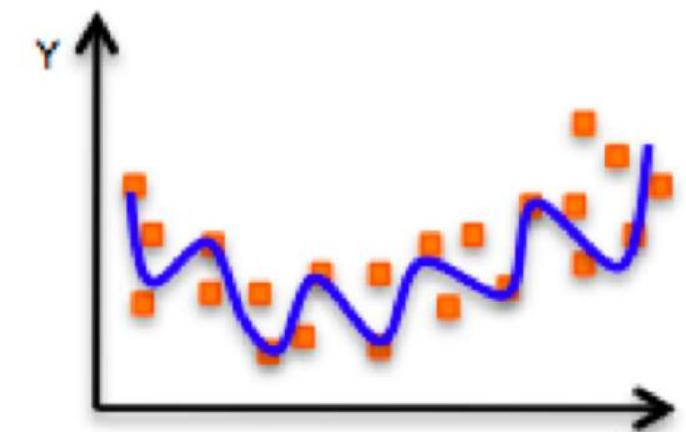
= High bias



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**

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

→ 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



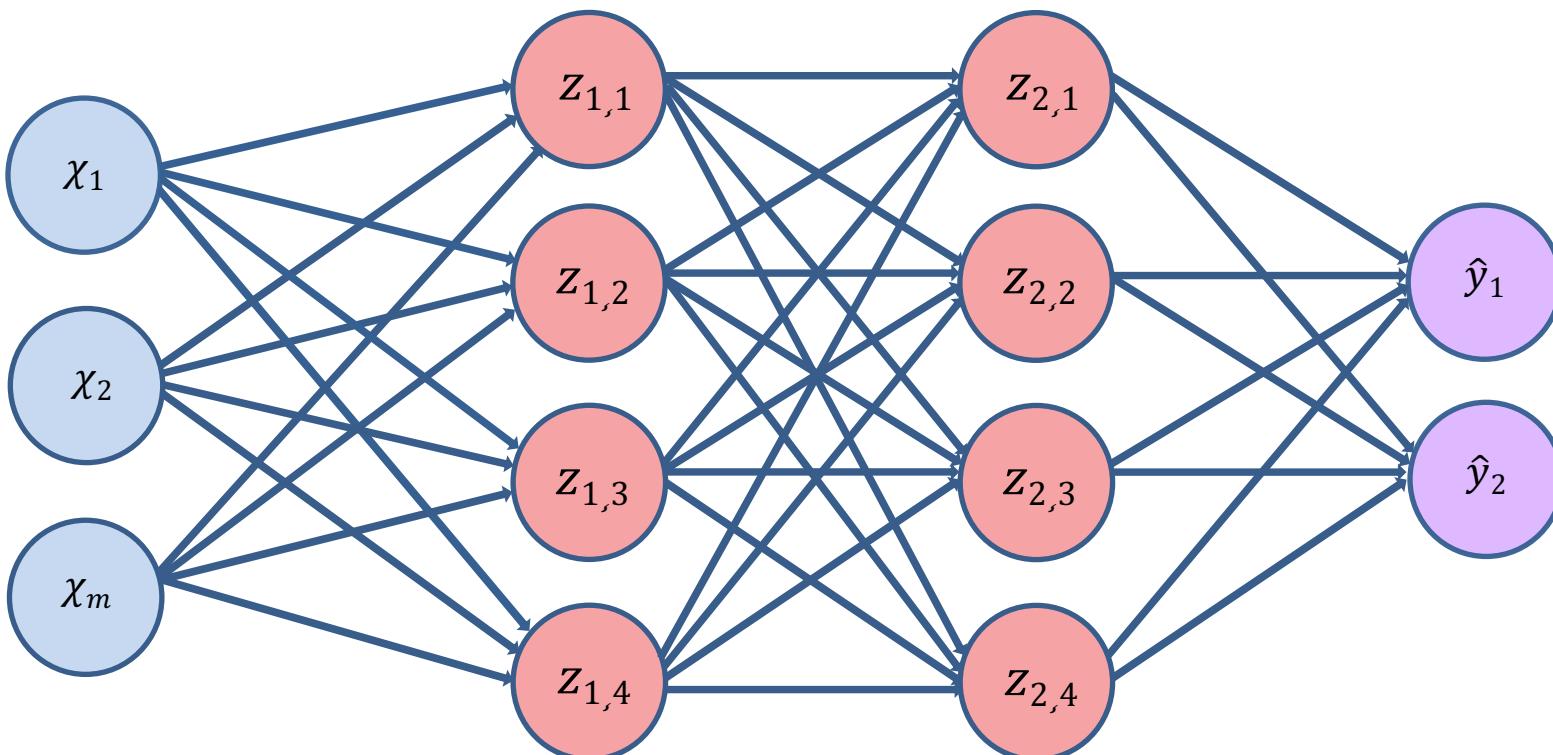
Tint



Regularization

Preventing overfitting: Dropout

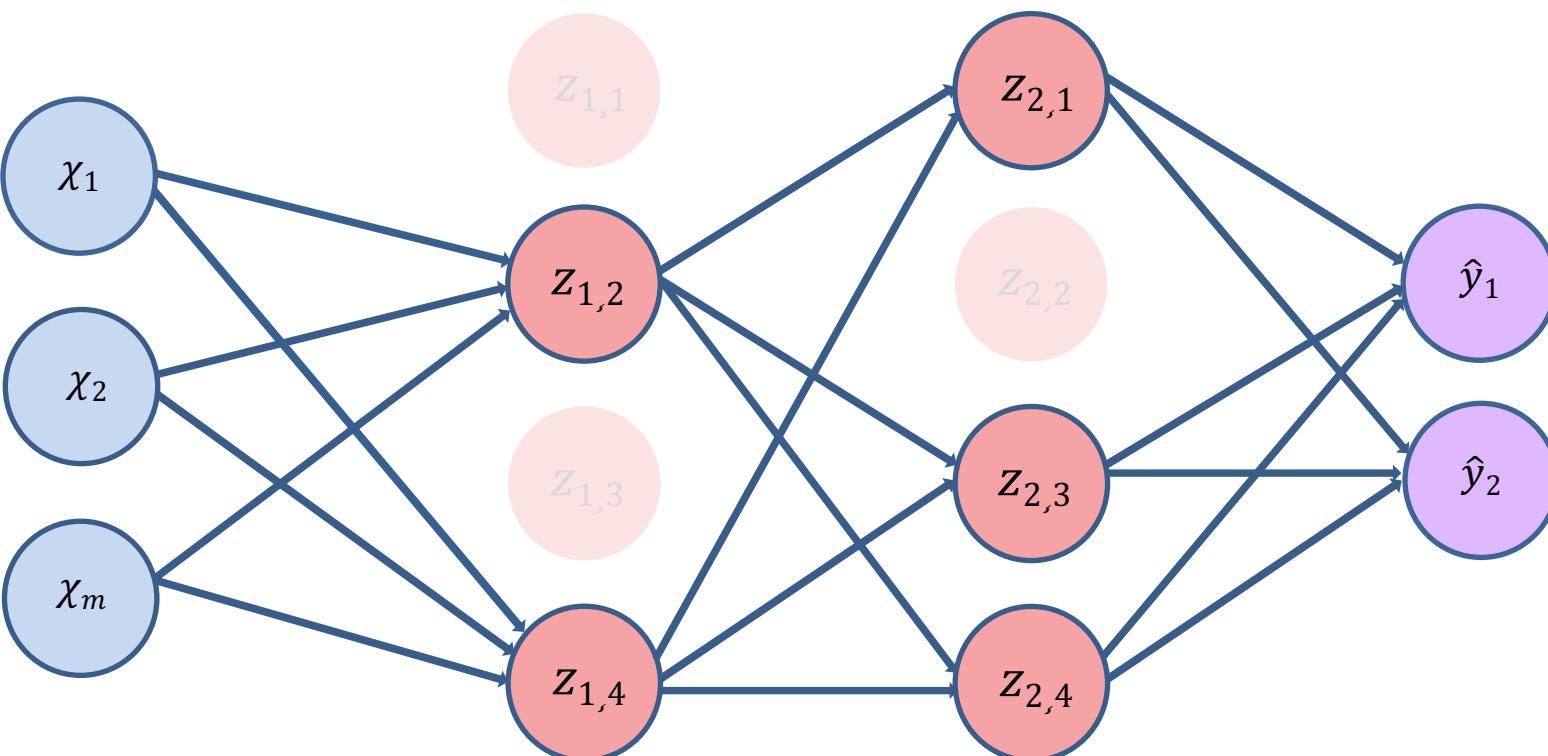
- During training, randomly set some activations to 0



Regularization

Preventing overfitting: Dropout

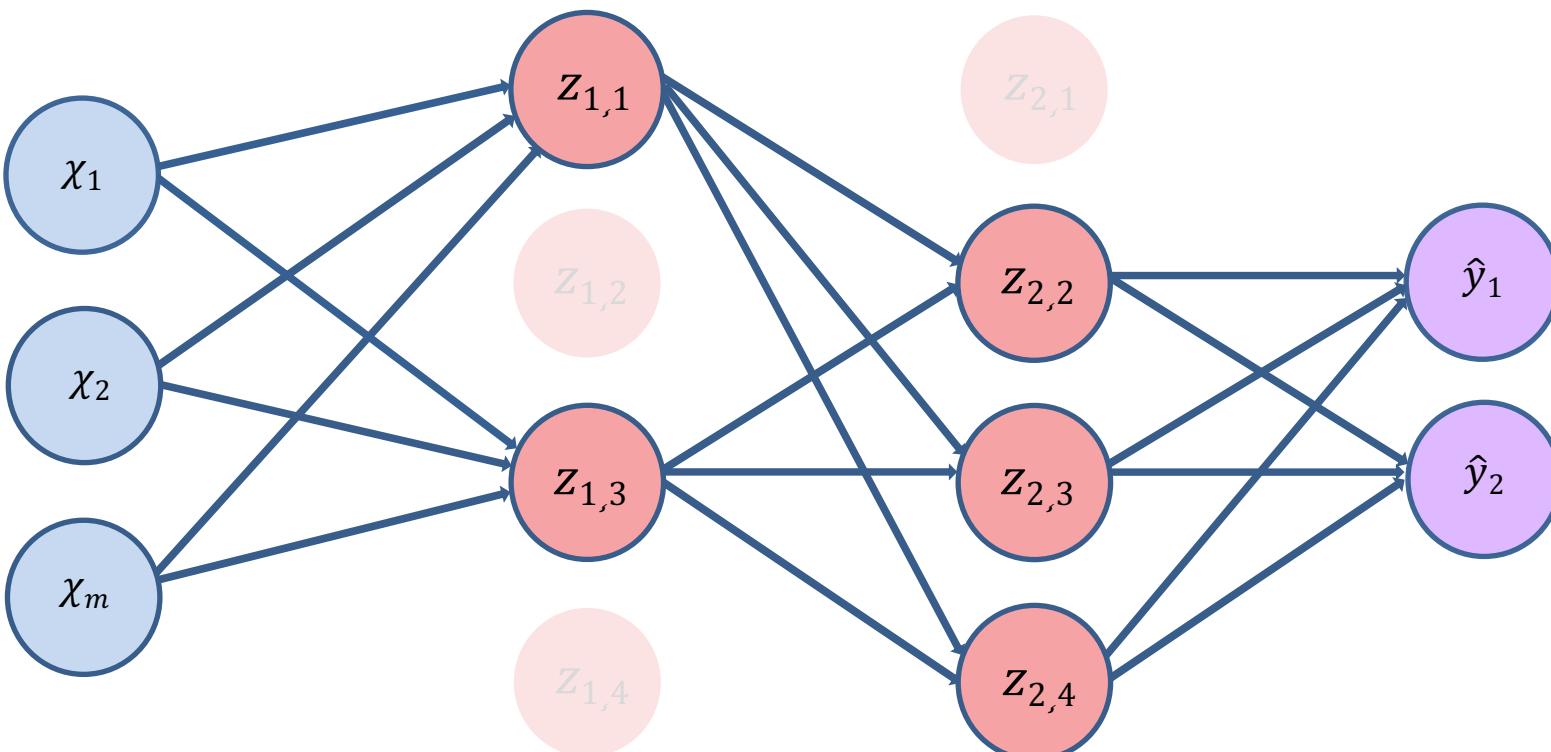
- During training, randomly set some activations to 0



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



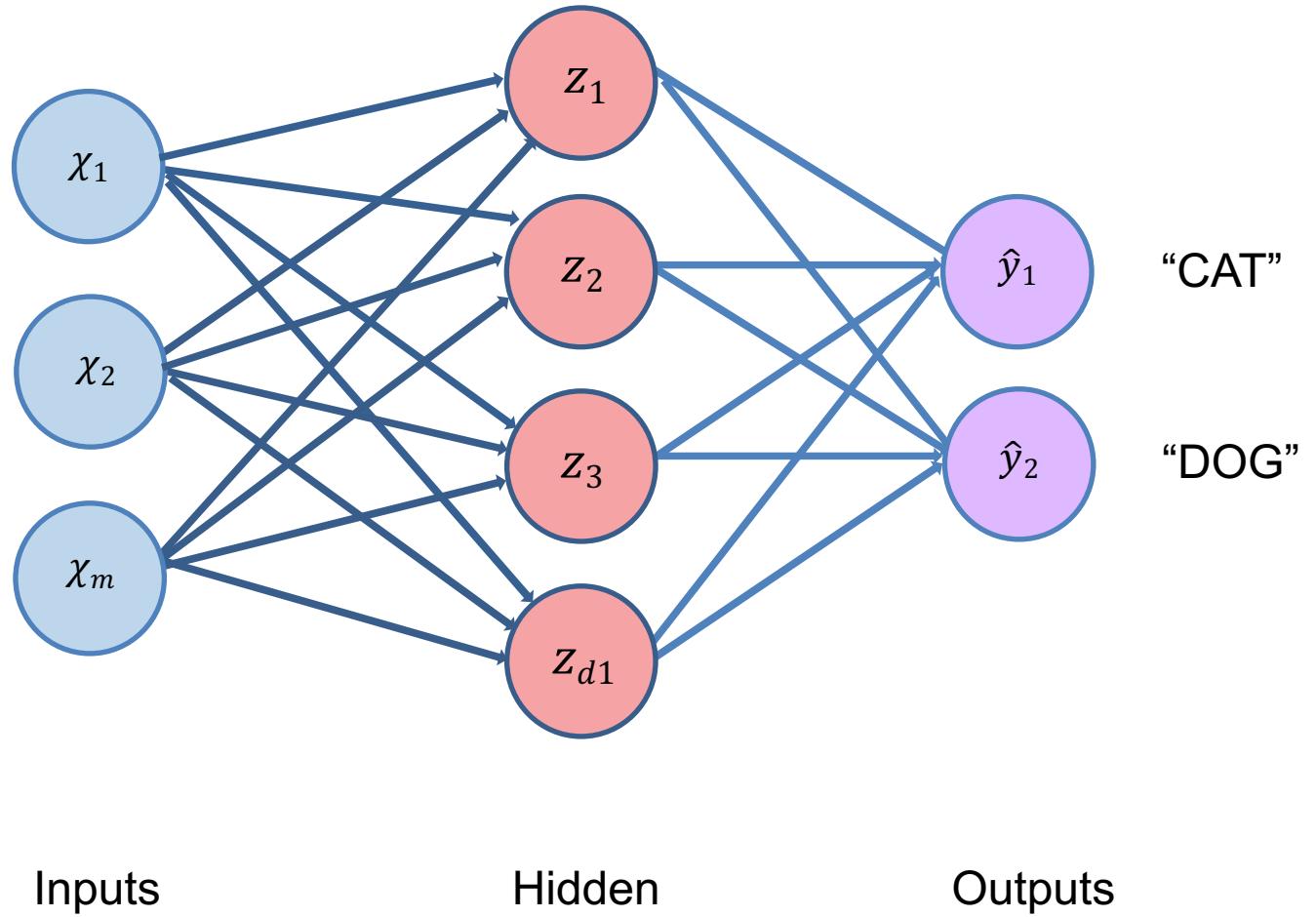
Minibatch composition

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

Variants of Neural Networks



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



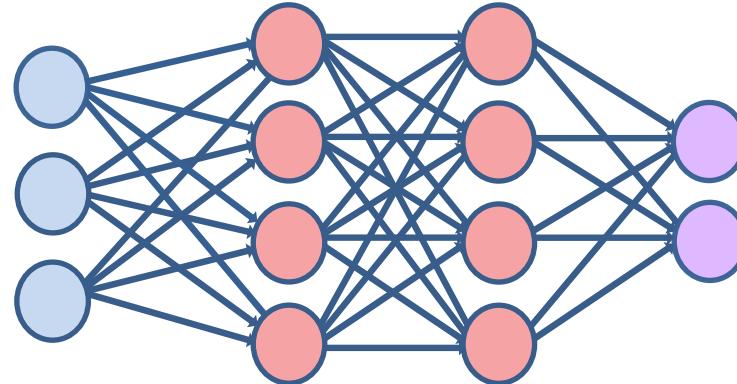
Types of Neural Networks Layer

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	✓	✓		✓

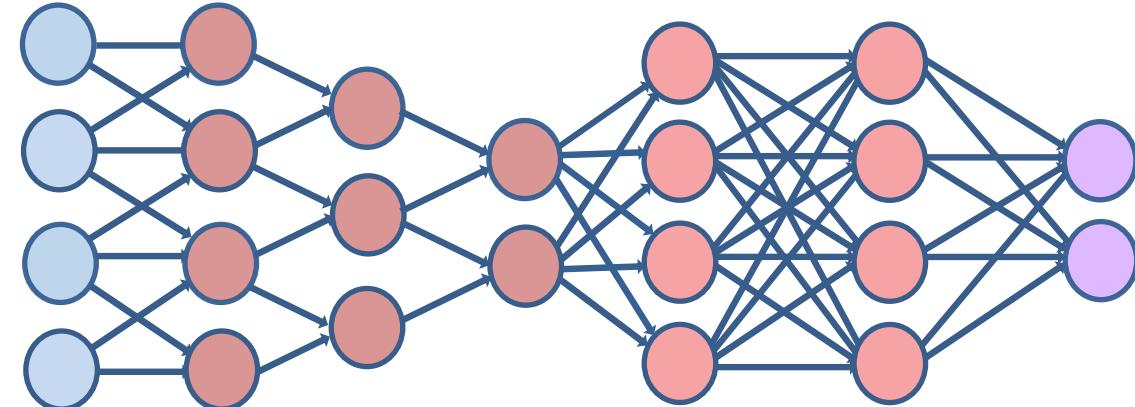
Variants of Neural Network - Examples

Blue circle = Input
Purple circle = Output

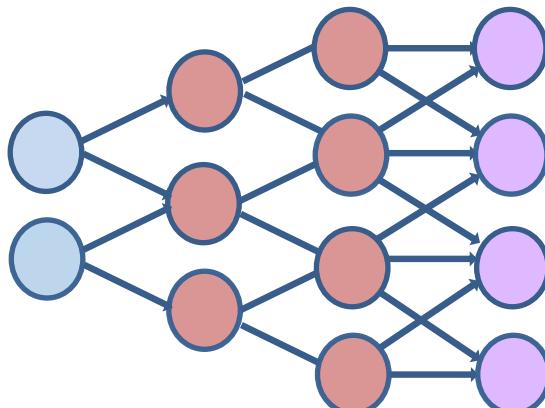
Deep Feed Forward (DFF) (fully connected)



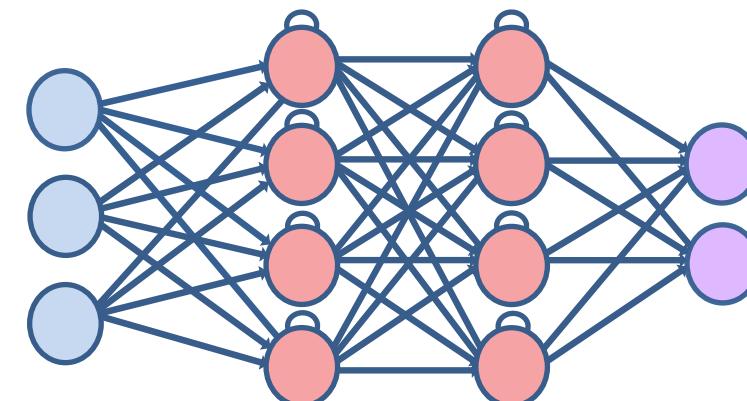
Deep convolution network (DCN)



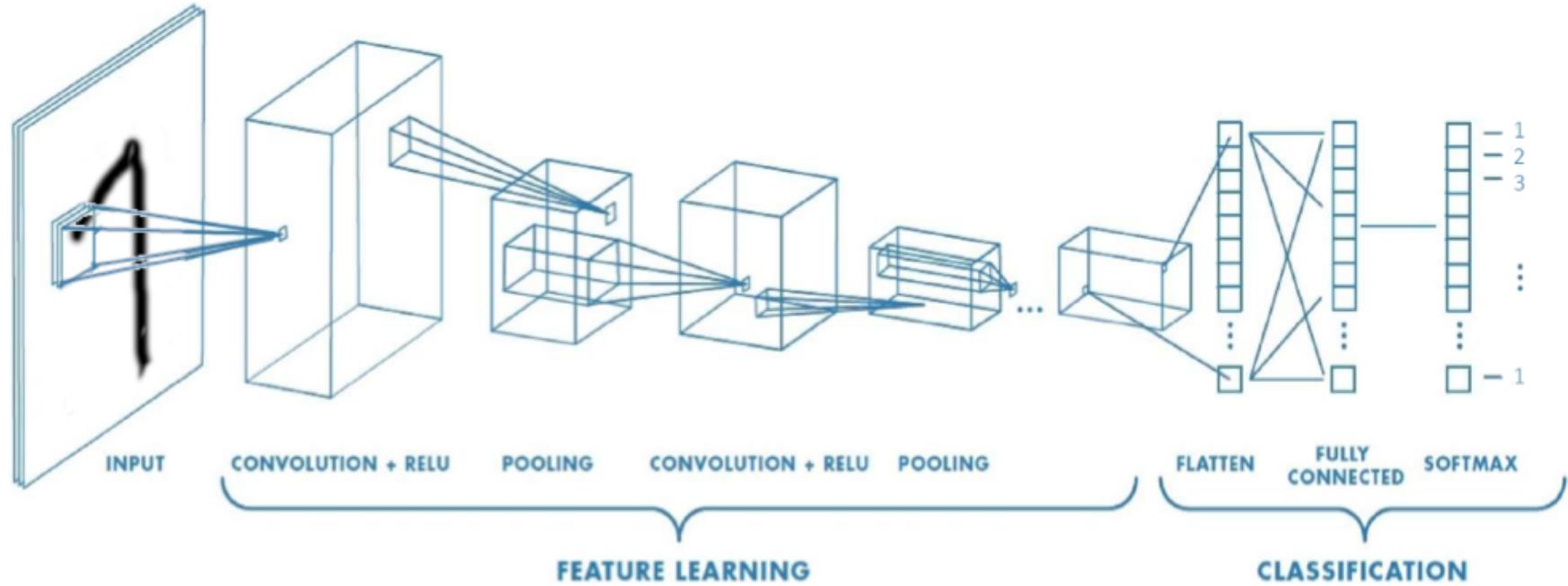
Deconvolution network (DN)



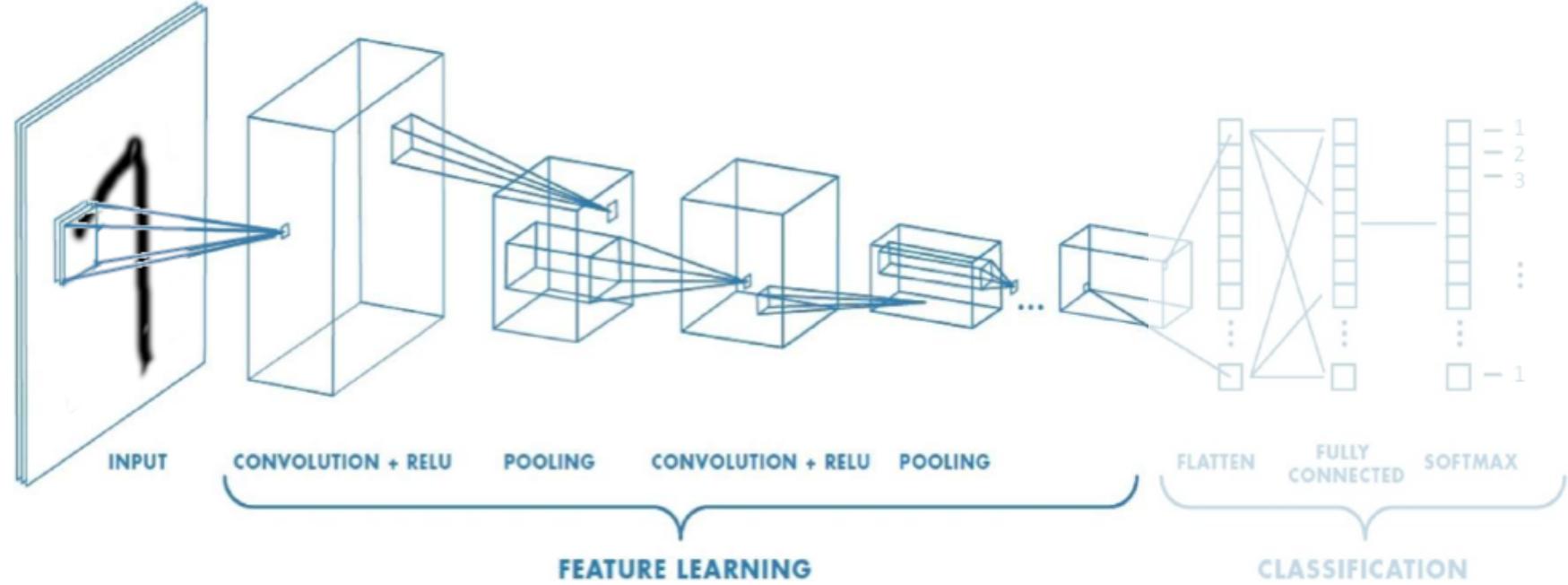
Recurrent neural network (RNN)



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

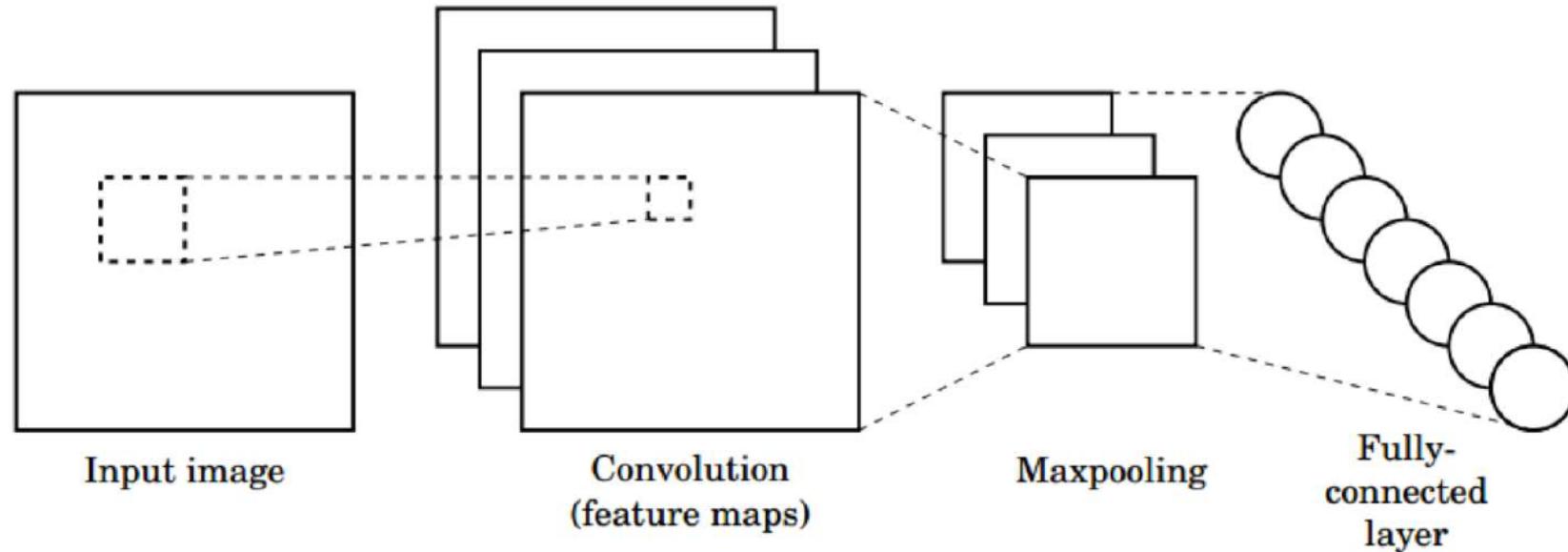


Edge Detect

“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.

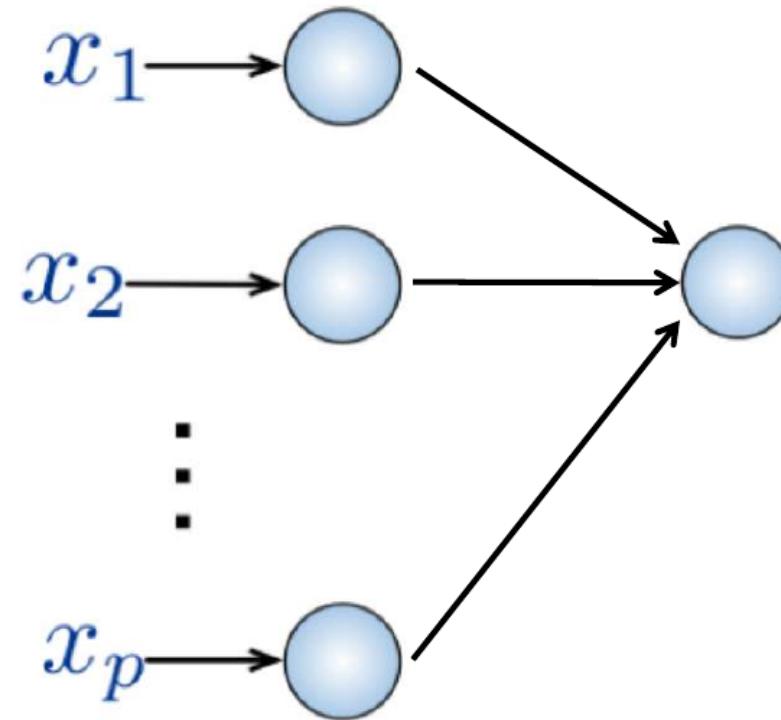
Learn weights of filters in convolutional layers.

Convolutional Neural Networks

Learning on image data

Input:

- 2D image
- Vector of pixel values



Fully Connected:

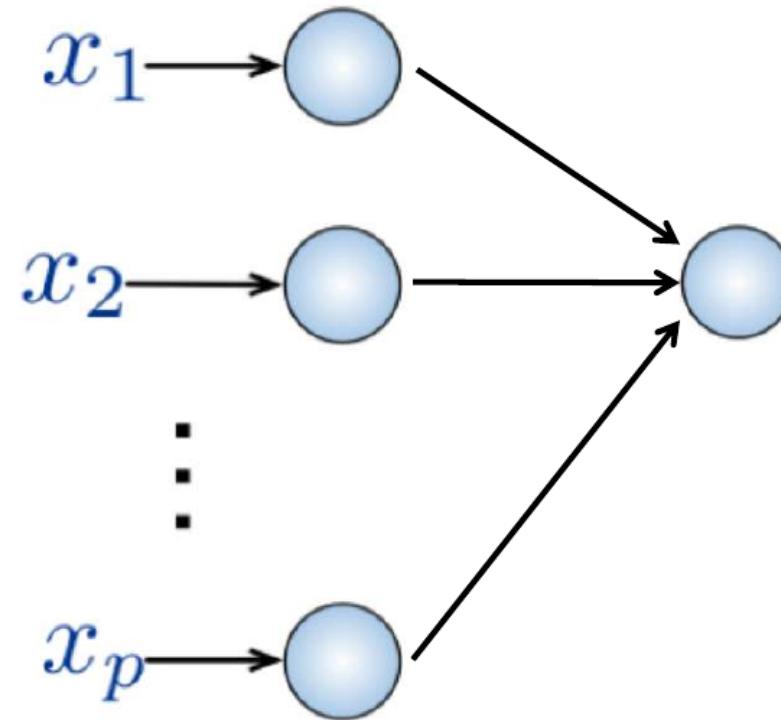
- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

Convolutional Neural Networks

Learning on image data

Input:

- 2D image
- Vector of pixel values



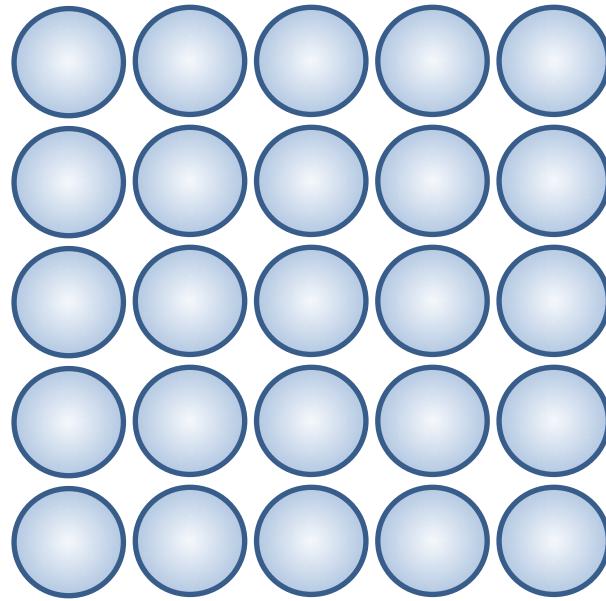
Fully Connected:

- 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

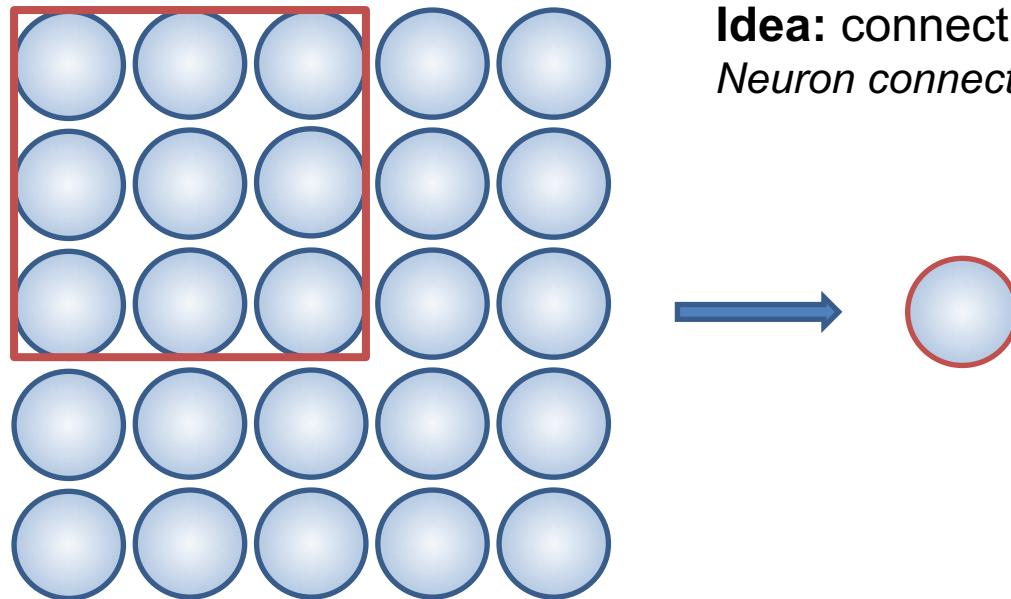


Input: 2D image (5x5 or 256x256 or ...)
Array of pixel values



Convolutional Neural Networks

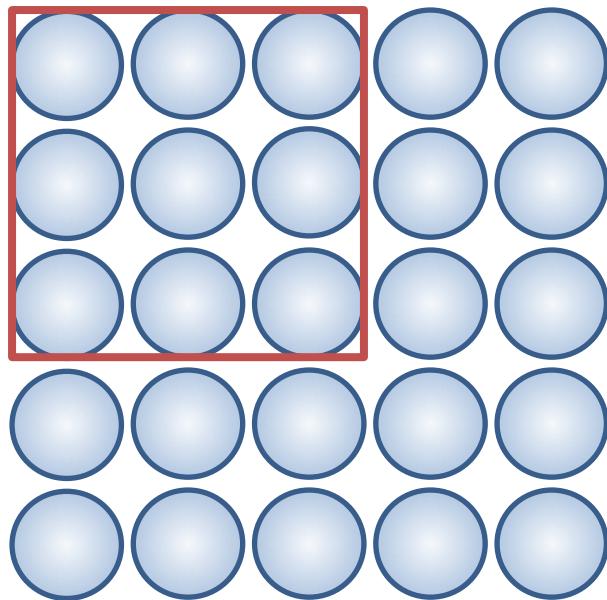
Learning on image data



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

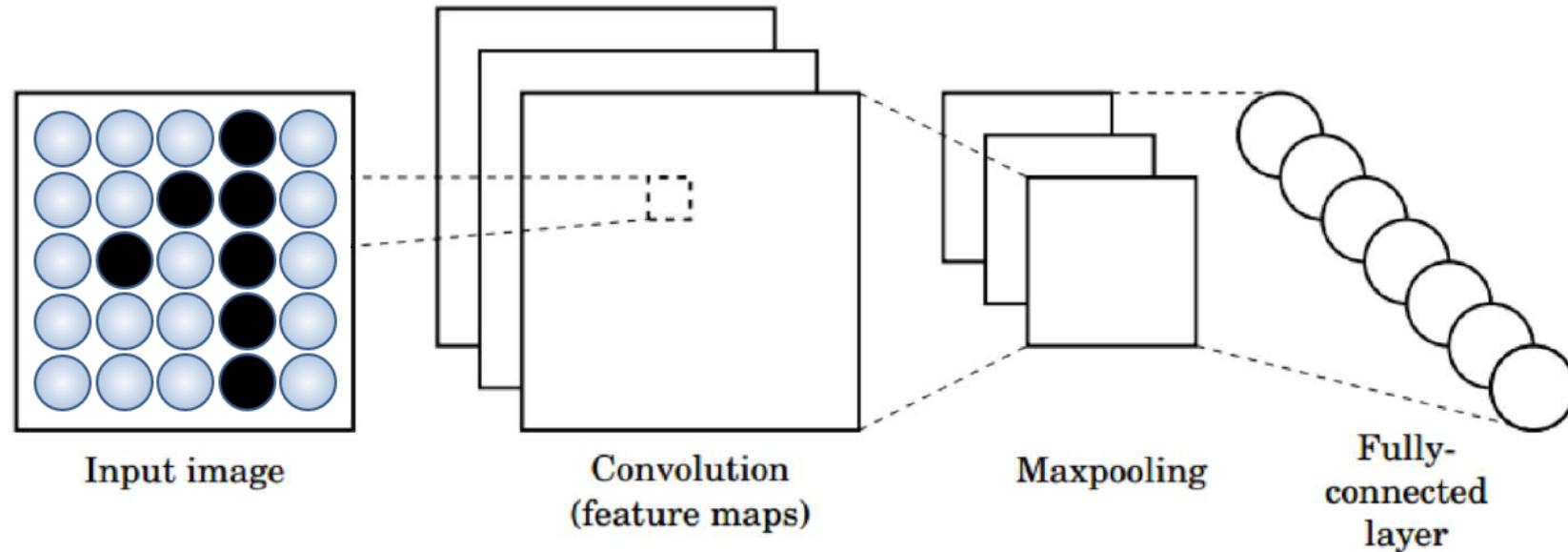
$$\sum_{i=1}^3 \sum_{j=1}^3 W_{ij} X_{i+p,j+q} + b$$

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



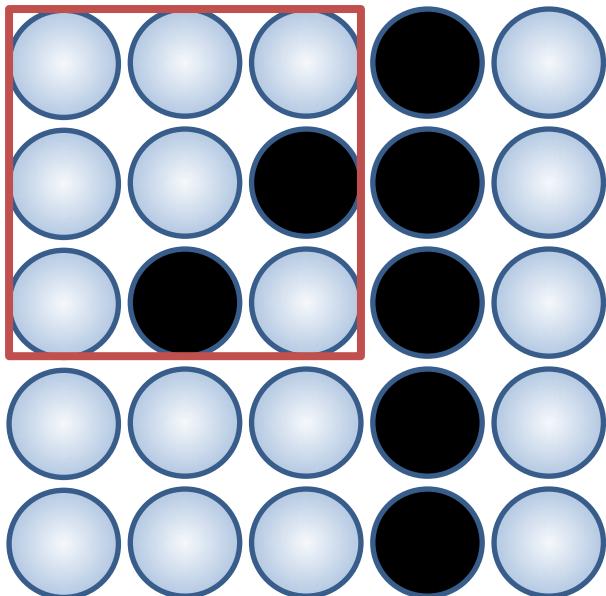
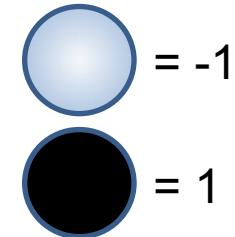
- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

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}

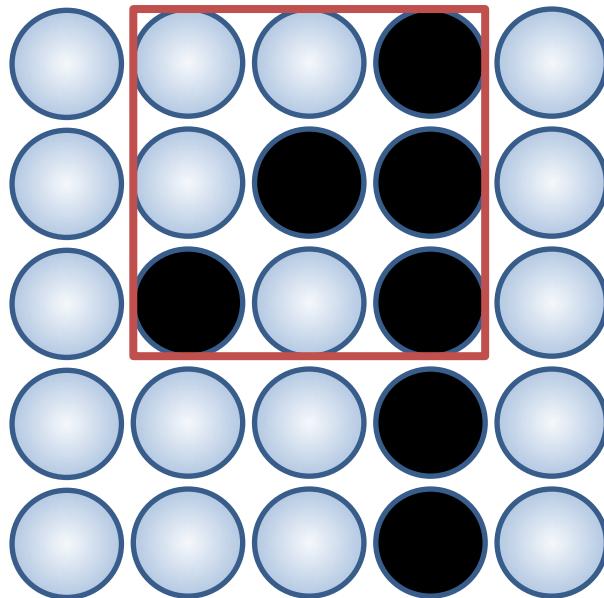
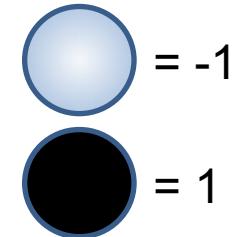


Multiply same indices of clipping and filter and sum it up.

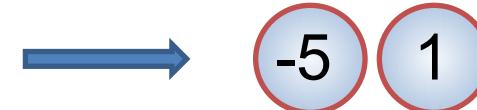
The diagram shows the convolution operation. On the left is a 3x3 input patch with values [-1, -1, -1; -1, -1, 1; -1, 1, -1]. On the right is a 3x3 filter with values [1, 1, 1; 1, 1, 1; 1, 1, 1]. Curved arrows point from the corresponding elements of the input patch and filter to the multiplication symbol between them. To the right of the multiplication is the equation $= (-1)+(-1)+(-1)+(-1)+(-1)+1+(-1)+1+(-1) = -5$.

Convolutional Neural Networks

Learning on image data - Example



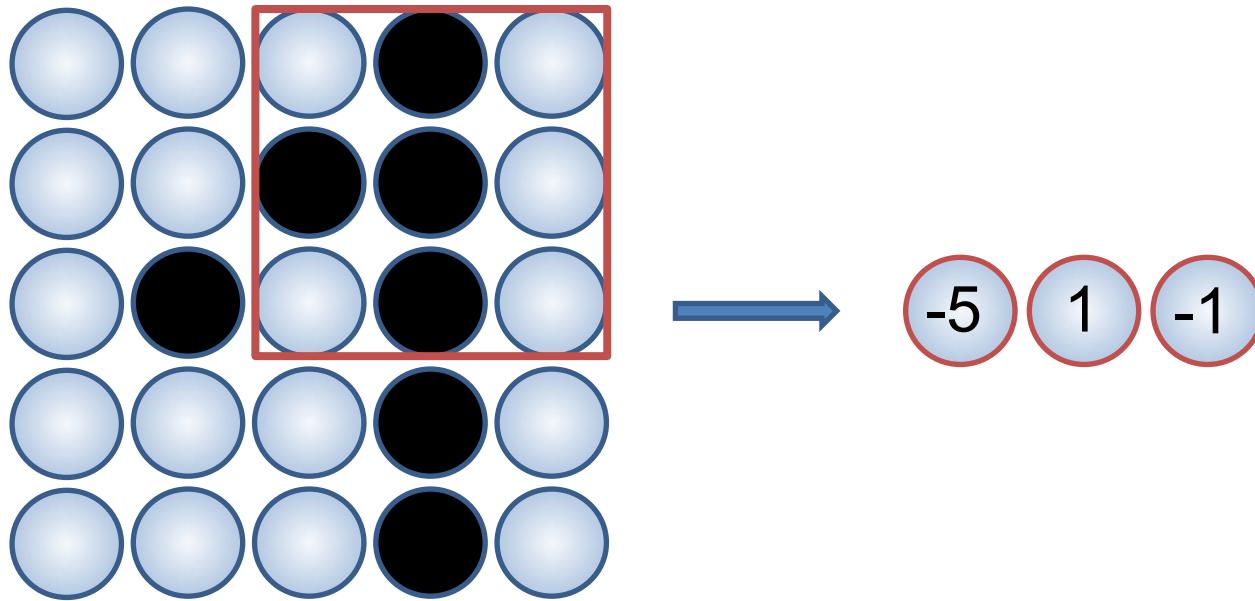
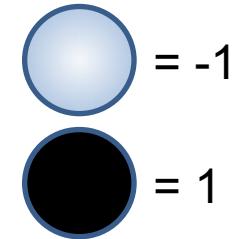
- **Step size here is one and the clipping is 3x3**



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

Convolutional Neural Networks

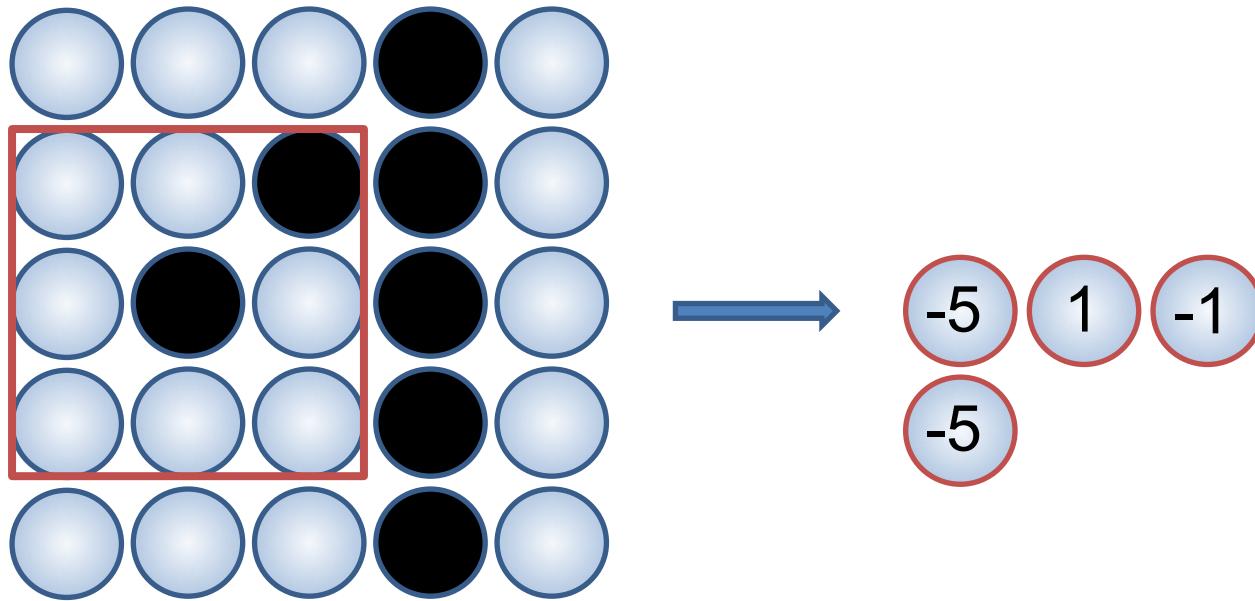
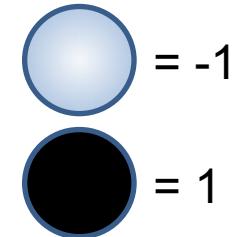
Learning on image data - Example



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

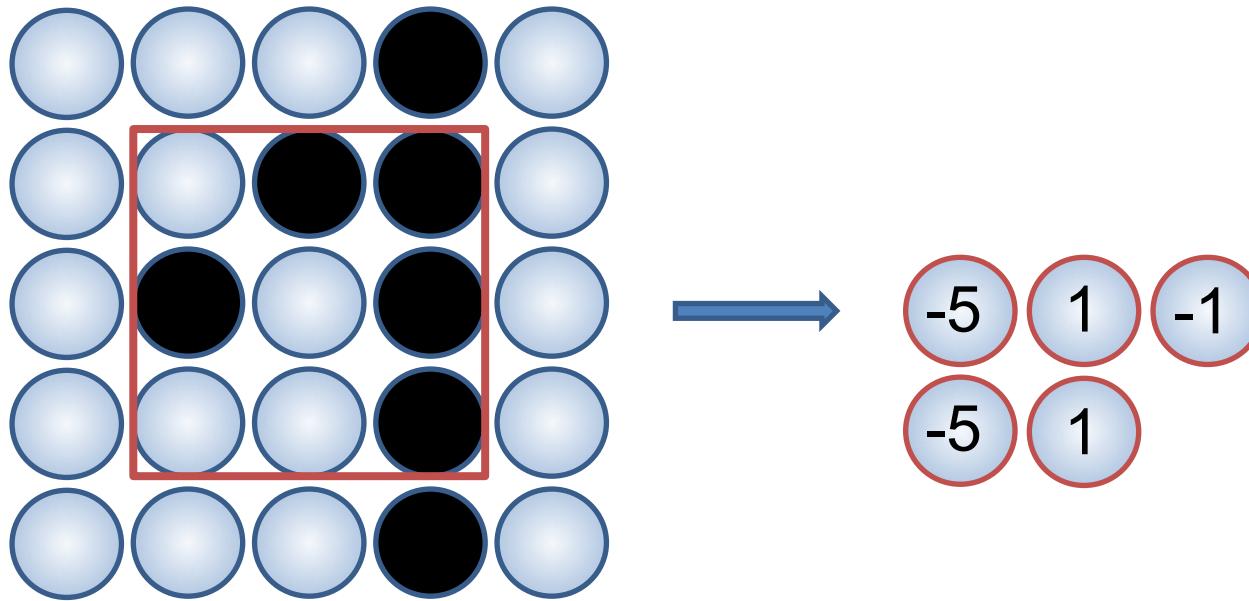
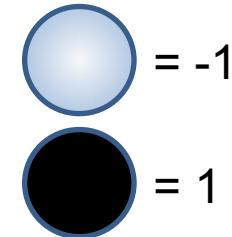
Convolutional Neural Networks

Learning on image data - Example



Convolutional Neural Networks

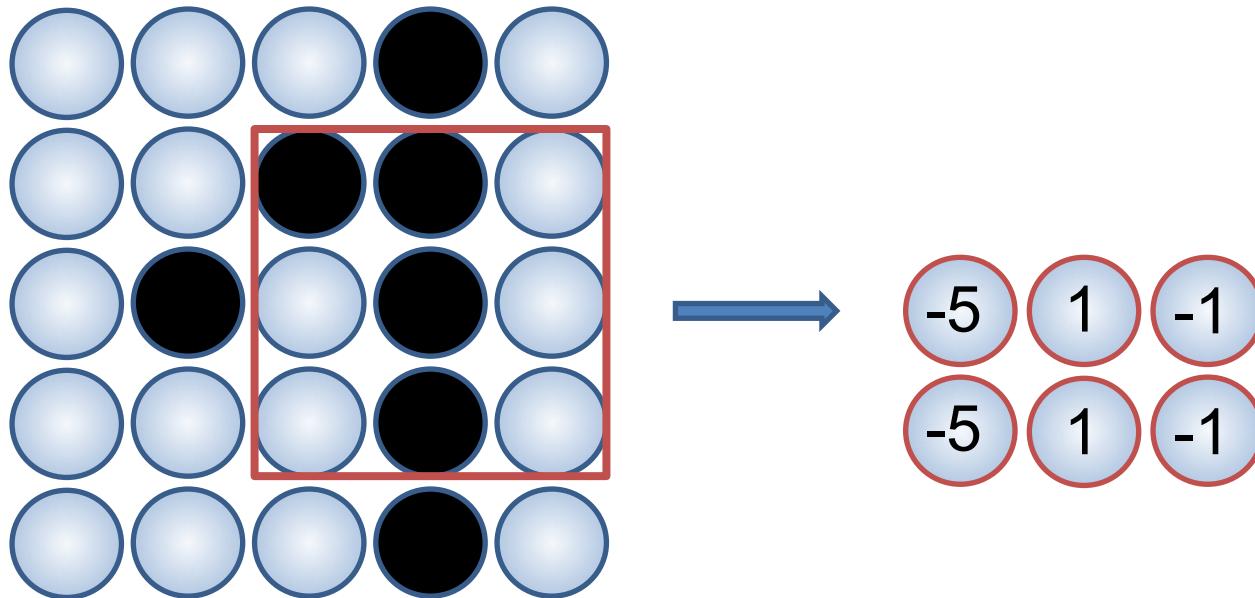
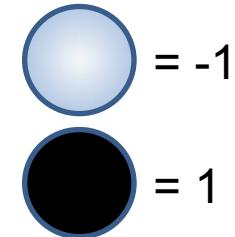
Learning on image data - Example



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

Convolutional Neural Networks

Learning on image data - Example

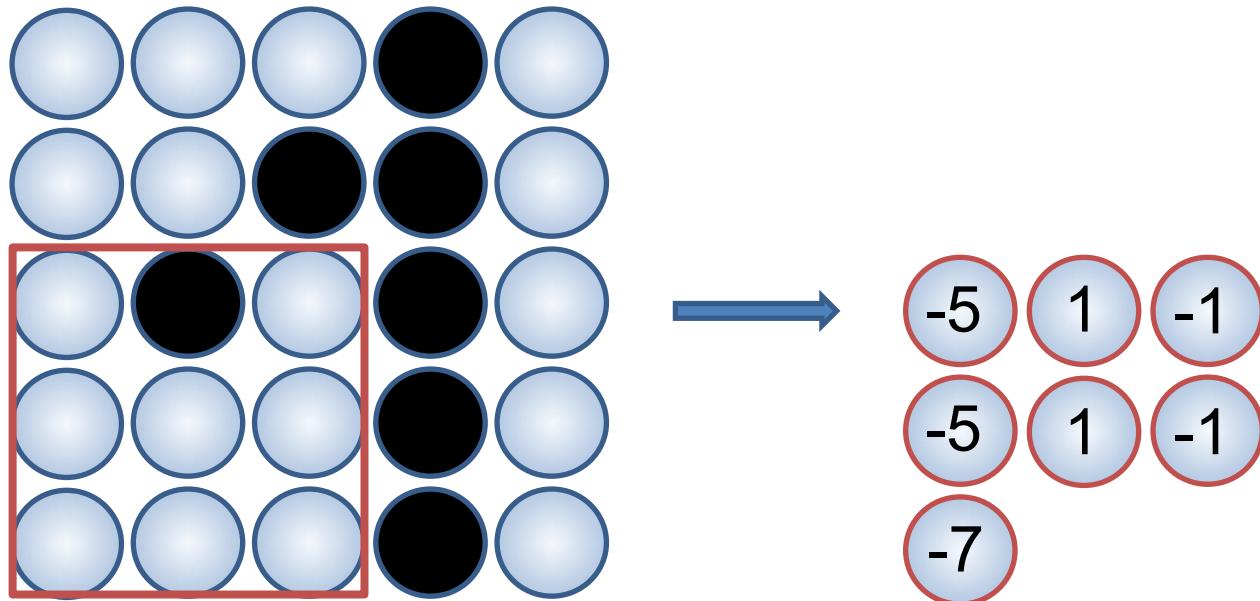


$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = -1$$

Convolutional Neural Networks

Learning on image data - Example

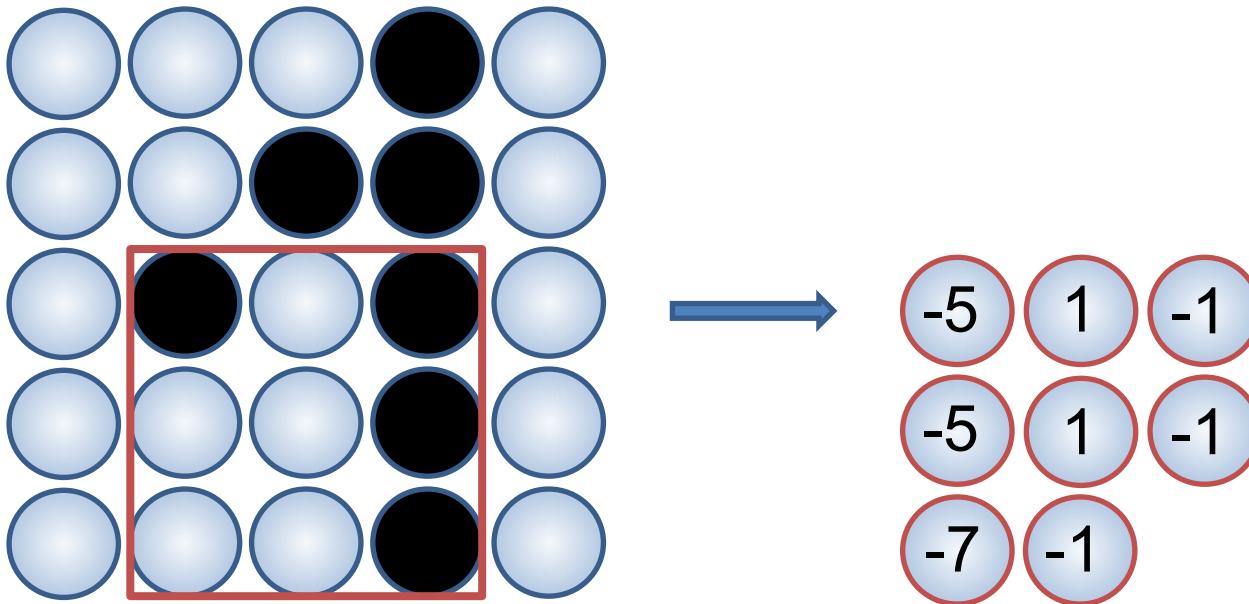
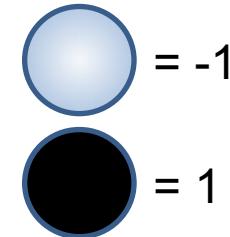
$$\begin{array}{c} \text{Light blue circle} \\ \text{Black circle} \end{array} = \begin{array}{l} -1 \\ 1 \end{array}$$



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

Convolutional Neural Networks

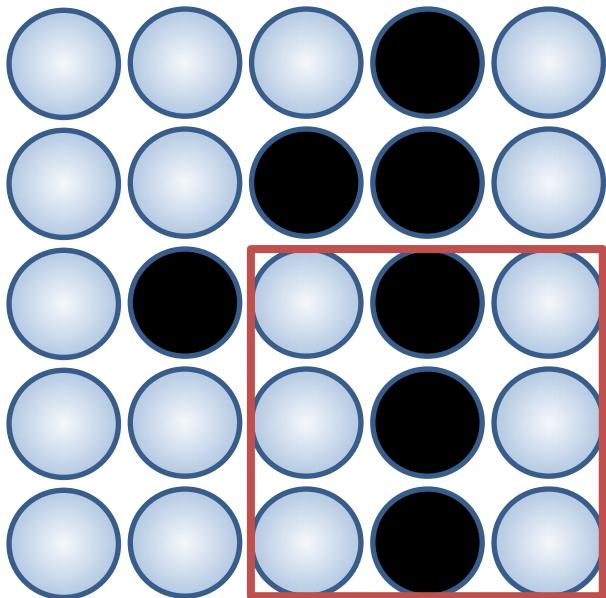
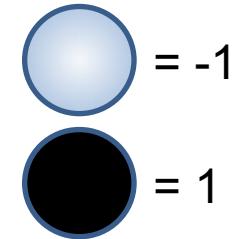
Learning on image data - Example



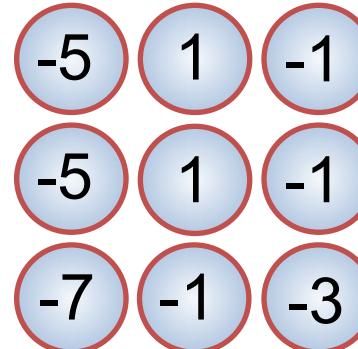
$$\dots \times \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = -1$$

Convolutional Neural Networks

Learning on image data - Example



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

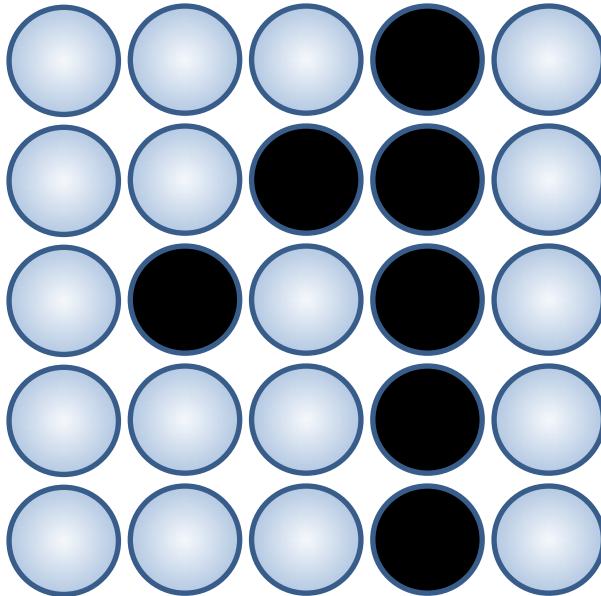
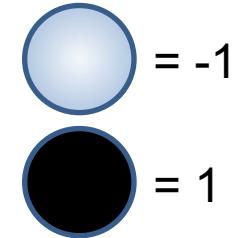


- Filter of size 3x3 → 9 separate weights
- Scan step wise the image with the filter
- That's **Convolution**

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

Convolutional Neural Networks

Learning on image data - Feature extraction



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

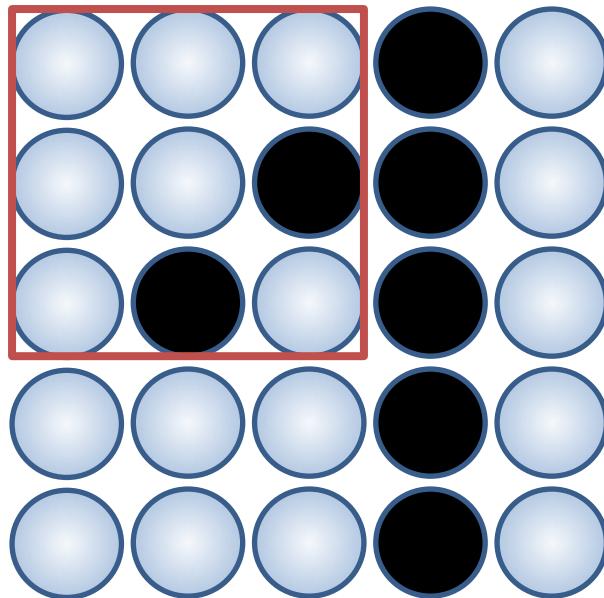
Next: extracting feature with filter $f_{i_1} =$



Convolutional Neural Networks

Learning on image data - Feature extraction

A legend showing two circles. The top circle is light blue and labeled = -1. The bottom circle is black and labeled = 1.



Low score !
barely matches



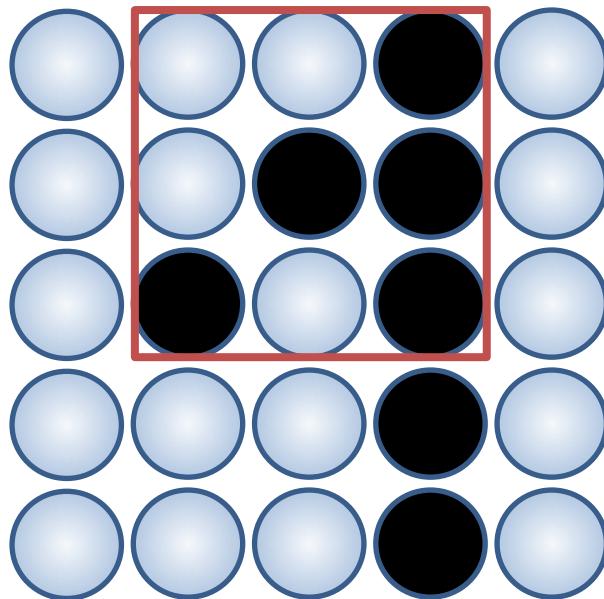
-1

A diagram illustrating a convolution step. It shows a 3x3 input kernel with values [-1, -1, -1; -1, 1, -1; -1, 1, -1] highlighted by a red border, multiplied by a 3x3 weight kernel with values [-1, -1, 1; -1, 1, 1; 1, 1, 1] highlighted by colored borders (blue for -1, yellow for 1). The result is a single value: $= 1+1+(-1)+1+(-1)+1+(-1)+(-1) = -1$.

Convolutional Neural Networks

Learning on image data - Feature extraction

A legend showing two circles. The top circle is light blue and labeled = -1. The bottom circle is black and labeled = 1.



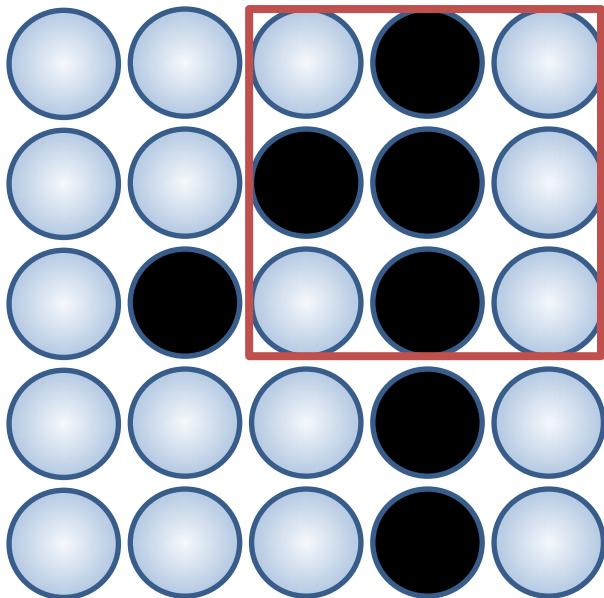
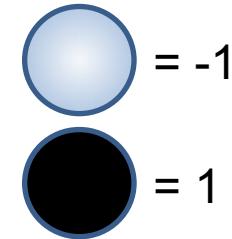
Maximum score !
Filter matches full

A blue arrow points from the input grid to a result row. The first circle contains the value -1 and the second circle contains the value 9. An arrow points down to this row from the text "Maximum score ! Filter matches full".

A diagram illustrating the convolution operation. It shows two 3x3 matrices being multiplied. The left matrix is a weight filter with values: -1, -1, 1; -1, 1, 1; 1, -1, 1. The right matrix is the receptive field of the highlighted unit, with values: -1, -1, 1; -1, 1, 1; 1, -1, 1. The result of the multiplication is shown as a sum: $= 1+1+1+1+1+1+1+1 = 9$.

Convolutional Neural Networks

Learning on image data - Feature extraction



Very low score !
Many mismatches

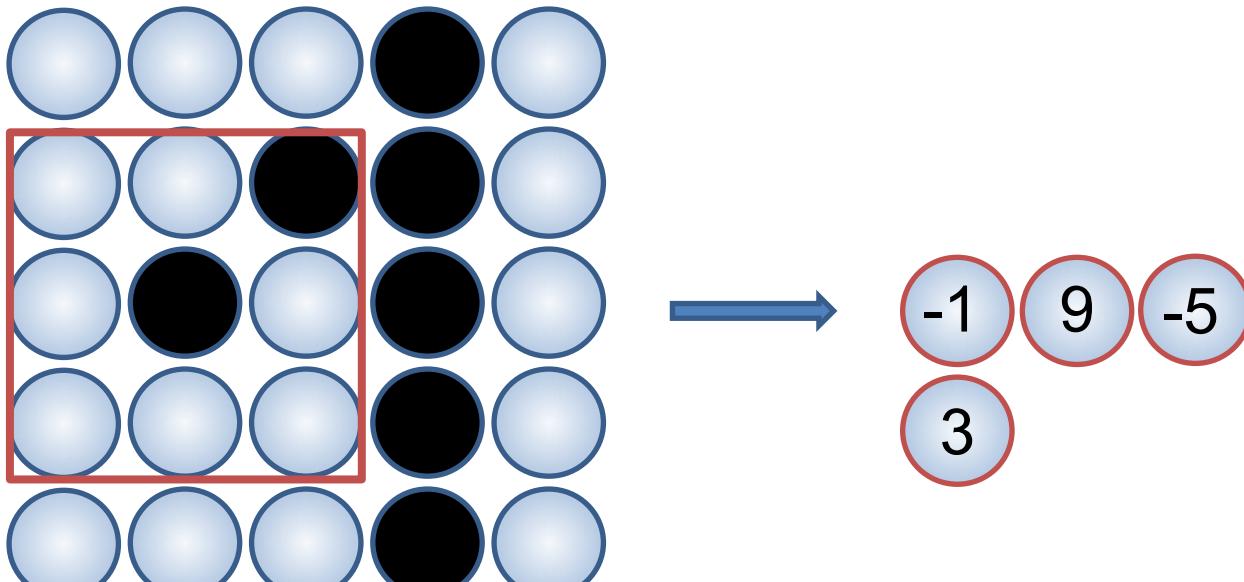
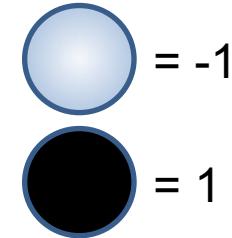
A horizontal row of three circles with red outlines. The first circle contains '-1', the second contains '9', and the third contains '-5'. An arrow points from the input image to this row. A downward arrow points from the text 'Very low score ! Many mismatches' to the circle containing '-5'.

A diagram illustrating a convolution operation. On the left, a 3x3 input kernel with red border and black centers is multiplied by a 3x3 weight kernel with yellow border and green centers. The result is a single value: $= 1+(-1)+(-1)+(-1)+1+(-1)+(-1)+(-1) = -5$.

$$\begin{matrix} -1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \end{matrix} \times \begin{matrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{matrix} = 1+(-1)+(-1)+(-1)+1+(-1)+(-1)+(-1) = -5$$

Convolutional Neural Networks

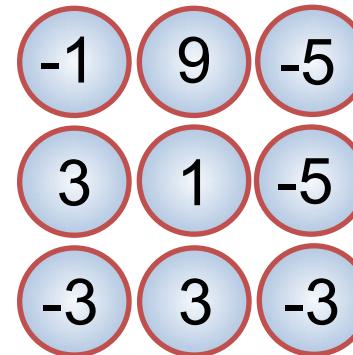
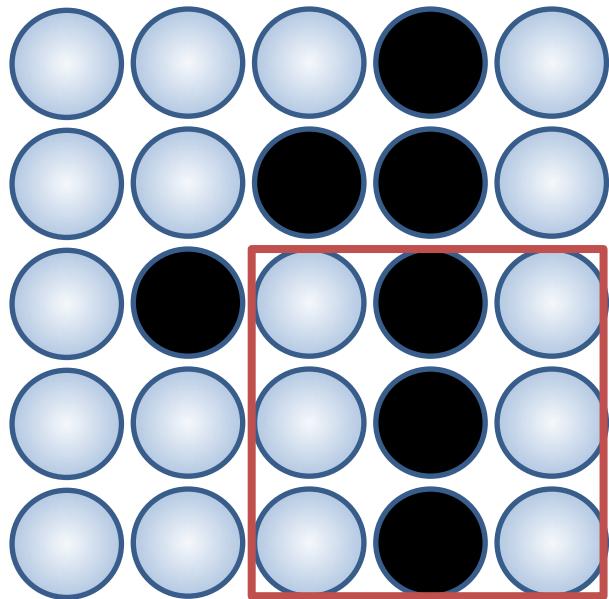
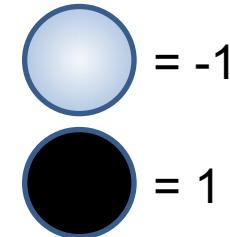
Learning on image data - Feature extraction



A diagram illustrating a convolution operation. It shows a red-bordered input patch with three dots (...), a green-bordered kernel with values -1, -1, 1, -1, 1, 1, 1, -1, 1, and the result of the multiplication, which is 3.

Convolutional Neural Networks

Learning on image data - Feature extraction

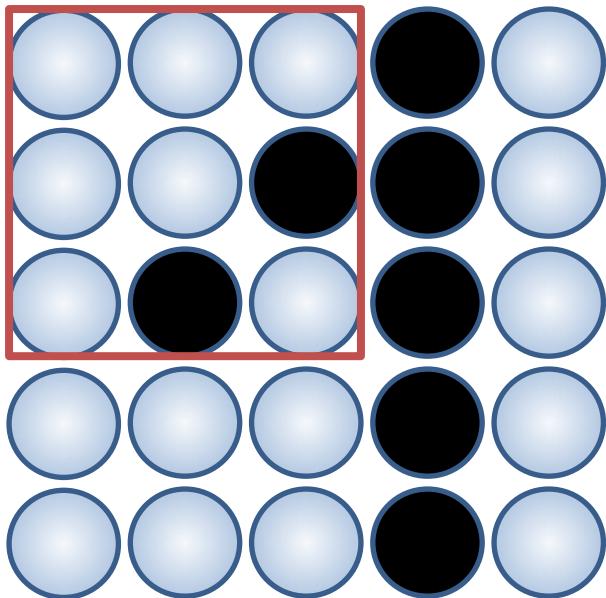


A diagram illustrating a convolution operation. On the left, a red box contains three dots (...), indicating multiple input channels. To its right is a multiplication sign (x). Next is a 3x3 kernel represented by a grid of colored circles: the top row is yellow (-1, -1, 1), the middle row is green (1, 1, 1), and the bottom row is green (1, -1, 1). An equals sign (=) followed by the number -3 indicates the result of the convolution step.

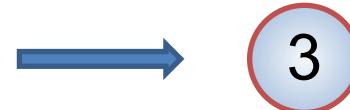
Convolutional Neural Networks

Learning on image data - Feature extraction

$$\begin{array}{l} \text{Light blue circle} = -1 \\ \text{Black circle} = 1 \end{array}$$



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



Next filter $\mathbf{f}_2 =$

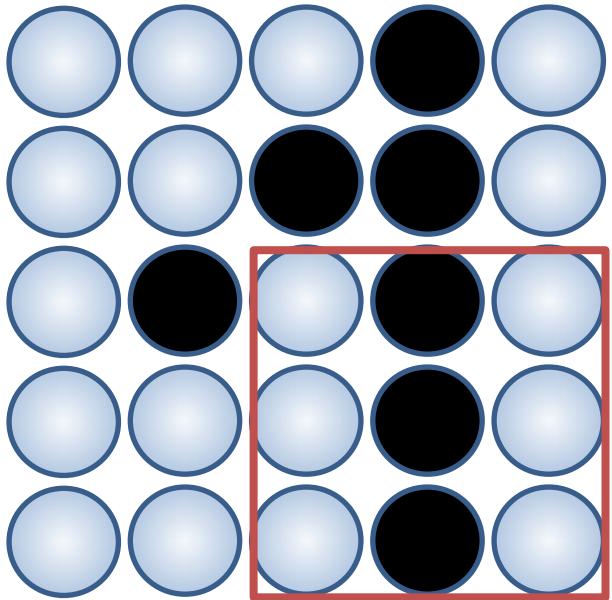
$$\begin{matrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{matrix}$$

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

Convolutional Neural Networks

Learning on image data - Feature extraction

$$\begin{array}{c} \text{Light blue circle} \\ \text{Dark blue circle} \end{array} = \begin{array}{l} -1 \\ 1 \end{array}$$



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

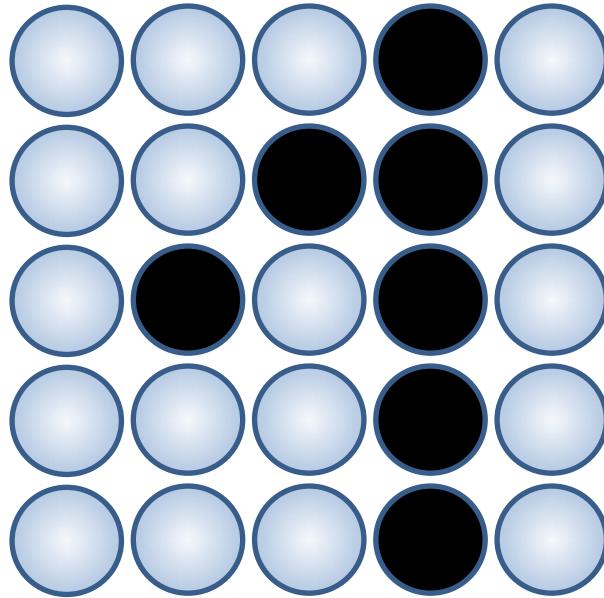
Next filter $\mathbf{f}_2 =$

$$\begin{array}{ccc} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{array}$$

$$\begin{matrix} & \begin{matrix} 3 & -3 & 7 \\ 3 & -3 & 7 \\ 5 & -5 & 9 \end{matrix} \\ \begin{matrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{matrix} & \times & \begin{matrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{matrix} & = 9 \end{matrix}$$

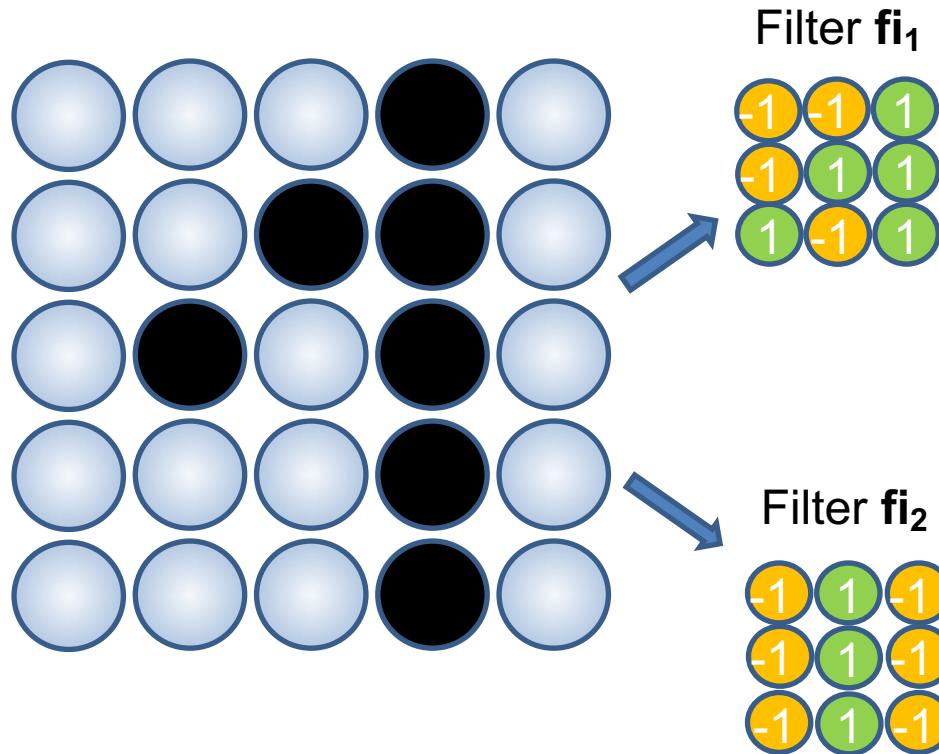
Convolutional Neural Networks

Learning on image data - Feature extraction – filter comparison



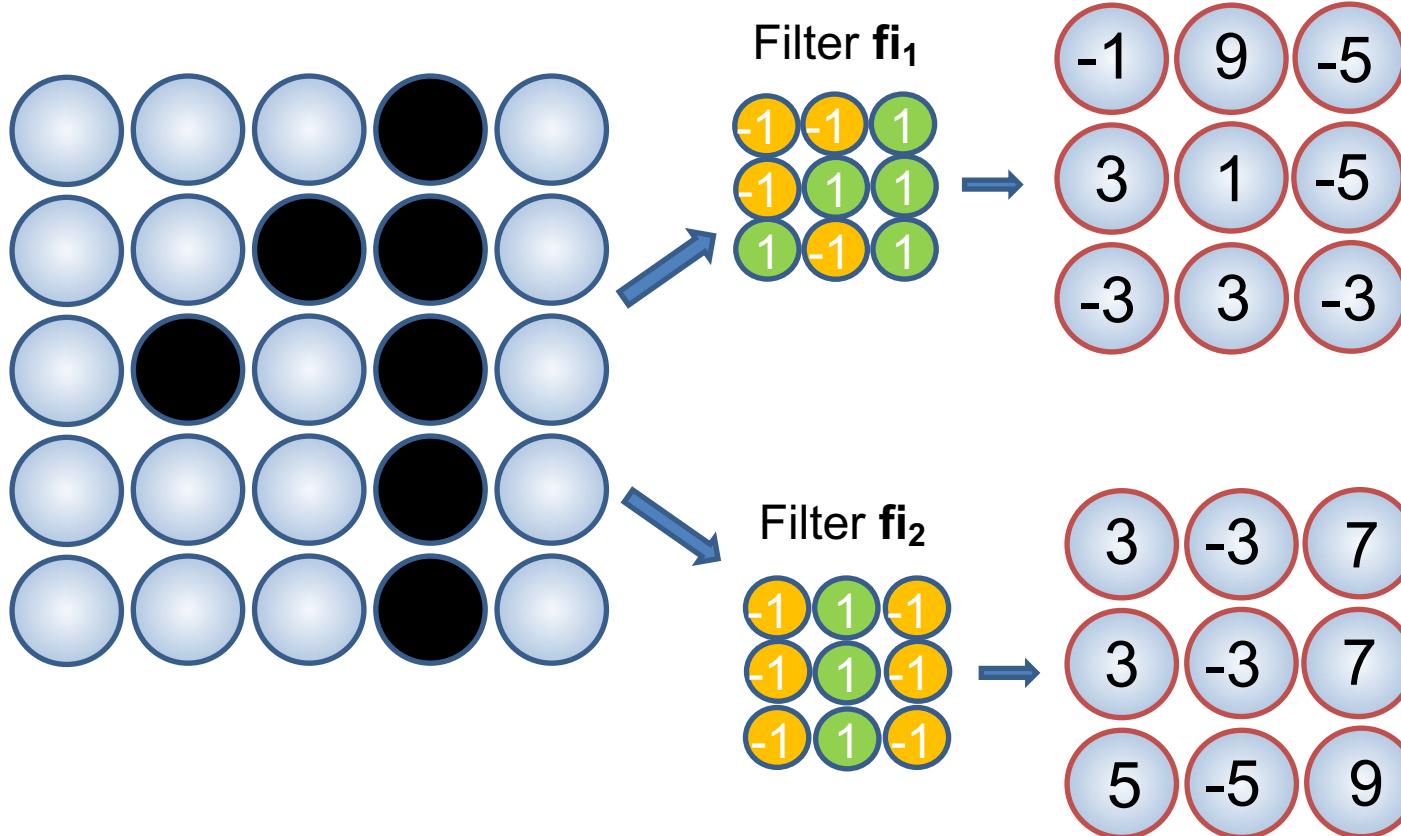
Convolutional Neural Networks

Learning on image data - Feature extraction – filter comparison



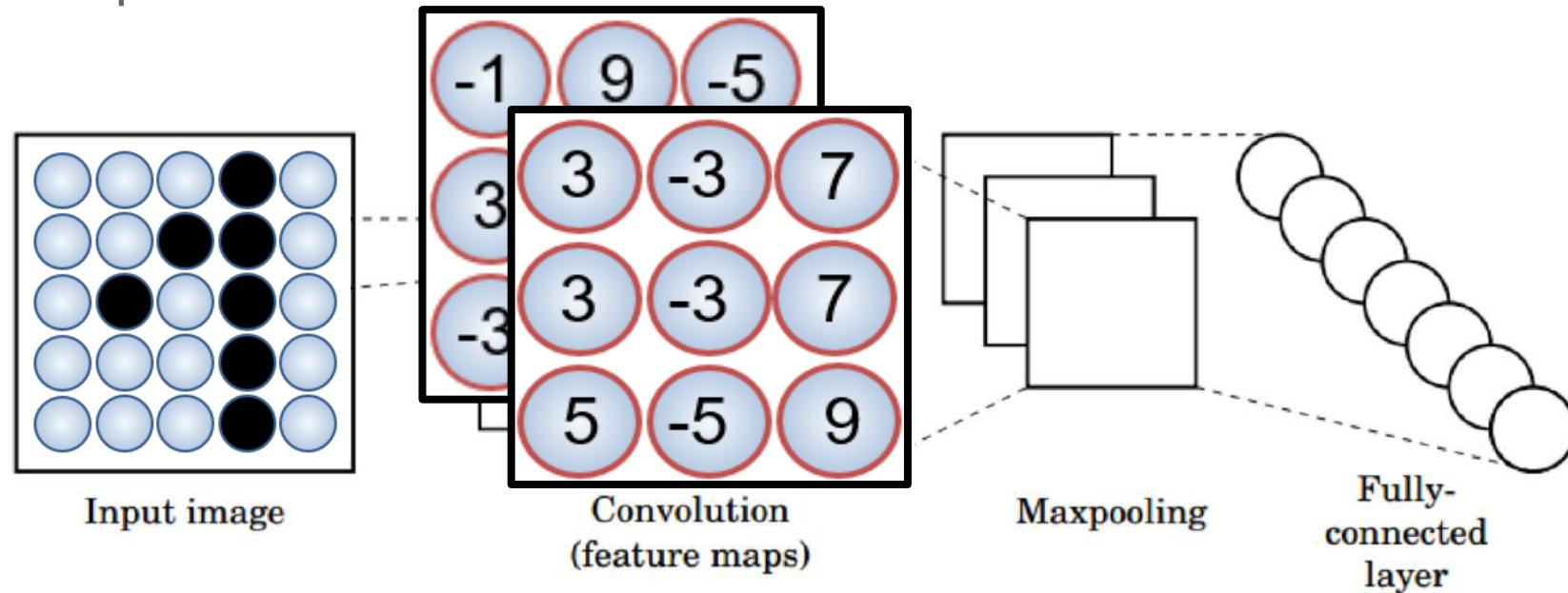
Convolutional Neural Networks

Learning on image data - Feature extraction – filter comparison



Convolutional Neural Networks

Building a CNN - Example



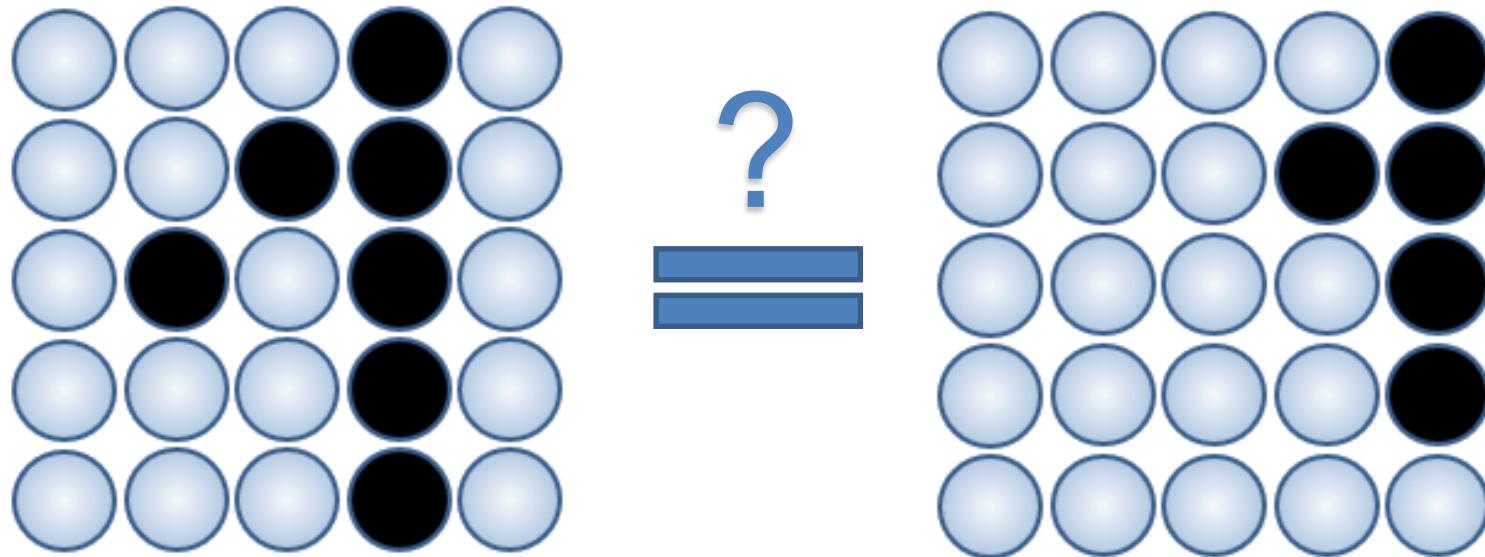
- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

Train model with image data.

Learn weights of filters in convolutional layers.

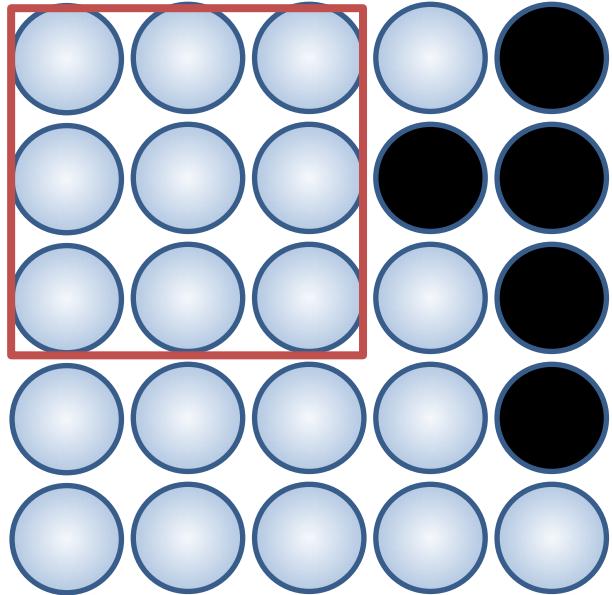
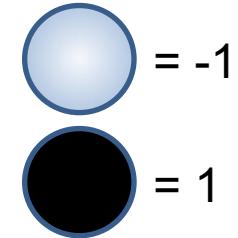
Convolutional Neural Networks

Filters to detect features



Convolutional Neural Networks

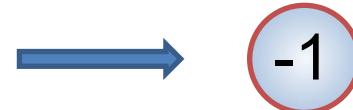
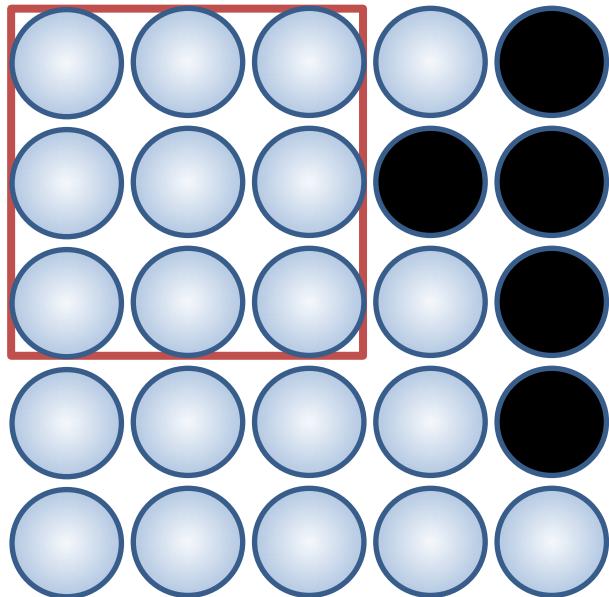
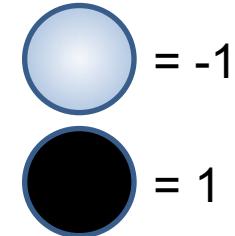
Filters to detect features



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**.

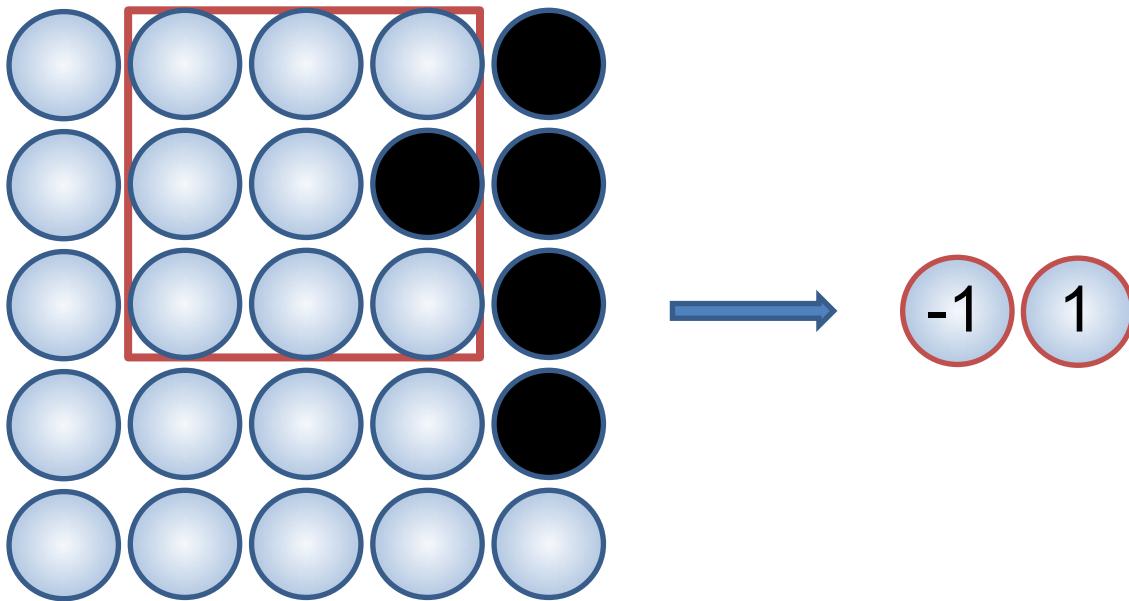
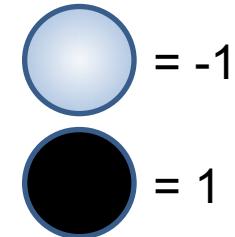
e.g. Filter $f_{i_1} =$

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

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

Convolutional Neural Networks

Filters to detect features



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

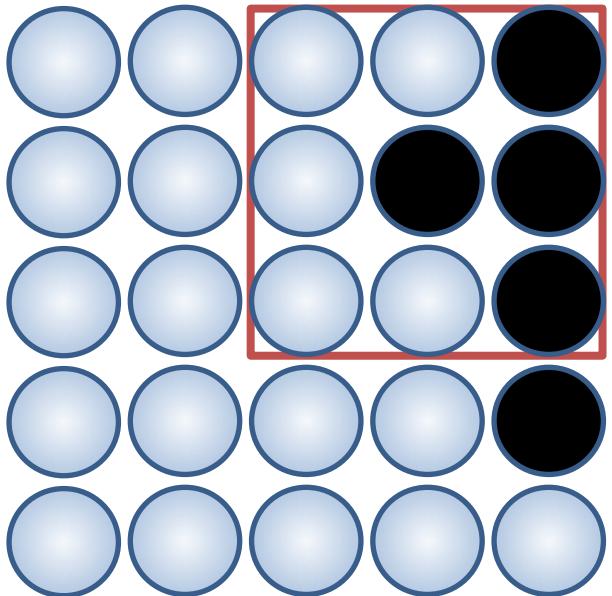
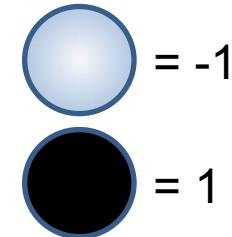


Feature/filter \mathbf{f}_1 still matches the deformed image well

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

Convolutional Neural Networks

Filters to detect features



Not a full match
but still a high score

→

-1 1 7

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

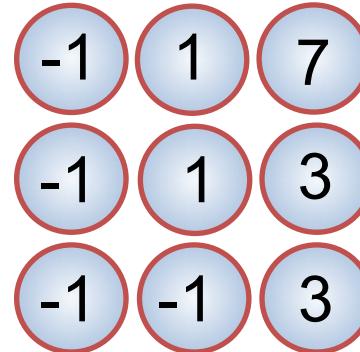
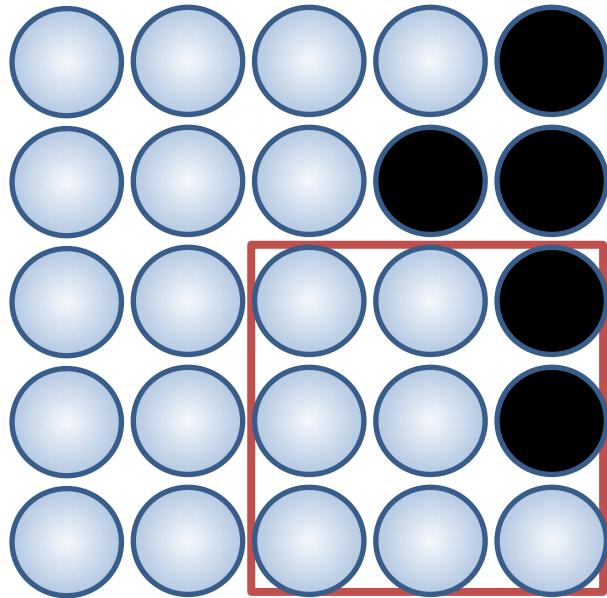
$$\begin{matrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{matrix}$$

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

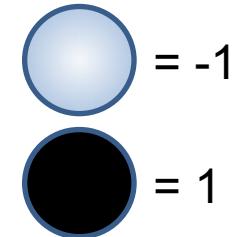
$$\begin{matrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \end{matrix} \times \begin{matrix} -1 & -1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{matrix} = 1+1+1+1+1+(-1)+1+1 = 7$$

Convolutional Neural Networks

Filters to detect features



Finished feature map
using filter \mathbf{f}_1

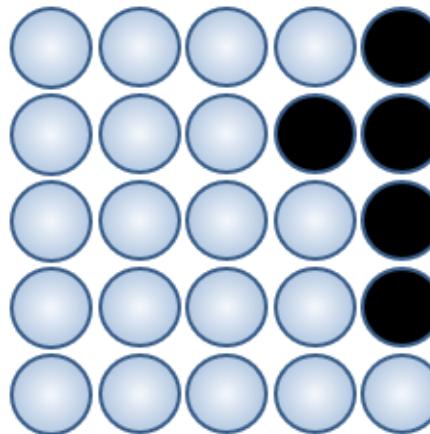
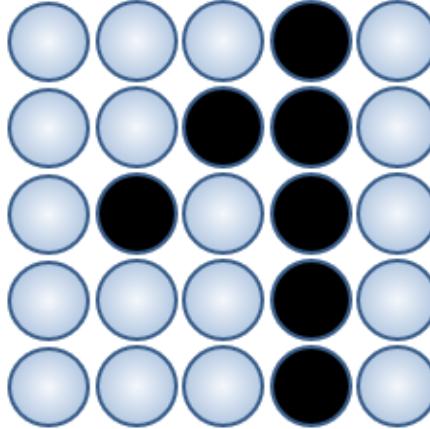


e.g. Filter $\mathbf{f}_1 =$



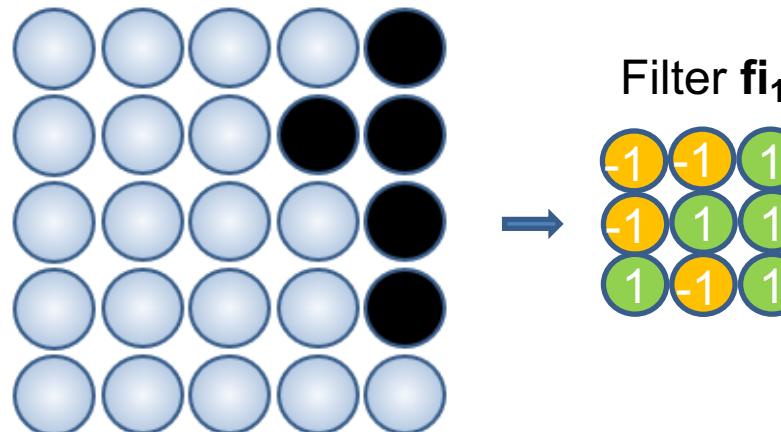
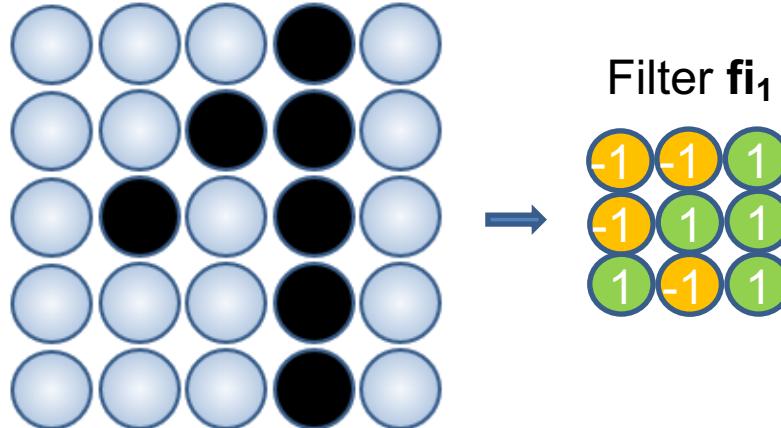
Convolutional Neural Networks

Filters to detect features – filter comparison on diff. images



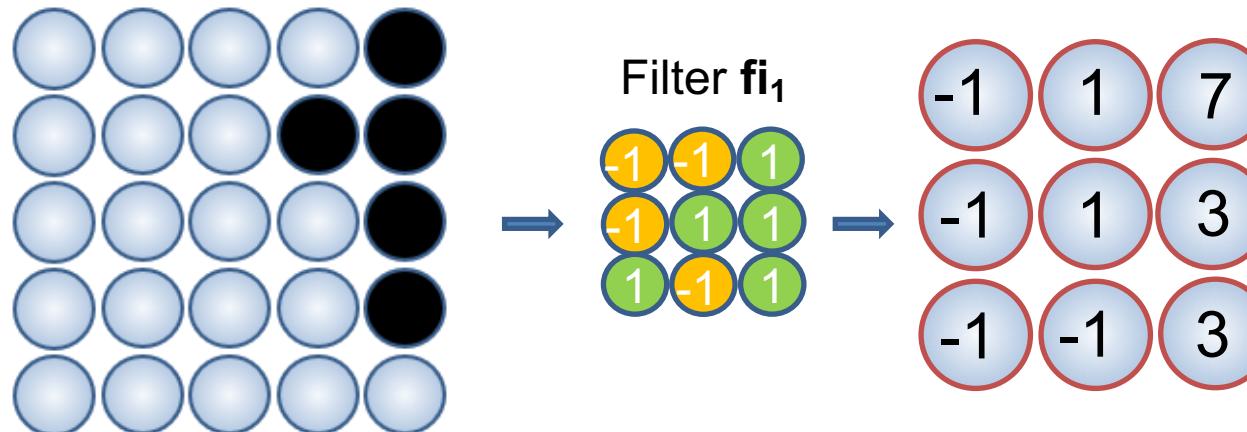
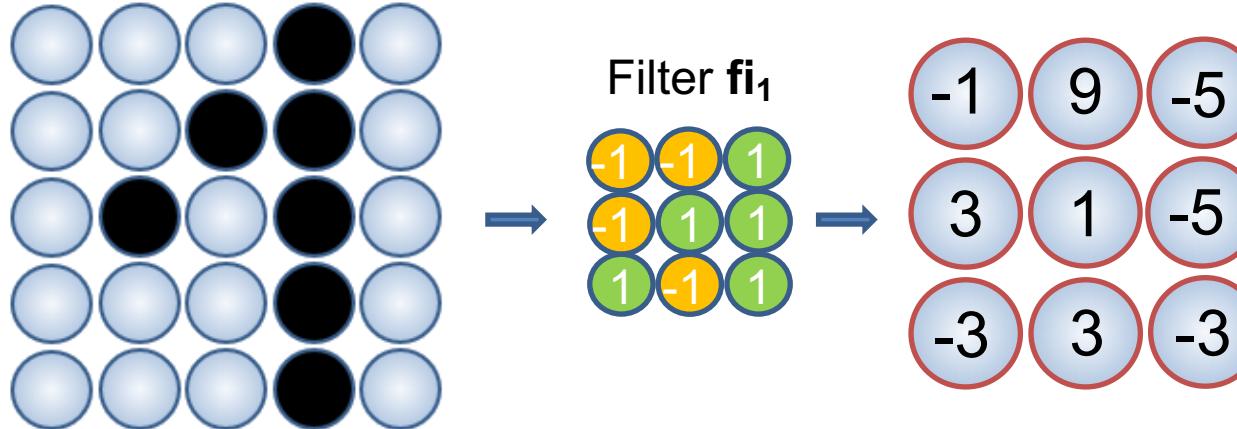
Convolutional Neural Networks

Filters to detect features – filter comparison on diff. images



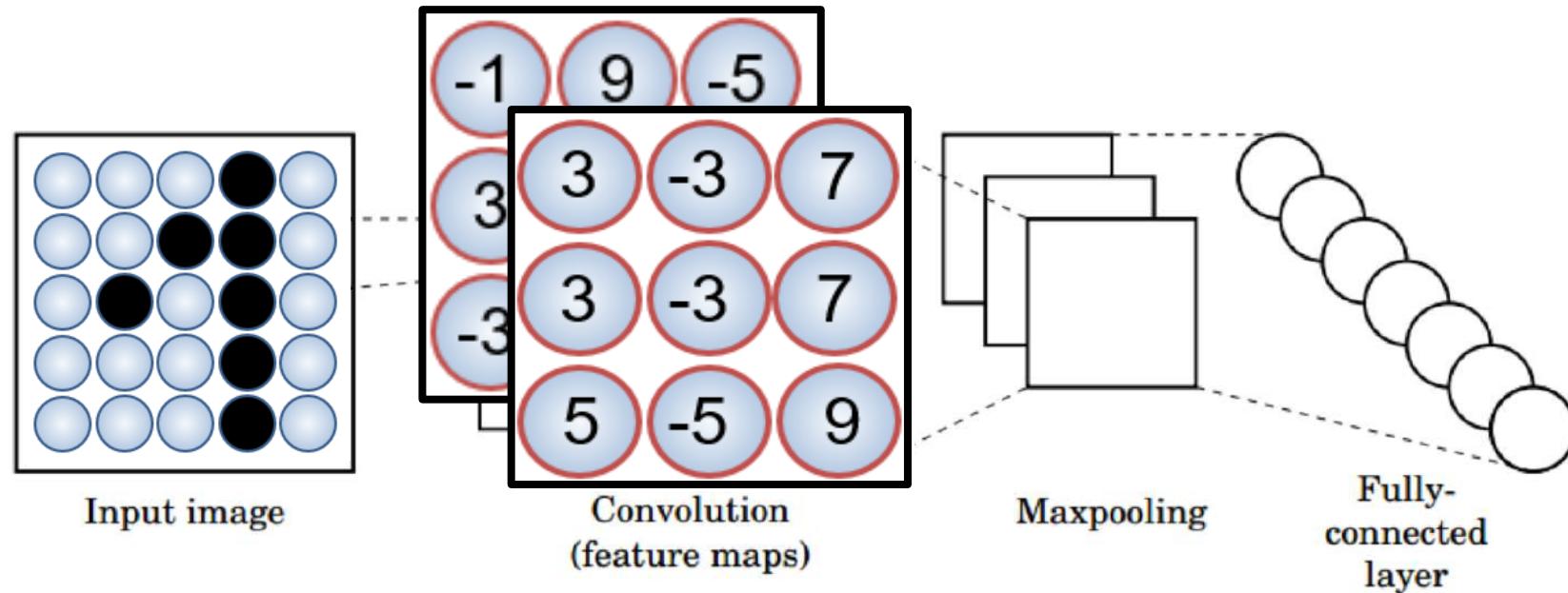
Convolutional Neural Networks

Filters to detect features – filter comparison on diff. images



Convolutional Neural Networks

Pooling



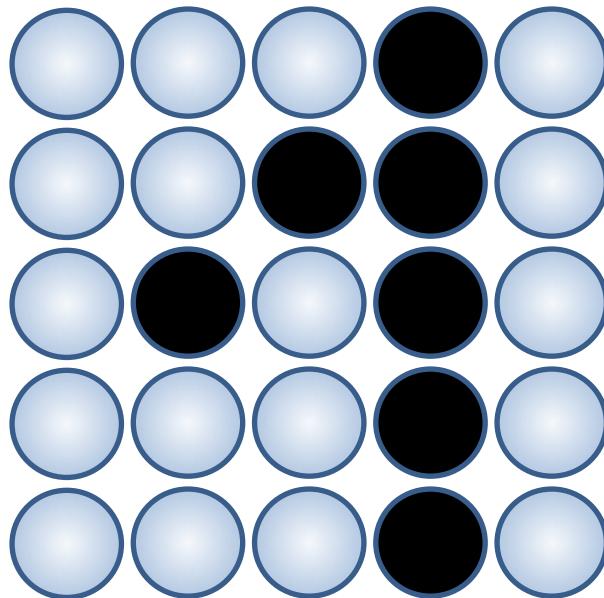
- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

Train model with image data.

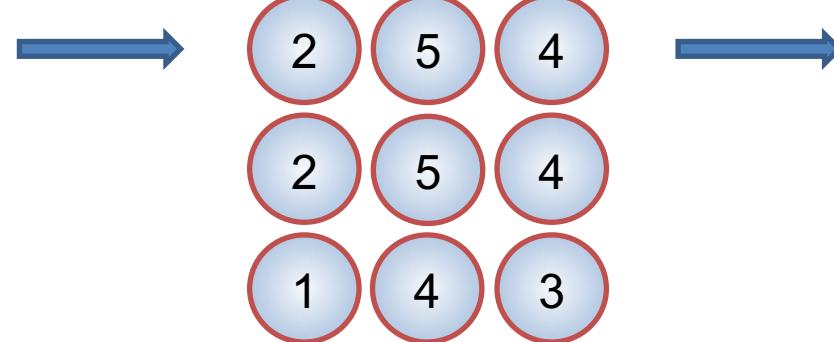
Learn weights of filters in convolutional layers.

Convolutional Neural Networks

Max Pool - Example

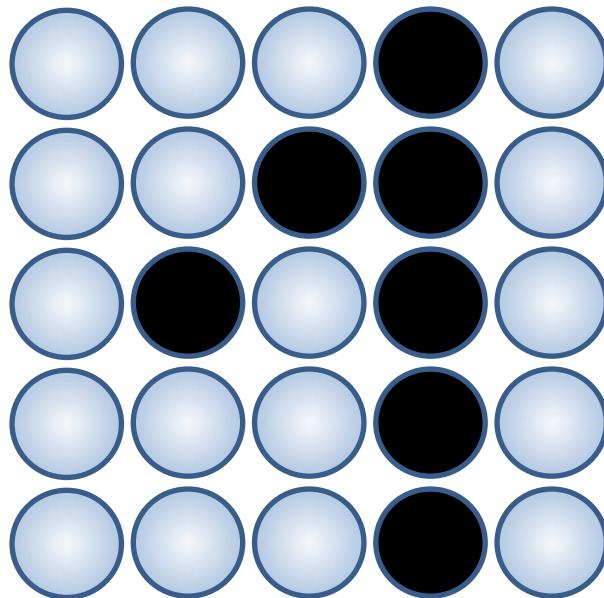


- We already have the layer with the 9 neurons for the 5x5 image

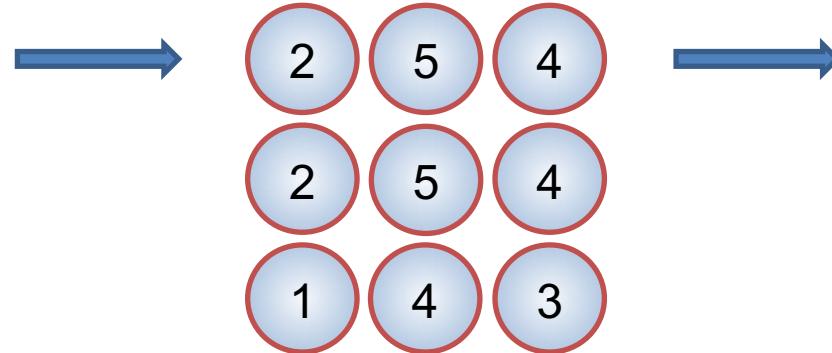


Convolutional Neural Networks

Max Pool - Example

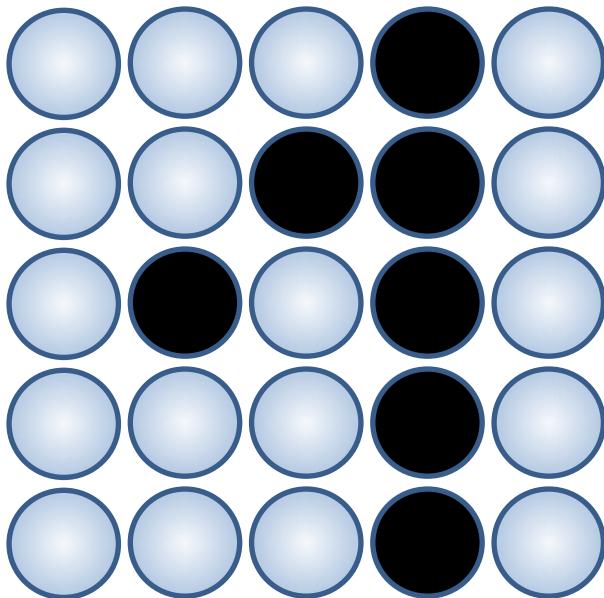


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance

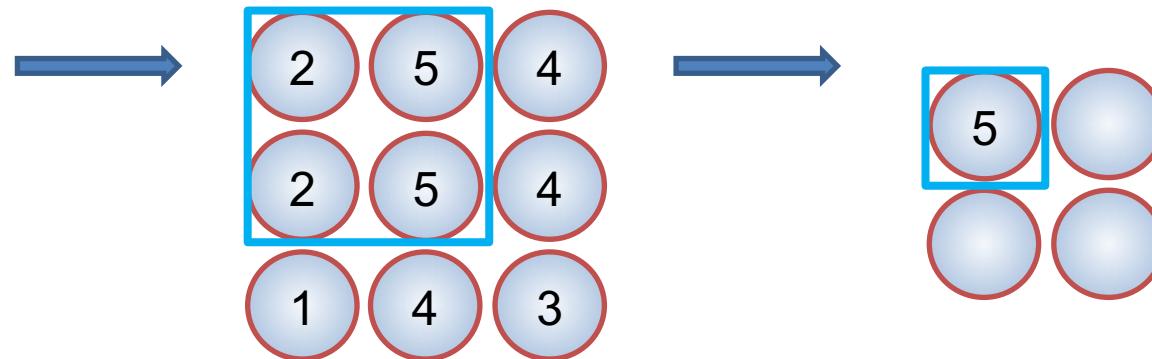


Convolutional Neural Networks

Max Pool - Example

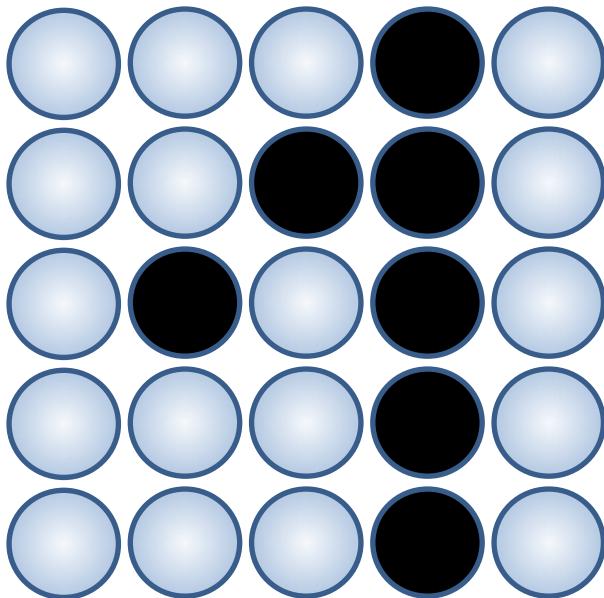


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1)

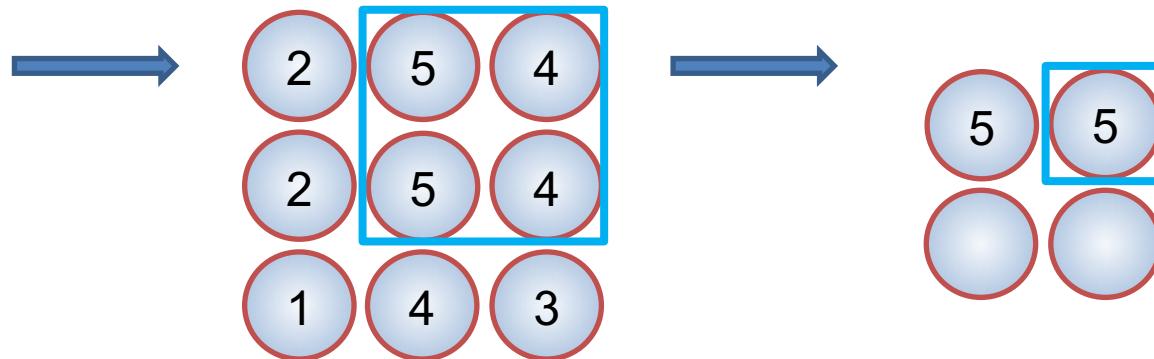


Convolutional Neural Networks

Max Pool - Example

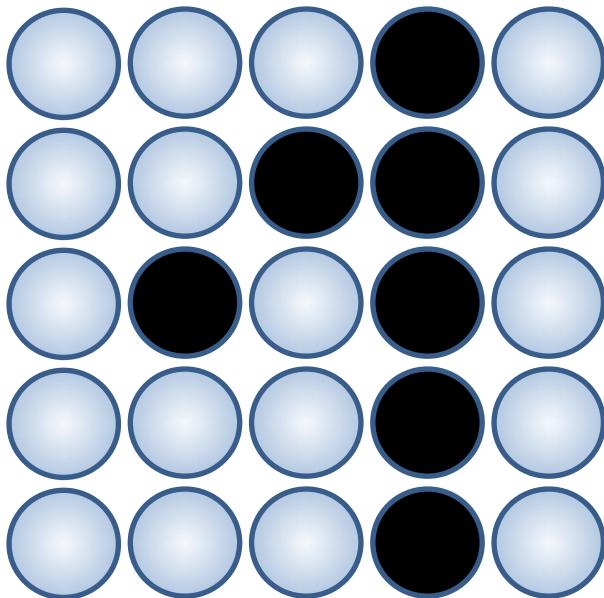


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1)

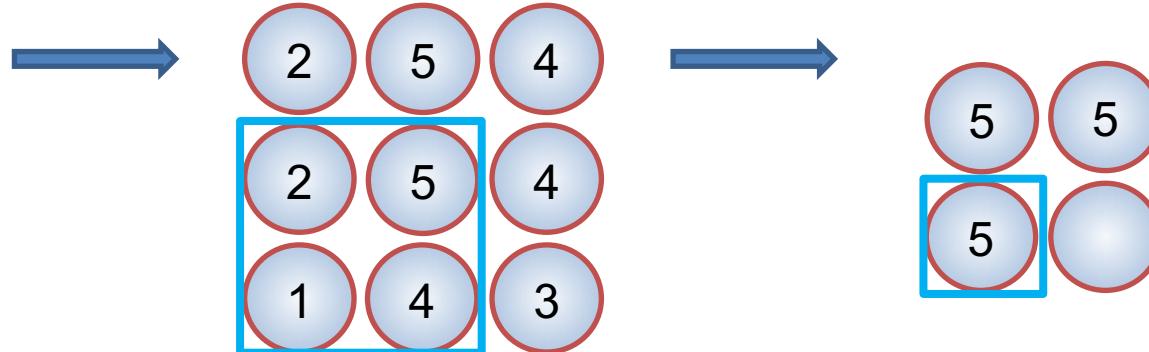


Convolutional Neural Networks

Max Pool - Example

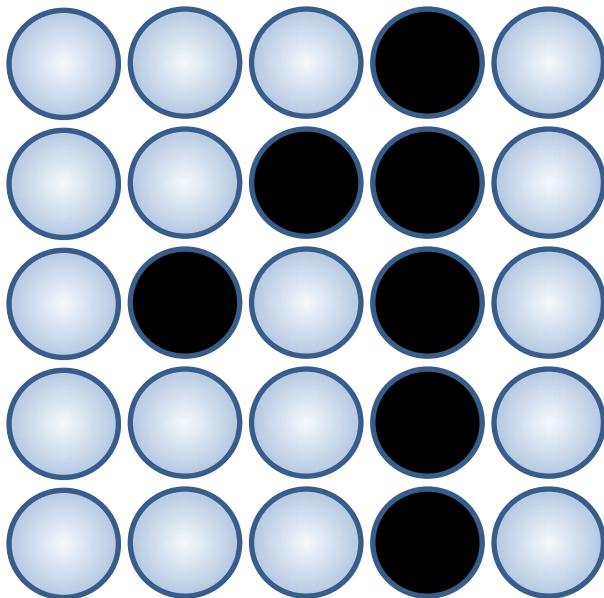


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1)

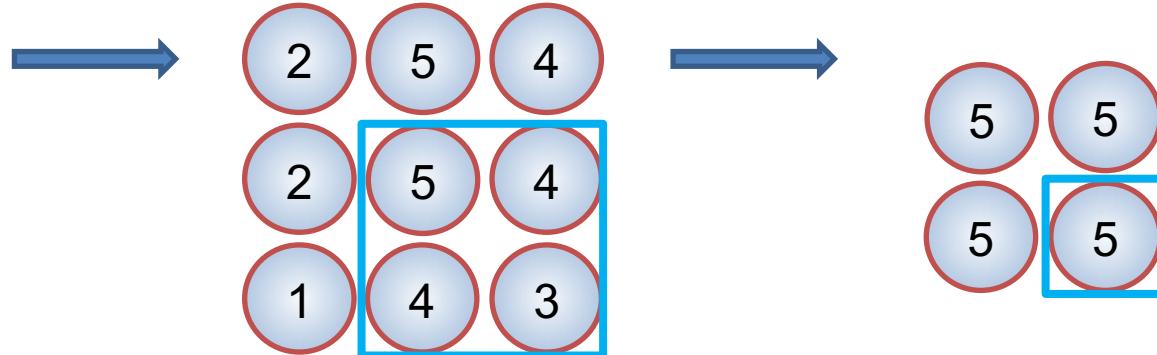


Convolutional Neural Networks

Max Pool - Example

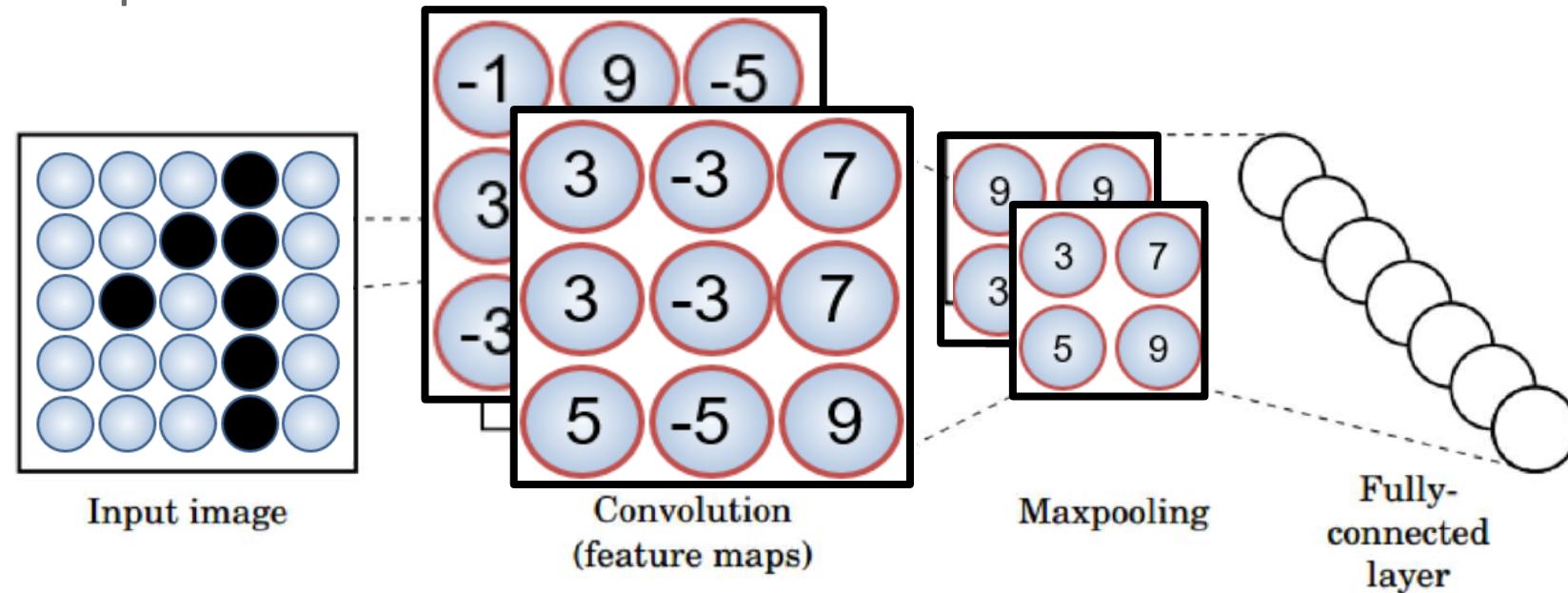


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1)



Convolutional Neural Networks

Building a CNN - Example

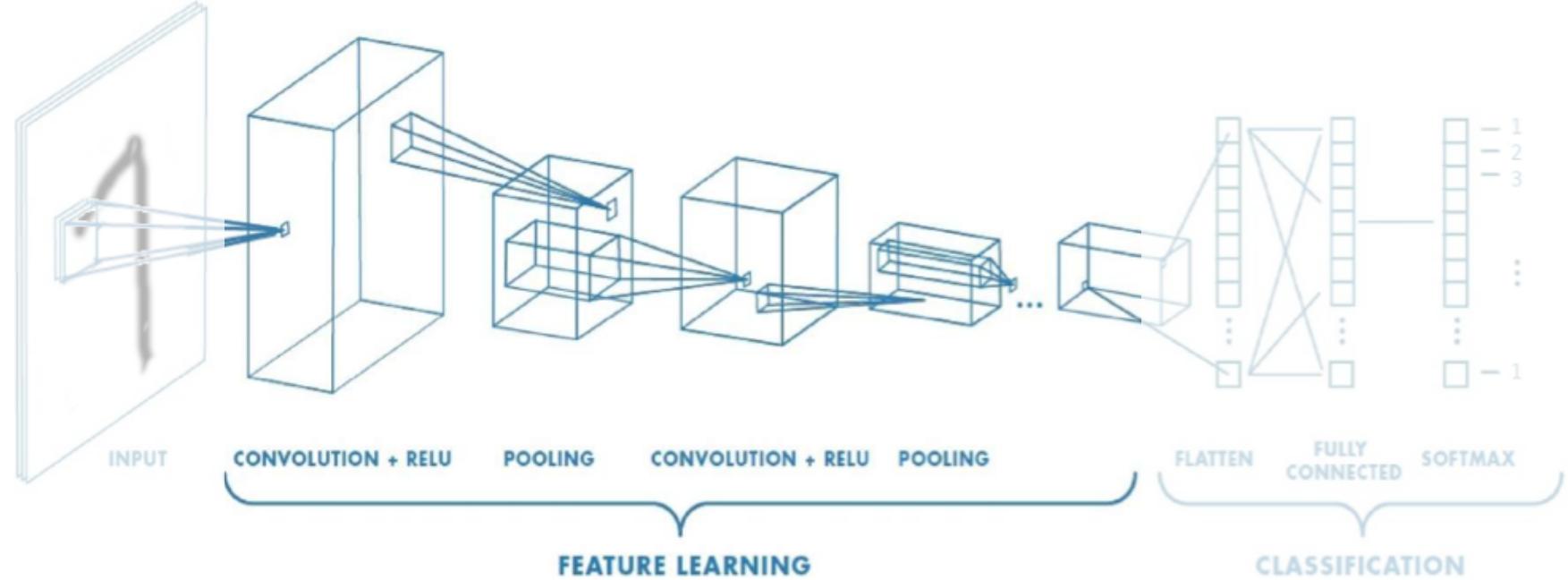


- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

Train model with image data.

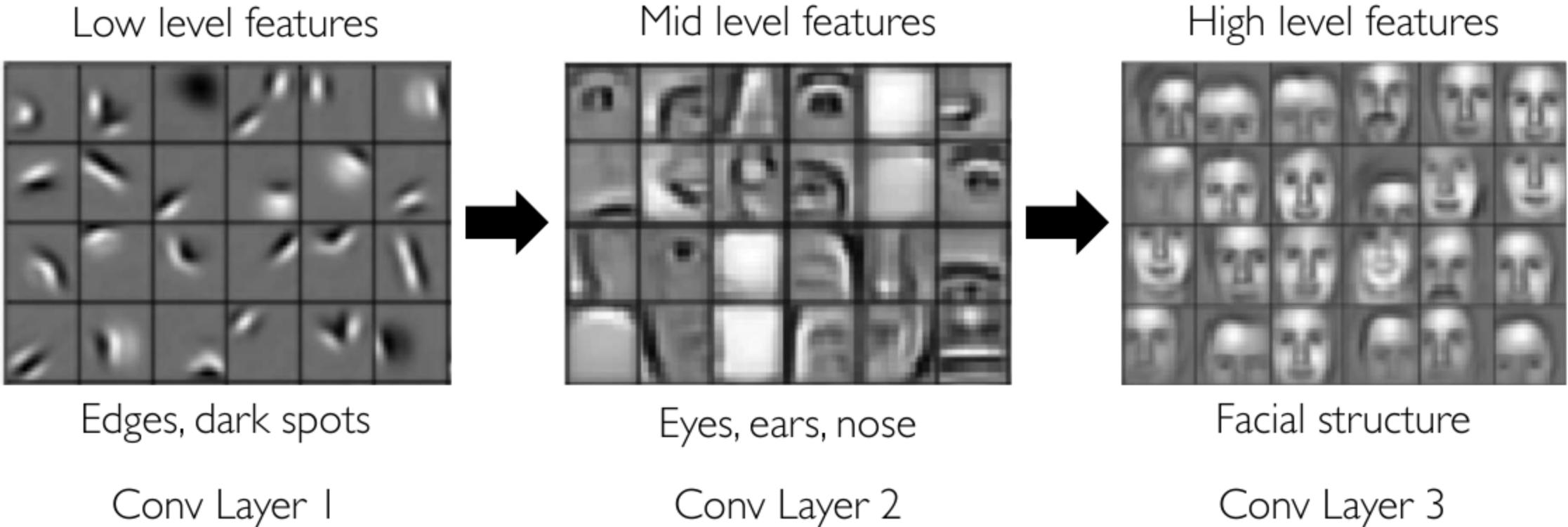
Learn weights of filters in convolutional layers.

Representation Learning

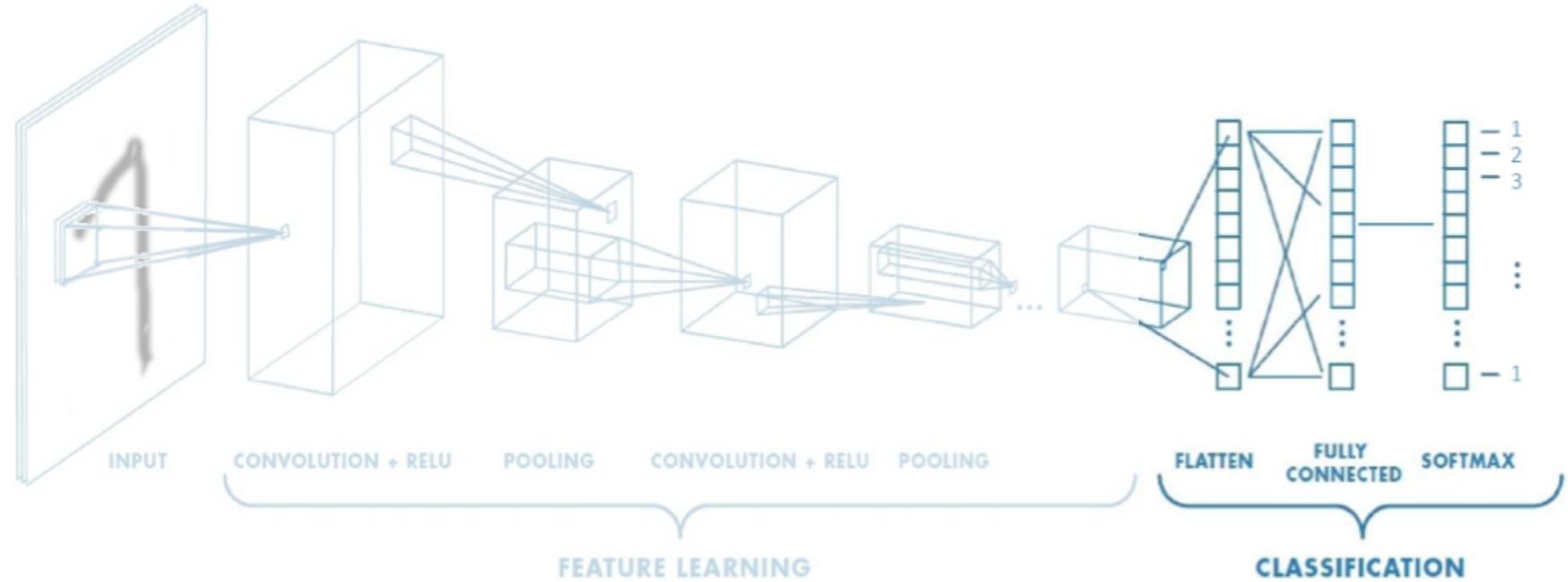


Convolutional Neural Networks

Representation learning



Classification

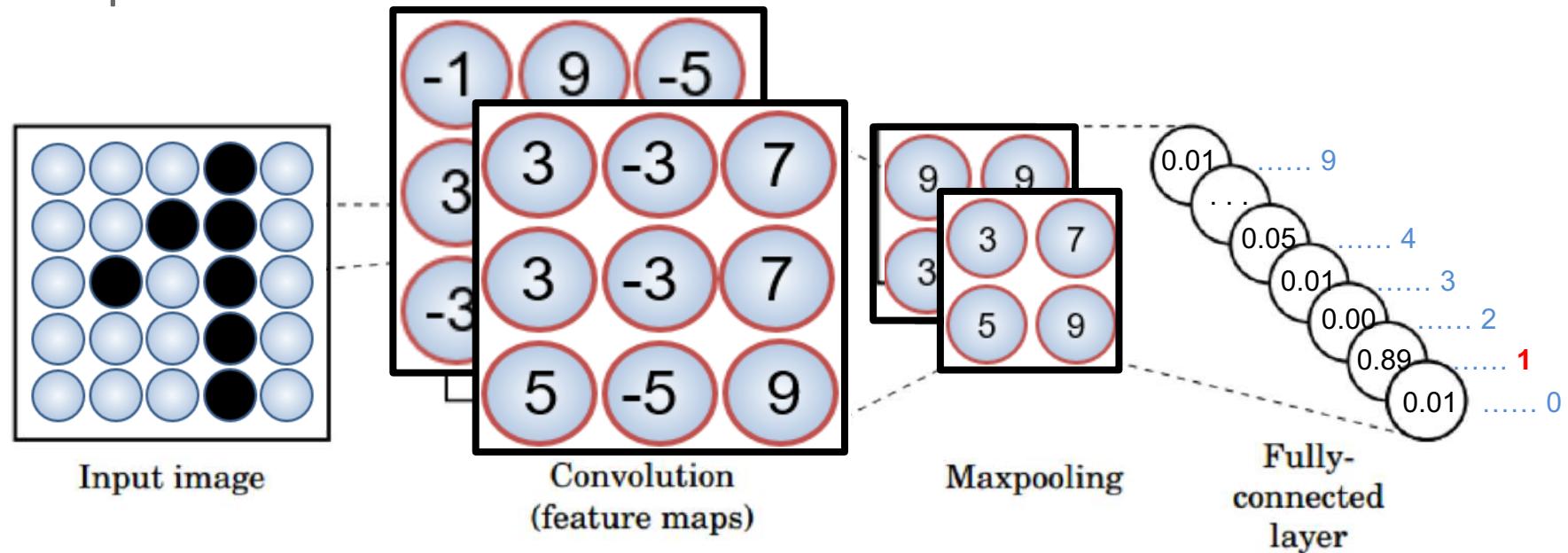


- CONV and POOL layers output high-level features of input
- Fully connected layer uses these features for classifying input image
- Express output as **probability** of image belonging to a particular class

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

Convolutional Neural Networks

Building a CNN - Example



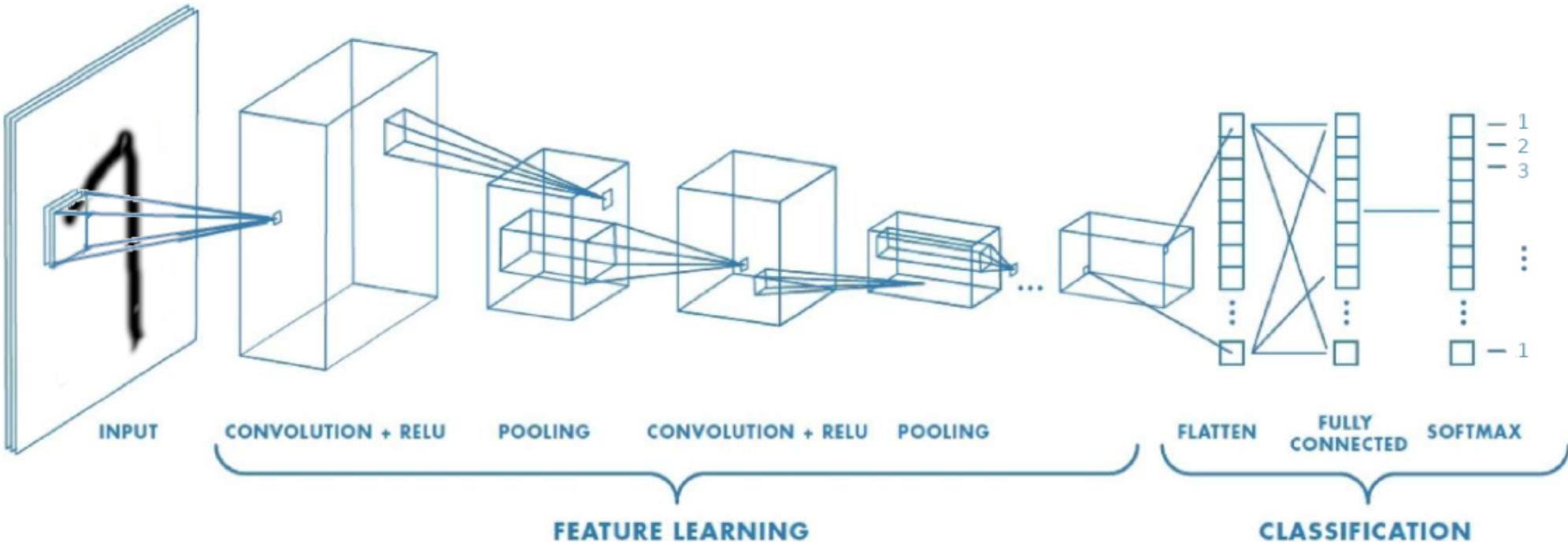
- 1. Convolution:** Apply filters with learned weights to generate feature maps.
- 2. Non-linearity:** Often ReLU.
- 3. Pooling:** Downsampling operation on each feature map.

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: III. Practical application

- Frameworks
- Pre-trained models
- Code & knowledge sources
- (Current research topics)

In B. Hands-on seminar – Try it out:

- Understand PyTorch's Tensor library and neural networks at a high level
- Train a small neural network for regression analysis
- Train a small neural network to classify images