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

Building the network

- Input layer
- Hidden layers
  - Initialize weights and biases
- Output layer

Forward propagate input

- Calculating weighted sum from input
- Add non-linearity using activation function
- Make prediction from input

Evaluate performance

- Compute the error and its gradient

Back propagate

- Update weights according to the optimization algorithm

Improving the network

- **Regularization**,
- Data preprocessing,
- Fine-tuning parameters, etc.

Improve the Network: Regularization

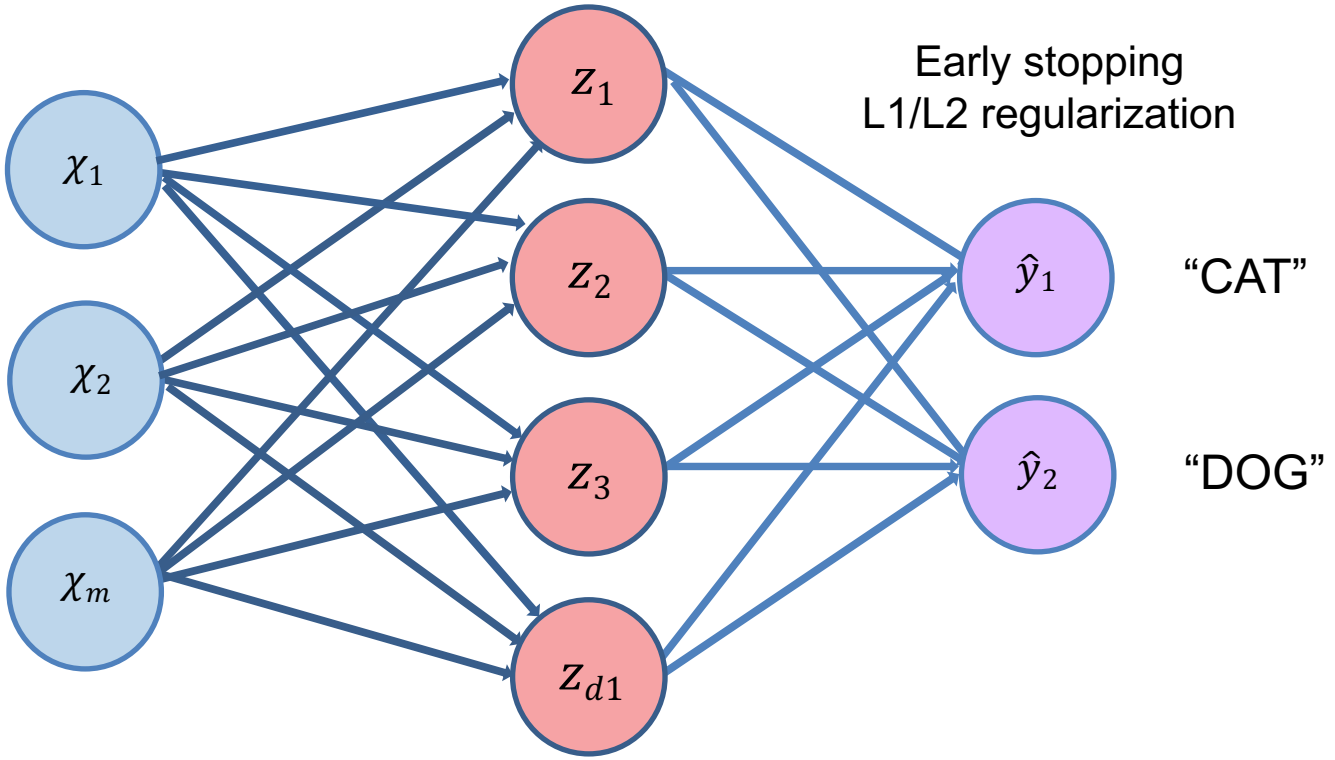
Data Augmentation

CAT

$$\begin{bmatrix} x_1 \\ x_2 \\ x_m \end{bmatrix}$$

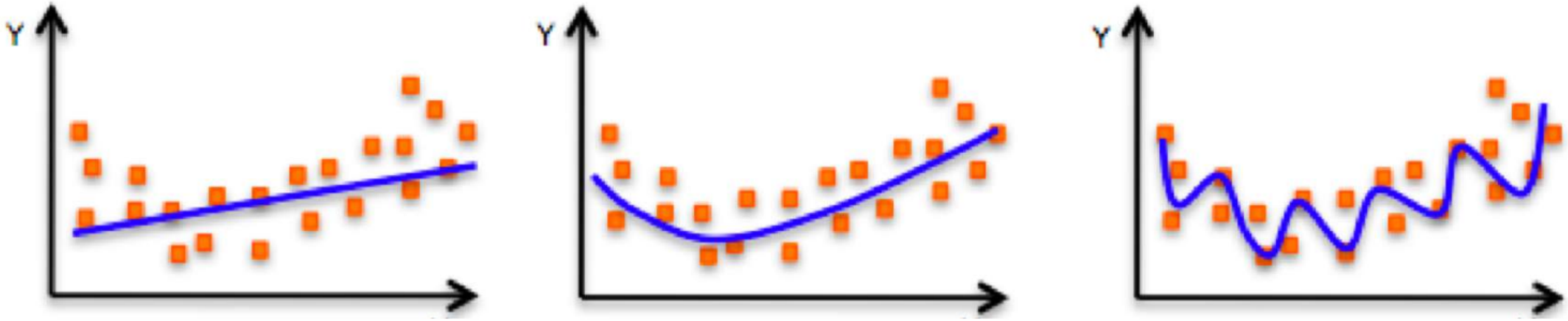
Dropout

Early stopping
L1/L2 regularization



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**

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

→ 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

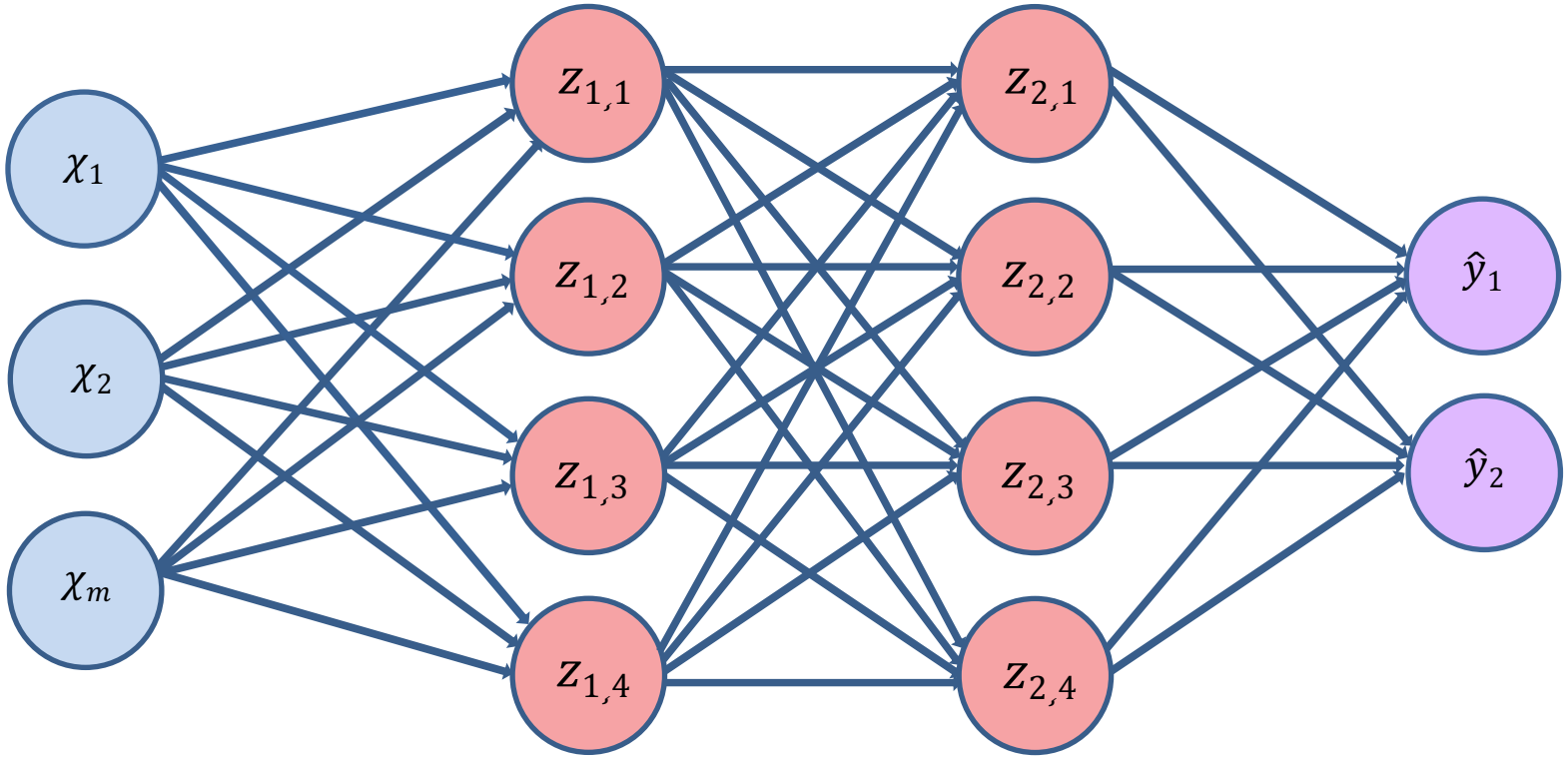
Contrast

Tint

Regularization

Preventing overfitting: Dropout

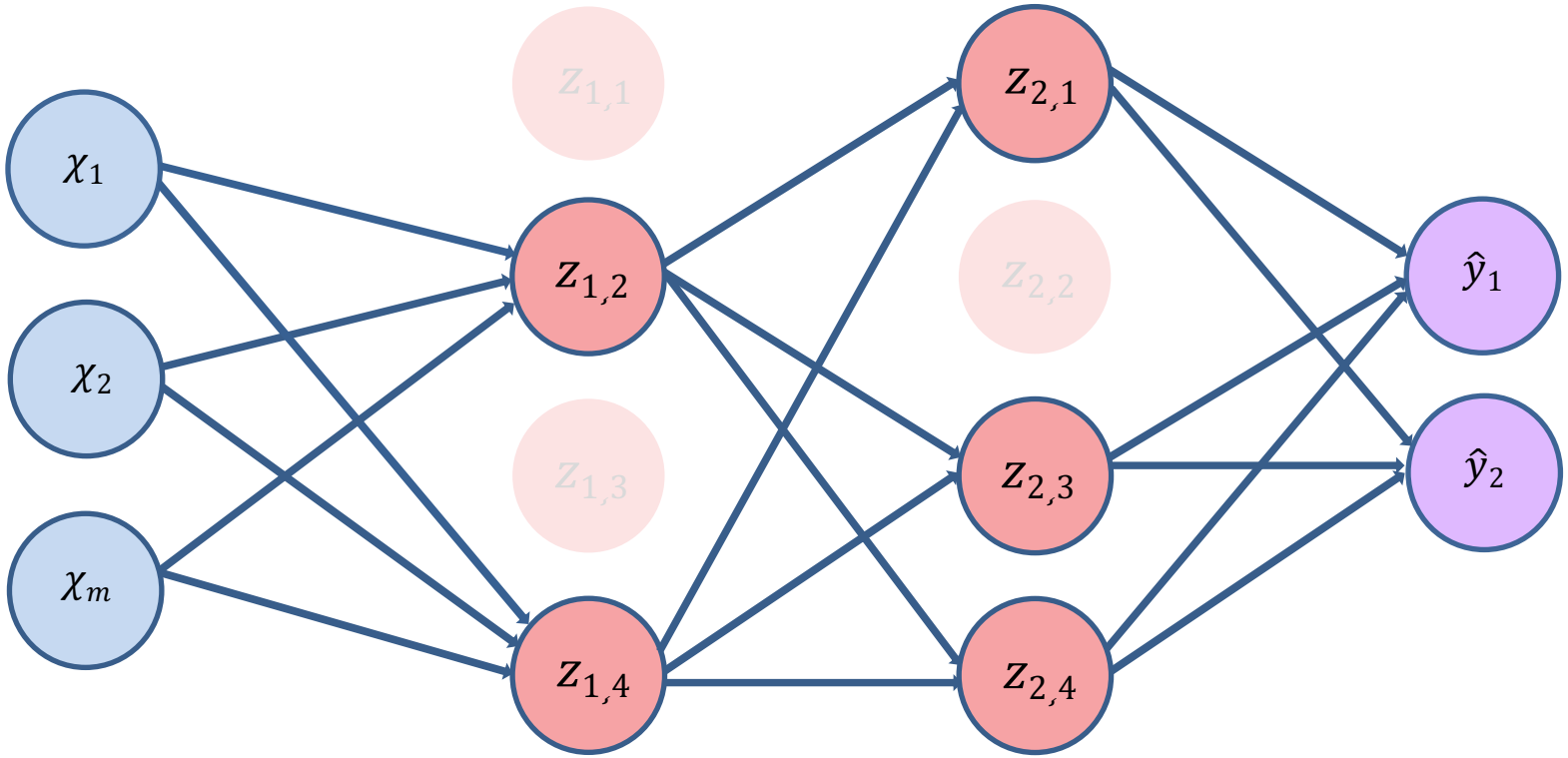
- During training, randomly set some activations to 0



Regularization

Preventing overfitting: Dropout

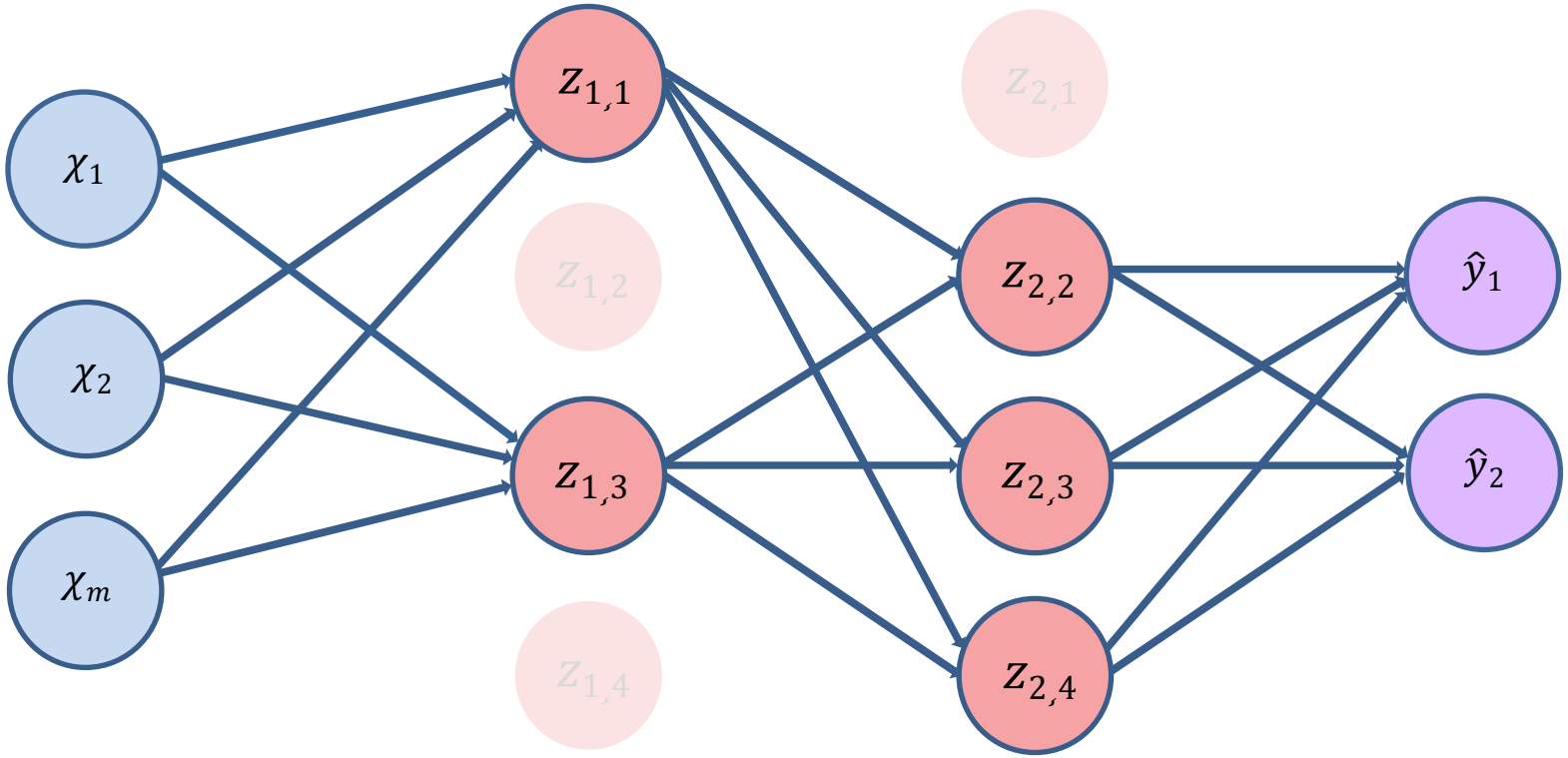
- During training, randomly set some activations to 0



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

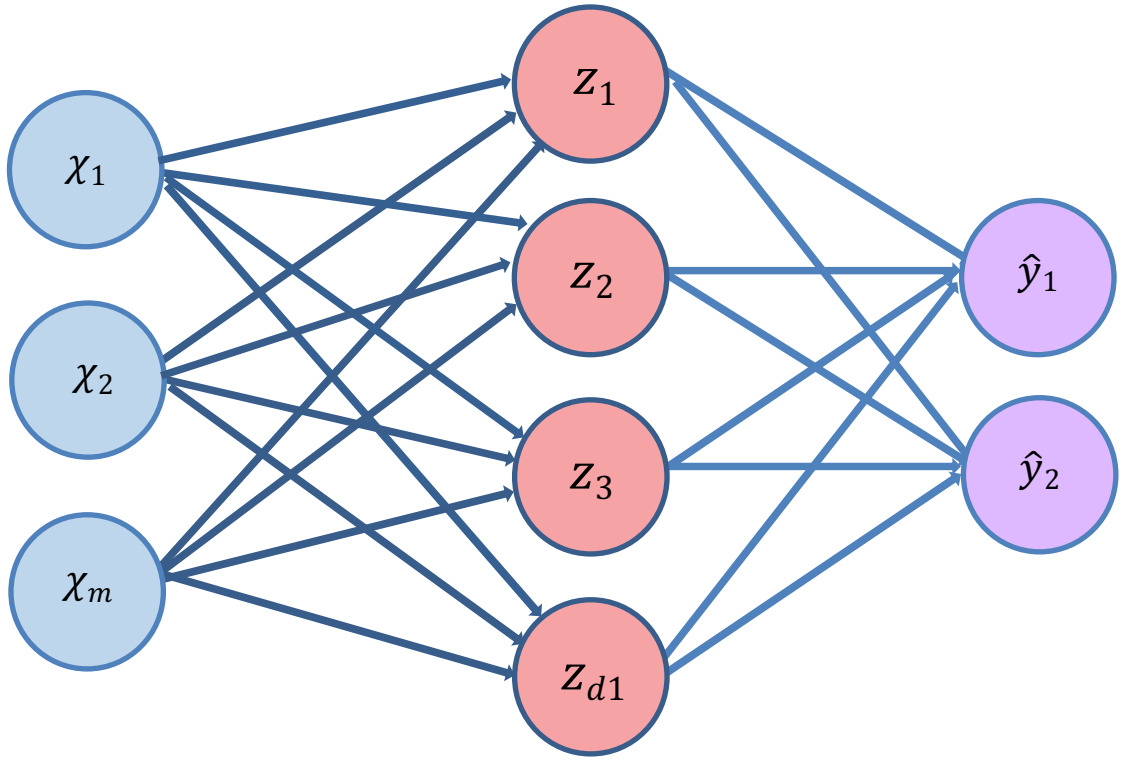
Minibatch composition

(shuffle data, take only a subset, …)

Variants of Neural Networks

CAT

$$\begin{bmatrix} x_1 \\ x_2 \\ x_m \end{bmatrix}$$



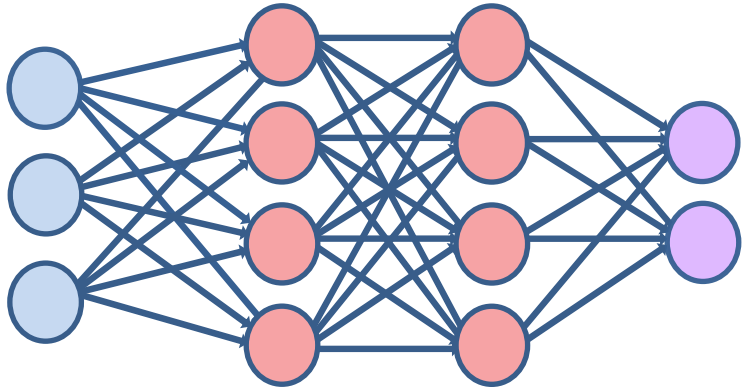
Types of Neural Networks Layer

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	✓	✓		✓

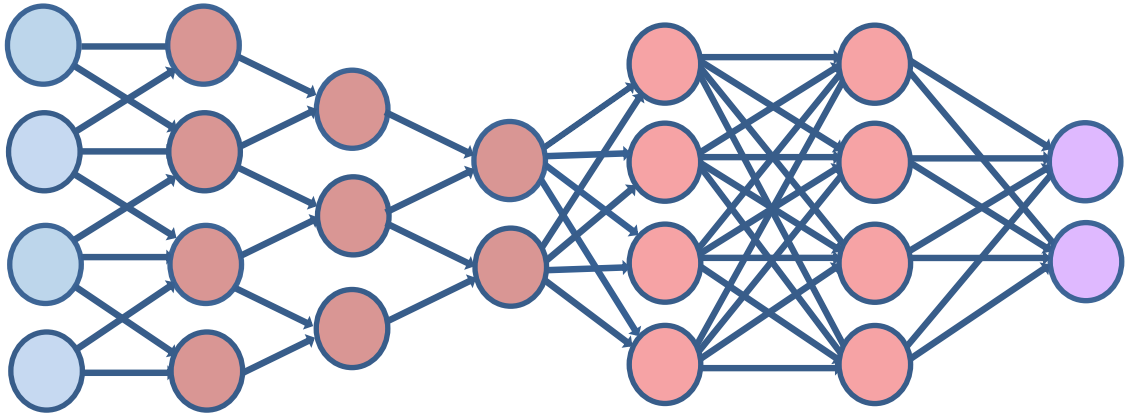
Variants of Neural Network - Examples

= Input
= Output

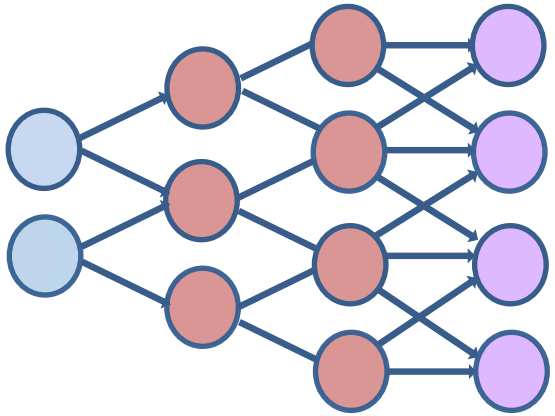
Deep Feed Forward (DFF) (fully connected)



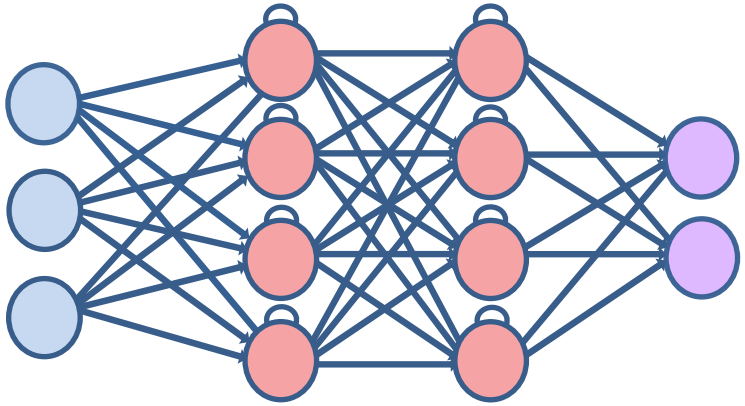
Deep convolution network (DCN)



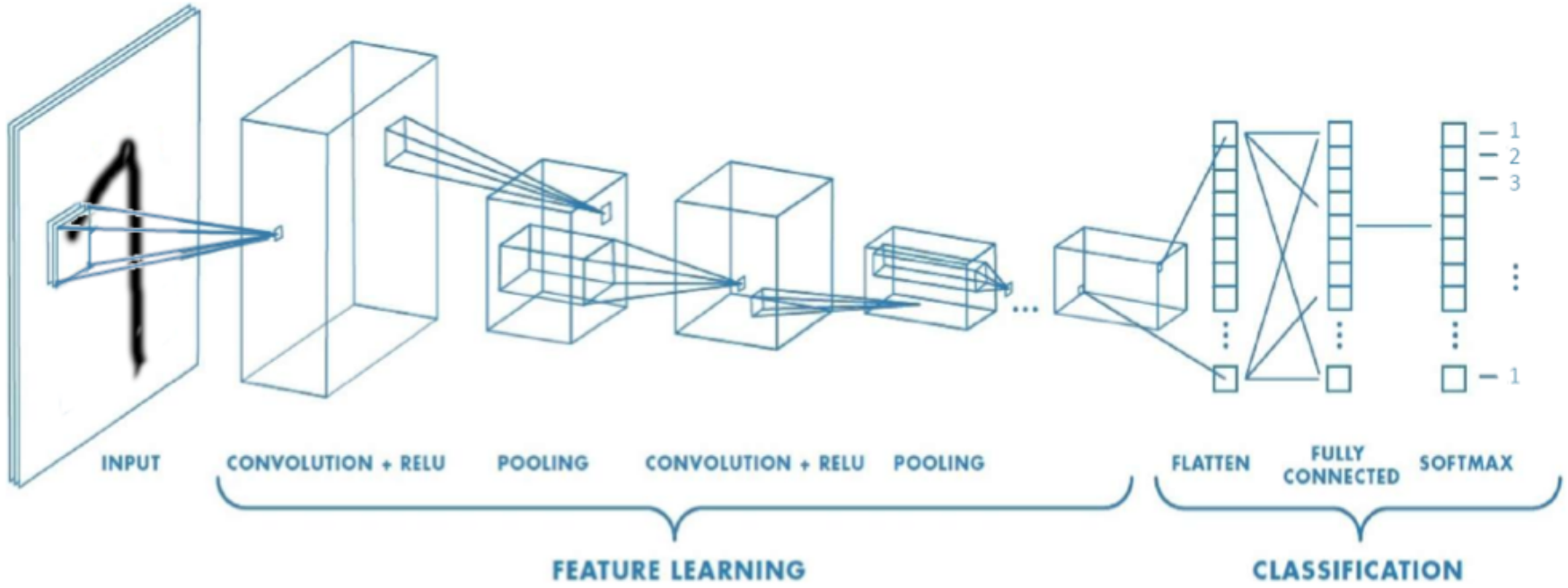
Deconvolution network (DN)



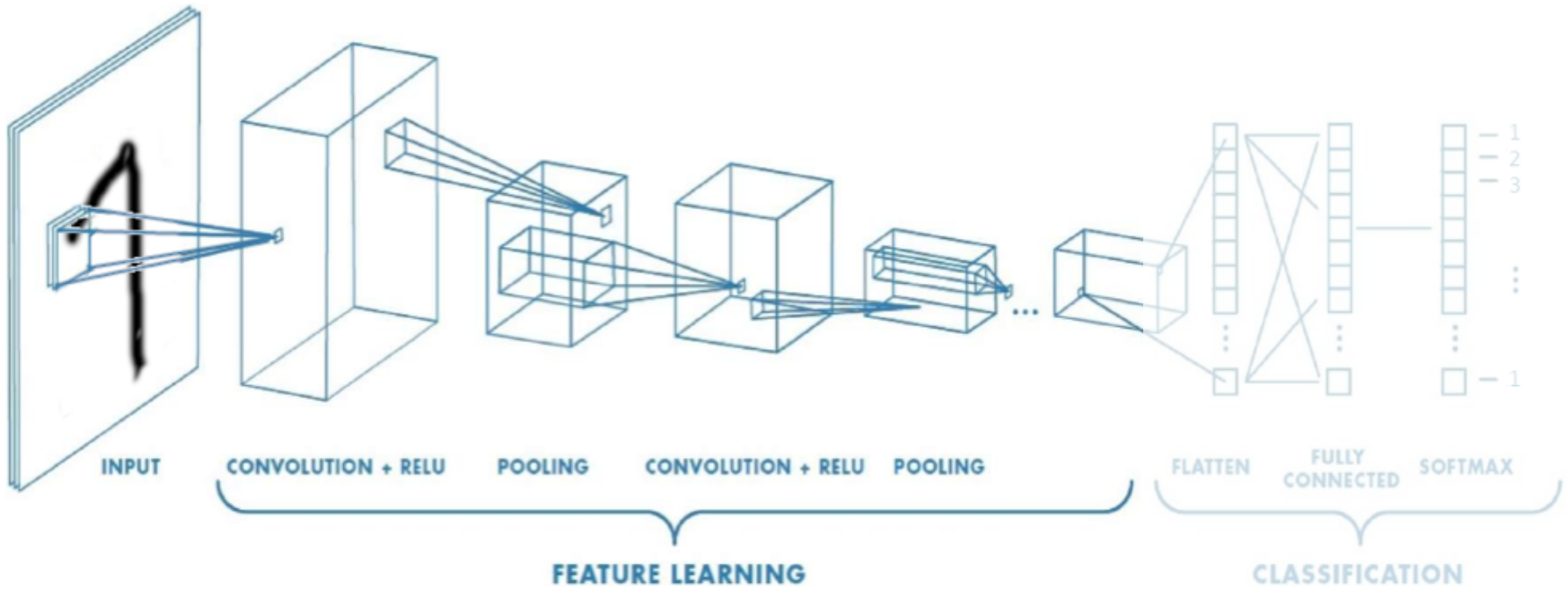
Recurrent neural network (RNN)



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

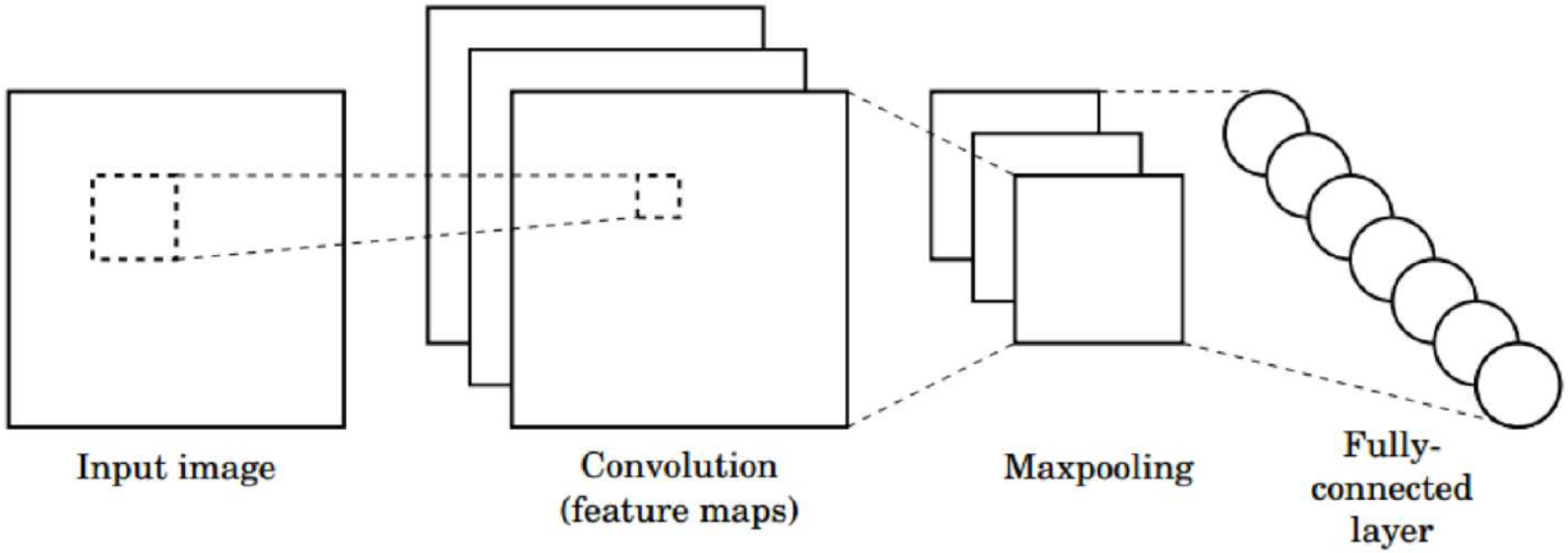
Sharpen

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.

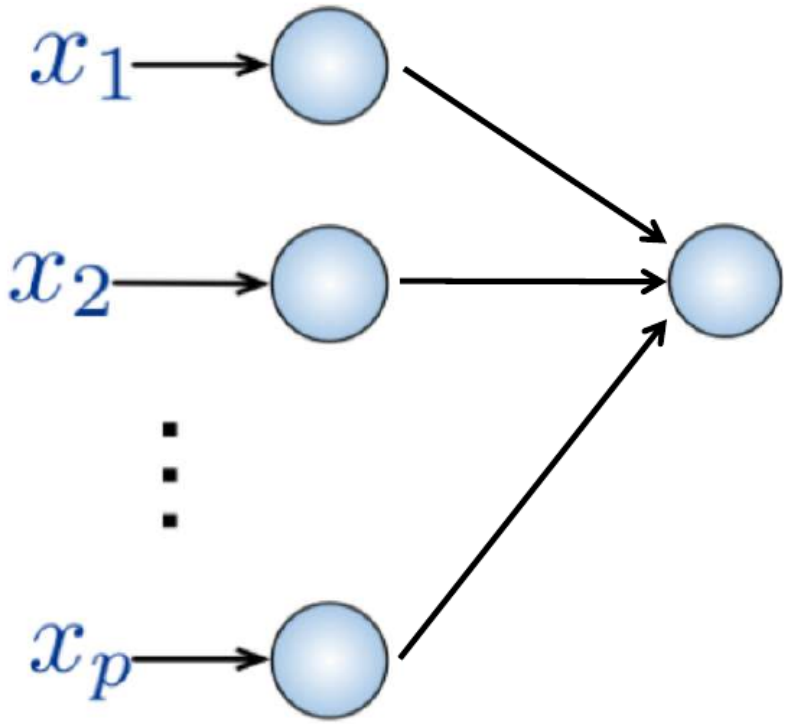
Train model with image data.
Learn weights of filters in convolutional layers.

Convolutional Neural Networks

Learning on image data

Input:

- 2D image
- Vector of pixel values



Fully Connected:

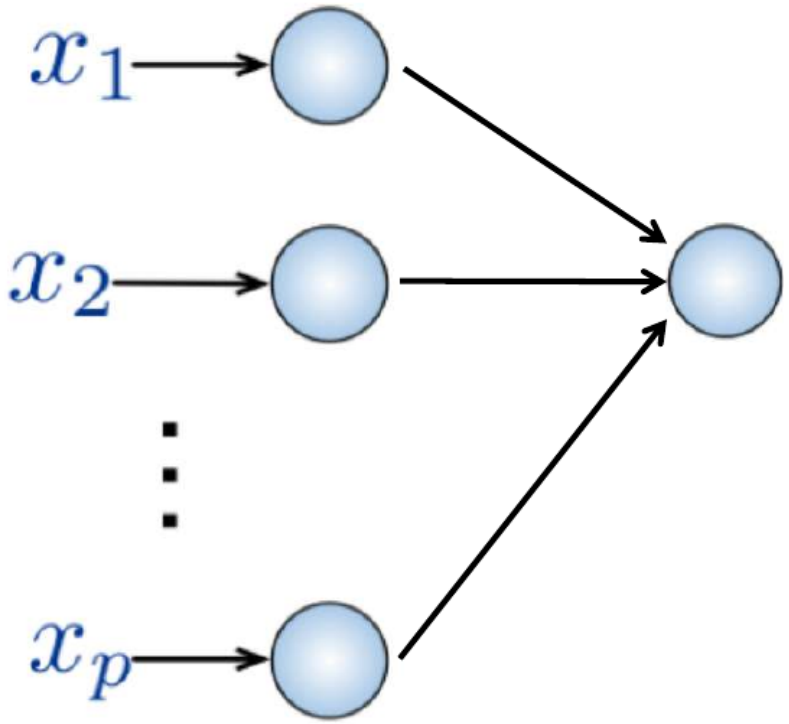
- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

Convolutional Neural Networks

Learning on image data

Input:

- 2D image
- Vector of pixel values



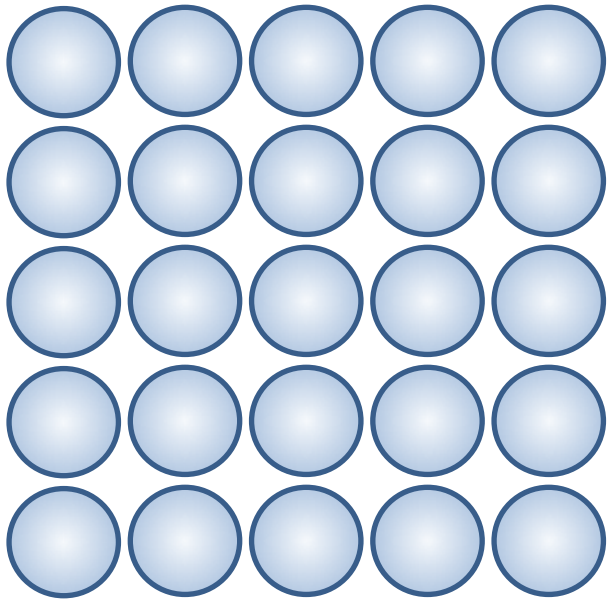
Fully Connected:

- 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

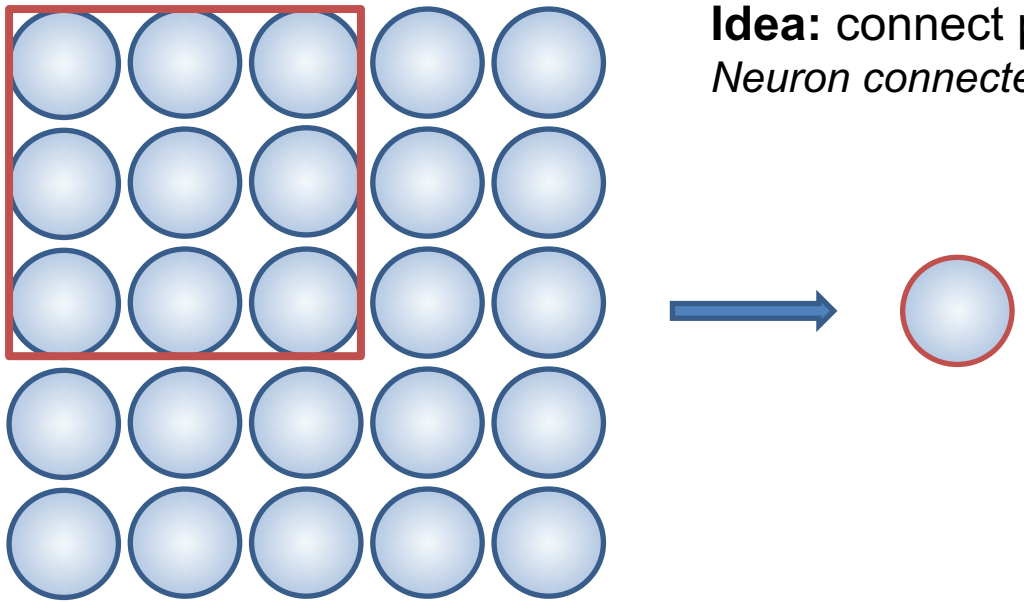
Learning on image data



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

Convolutional Neural Networks

Learning on image data

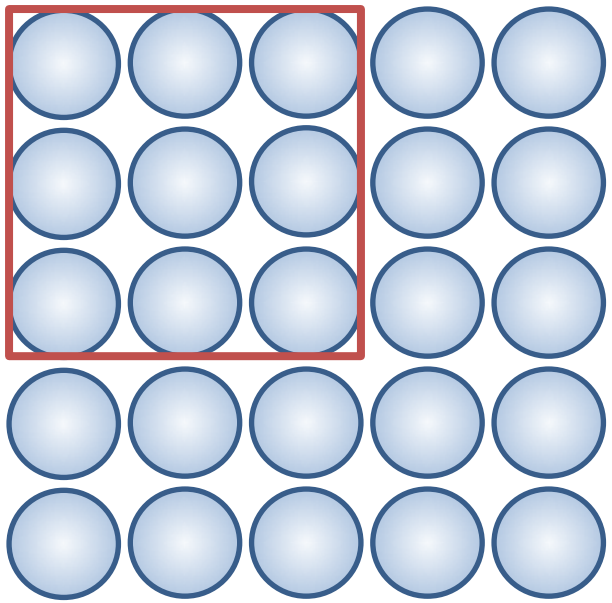


Idea: connect patches of input to neurons in hidden layer

Neuron connected to the region of input only „sees“ these values.

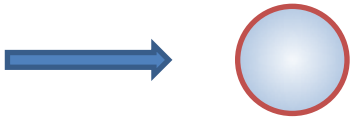
Convolutional Neural Networks

Learning on image data



For a neuron in hidden layer:

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



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

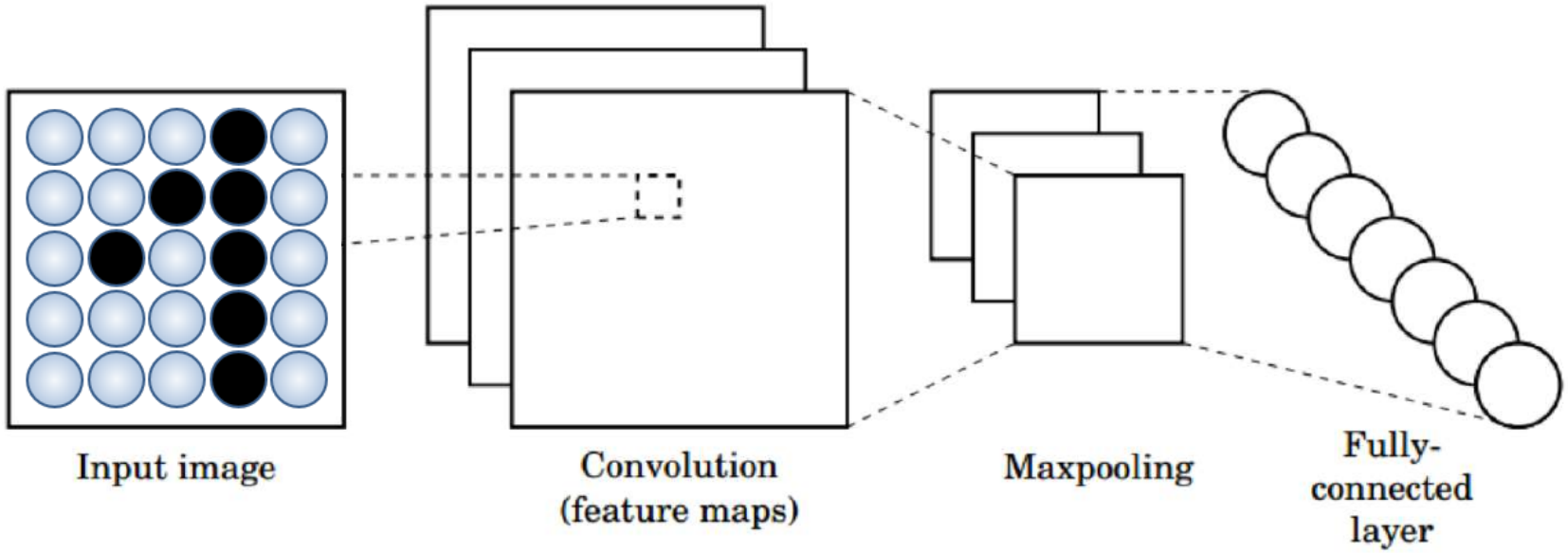
For neuron (p,q) in hidden layer

3x3 filter: matrix of weights W_{ij}

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

Convolutional Neural Networks

Building a CNN - Example

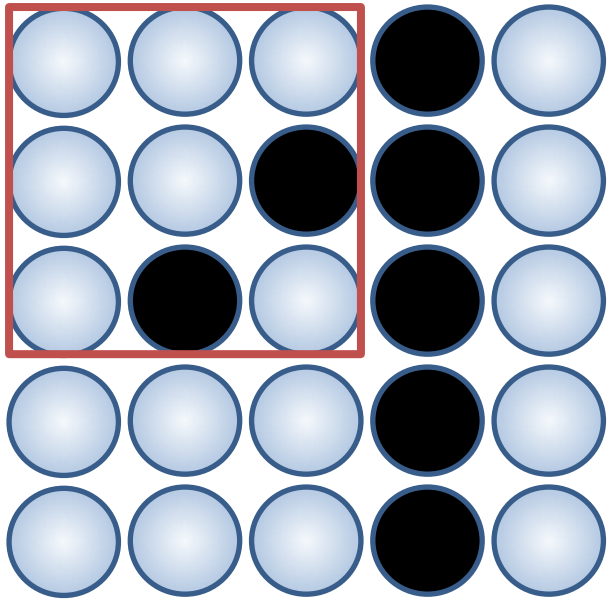


1. **Convolution**: Apply filters with learned weights to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

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

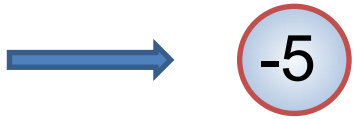
Convolutional Neural Networks

Learning on image data - Example

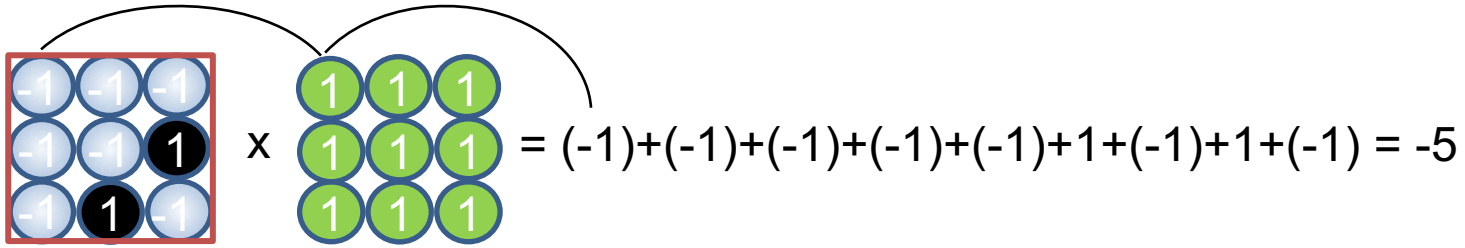


- Connect patch in input layer to a single neuron in subsequent layer
- Use a **sliding window** to define connections

- Ex: extracting feature with filter fi_0

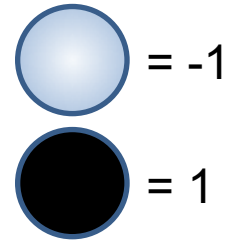


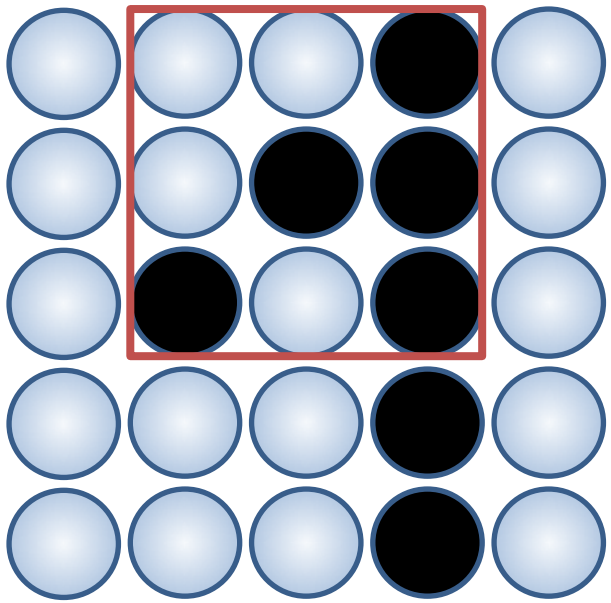
Multiply same indices of clipping and filter and sum it up.



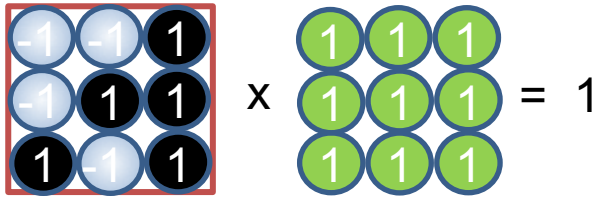
Convolutional Neural Networks

Learning on image data - Example



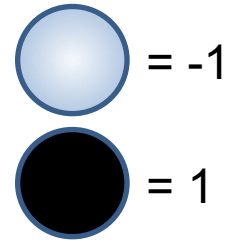


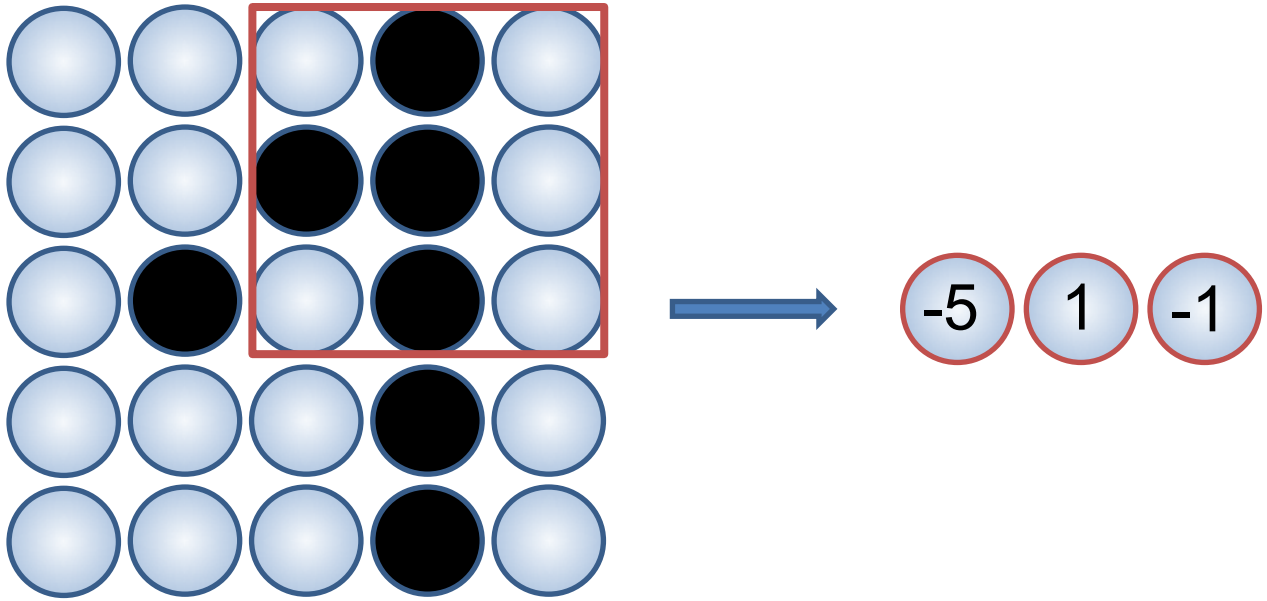
- **Step size** here is one and the clipping is 3x3

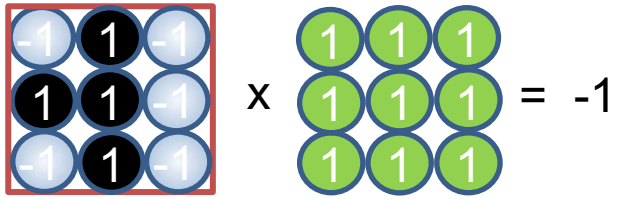


Convolutional Neural Networks

Learning on image data - Example

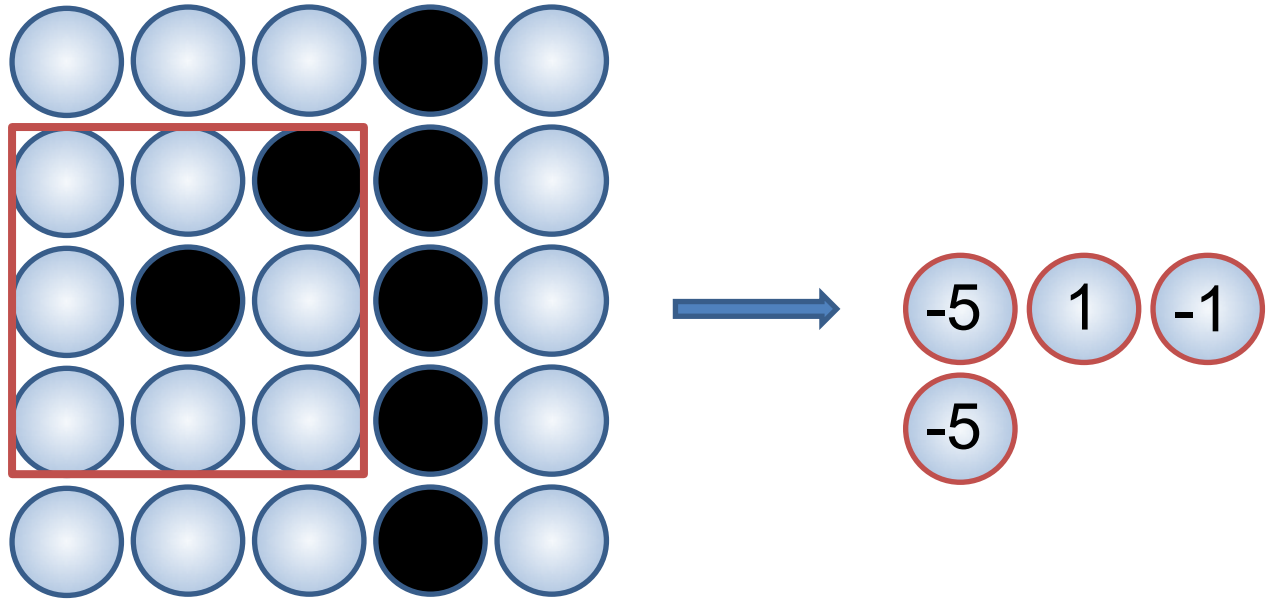






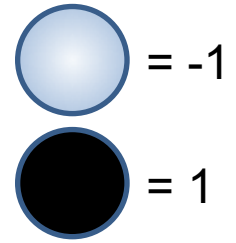
Convolutional Neural Networks

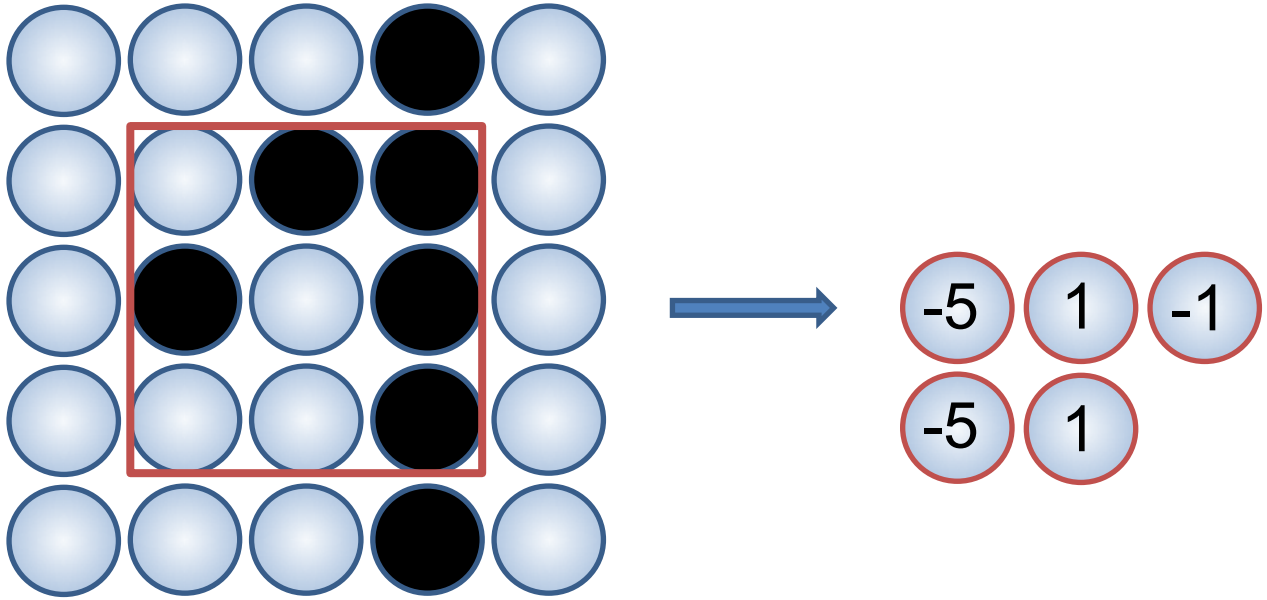
Learning on image data - Example

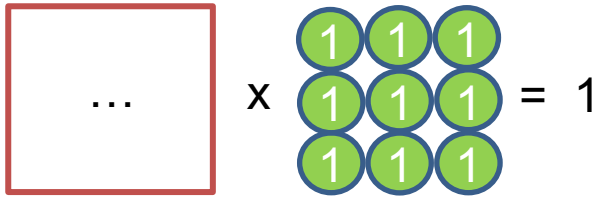


Convolutional Neural Networks

Learning on image data - Example

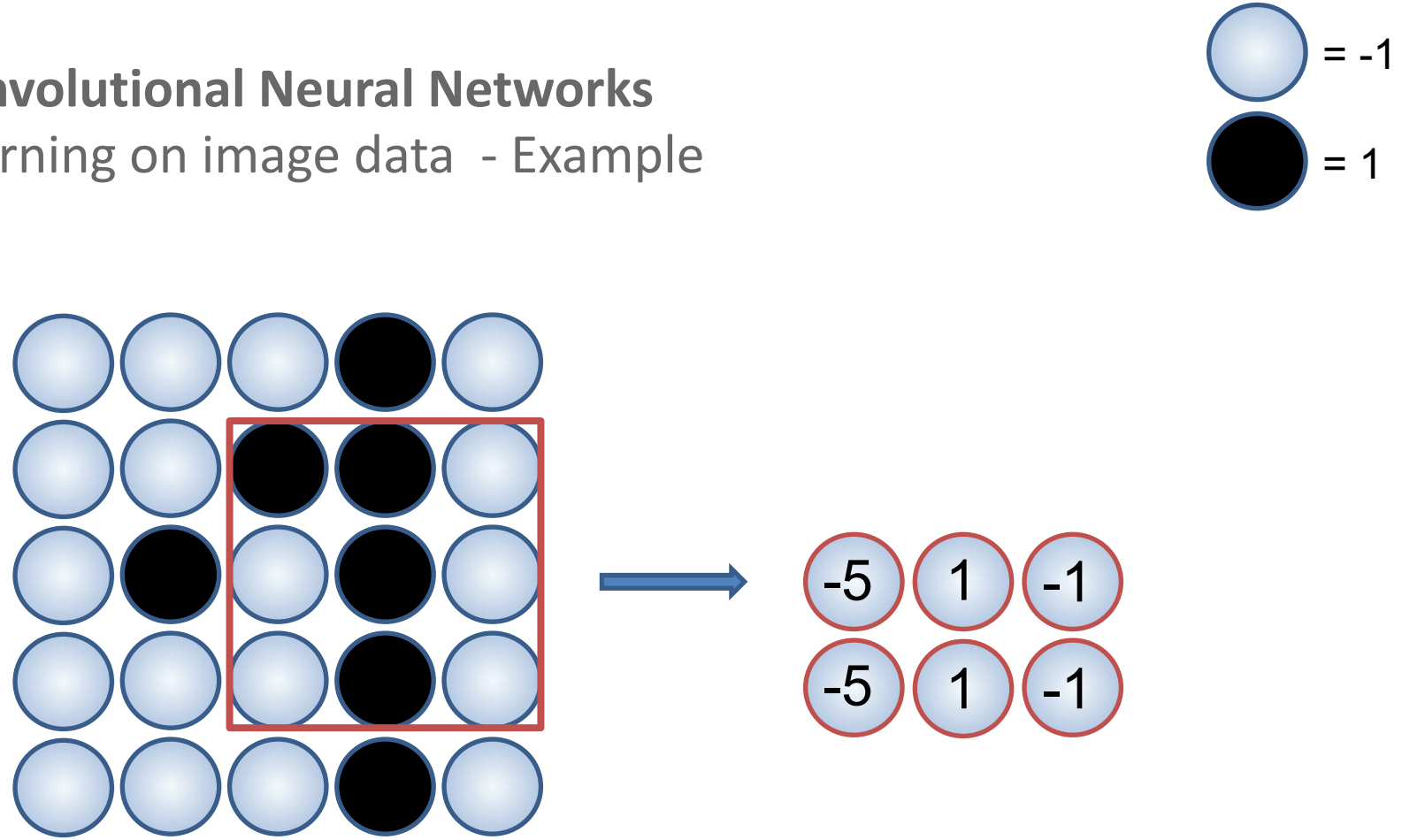






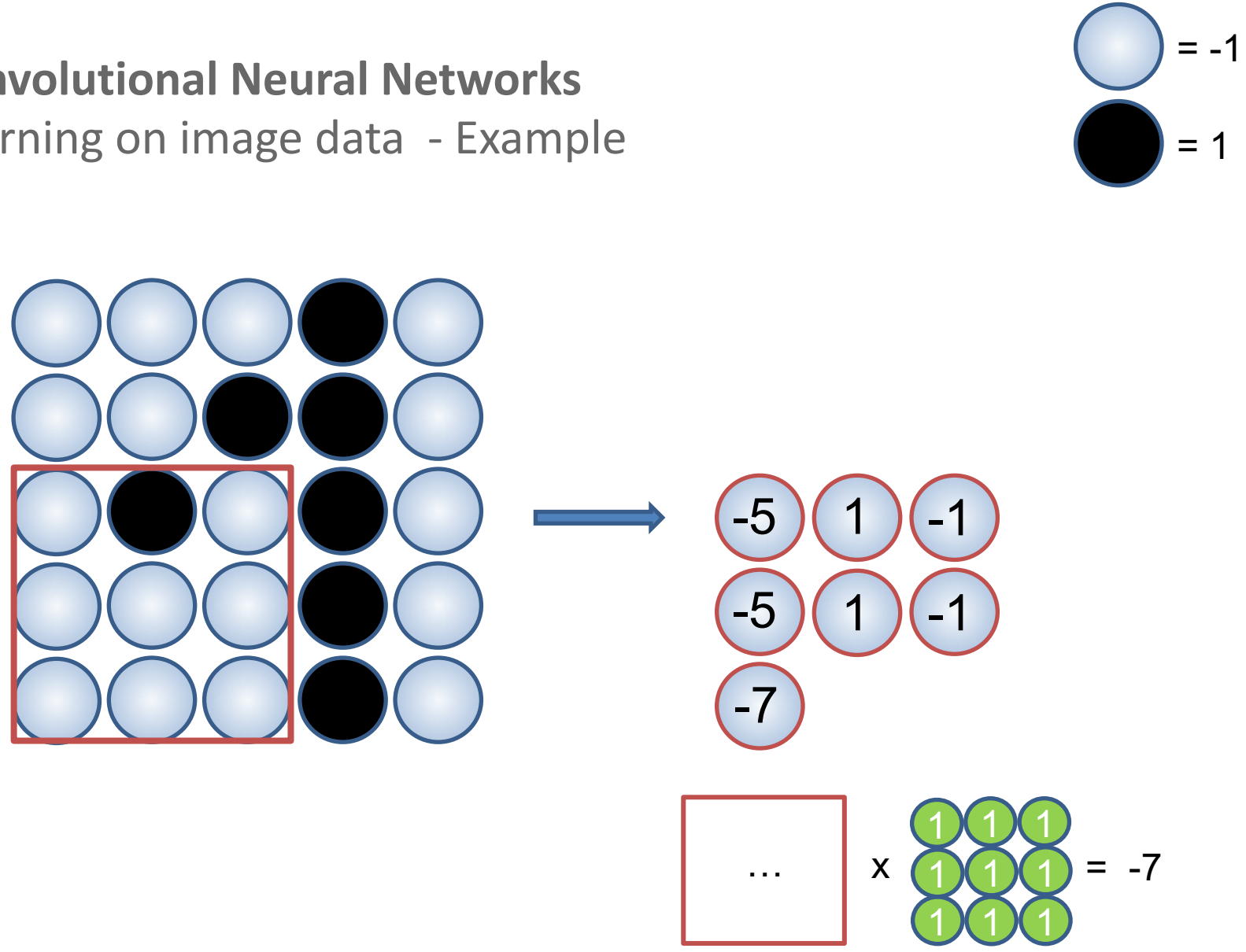
Convolutional Neural Networks

Learning on image data - Example



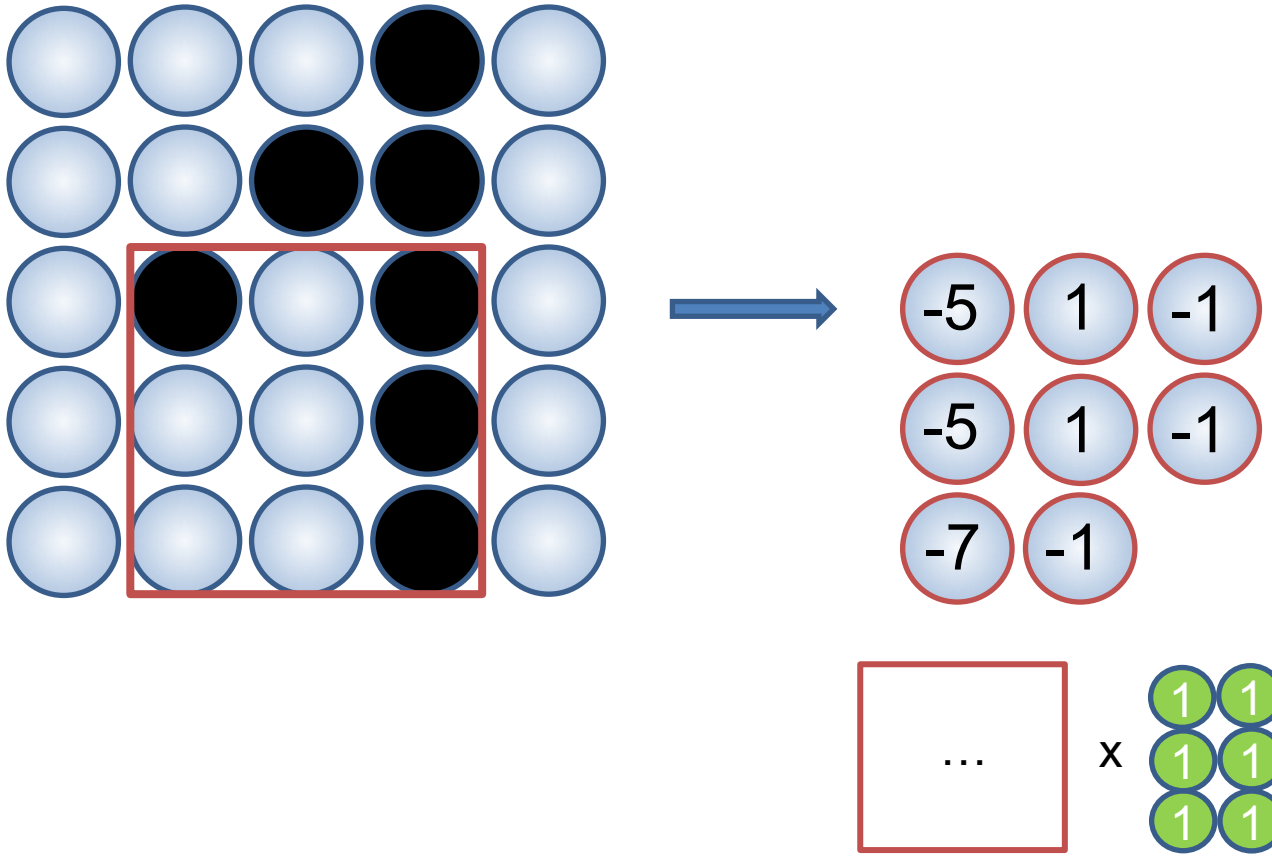
Convolutional Neural Networks

Learning on image data - Example



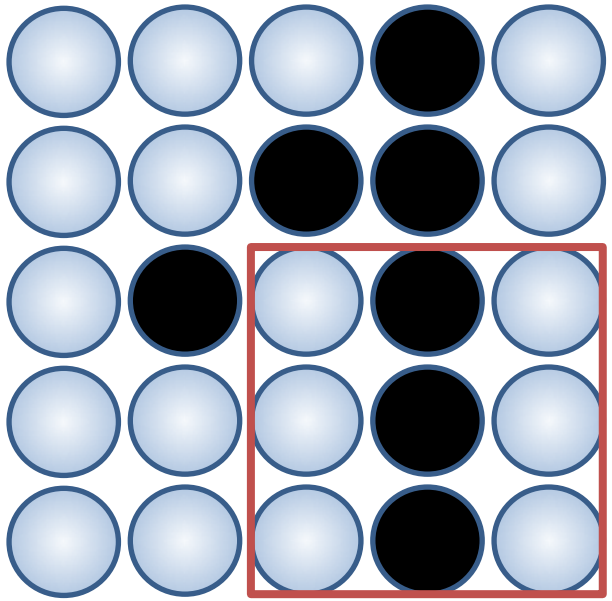
Convolutional Neural Networks

Learning on image data - Example

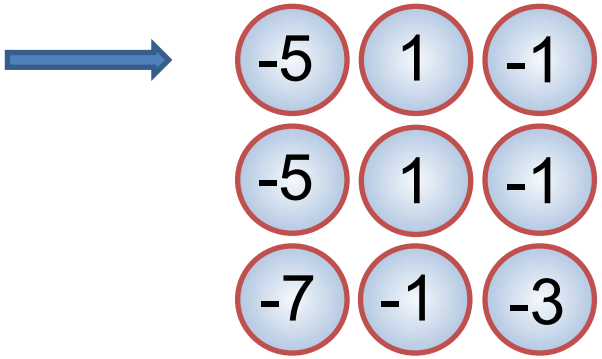


Convolutional Neural Networks

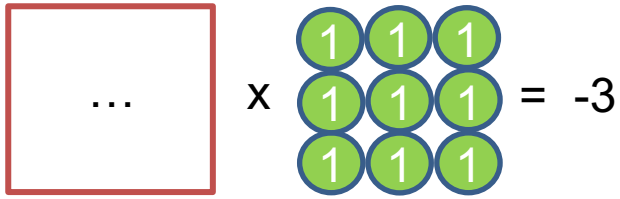
Learning on image data - Example



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

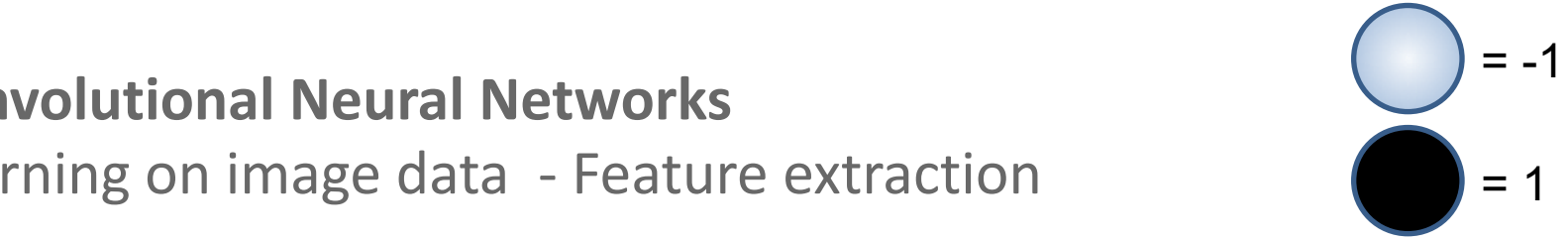


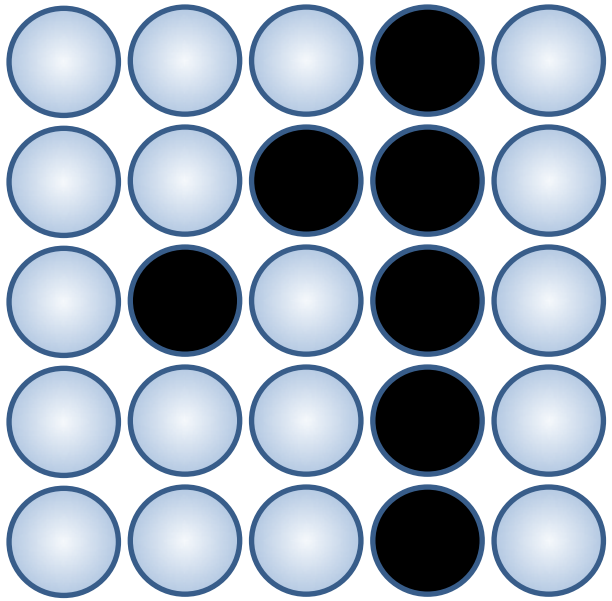
- Filter of size 3x3 → 9 separate weights
- Scan step wise the image with the filter
- That’s **Convolution**



Convolutional Neural Networks

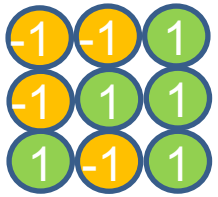
Learning on image data - Feature extraction





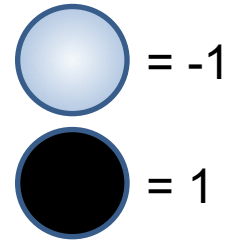
- Repeat the scan process with **multiple filters**

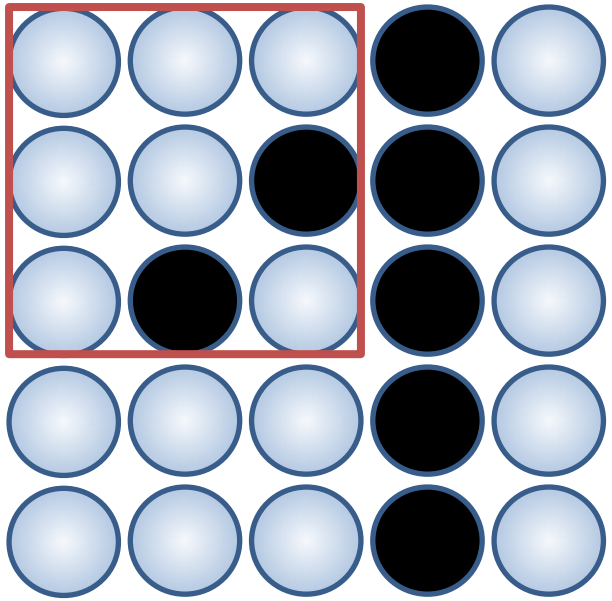
Next: extracting feature with filter fi_1 =



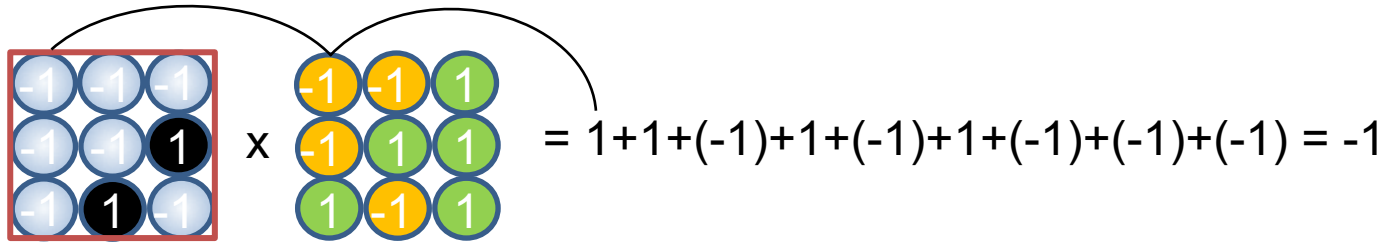
Convolutional Neural Networks

Learning on image data - Feature extraction



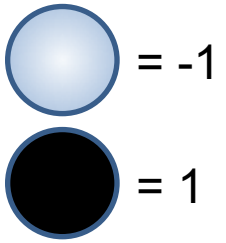


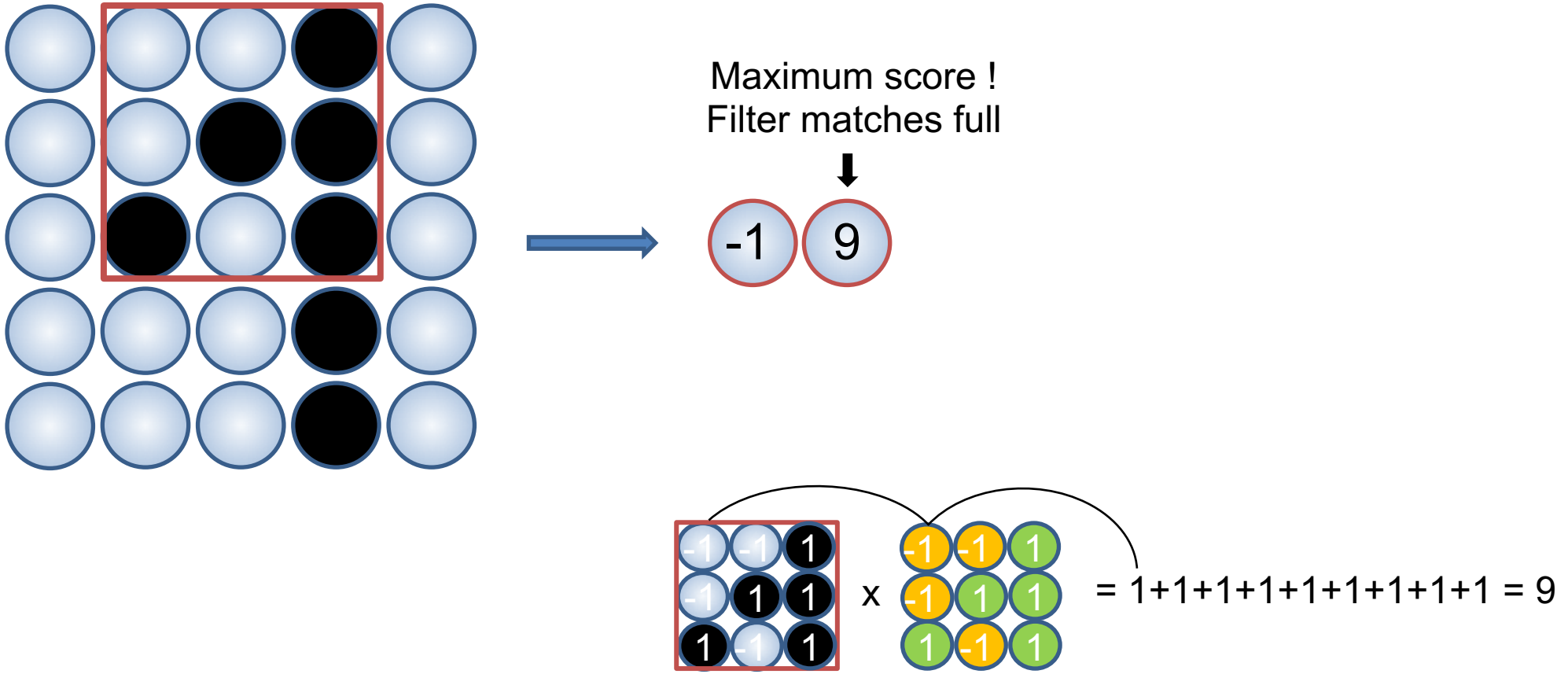
Low score !
barely matches



Convolutional Neural Networks

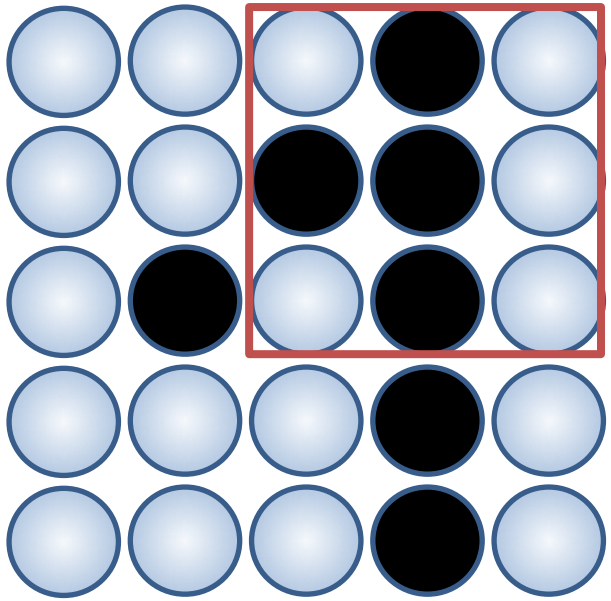
Learning on image data - Feature extraction



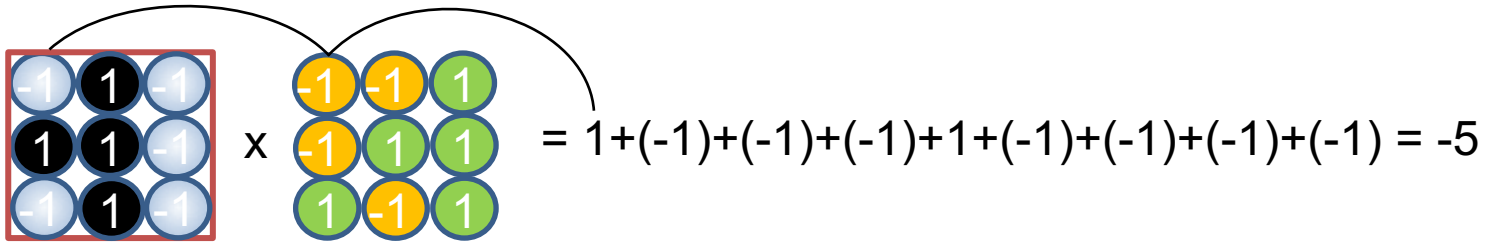


Convolutional Neural Networks

Learning on image data - Feature extraction

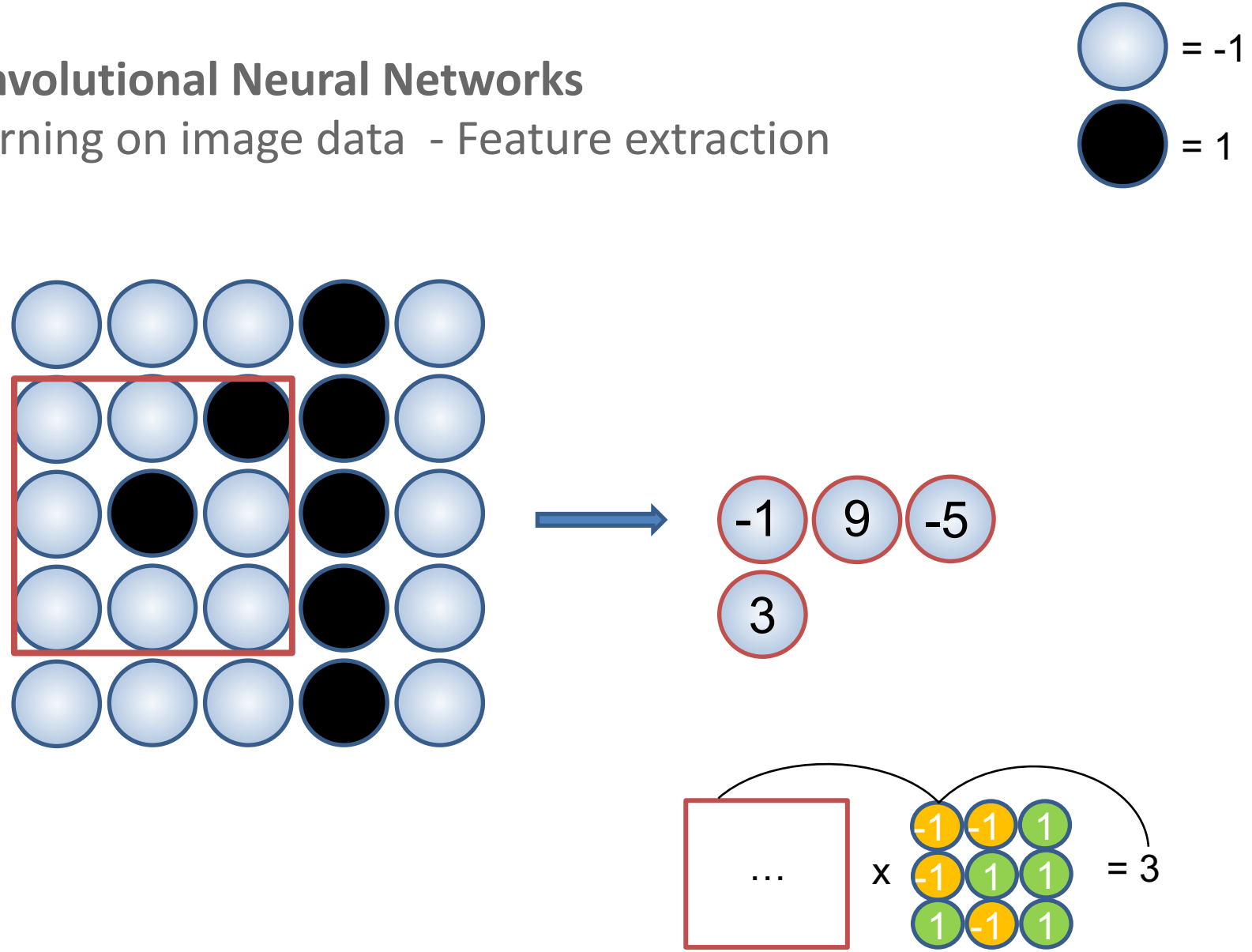


Very low score !
Many mismatches



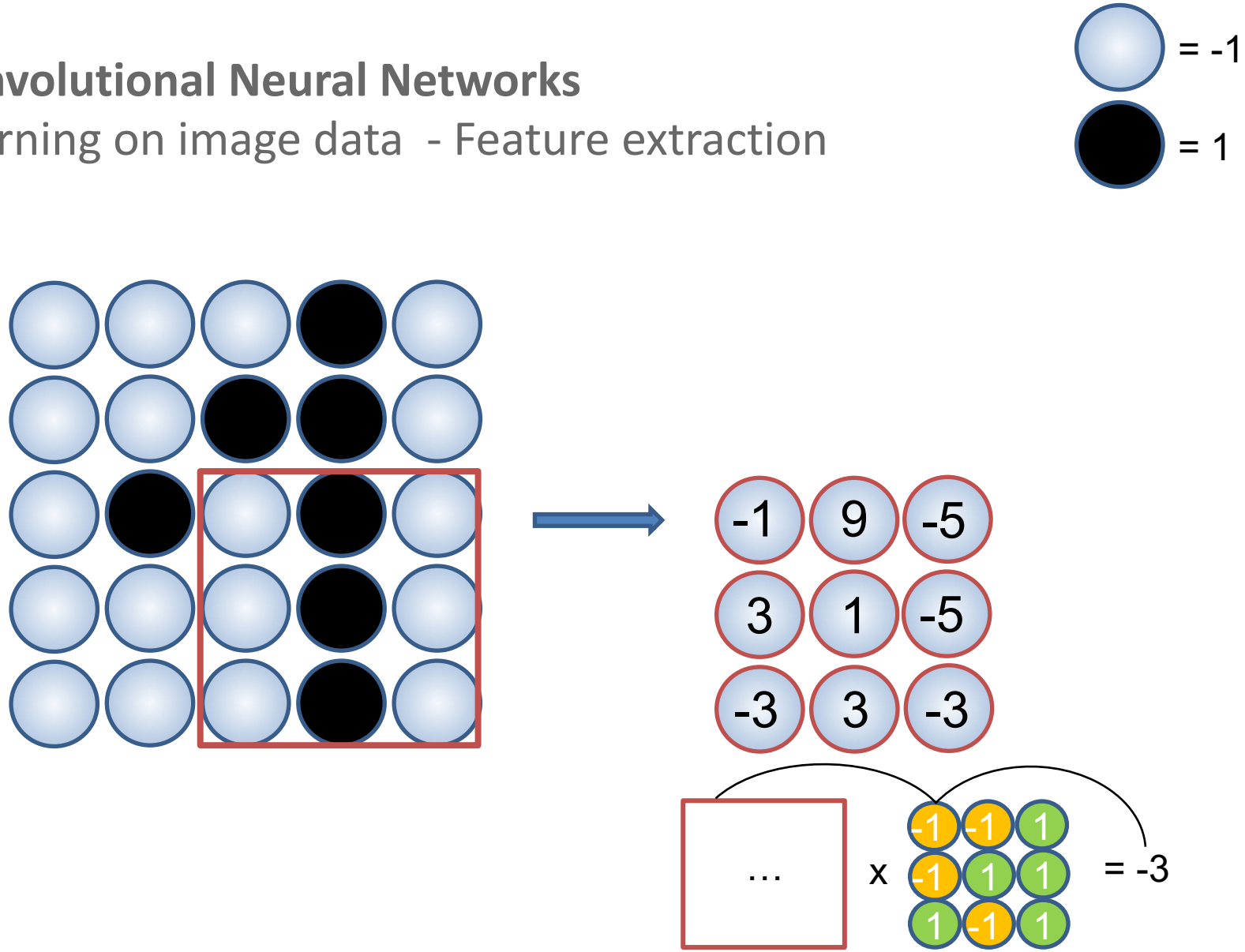
Convolutional Neural Networks

Learning on image data - Feature extraction



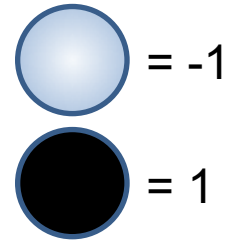
Convolutional Neural Networks

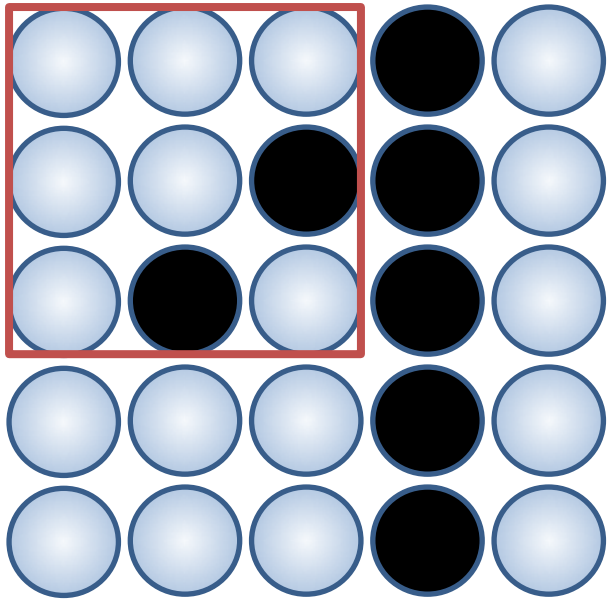
Learning on image data - Feature extraction



Convolutional Neural Networks

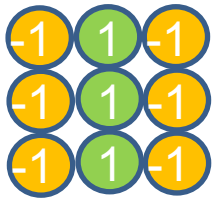
Learning on image data - Feature extraction

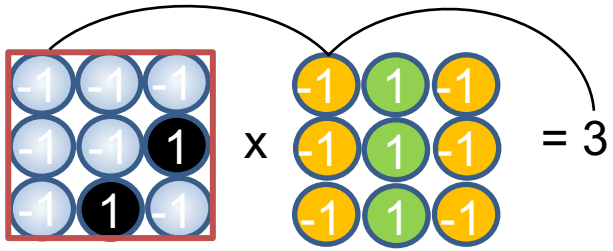




- Different feature will extract different features
- **Spatially share** parameters of each filter

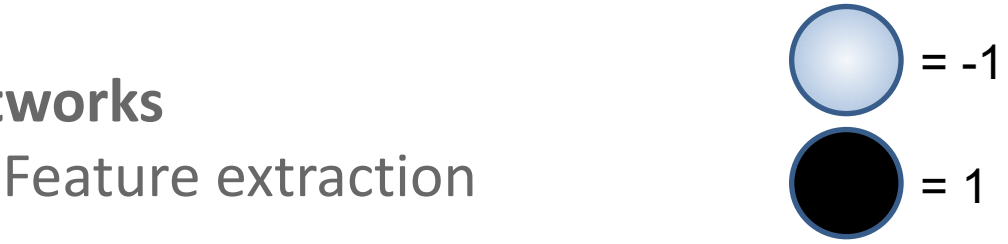
Next filter $\mathbf{fi}_2$ =

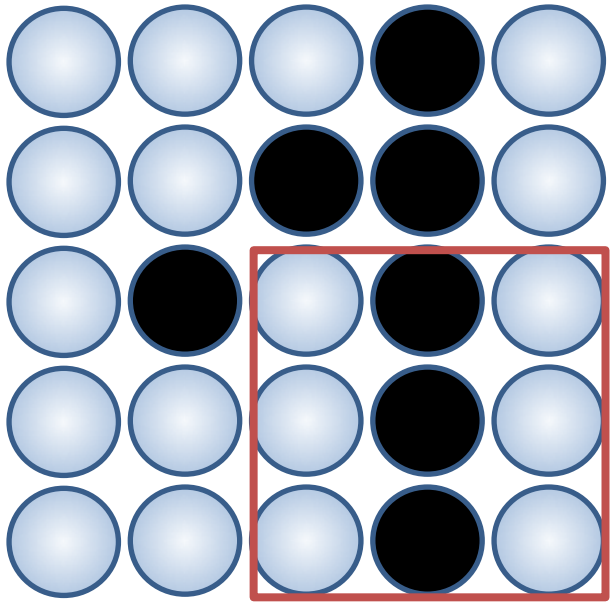




Convolutional Neural Networks

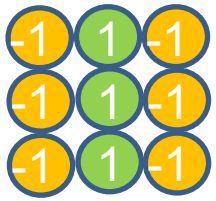
Learning on image data - Feature extraction

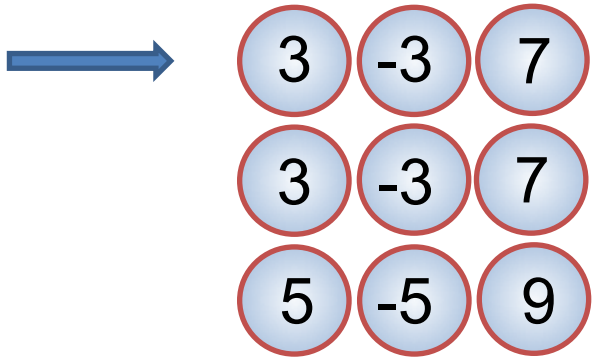


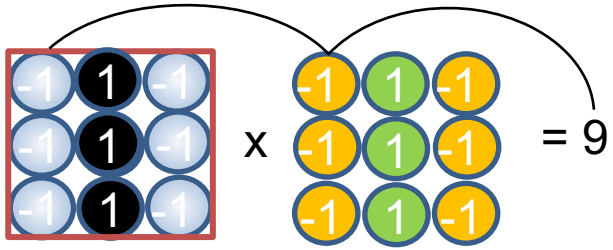


- Different feature will extract different features
- **Spatially share** parameters of each filter

Next filter **fi_2** =

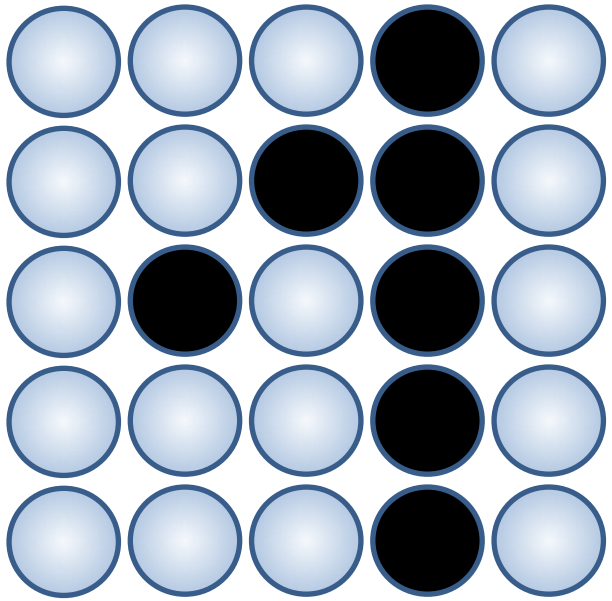






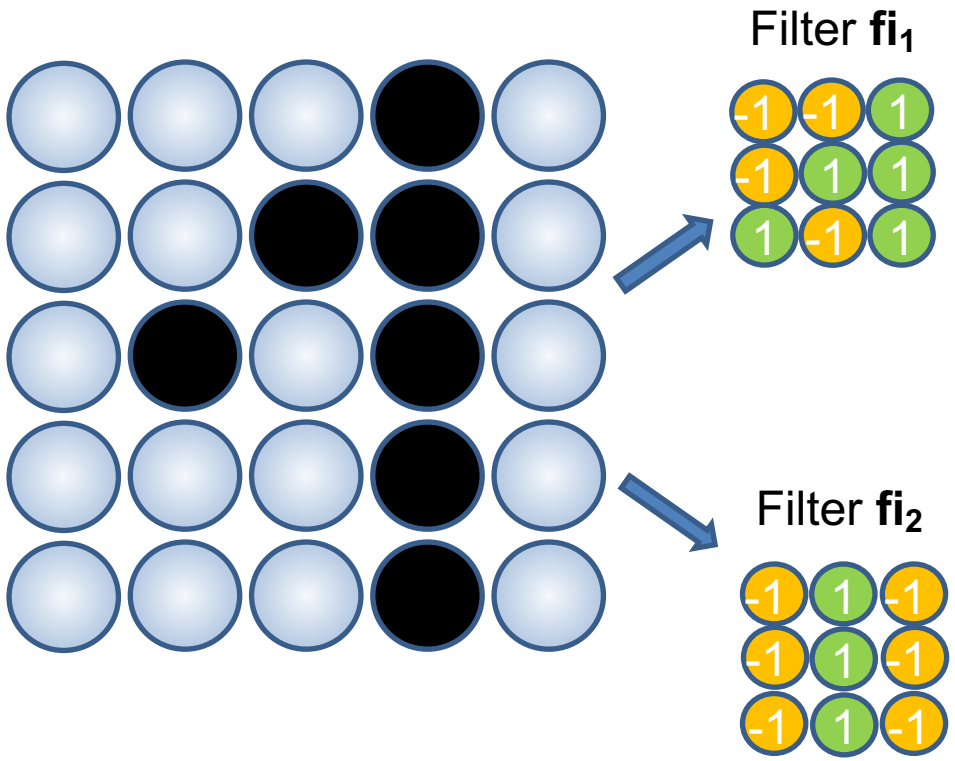
Convolutional Neural Networks

Learning on image data - Feature extraction – filter comparison



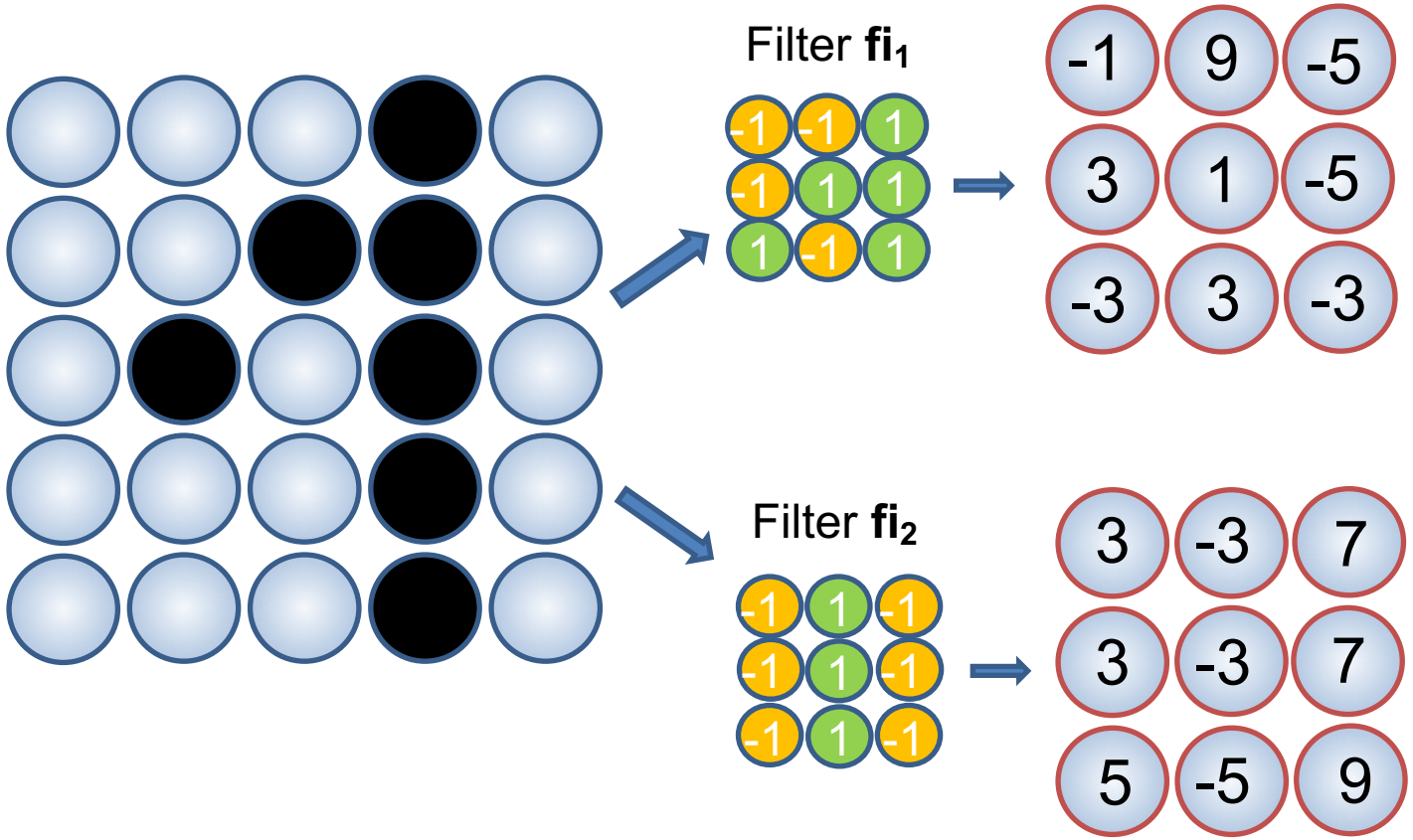
Convolutional Neural Networks

Learning on image data - Feature extraction – filter comparison



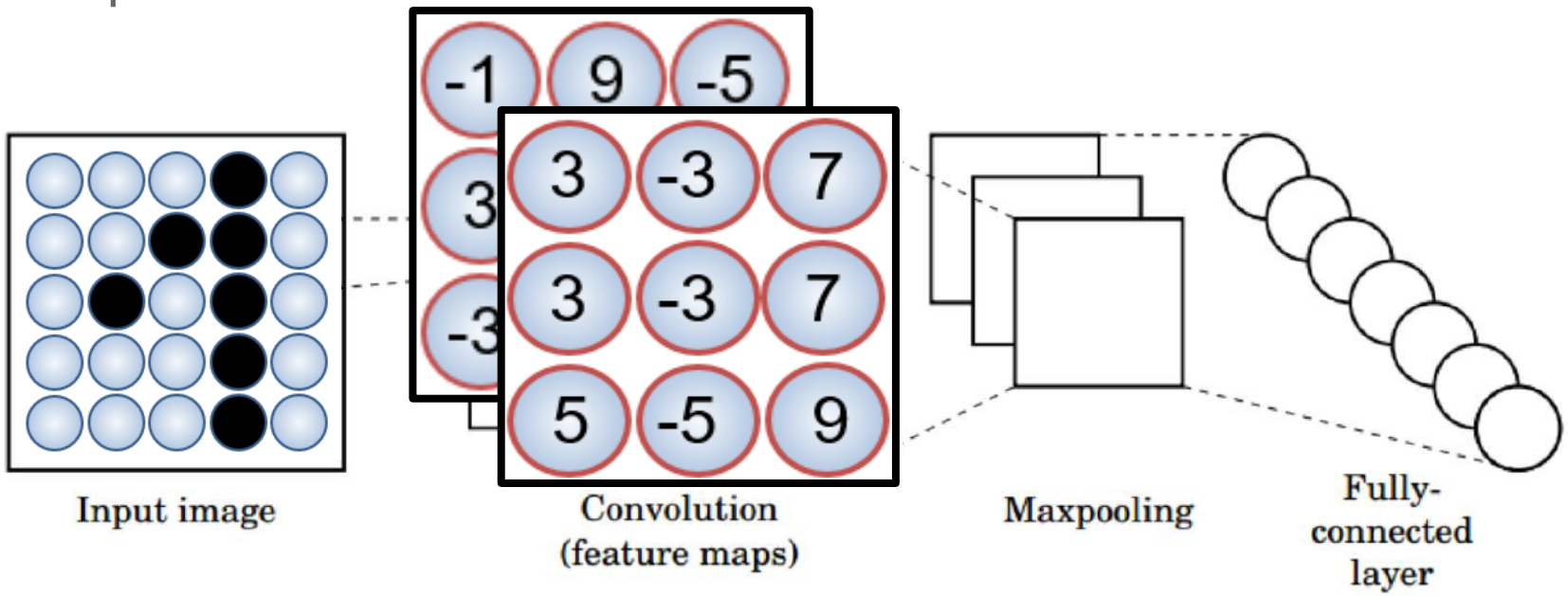
Convolutional Neural Networks

Learning on image data - Feature extraction – filter comparison



Convolutional Neural Networks

Building a CNN - Example

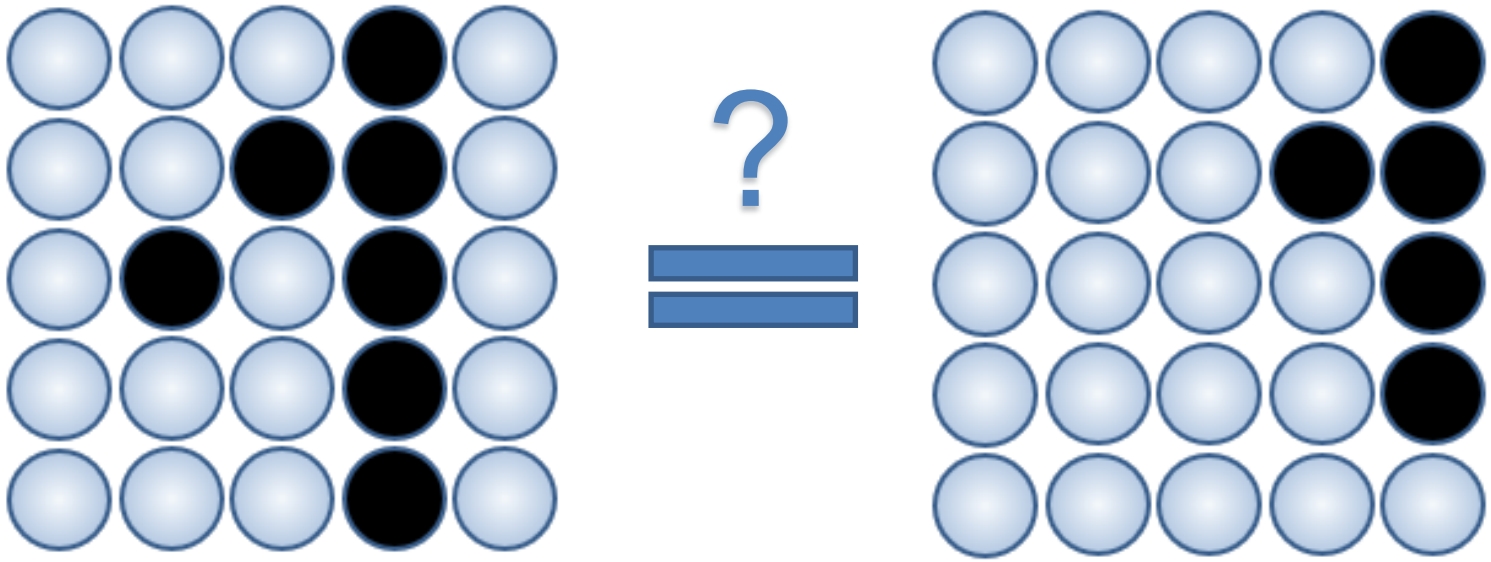


1. **Convolution**: Apply filters with learned weights to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

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

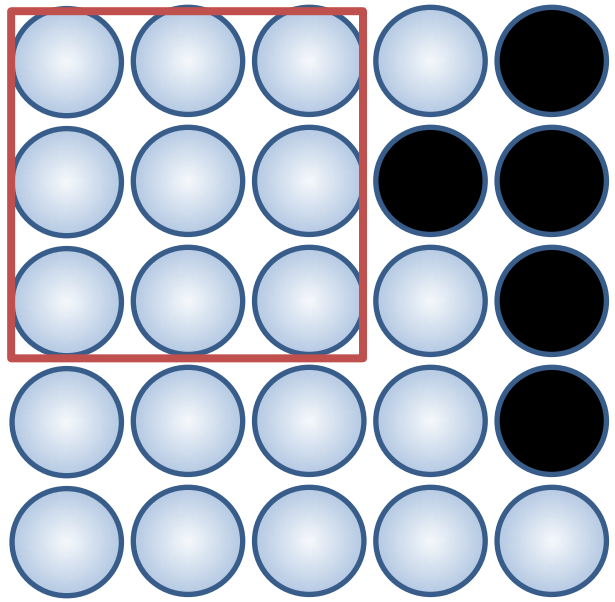
Convolutional Neural Networks

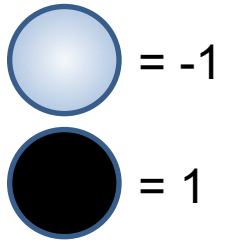
Filters to detect features



Convolutional Neural Networks

Filters to detect features

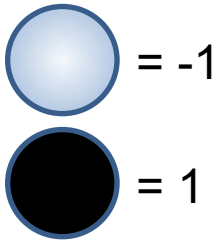


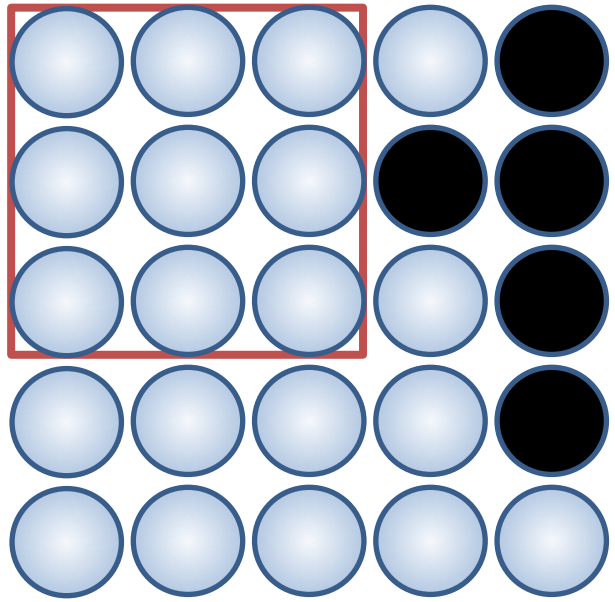


The usage of filters in detecting features allow good identification even in the event of **displacement, shrinkage, rotation or deformation**.

Convolutional Neural Networks

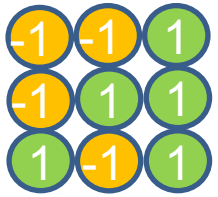
Filters to detect features

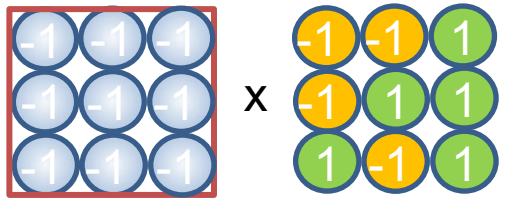




The usage of filters in detecting features allow good identification even in the event of **displacement, shrinkage, rotation or deformation**.

e.g. Filter $\mathbf{fi_1}$ =

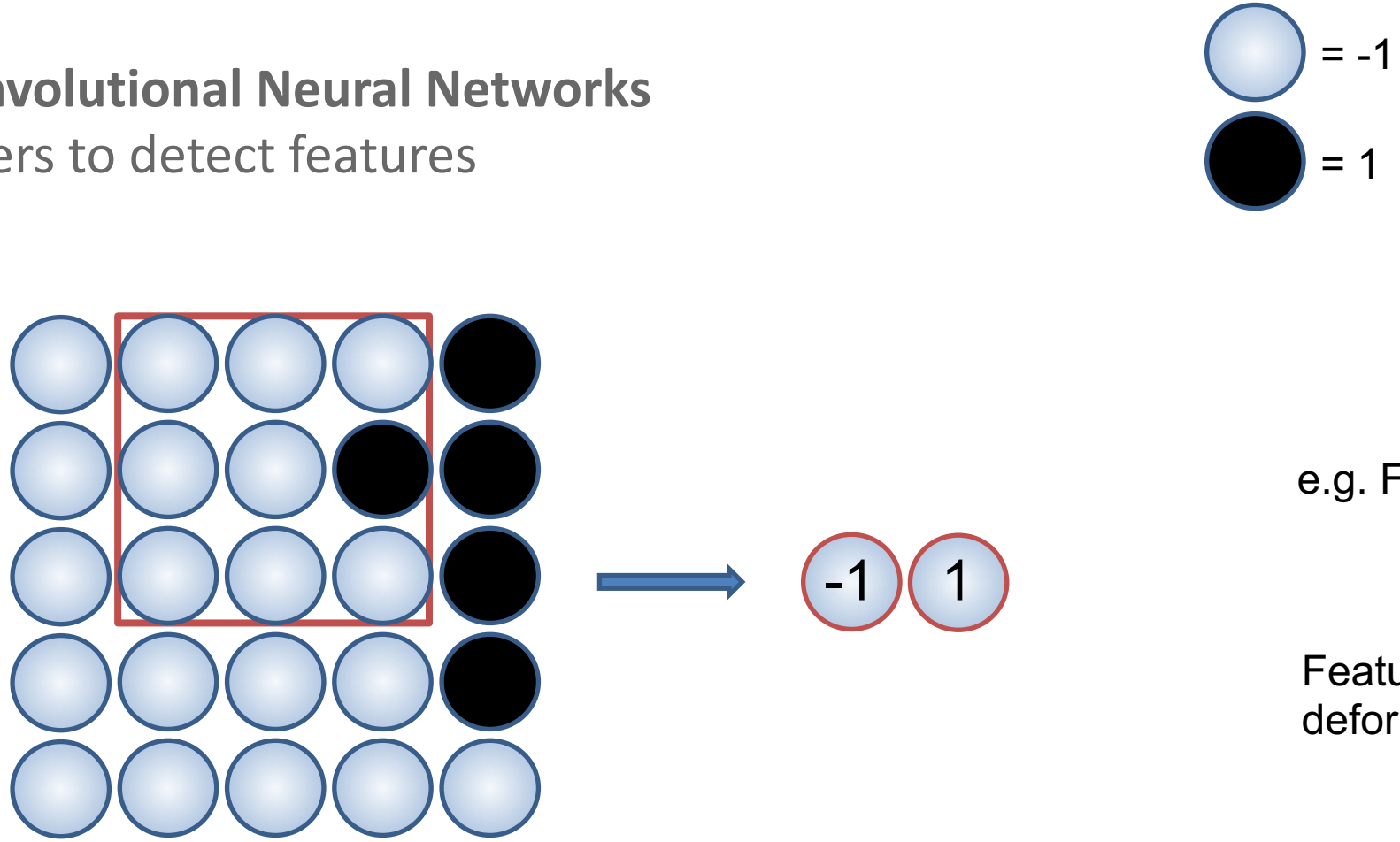




= …

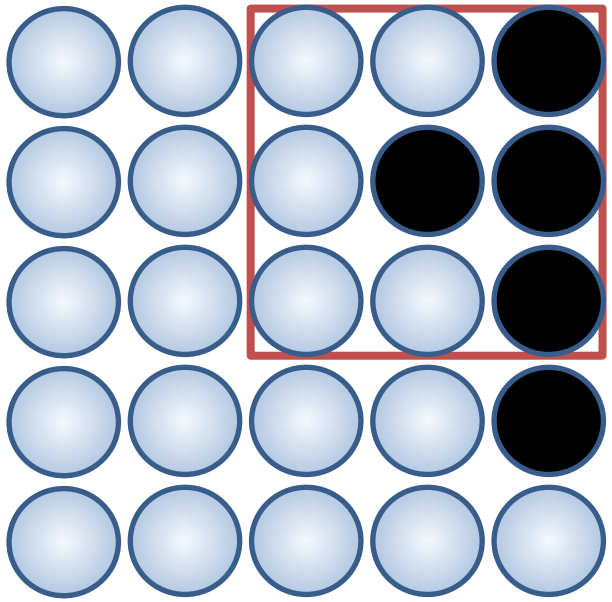
Convolutional Neural Networks

Filters to detect features



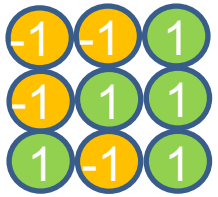
Convolutional Neural Networks

Filters to detect features

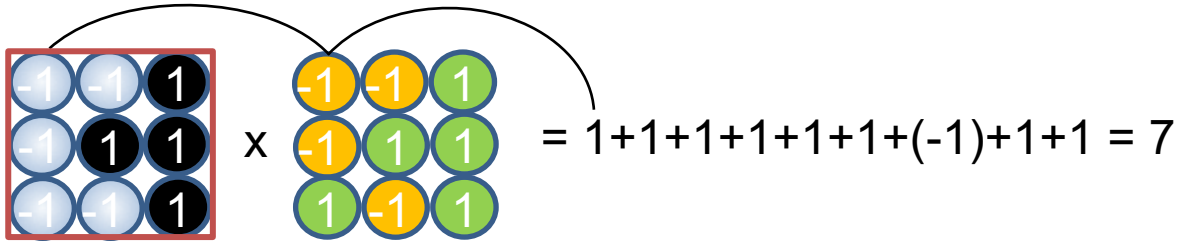


Not a full match
but still a high score

e.g. Filter $\mathbf{fi_1}$ =

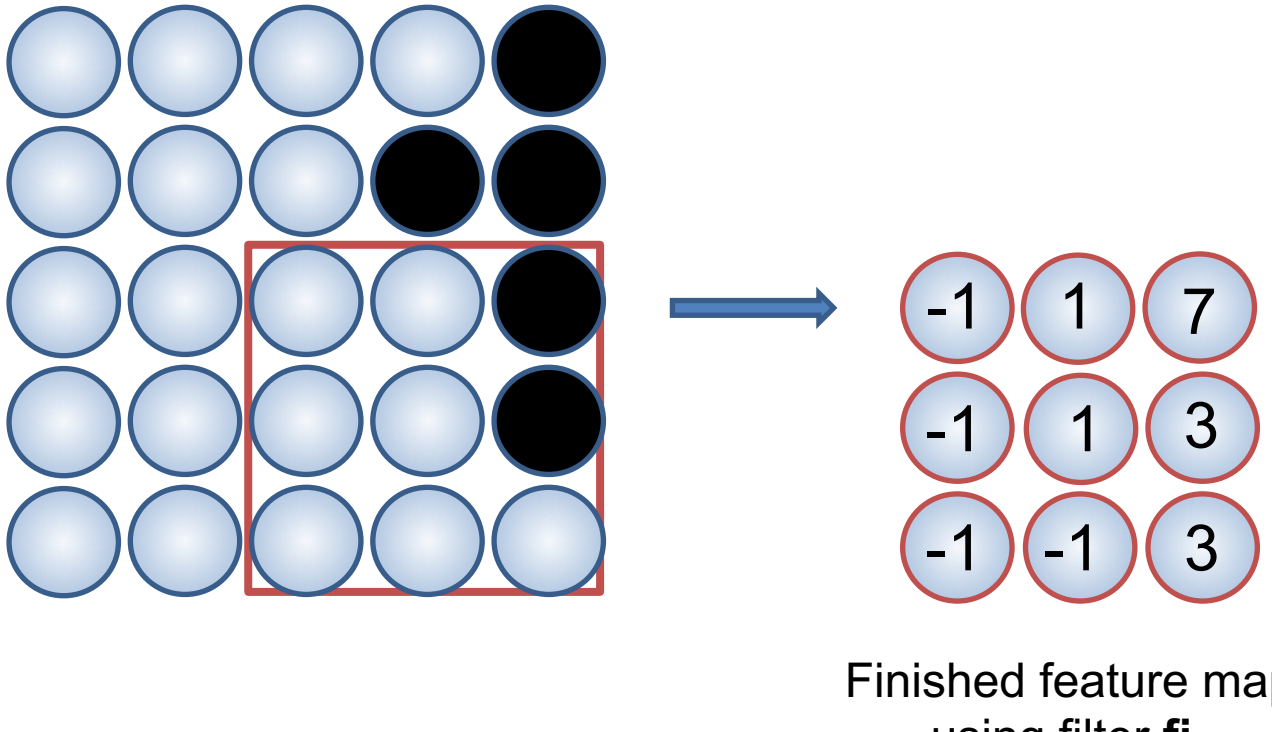


Feature/filter fi_1 still matches the deformed image well
Remember! **A full match would be 9**



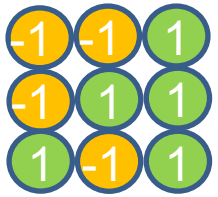
Convolutional Neural Networks

Filters to detect features



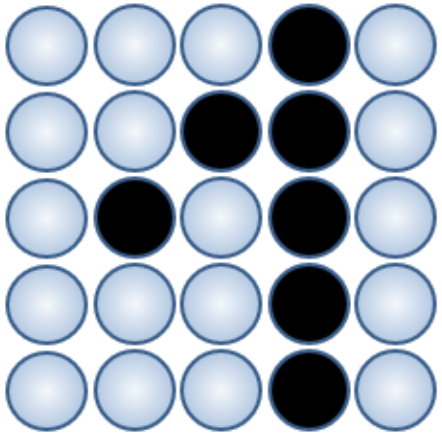
Finished feature map
using filter $\mathbf{fi}_1$

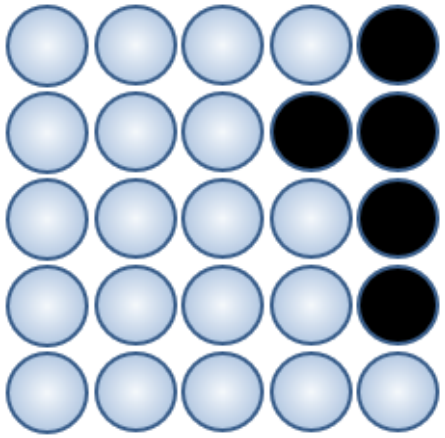
e.g. Filter $\mathbf{fi}_1$ =



Convolutional Neural Networks

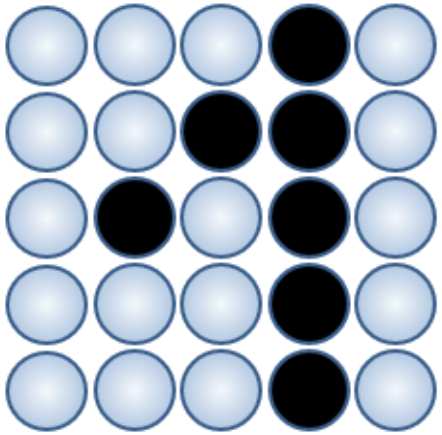
Filters to detect features – filter comparison on diff. images





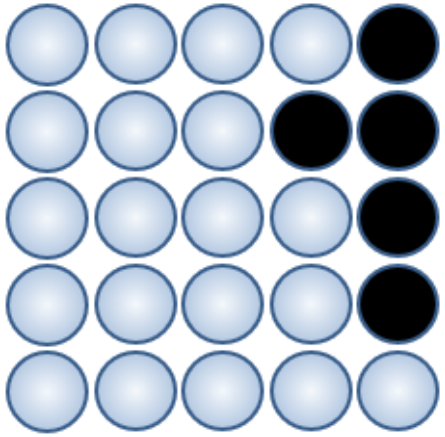
Convolutional Neural Networks

Filters to detect features – filter comparison on diff. images



Filter $\mathbf{fi}_1$

-1	-1	1
-1	1	1
1	-1	1

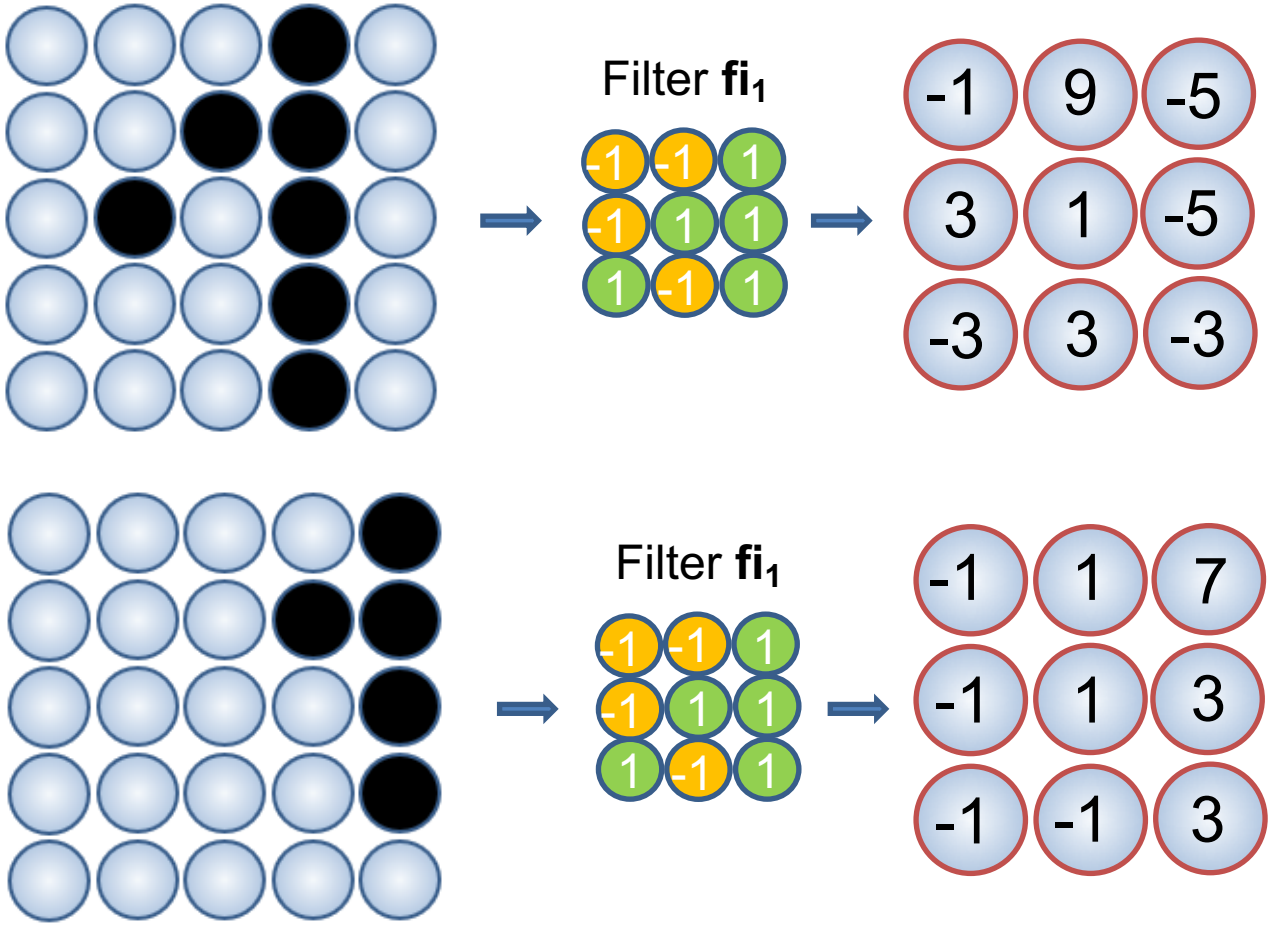


Filter $\mathbf{fi}_1$

-1	-1	1
-1	1	1
1	-1	1

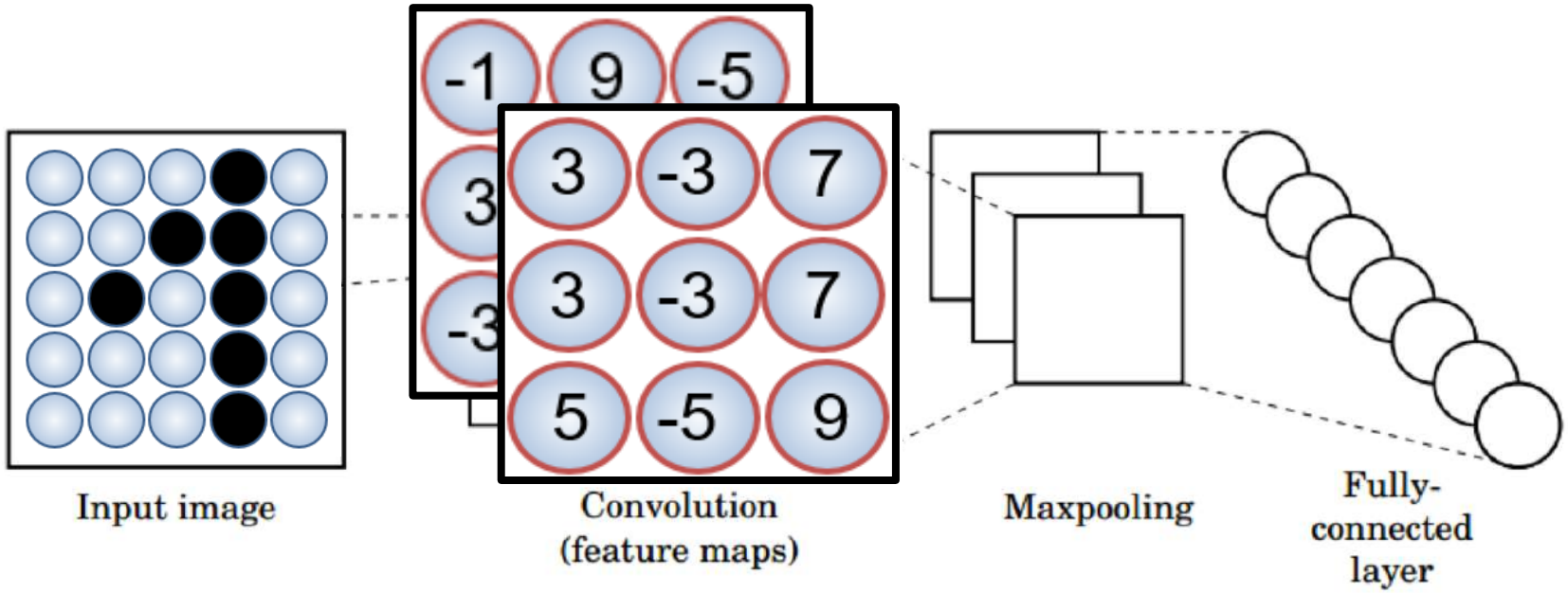
Convolutional Neural Networks

Filters to detect features – filter comparison on diff. images



Convolutional Neural Networks

Pooling

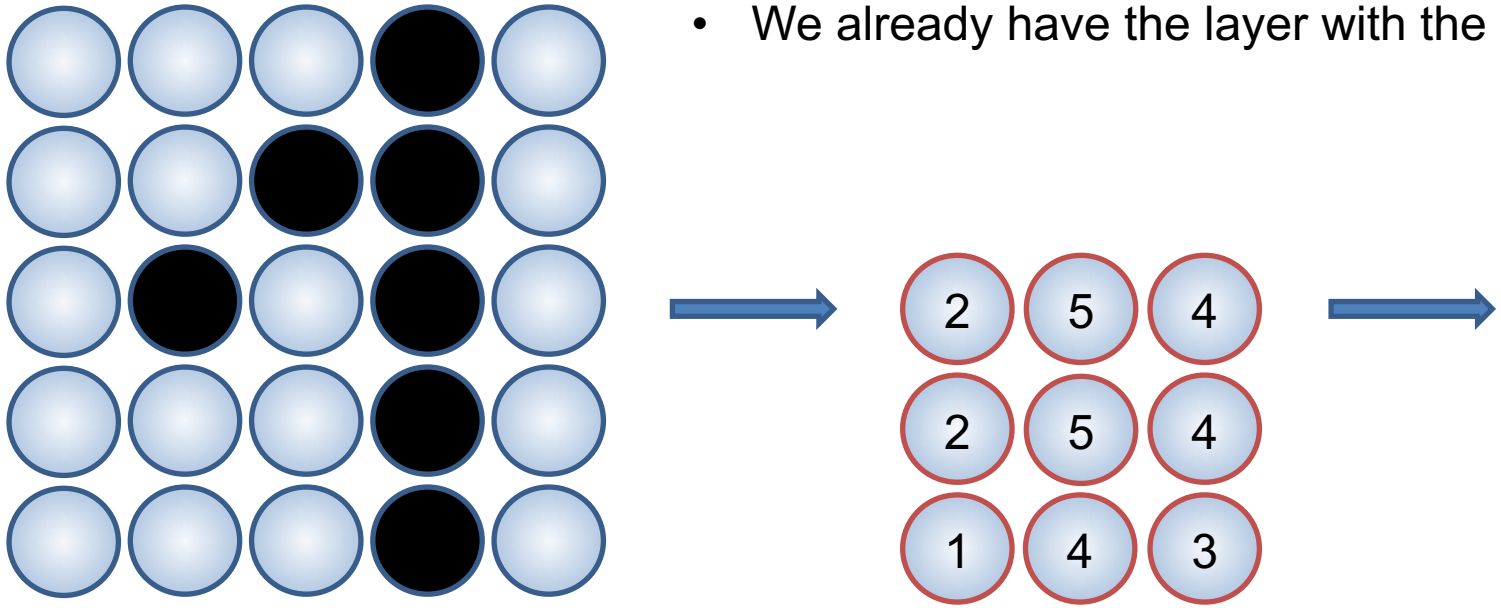


1. **Convolution**: Apply filters with learned weights to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

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

Convolutional Neural Networks

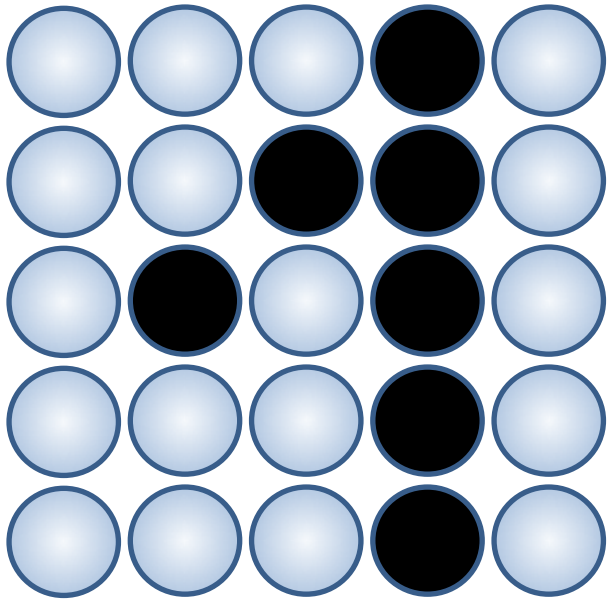
Max Pool - Example



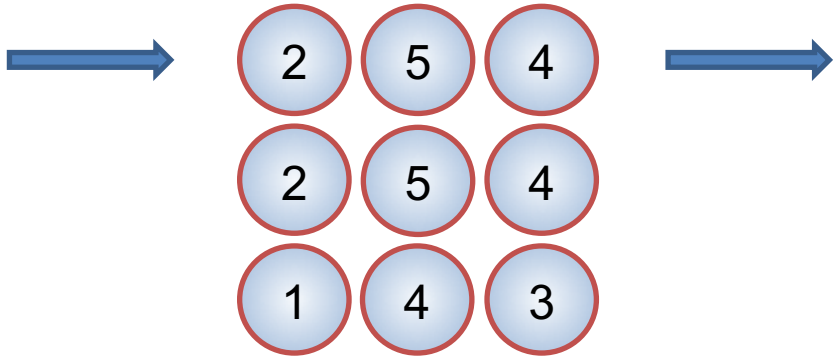
- We already have the layer with the 9 neurons for the 5x5 image

Convolutional Neural Networks

Max Pool - Example

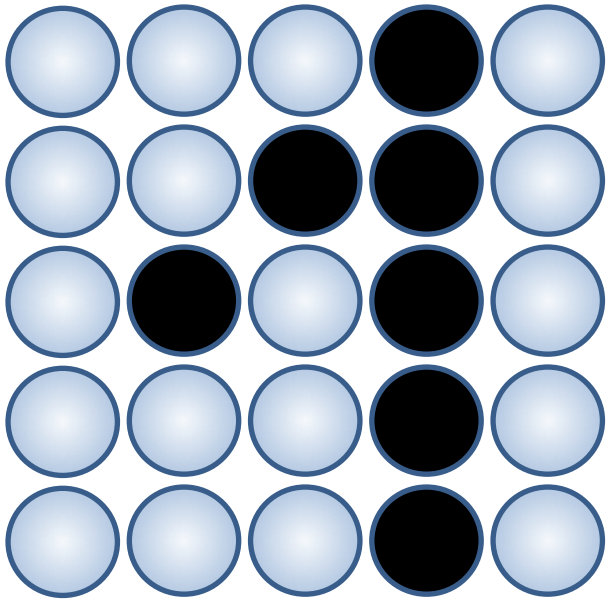


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance

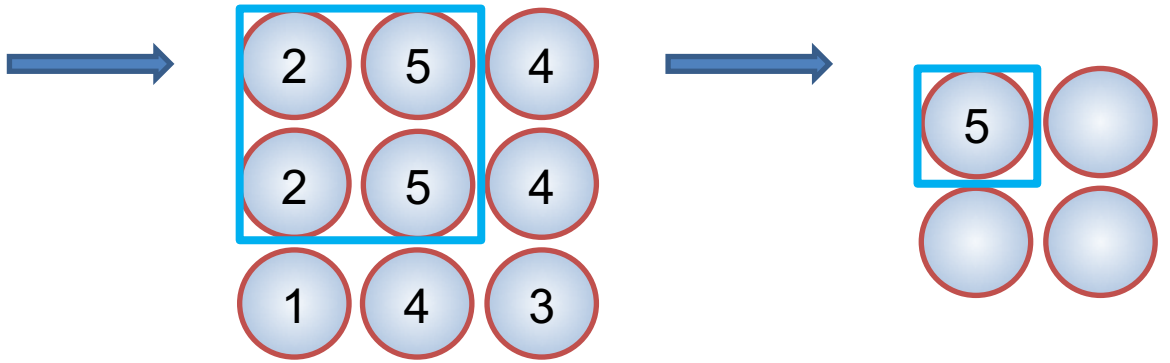


Convolutional Neural Networks

Max Pool - Example

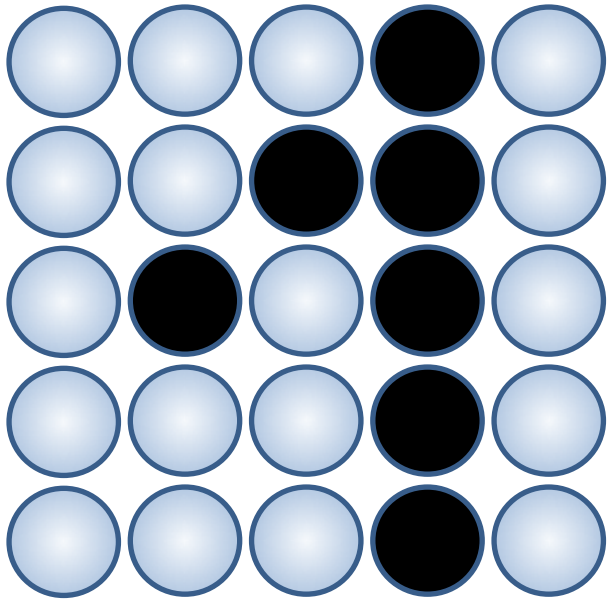


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1

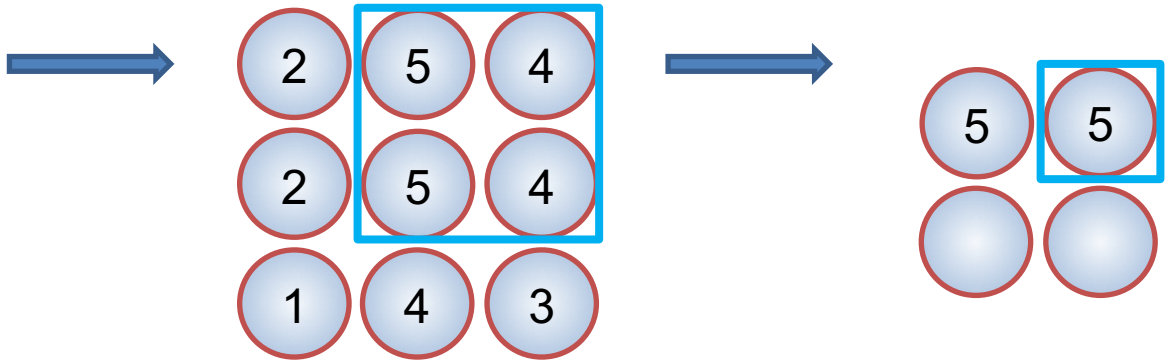


Convolutional Neural Networks

Max Pool - Example

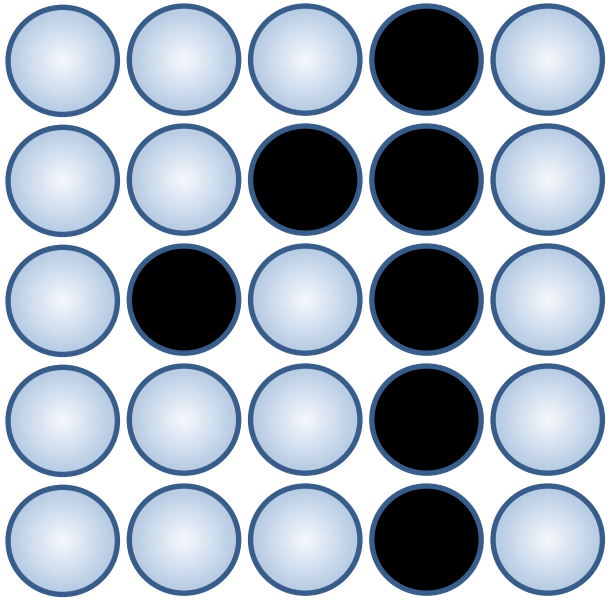


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1

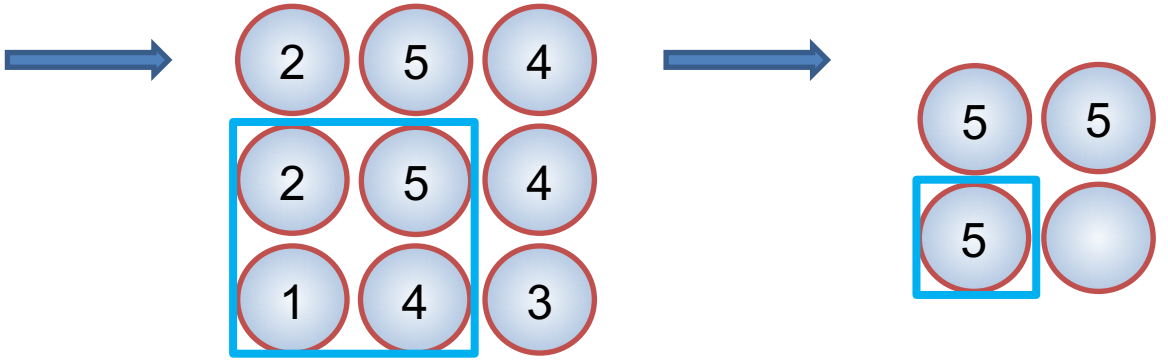


Convolutional Neural Networks

Max Pool - Example

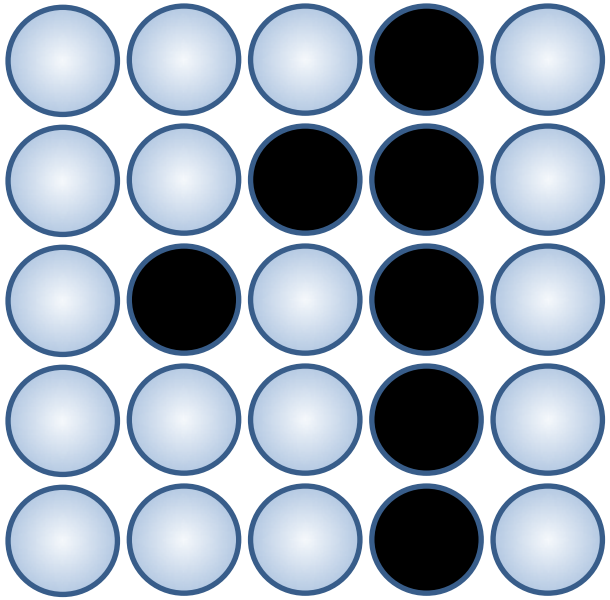


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1

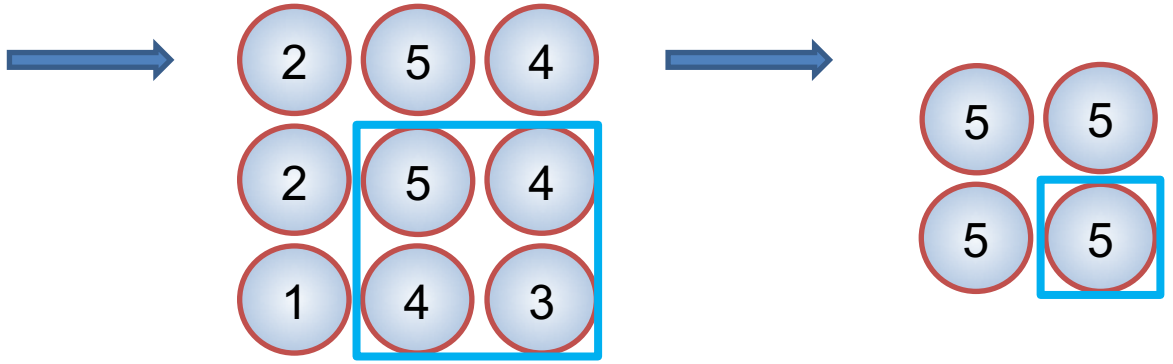


Convolutional Neural Networks

Max Pool - Example

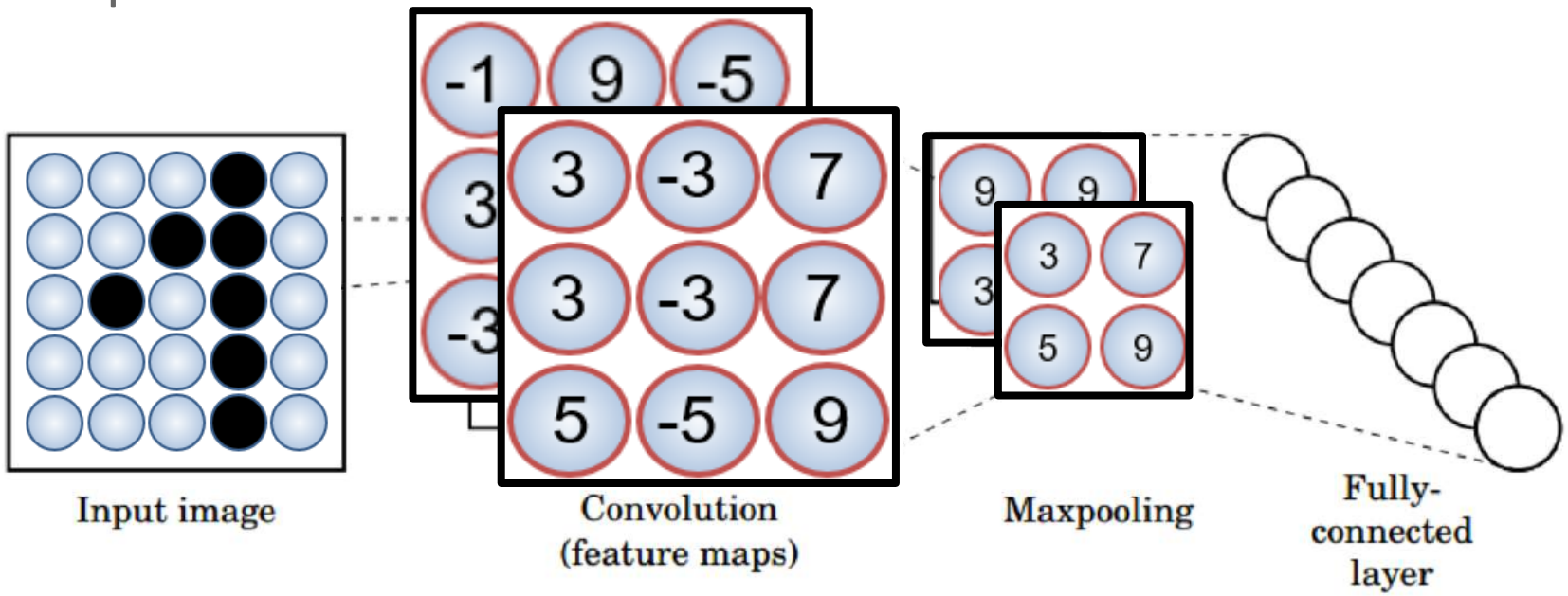


- We already have the layer with the 9 neurons for the 5x5 image
- Now we want to reduce dimensionality and spatial invariance
- Using max pool (find max with 2x2 filters and stride 1



Convolutional Neural Networks

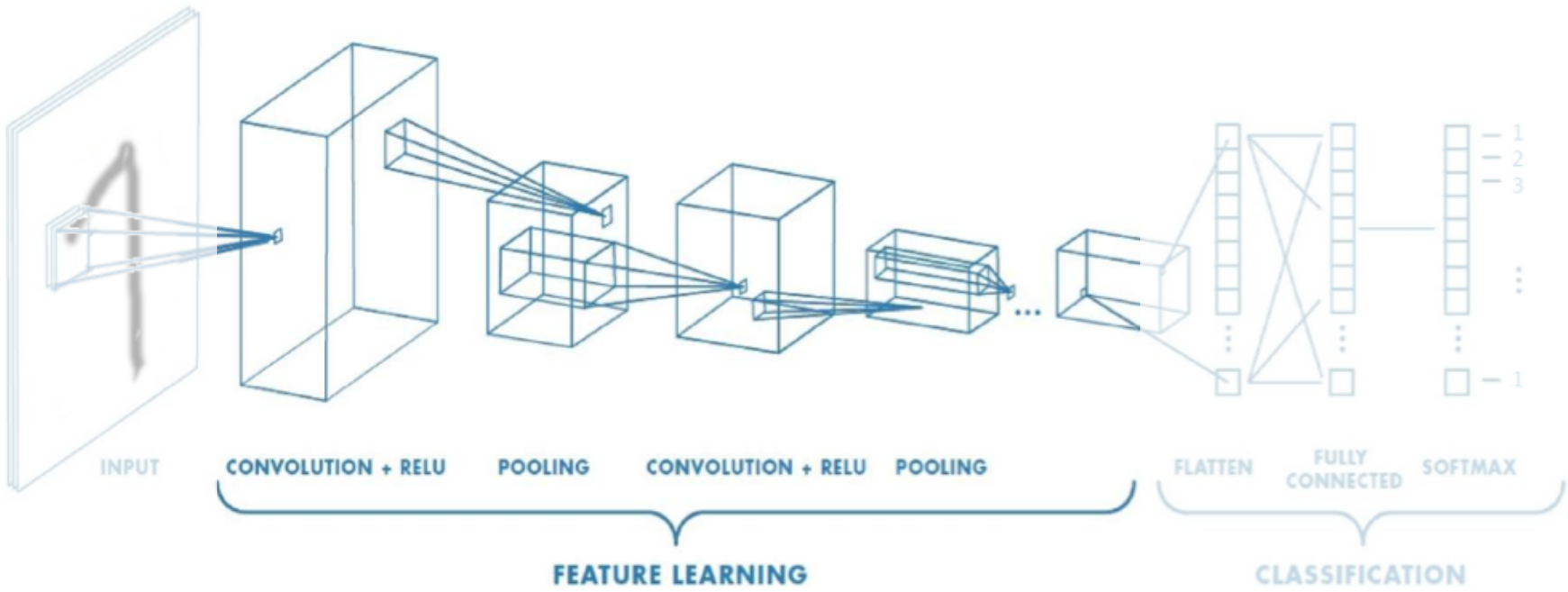
Building a CNN - Example



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

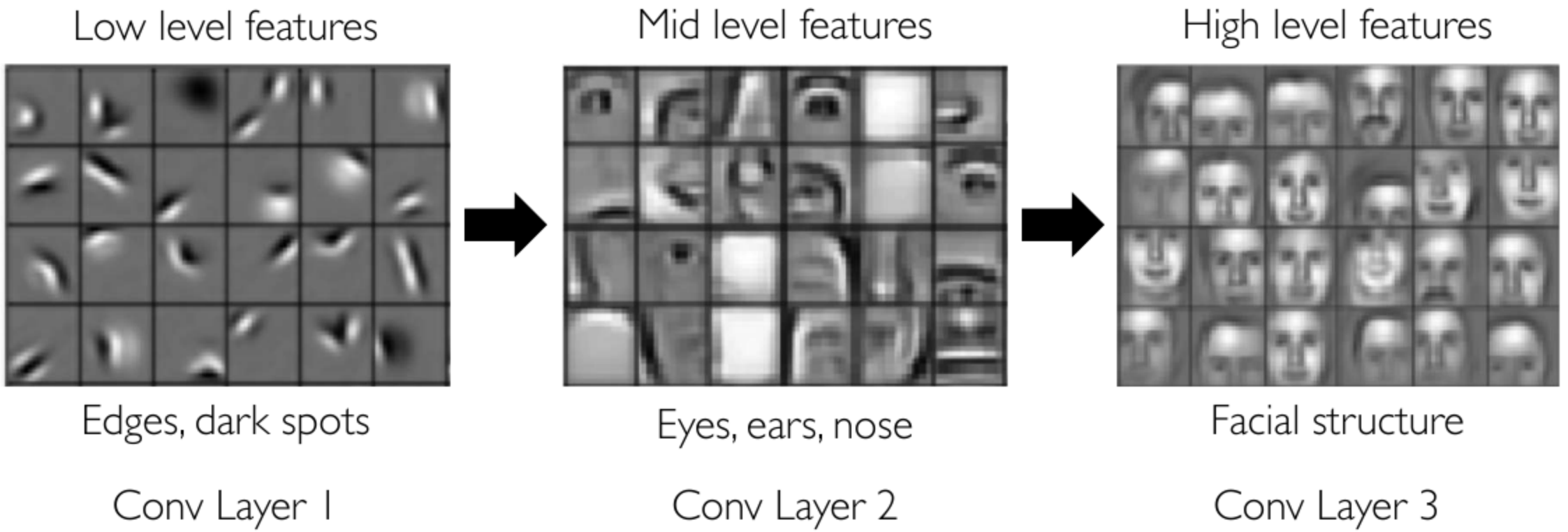
Train model with image data.
Learn weights of filters in convolutional layers.

Representation Learning

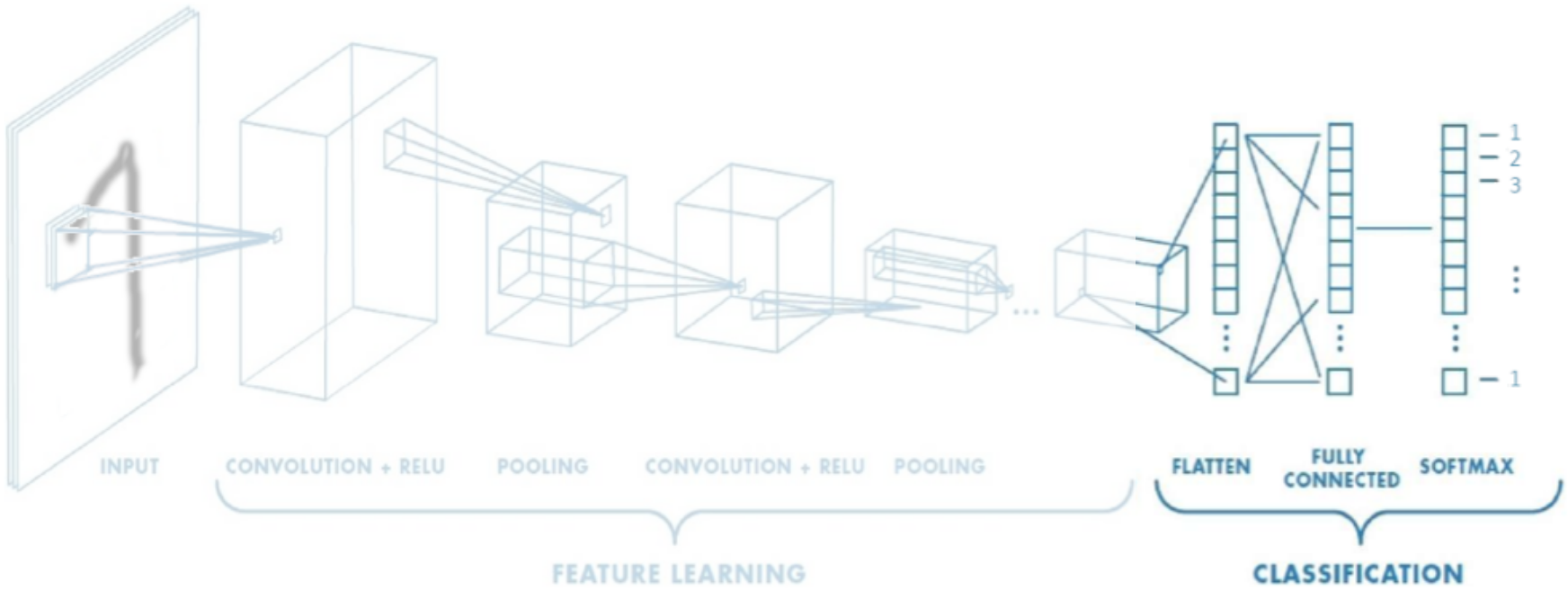


Convolutional Neural Networks

Representation learning



Classification

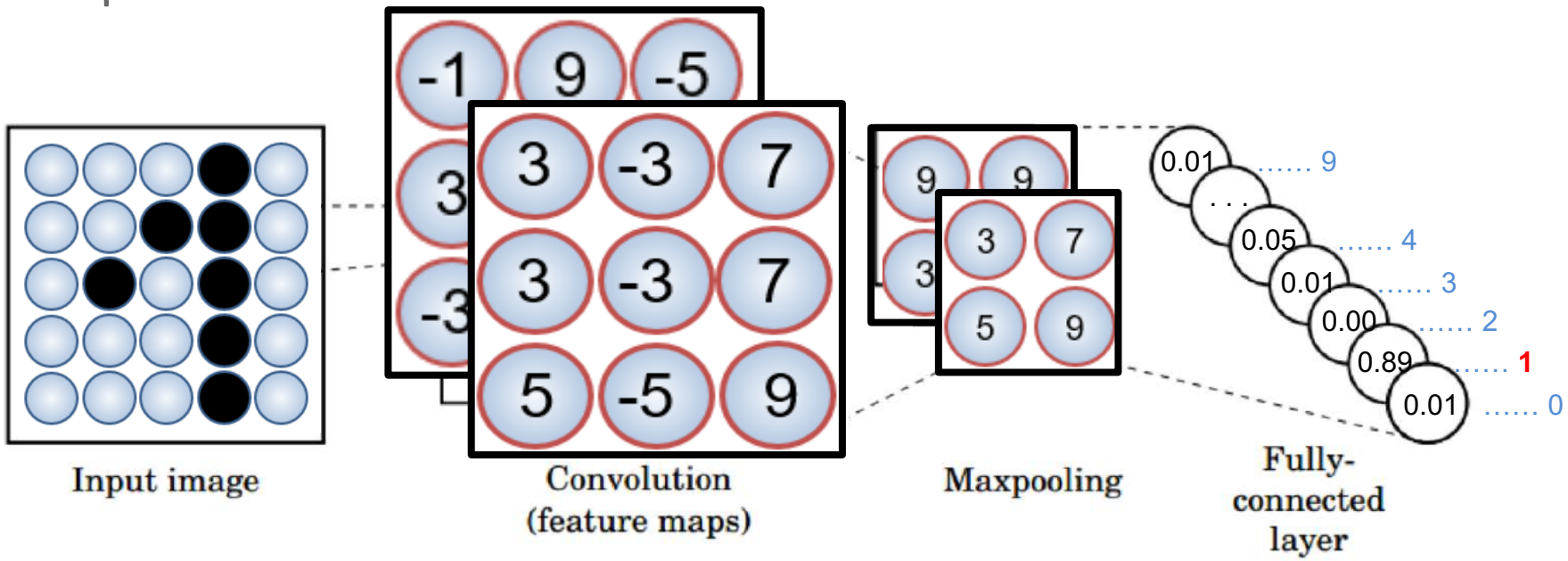


- CONV and POOL layers output high-level features of input
- Fully connected layer uses these features for classifying input image
- Express output as **probability** of image belonging to a particular class

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

Convolutional Neural Networks

Building a CNN - Example

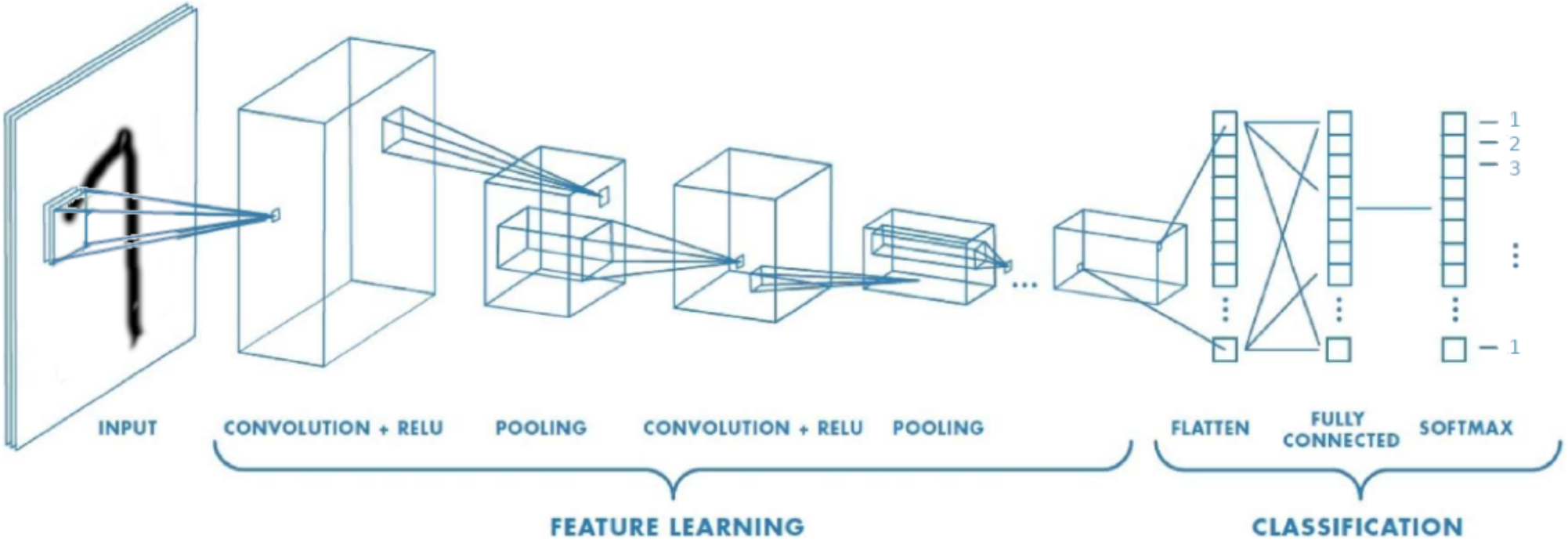


1. **Convolution**: Apply filters with learned weights to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

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

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

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