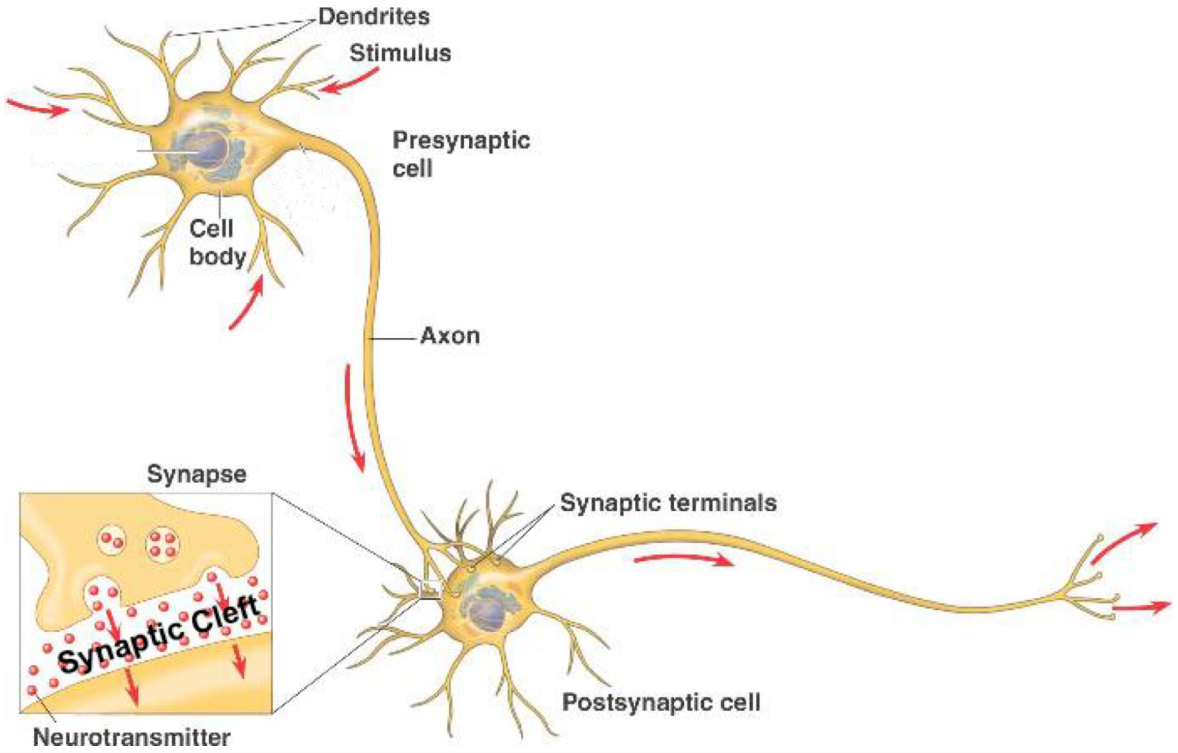
Overview of Machine Learning and Pattern Recognition

Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, ITALY

Neural Networks

- A NN is a machine learning approach inspired by the way in which the brain performs a particular learning task
- Network of simple computational units (neurons) connected by links (synapses)



Neural Networks

- A NN is a machine learning approach inspired by the way in which the brain performs a particular learning task
- Network of simple computational units (neurons) connected by links (synapses)
  - Knowledge about the learning task is given in the form of training examples (training vectors).
  - Inter neuron connection strengths (weights) are used to store the acquired information (the training examples).
  - During the learning process the weights are modified in order to model the particular learning task correctly on the training examples.

NN - Learning

- Supervised Learning
  - Pattern recognition, regression, etc.
  - Labeled training examples (input + desired output)
  - Neural Network models: perceptron, feed-forward, etc.

- Unsupervised Learning
  - Clustering.
  - Unlabeled training examples (different realizations of the input alone)
  - Neural Network models: self organizing maps (SOMs), Hopfield networks, etc.

NN - architectures

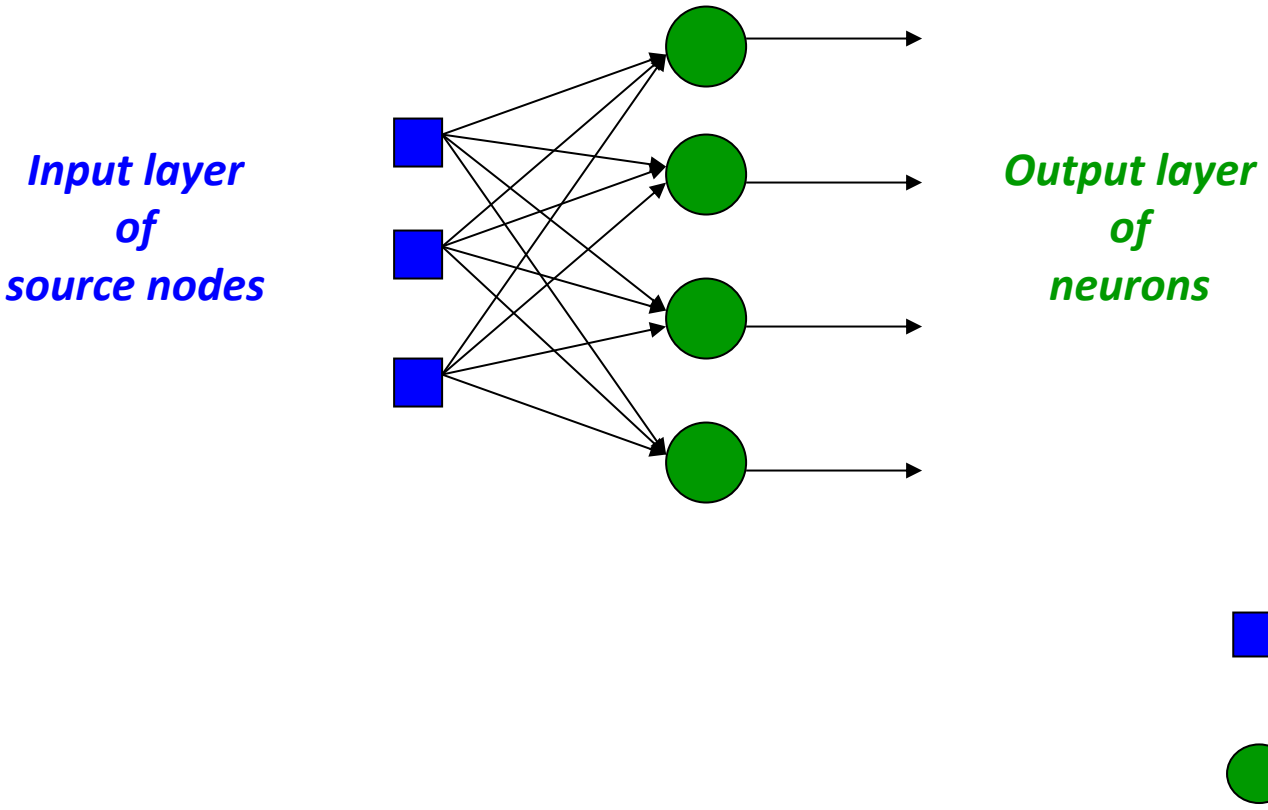
- Three different classes of network architectures
  - single-layer feed-forward
  - multi-layer feed-forward
  - Recurrent

  } neurons are organized in acyclic layers (links have only one direction)

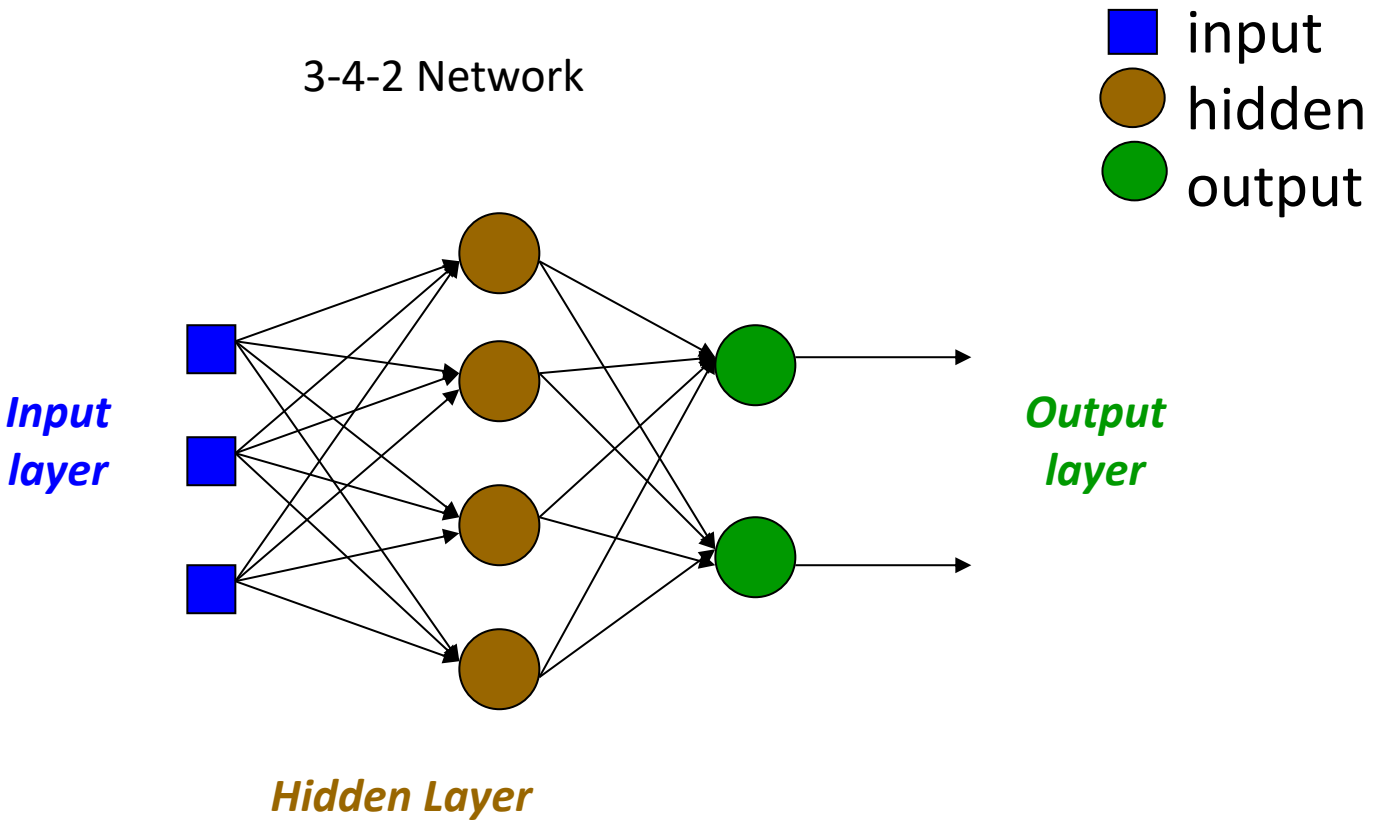
- A standard architecture consists of:
  - Input units
    - represent the input as a fixed-length vector of numbers (user-defined)
  - Hidden units (optional)
    - thresholded weighted sums of the inputs
    - represent intermediate calculations that the network learns
  - Output units
    - represent the output as a fixed length vector of numbers

- The **architecture** of a neural network is linked with the **learning algorithm** used to train

Feed-forward single layer (perceptron)



Feed-forward multi layer



The neuron

- The neuron is the basic information processing unit of a NN. It consists of:

1 A set of synapses or **connecting links**, each link characterized by a numeric weight:

$$W_1, W_2, \ldots, W_m$$

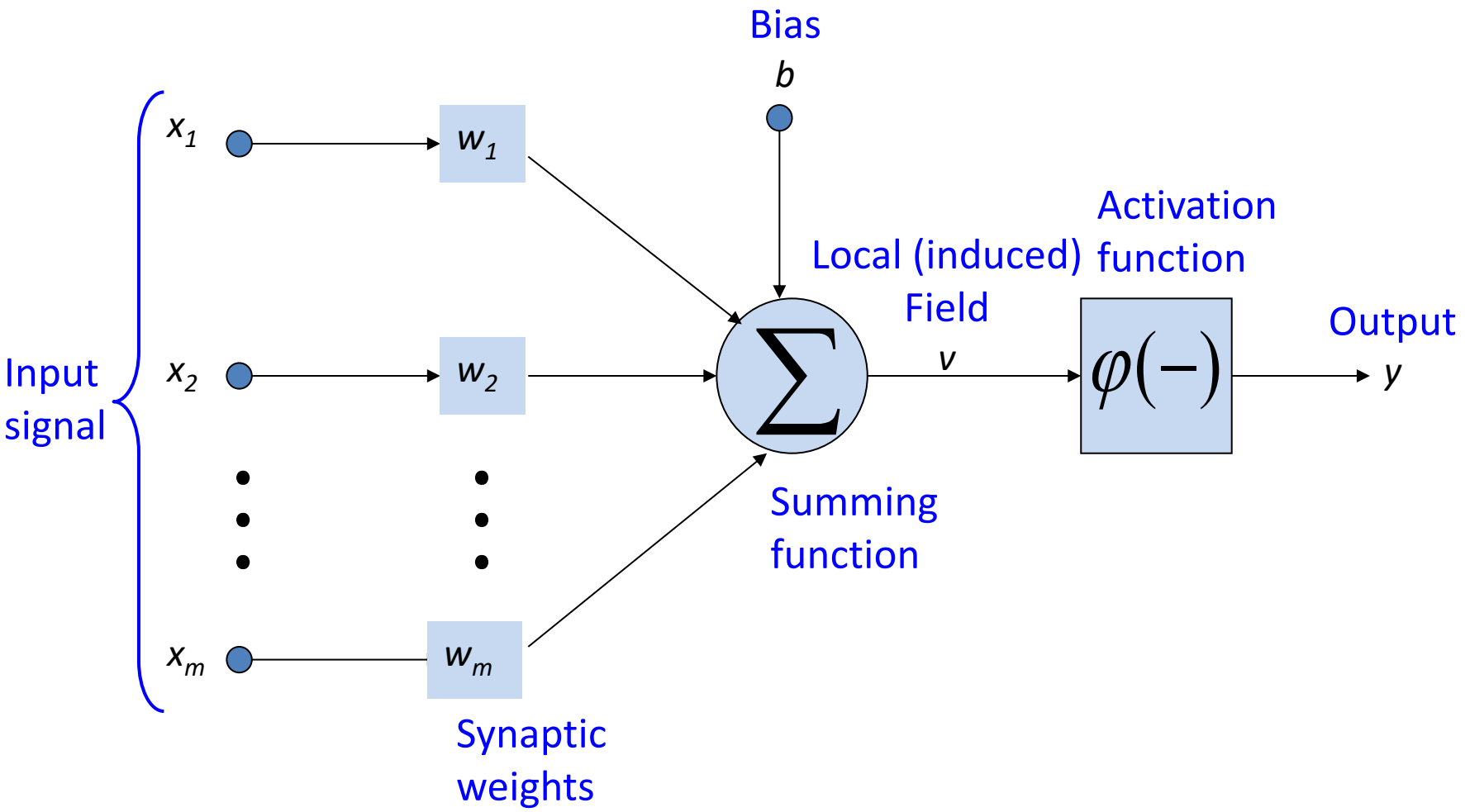
2 An adder function (linear combiner) which computes the **weighted sum of the inputs**:

$$u = \sum_{j=1}^{m} w_j x_j$$

3 **Activation function** (squashing function) φ for limiting the amplitude of the output of the neuron.

$$y = \varphi(u + b)$$

The neuron

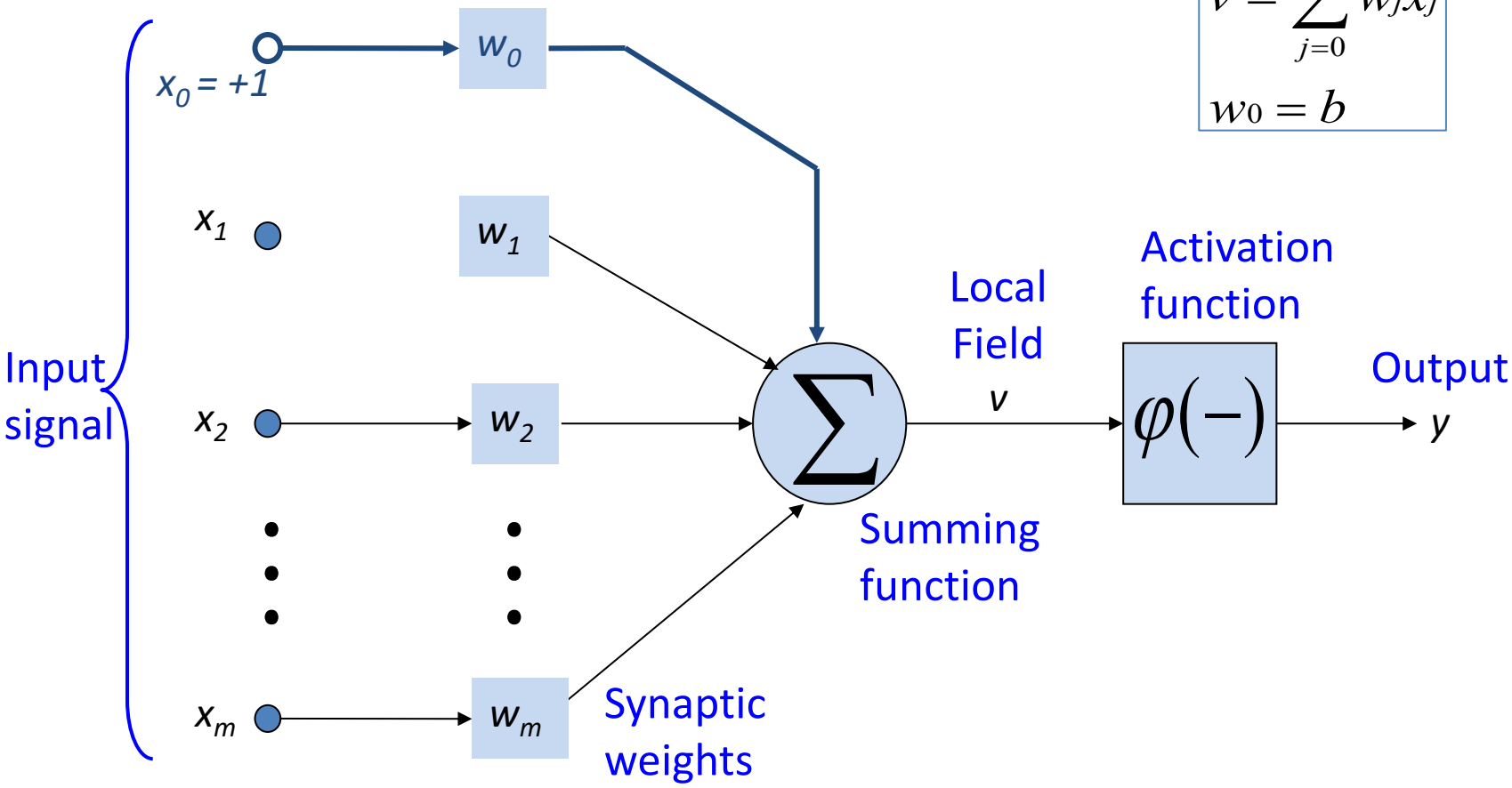

Bias
b
x_1
w_1
Activation function
Local (induced) Field
v
Output
y
Input signal
x_2
w_2
$\sum$
$\varphi(-)$
Summing function
x_m
w_m
Synaptic weights

The neuron

Bias is an external parameter of the neuron. It can be modeled as an extra input.

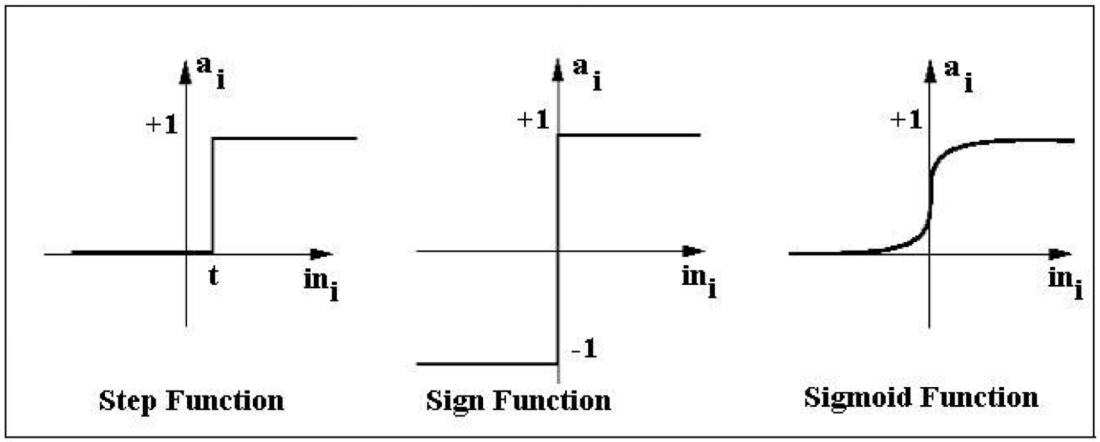
$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$



The neuron

- Typical activation functions:
  1. Step function
  2. Sign function
  3. Sigmoid function



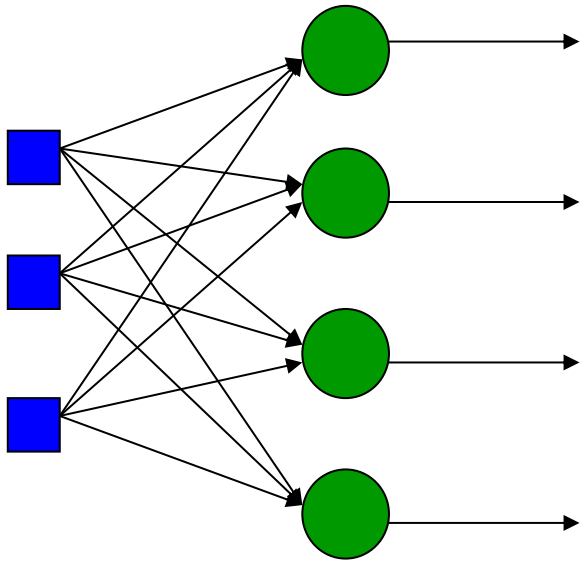
- Emulates the typical response of a biological neuron: the neuron “fires” only if the input signal is above the *threshold potential*

Designing a Neural Network

- Various types of **neurons**
- Various network **architectures**
- Various learning **algorithms**
- Various **applications**

Feed-forward single layer NN

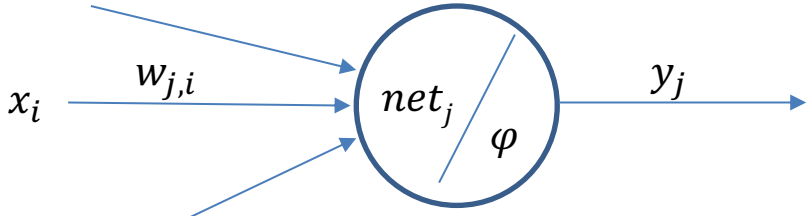
- The simplest architecture is the **perceptrons network**
  - No hidden units: synonym for a single-layer, feed-forward network
  - It can be used for multi-class problems: each neuron learns to recognize one particular class (i.e., output 1 if the input is in that class, and 0 otherwise)
  - It is a linear classifier: it can only classify **linearly separable instances**



Perceptrons network

Perceptron: learning rule

- The teacher (or target) specifies the **desired output** for a given input
- Network calculates the output based on its current weights (random initialization). Then it iteratively changes its weights in proportion to the **error between the desired output & calculated output:**



For a neuron j:

$$w_{j,i}(t+1) = w_{i,j}(t) + \Delta w_{i,j}$$

$$\Delta w_{j,i} = \eta * [t_j - y_j] * \varphi'(net_j) * x_i ,$$ where:

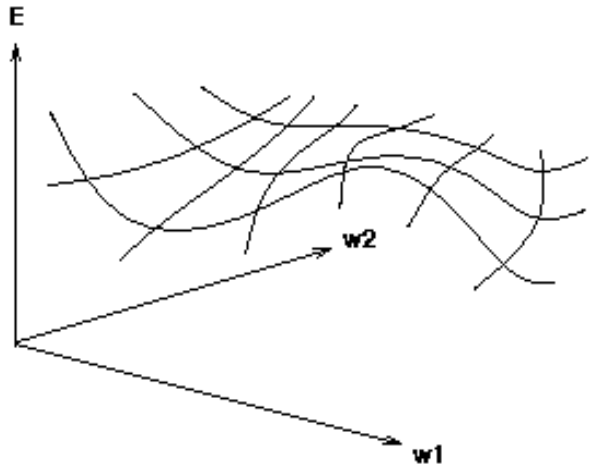
- η is the **learning rate**;
- $t_j - y_j$ is the **error term**
- net_j is the weighted sum of the inputs
- φ is the activation function
- x_i is the i-th input component

Delta rule

Gradient descent

- Delta rule is a derivation of a gradient descent optimization algorithm, attempting to **minimize the total error in the output**: $E = \sum_j \frac{1}{2}(t_j - y_j)^2$
- The aim is to obtain adjust weights in order to minimize E. The fastest procedure is to compute per each neuron:

  $Grad_E$ = [dE/dw1, dE/dw2, . . ., dE/dwn]
- Change i-th weight by
  - $\Delta \mathbf{w_i} = -\boldsymbol{\eta} * dE/dwi$

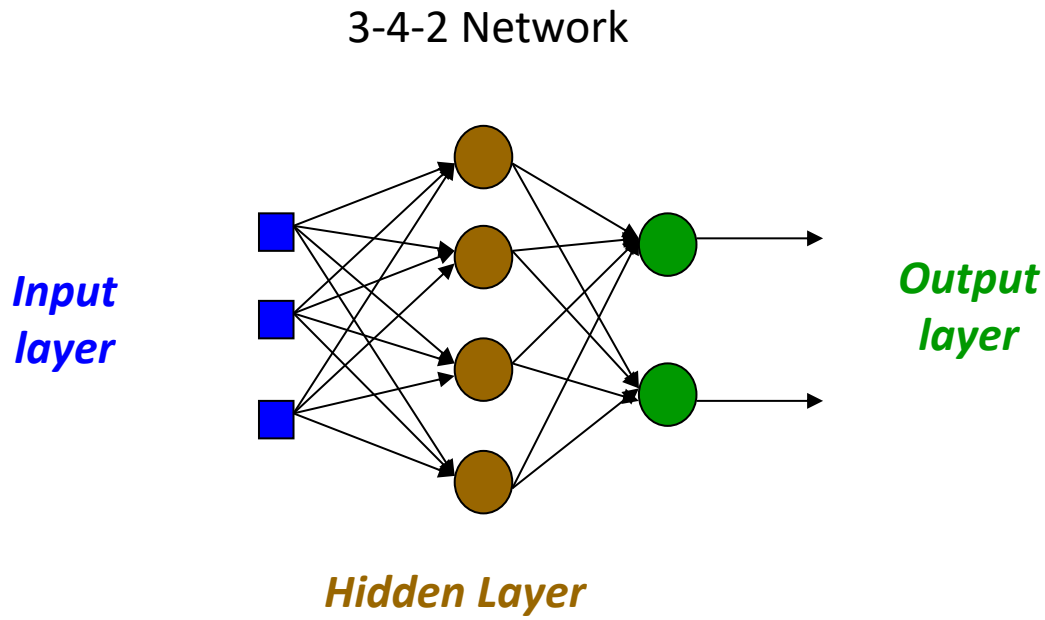


Error as function of weights in multidimensional space

Perceptron convergence theorem

- Provided that the classes are linearly separable, the final weights of a perceptron's network can be obtained in a finite number of steps and independent of initialization.
- However, a single layer perceptron **can only learn linearly separable classes**

Feed-forward multi layer NN



- Layer 0 is input nodes, Layers 1 to N-1 are hidden nodes, Layer N is output nodes
- All nodes at any layer k are connected to all nodes at layer k+1
- There are no cycles

- Can compute functions with convex regions
- Each hidden node acts like a perceptron, learning a separating line
- Combination of multiple linear classifiers

Backpropagation

- Can't use Delta Learning Rule
  - Target values for hidden units are not available! How to compute error term?
- Use backpropagation algorithm:
  1. Compute the error term for the output units, as with perceptron
  2. From output layer, repeat
     - *propagating the error term back to the previous layer and*
     - *updating the weights between the two layers*

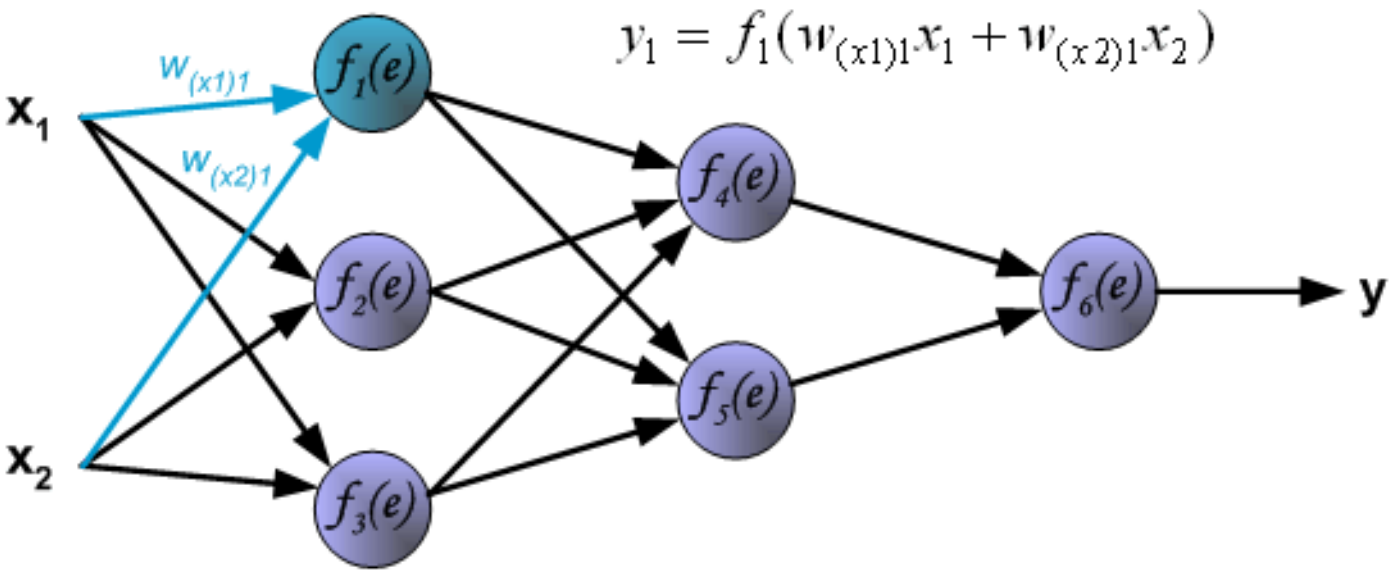
     until the earliest hidden layer is reached
- Backpropagation is a derivation of gradient descent (delta rule is a special case).

Backpropagation algorithm

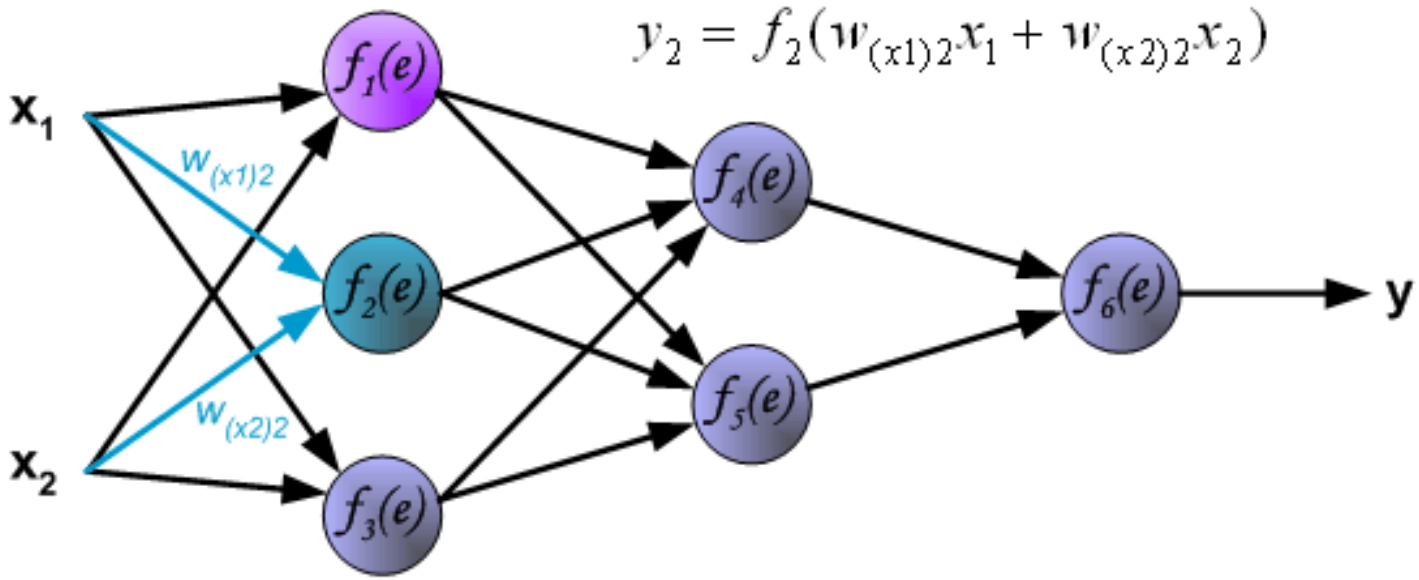
- Initialize weights (typically random!)
- Keep doing epochs:
  - For each example **e** in training set do
    - **forward pass** to compute
      - O = neural-net-output
      - miss = (T-O) at each output unit
    - **backward pass** to calculate deltas to weights
    - update all weights
  - end
- Until error stops improving or max number of epochs reached

Backpropagation algorithm

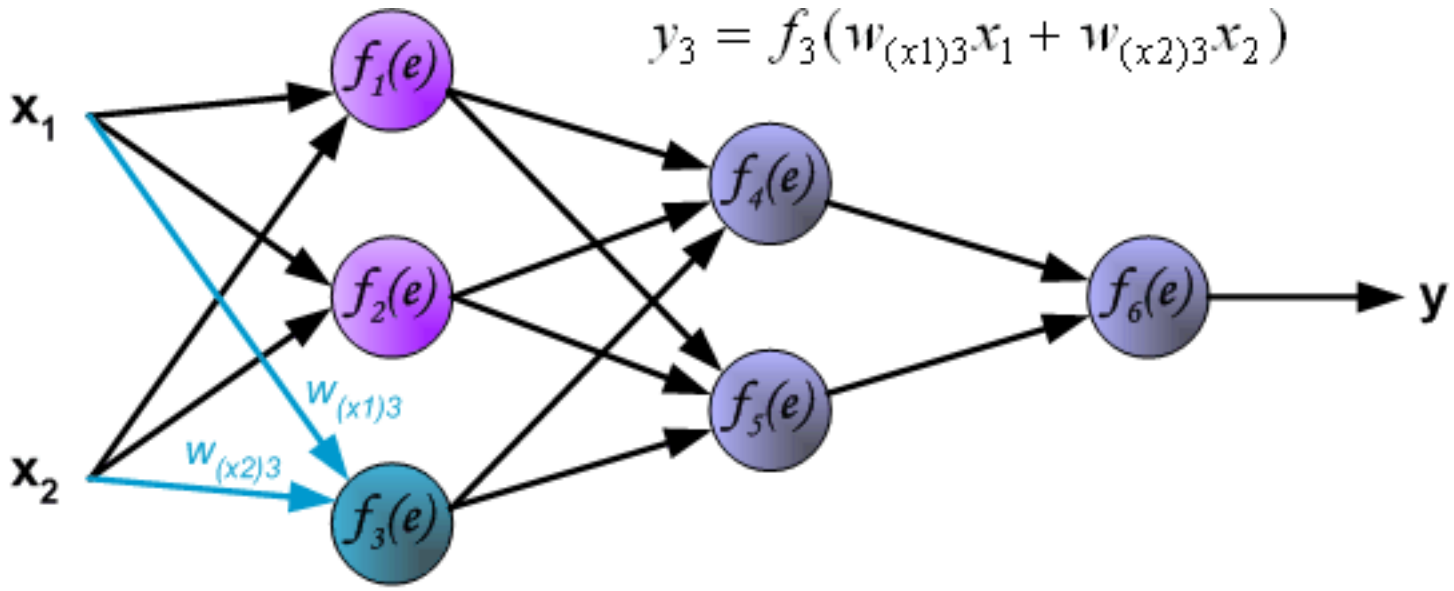
Pictures below illustrate how signal is propagating through the network, Symbols $w_{(xm)n}$ represent weights of connections between network input x_m and neuron n in input layer. Symbols y_n represents output signal of neuron n.



Backpropagation algorithm

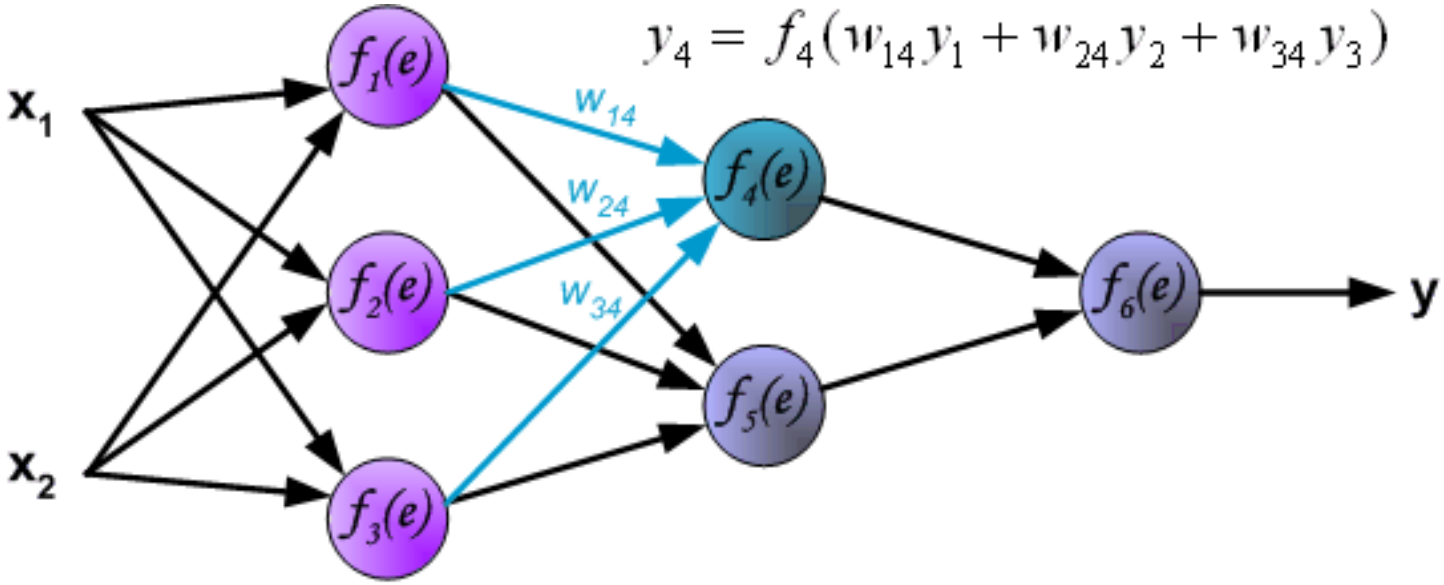


Backpropagation algorithm

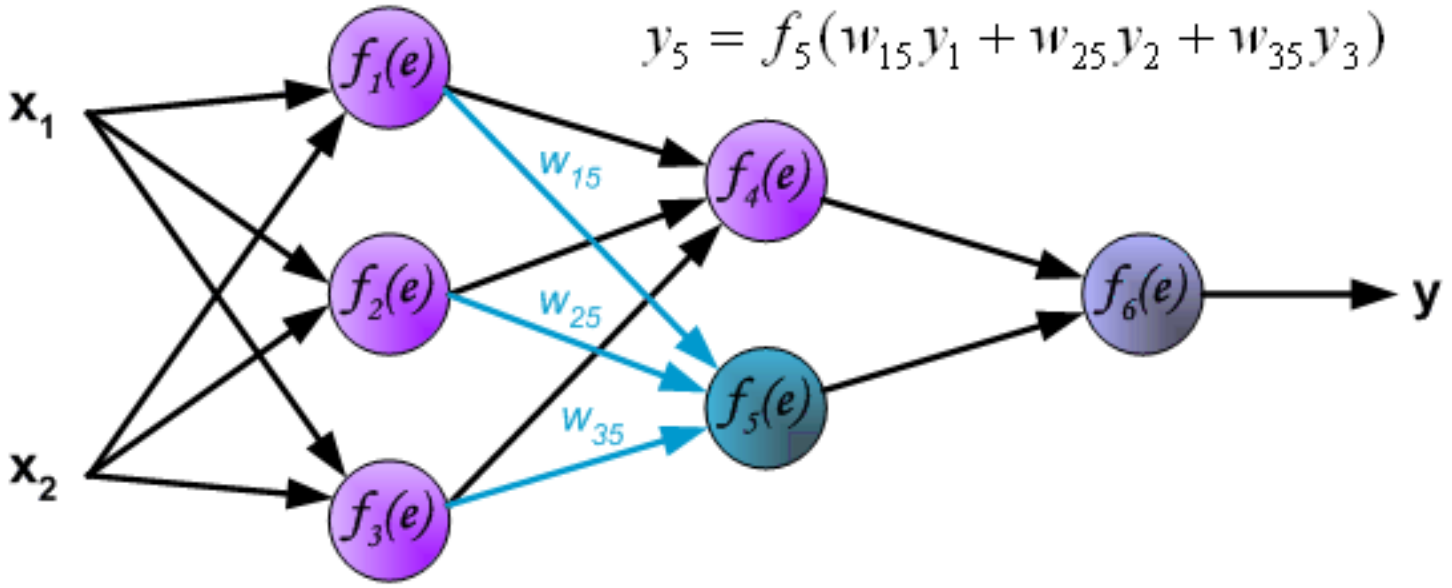


Backpropagation algorithm

Propagation of signals through the hidden layer. Symbols w_{mn} represent weights of connections between output of neuron *m* and input of neuron *n* in the next layer.

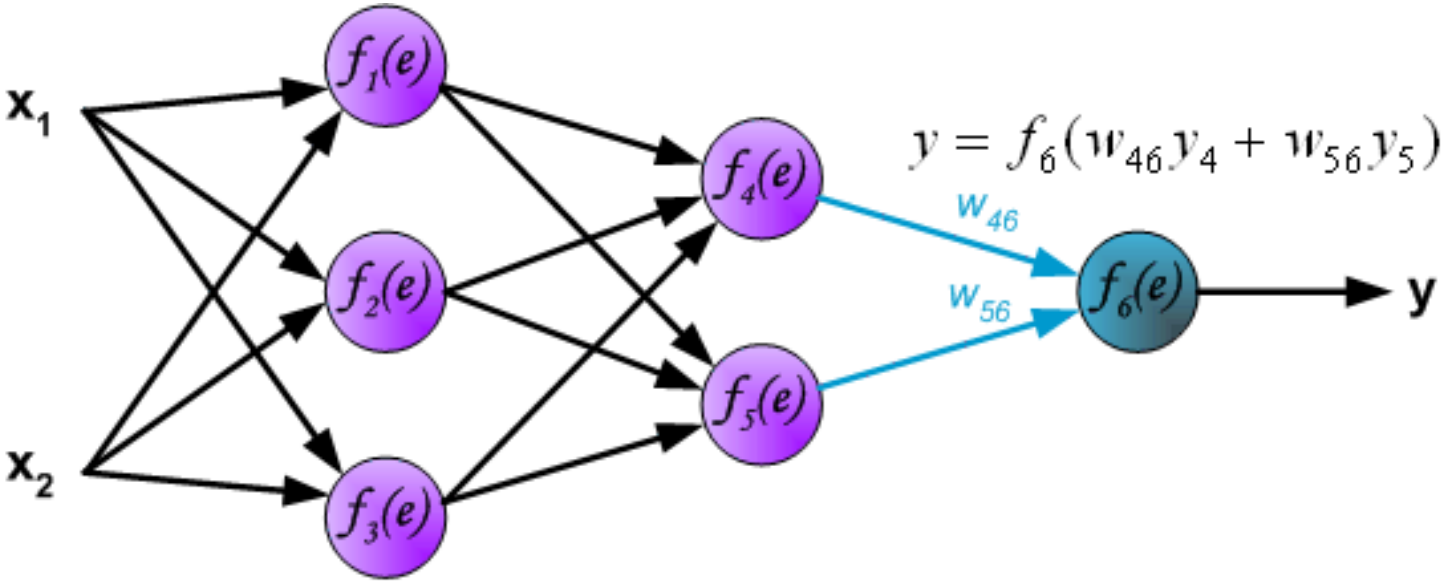


Backpropagation algorithm



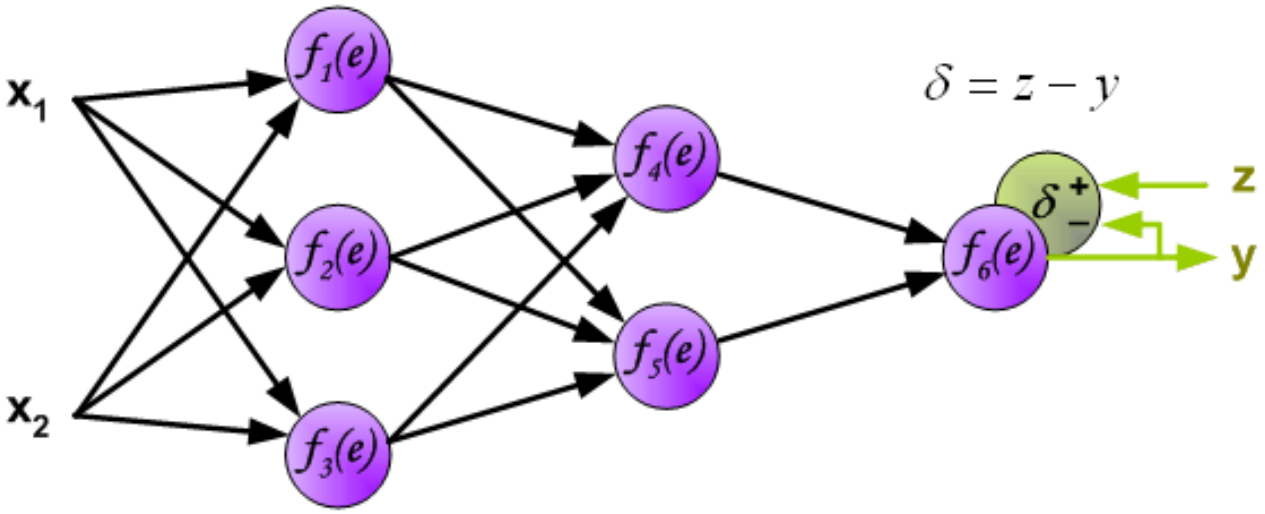
Backpropagation algorithm

Propagation of signals through the output layer.



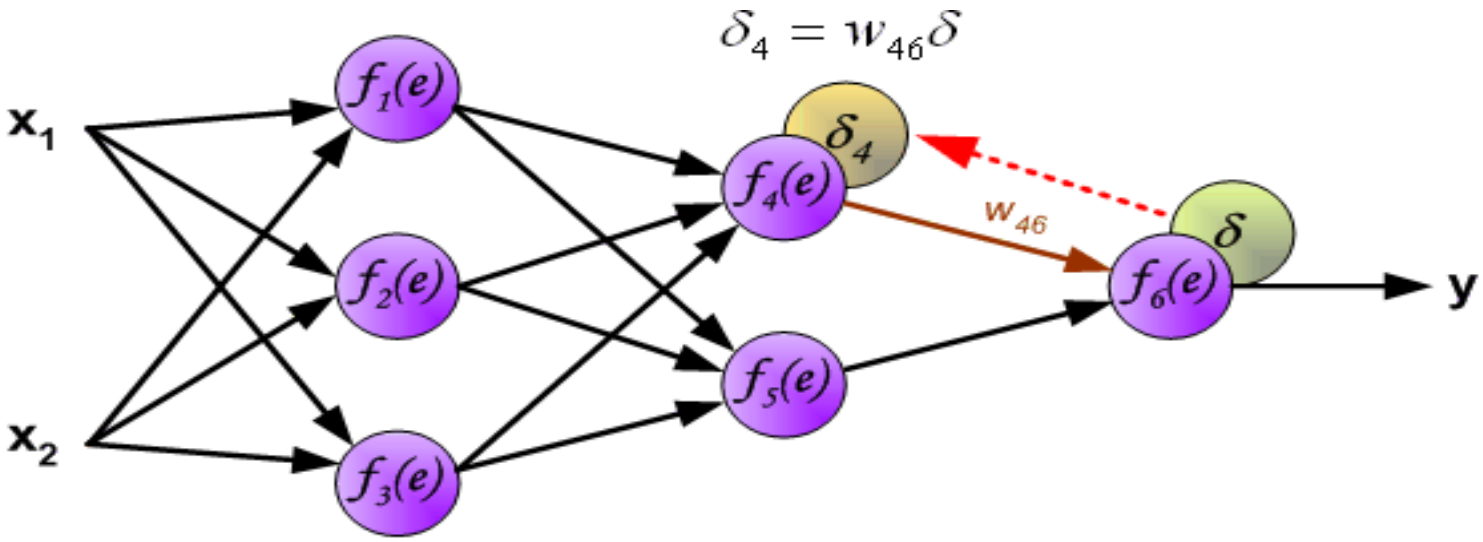
Backpropagation algorithm

In the next algorithm step the output signal of the network *y* is compared with the desired output value (the target), which is found in training data set. The difference is called error signal δ of output layer neuron



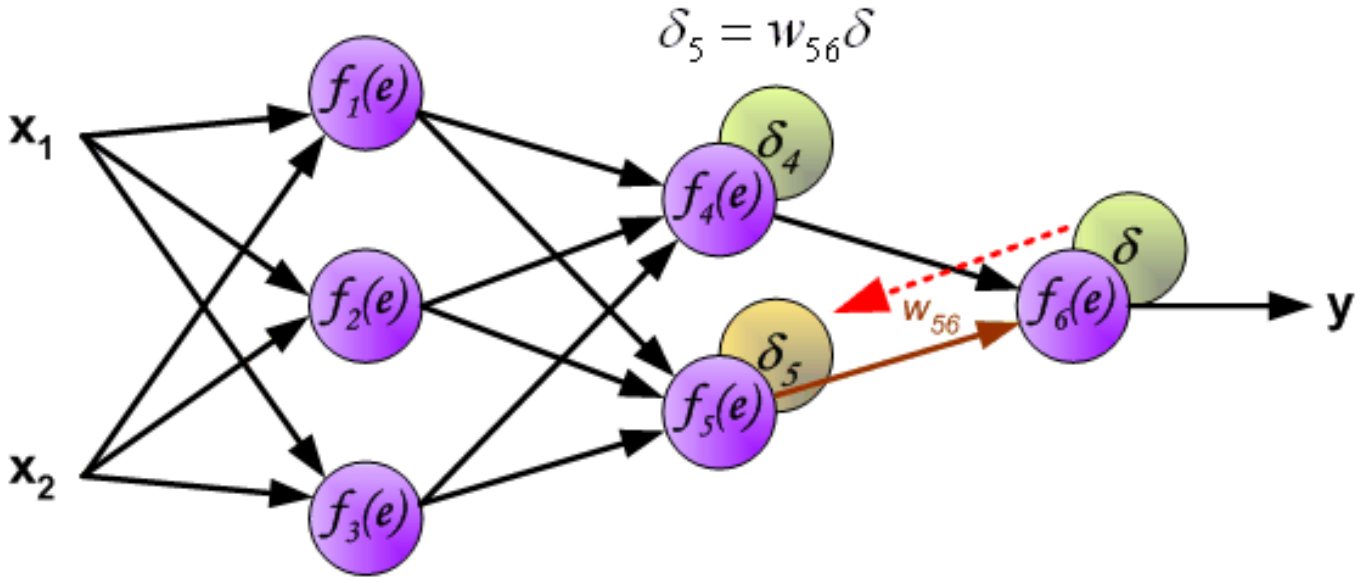
Backpropagation algorithm

The idea is to propagate the error signal δ back to all neurons, which output signals were input for discussed neuron.



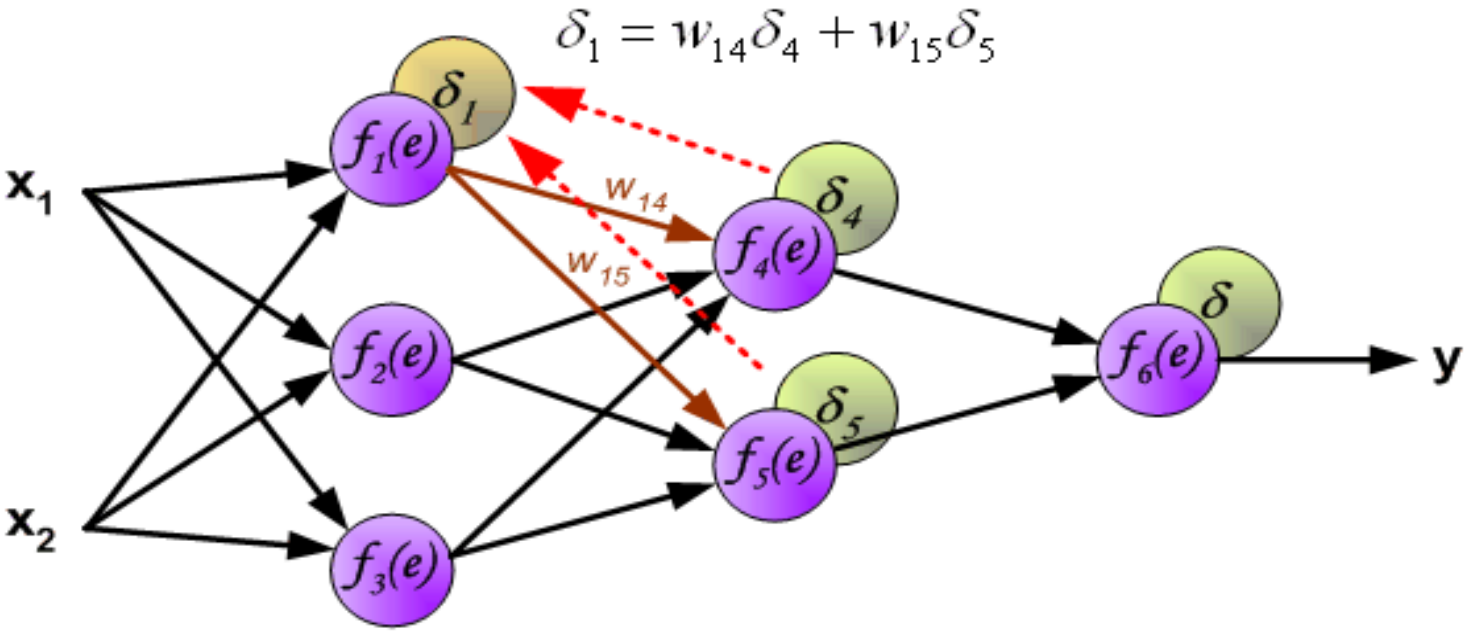
Backpropagation algorithm

The idea is to propagate the error signal δ back to all neurons, which output signals were input for discussed neuron.



Backpropagation algorithm

The weights' coefficients w_{mn} used to propagate errors back are equal to this used during computing output value. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). This technique is used for all network layers.

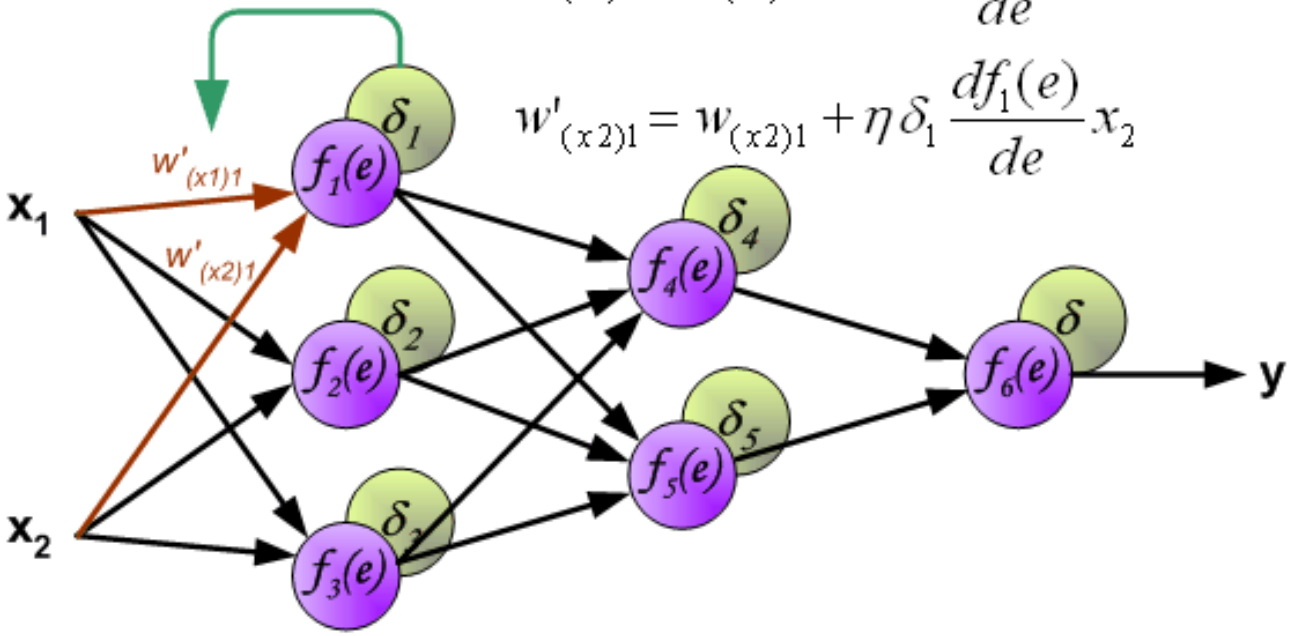


Backpropagation algorithm

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below *df(e)/de* represents derivative of neuron activation function (which weights are modified).

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$

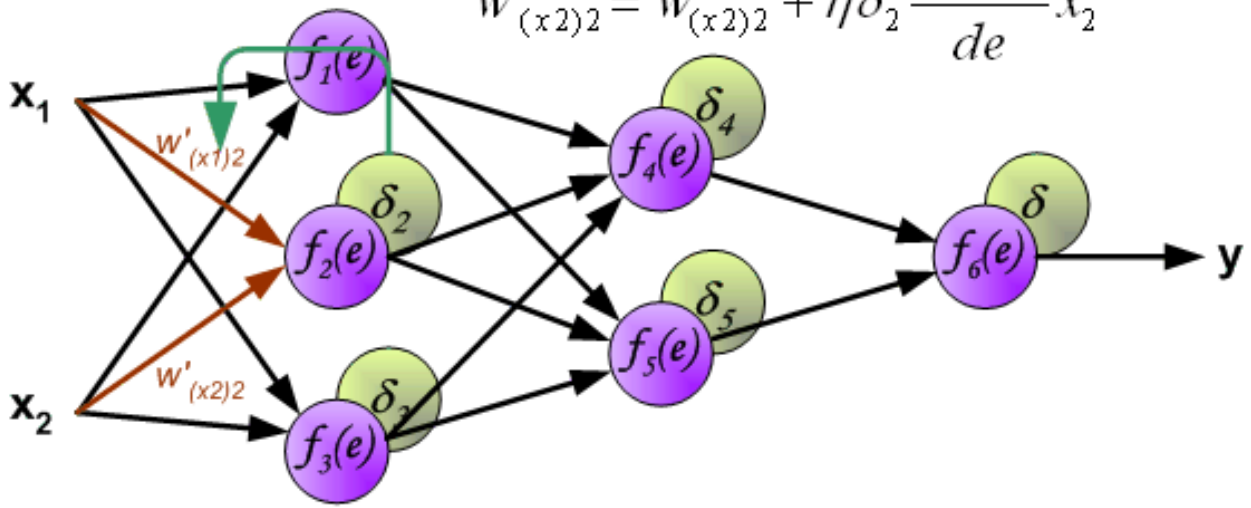


Backpropagation algorithm

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below *df(e)/de* represents derivative of neuron activation function (which weights are modified).

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$

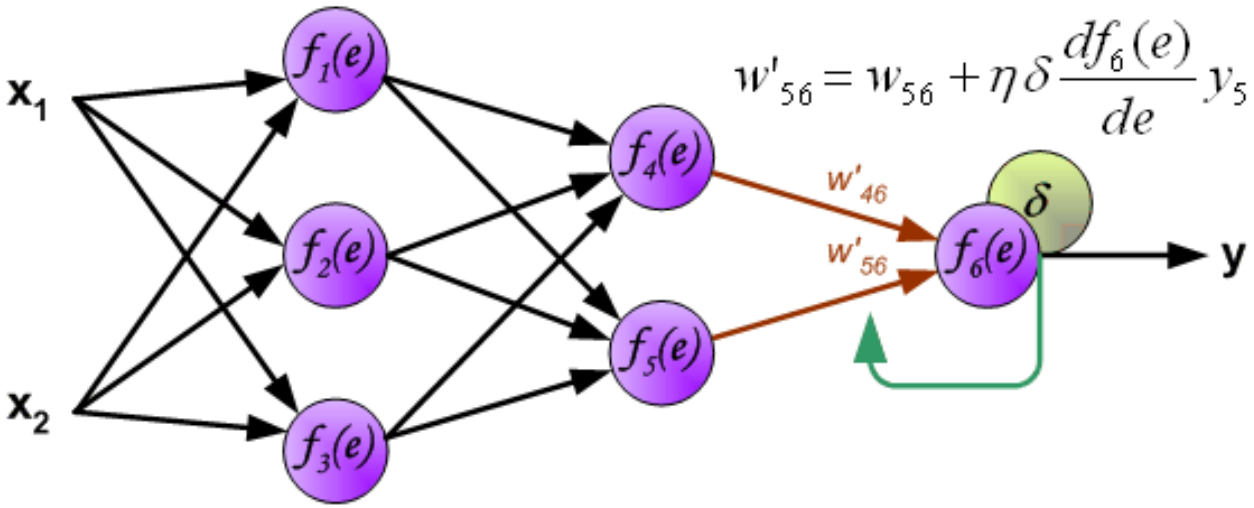


Backpropagation algorithm

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below *df(e)/de* represents derivative of neuron activation function (which weights are modified).

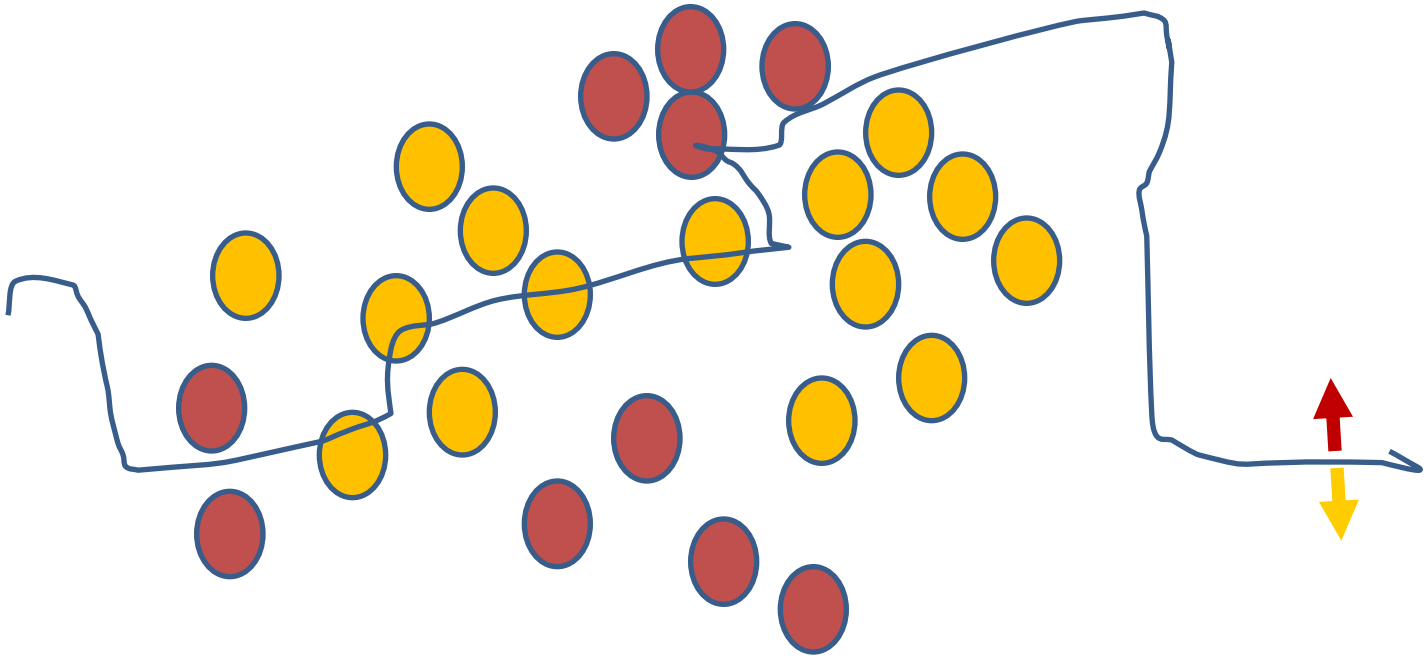
$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$



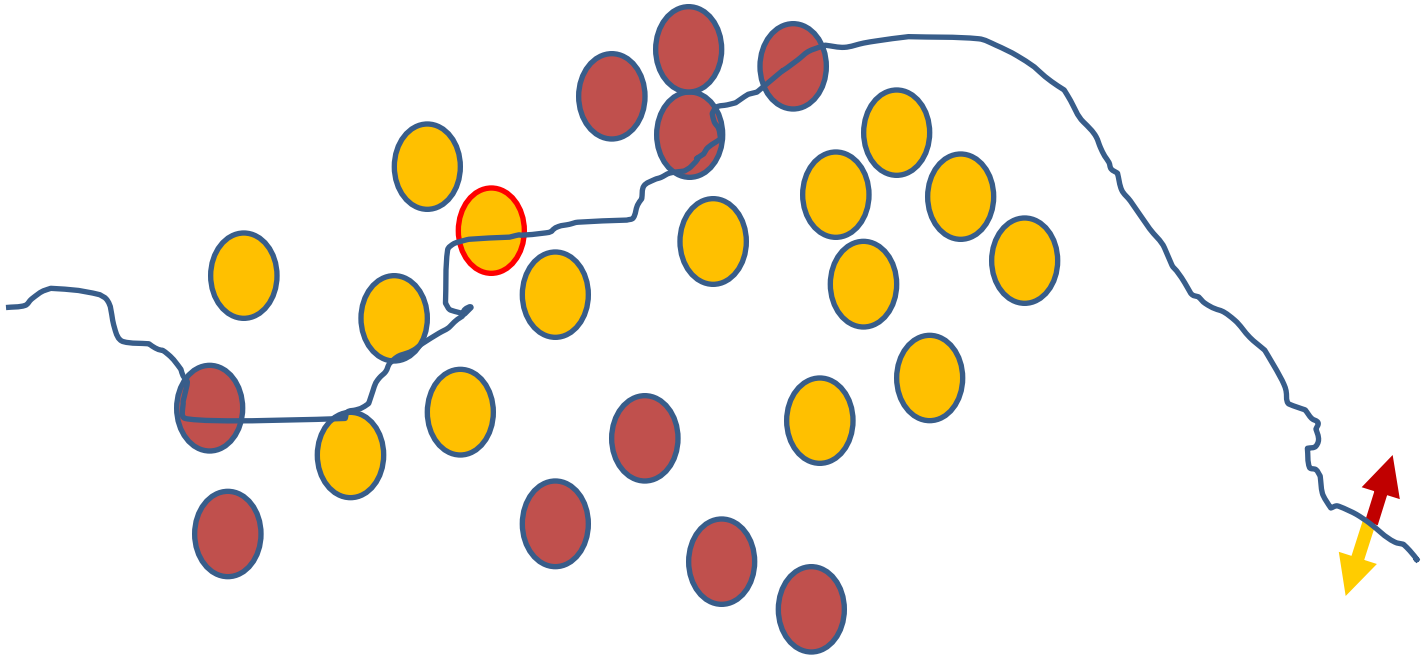
The decision boundary perspective

Initial random weights



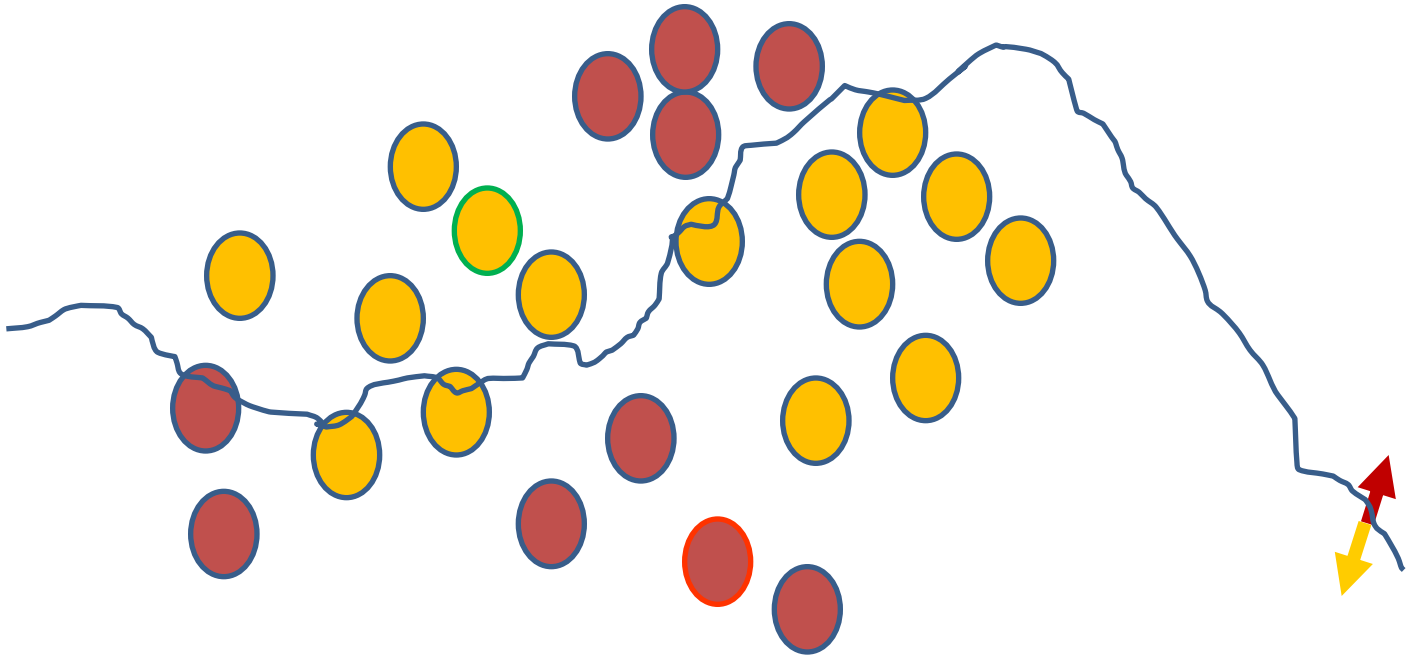
The decision boundary perspective

Present a training instance / adjust the weights



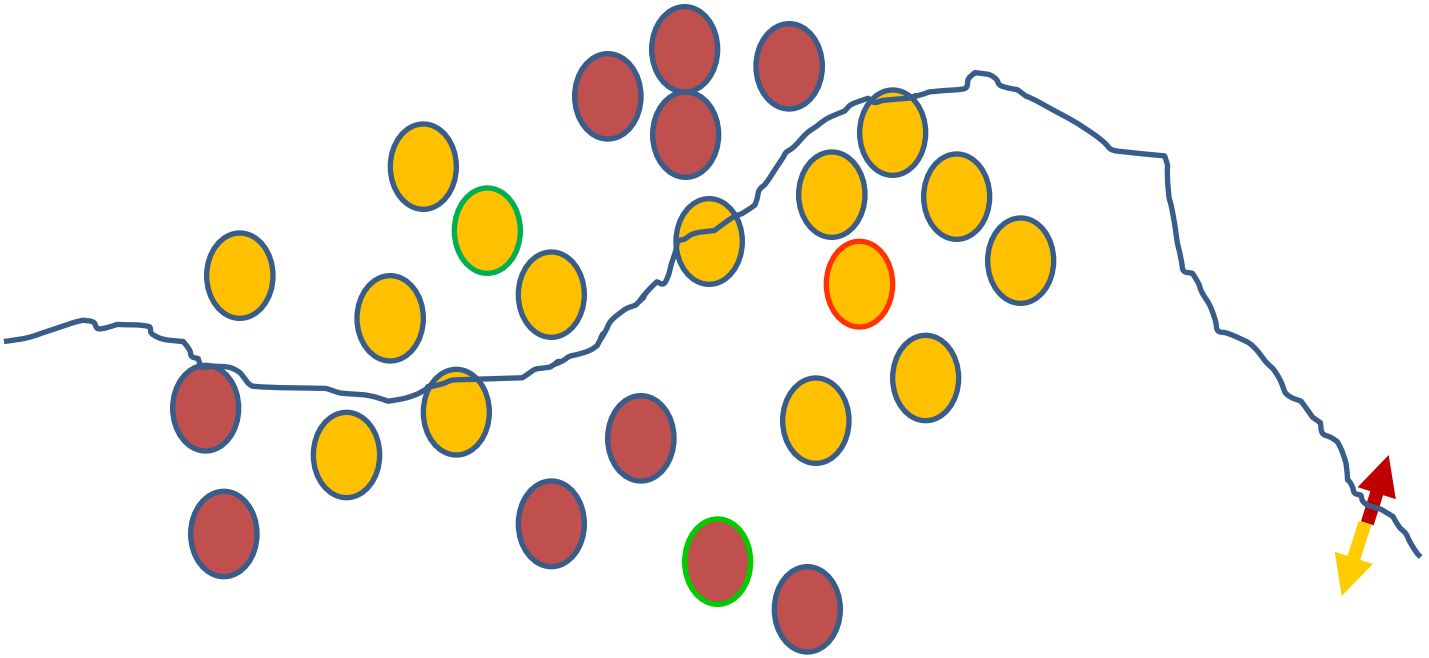
The decision boundary perspective

Present a training instance / adjust the weights



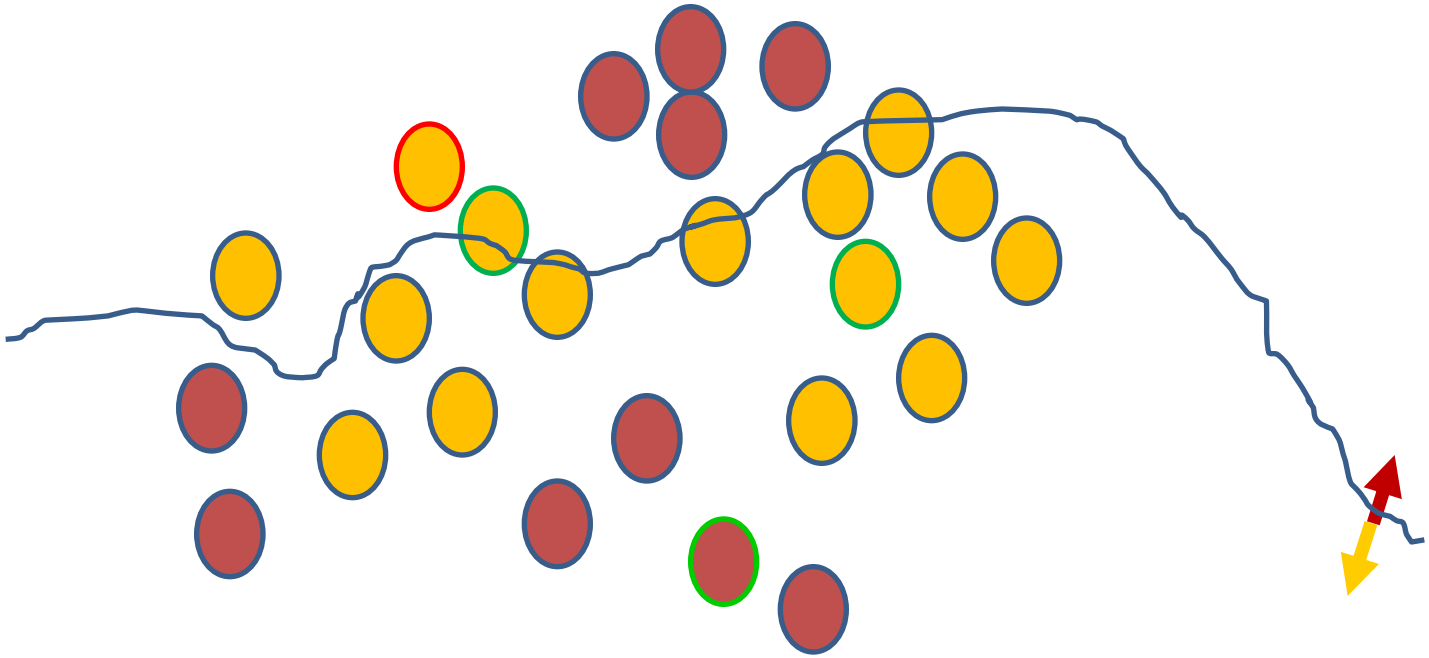
The decision boundary perspective

Present a training instance / adjust the weights



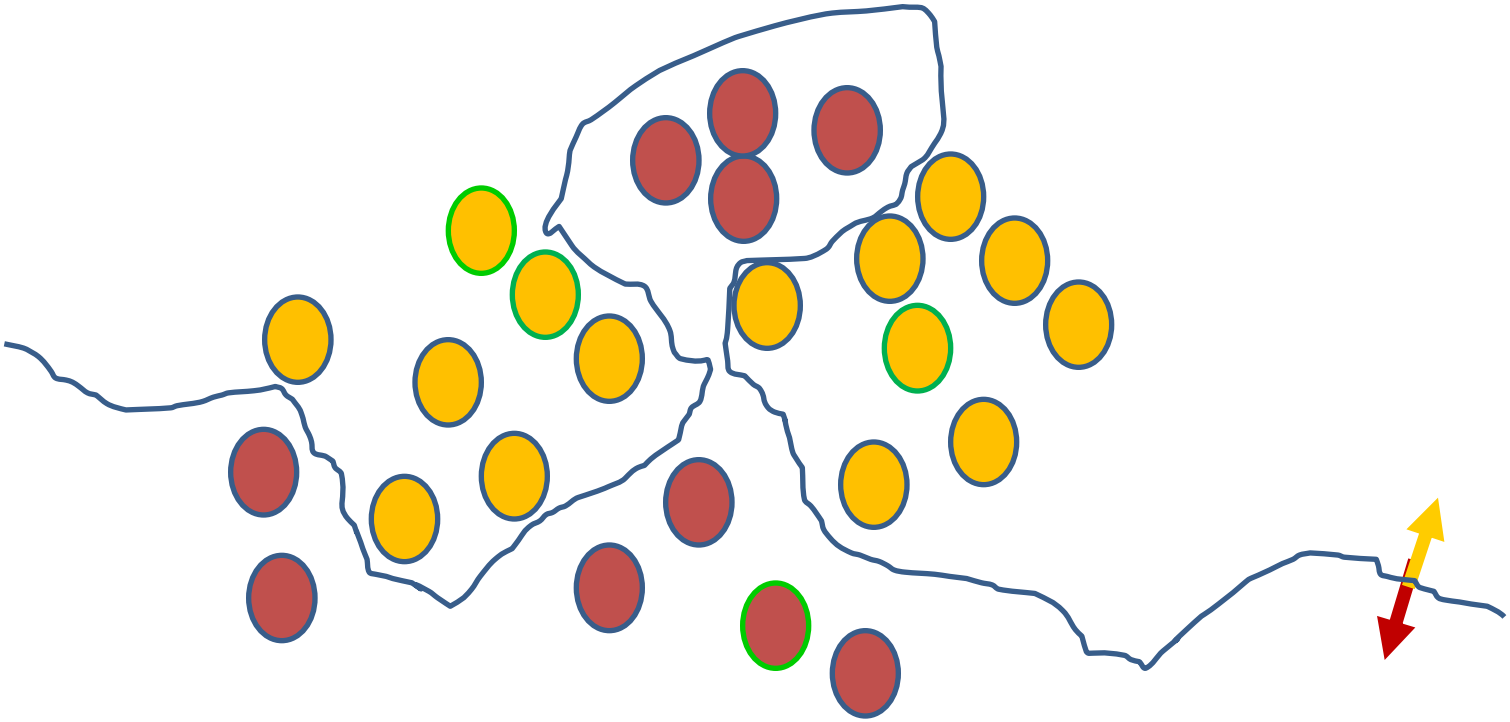
The decision boundary perspective

Present a training instance / adjust the weights



The decision boundary perspective

Eventually ….



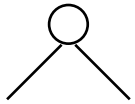
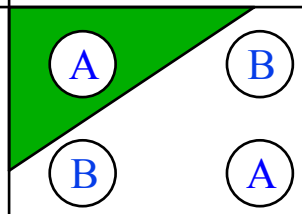
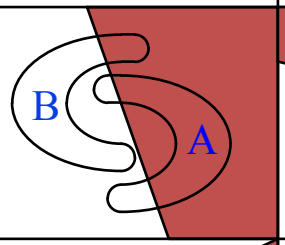
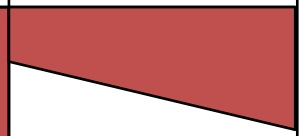
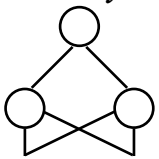
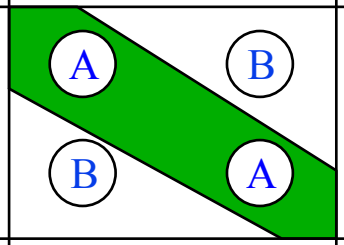
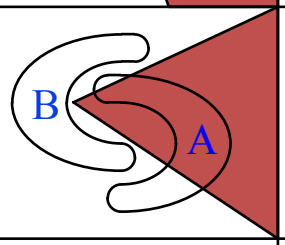
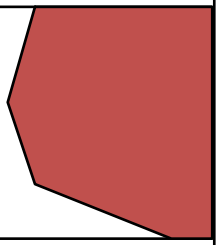
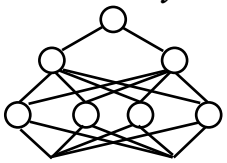
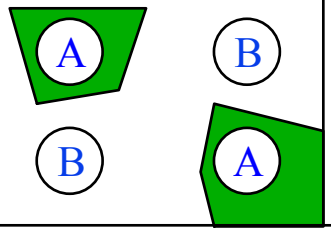
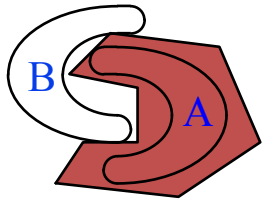
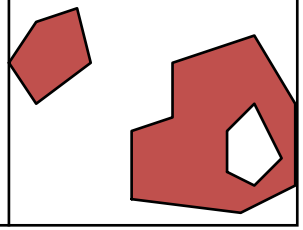
What learning rate?

1. Tuning set, or

2. Cross validation, or

3. **Small** for slow, conservative learning

How many layers?

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Abitrary (Complexity Limited by No. of Nodes)			

Neural Networks – An Introduction Dr. Andrew Hunter

How big a training set?

- Mostly, empirical rules…
- Determine your **target error rate**, *e*

  (success rate is 1- *e*)
- Typical training set approx. n/*e*, where n is the number of weights in the net
- Example:
  - *e* = 0.1, n = 80 weights
  - training set size 800

    trained until 95% correct training set classification

    should produce 90% correct classification

    on testing set (typical)

Limitations of Neural Networks

Random initialized densely connected networks lead to:

- High training cost
  - Each neuron in the neural network can be considered as a regression algorithm. Training the entire neural network is to train all the interconnected regressions.
- Difficult to train as the number of hidden layers increases
  - In backpropagation, gradient is progressively getting more dilute. That is, below top layers, the correction signal δ_n is minimal.
- Stuck in local optima
  - The random initialization does not guarantee starting from the proximity of global optima.

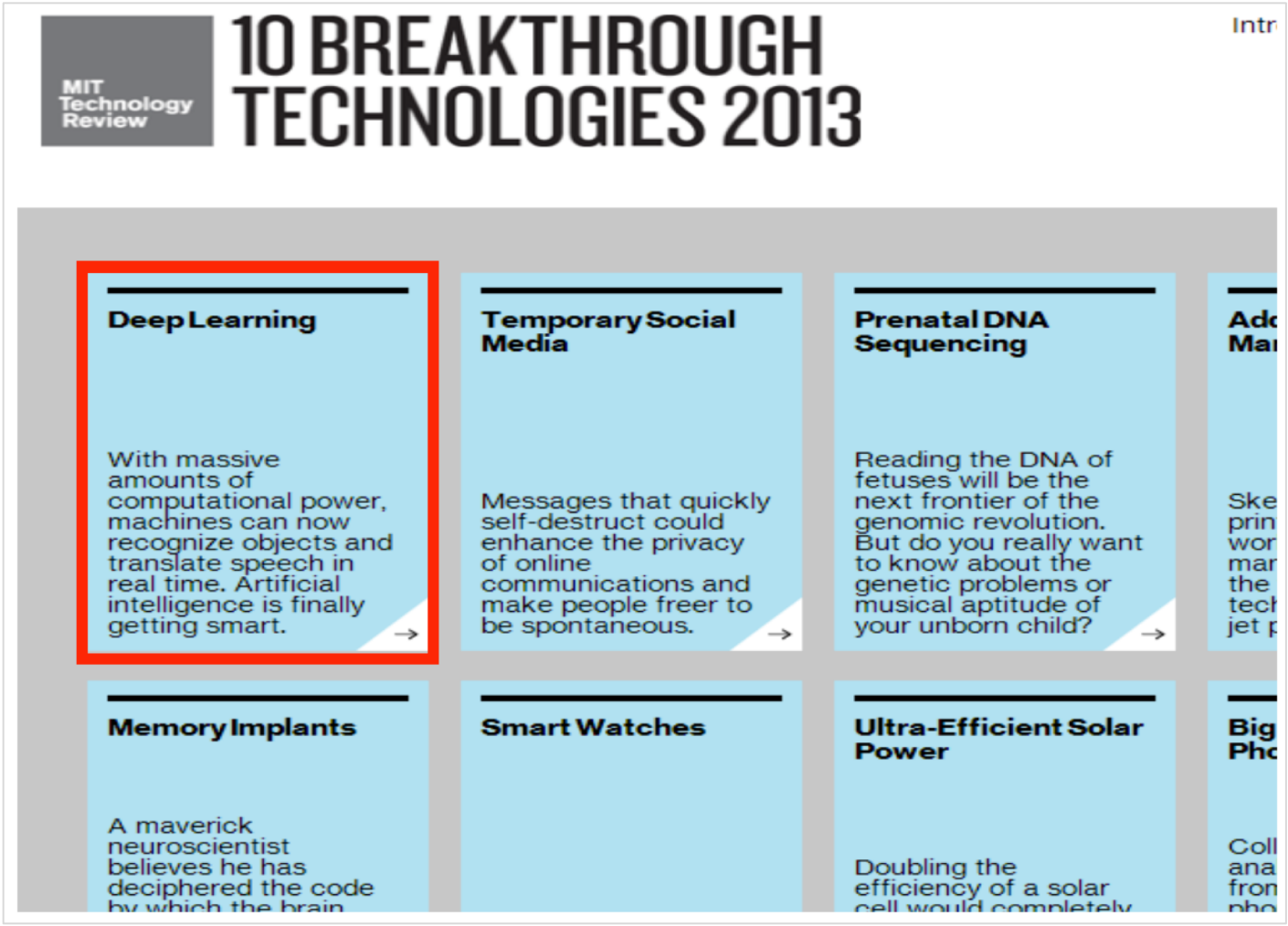
Solution:

- **Deep Learning**/Learning multiple levels of representation

Deep learning

- First conceived for image classification tasks, as a replication of mammalians' visual cortex
- Cascade of **many hidden layers of locally connected units,** for feature extraction and transformation.
- The hidden layers of a deep network **learn multiple levels of representations** that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

Deep learning: the next big thing?



Deep learning: the next big thing?

In "Nature" 27 January 2016:

- "DeepMind's program AlphaGo beat Fan Hui, the European Go champion, five times out of five..."
- "AlphaGo was not preprogrammed to play Go: it used a general-purpose algorithm to interpret the game's patterns."
- "...AlphaGo program applied **deep learning** to neural networks (convolutional NN)."

Deep Learning Today

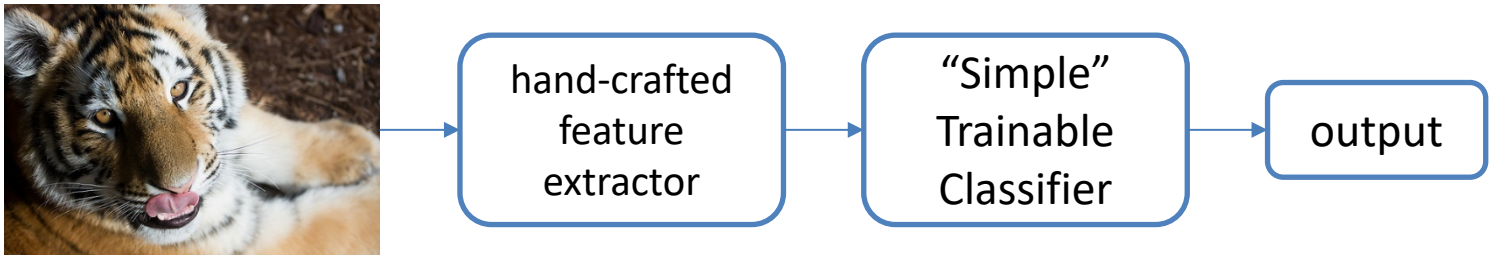
- **Computer Vision and Image Processing**
  - Feature engineering is the bread-and-butter of a large portion of the CV community, which creates some resistance to feature learning
  - But the record holders on ImageNet and Semantic Segmentation are convolutional nets
- Speech recognition
  - A few long-standing performance records were broken with deep learning methods
  - Microsoft and Google have both deployed DL-based speech recognition systems in their products
- Advancement in Natural Language Processing
  - Fine-grained sentiment analysis, syntactic parsing
  - Language model, machine translation, question answering
- … potentially any field, including bioinformatics

Motivations for Deep Architectures

- Insufficient depth can hurt
  - With shallow architecture (SVM, NB, KNN, etc.), the required number of nodes in the graph (i.e. computations, and also number of parameters, when we try to learn the function) may grow very large.
  - Many functions that can be represented efficiently with a deep architecture cannot be represented efficiently with a shallow one.
- The brain has a deep architecture
  - The visual cortex shows a sequence of areas each of which contains a representation of the input, and signals flow from one to the next.
  - Note that representations in the brain are in between dense distributed and purely local: they are **sparse**: about 1% of neurons are active simultaneously in the brain.
- Cognitive processes seem deep
  - Humans organize their ideas and concepts hierarchically.
  - Humans first learn simpler concepts and then compose them to represent more abstract ones.
  - Engineers break-up solutions into multiple levels of abstraction and processing

Deep Learning vs Traditional ML

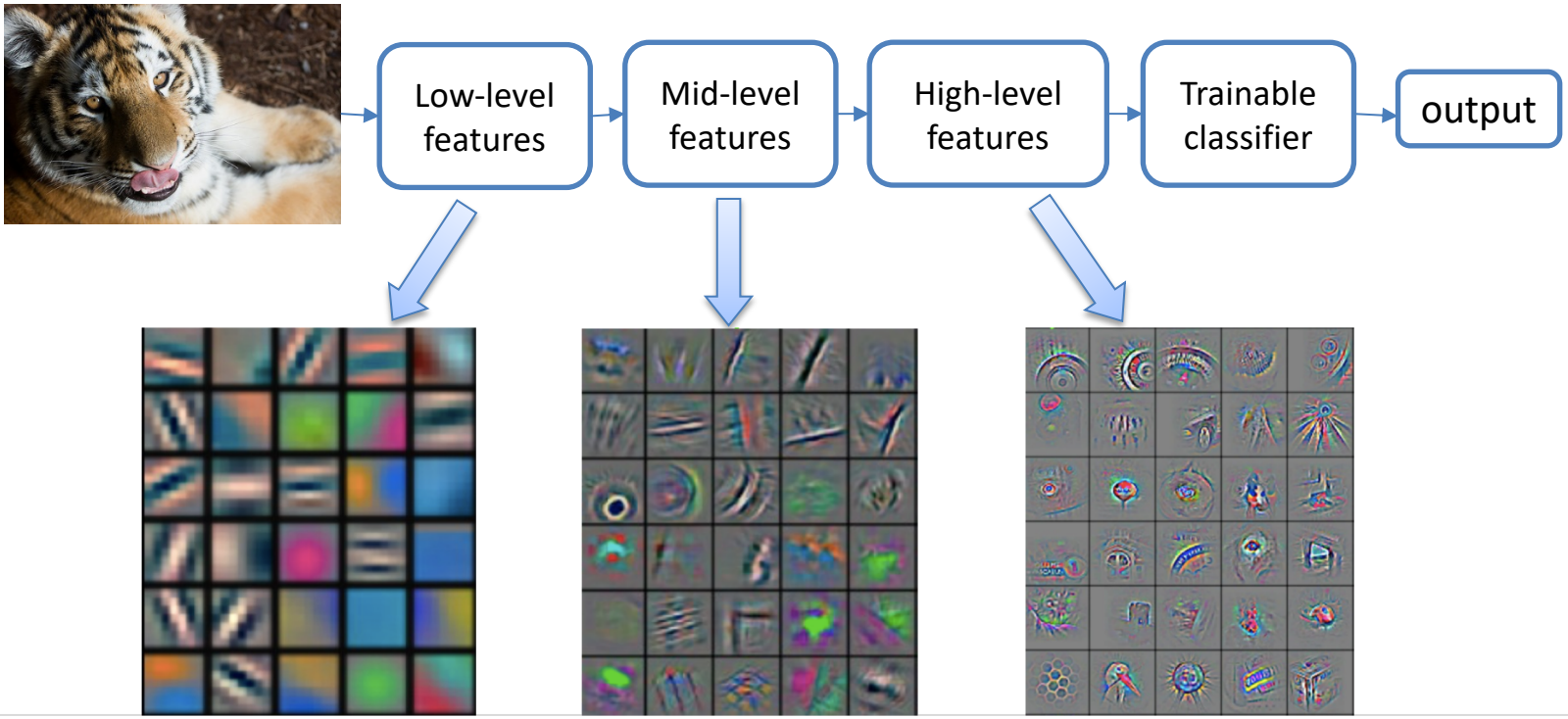
- Traditional pattern recognition models use hand-crafted features and relatively simple trainable classifier.



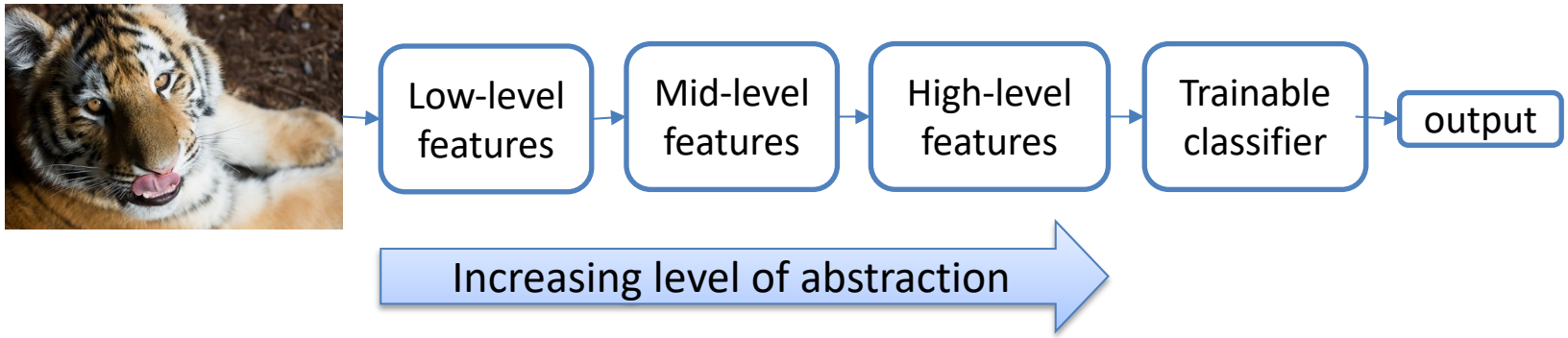
- This approach has the following limitations:
  - It is very tedious and costly to develop hand-crafted features
  - The hand-crafted features are usually **highly dependent on one application**, and cannot be transferred easily to other applications

Deep Learning vs Traditional ML

- Deep learning (a.k.a. representation learning) seeks to learn rich hierarchical representations (i.e. features) automatically through multiple stage of feature learning process.



Learning Hierarchical Representations



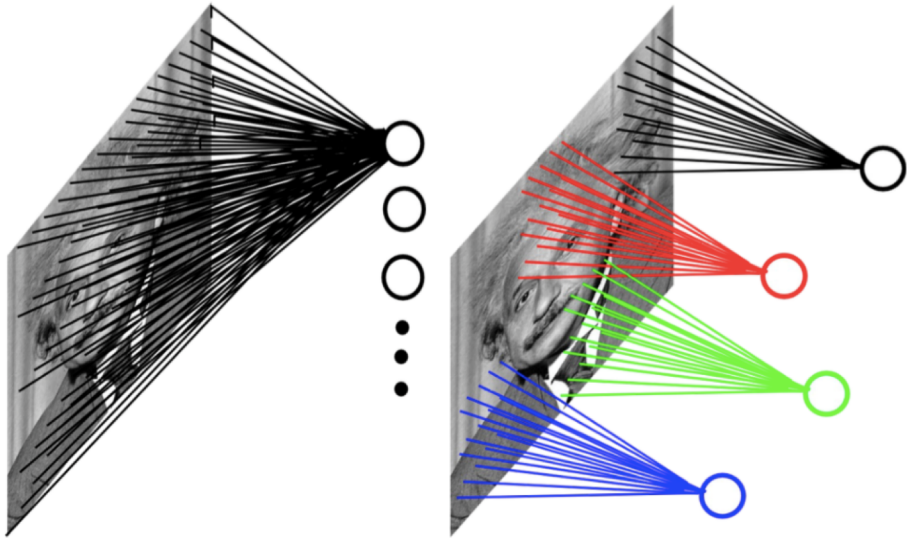
- Hierarchy of representations with increasing level of abstraction. Each stage is a kind of trainable nonlinear feature transform
- Image recognition:
  - Pixel → edge → texton → motif → part → object
- Text:
  - Character → word → word group → clause → sentence → story
- Virtually, any kind of application requiring PR (e.g. DNA sequence classification)

Convolutional Neural Network (CNN)

- Convolutional Neural Networks are inspired by mammalian visual cortex.
  - The visual cortex contains a complex arrangement of cells, which are sensitive to small sub-regions of the visual field, called a **receptive field**. These cells act as local filters over the input space and are well-suited to exploit the strong spatially local correlation present in natural images.
  - Two basic cell types:
    - Simple cells respond maximally to specific edge-like patterns within their receptive field.
    - Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.

Convolutional Neural Networks (CNN)

- Inspired by the neurophysiological experiments conducted by [Hubel & Wiesel 1962], CNNs are a special type of neural network whose hidden units are only connected to local receptive field. The number of parameters needed by CNNs is much smaller.
- Input can have very high dimension. Using a fully-connected neural network would need a large amount of parameters.



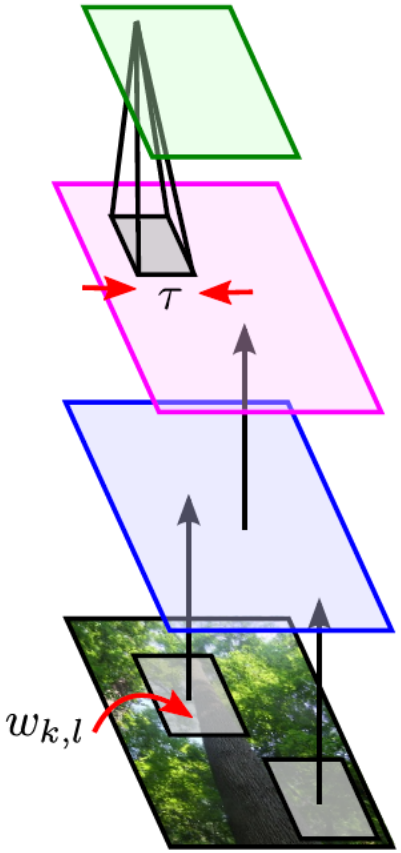
Example: 200x200 image

a) fully connected: 40,000 hidden units => 1.6 billion parameters
b) CNN: 5x5 kernel, 100 feature maps => 2,500 parameters

CNN Architecture

- Intuition: Neural network with specialized connectivity structure,
  - Stacking multiple layers of feature extractors
  - Low-level layers extract local features.
  - High-level layers extract global patterns.
- A CNN is a list of layers that transform the input data into an output class/prediction.
- There are a few distinct types of layers:
  - Convolutional layer
  - Non-linear layer
  - Pooling layer

Building-blocks for CNN



$$x_{i,j} = \max_{|k|<\tau,|l|<\tau} y_{i-k,j-l}$$

mean or subsample also used

pooling stage

$$y_{i,j} = f(a_{i,j})$$

e.g. $f(a) = [a]_+$

$f(a) = \text{sigmoid}(a)$

non-linear stage

Feature maps of a larger region are combined.

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

convolutional stage

Shared weights

Feature maps are trained with neurons.

Each sub-region yields a feature map, representing its feature.

$z_{i,j}$

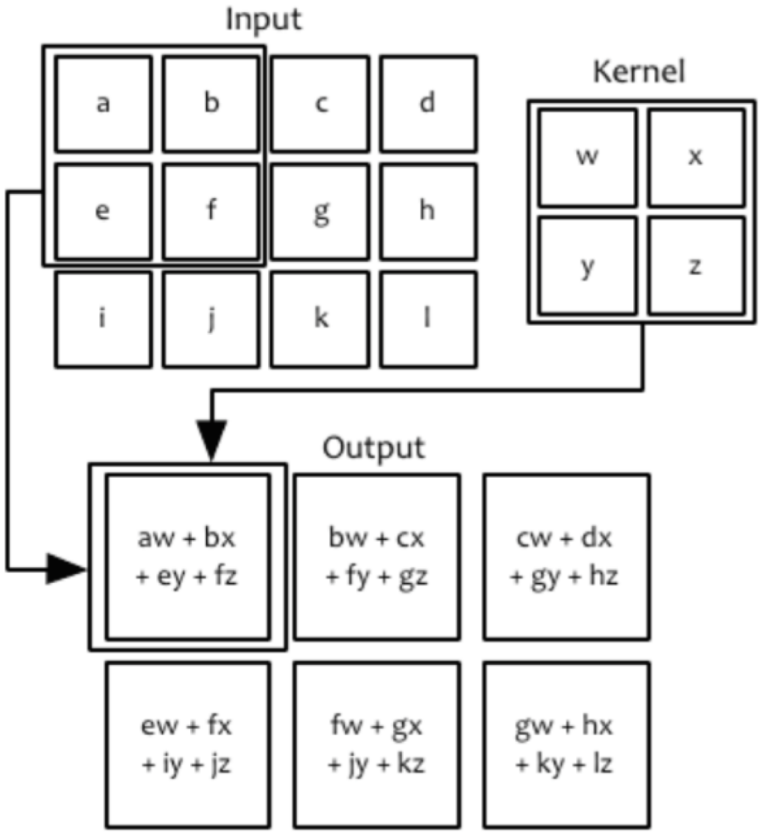
Images are segmented into sub-regions.

input image

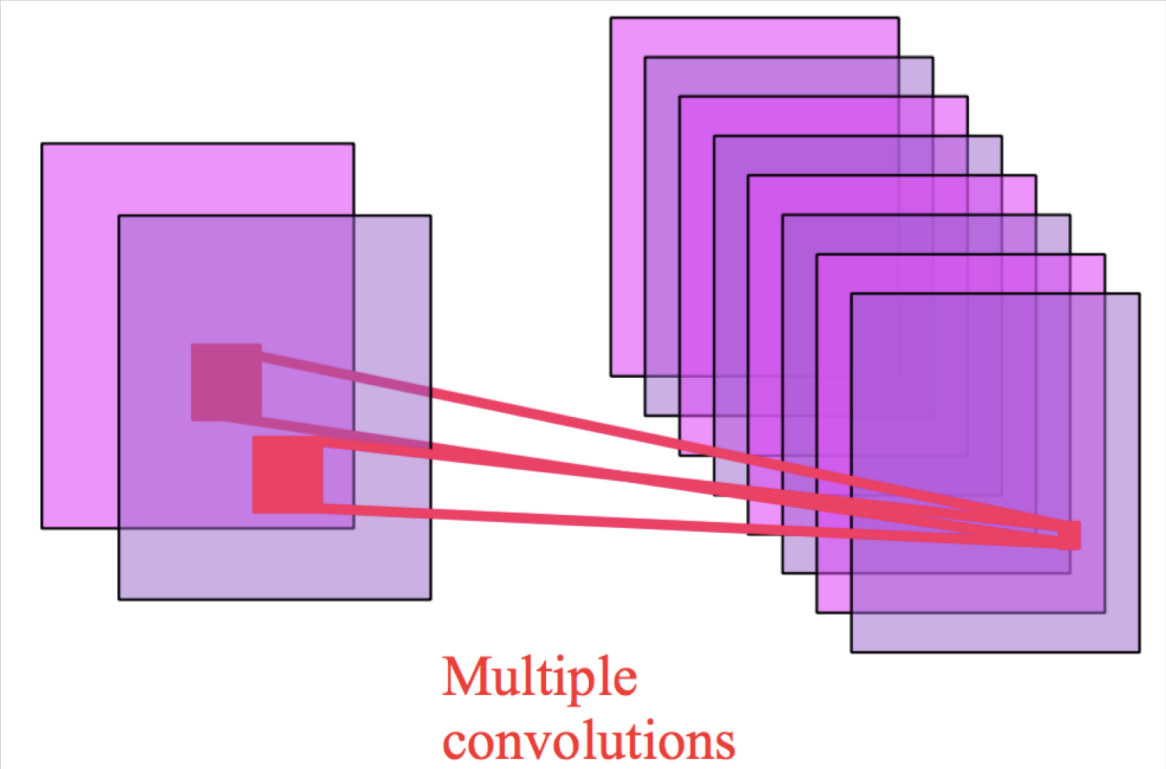
Convolution operation

Input: an image (2-D array) x
Convolution kernel/operator (2-D array of learnable parameters): w
Feature map (2-D array of processed data): s
Convolution operation in 2-D domains:

$$s[i,j] = (x * w)[i,j] = \sum_{m=-M}^{M} \sum_{n=-N}^{N} x[i+m, j+n]\, w[m,n]$$



Multiple Convolutions

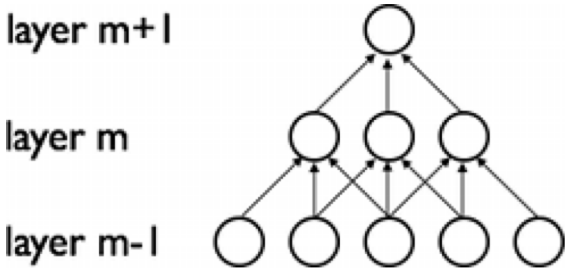


Usually there are multiple feature maps, one for each convolution operator.

CNN Architecture: Convolutional Layer

- The core layer of CNNs
- The convolutional layer consists of a set of filters.
  - Each filter covers a spatially small portion of the input data.
- Each filter is convolved across the dimensions of the input data, producing a multidimensional feature map.
  - As we convolve the filter, we are computing the dot product between the parameters of the filter and the input.
- Intuition: the network will learn filters that activate when they see some specific type of feature at some spatial position in the input.
- The key architectural characteristics of the convolutional layer is
  - local connectivity
  - and shared weights.

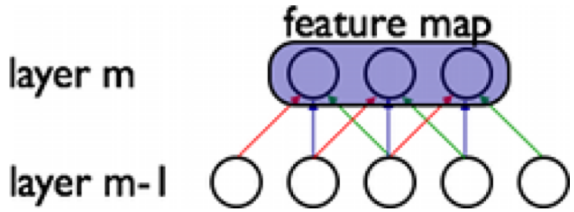
CNN Convolutional Layer: Local Connectivity



- Neurons in layer **m** are only connected to 3 adjacent neurons in the **m-1** layer.
- Neurons in layer **m+1** have a similar connectivity with the layer below.
- Each neuron is unresponsive to variations outside of its ***receptive field*** with respect to the input.
  - Receptive field: small neuron collections which process portions of the input data
- The architecture thus ensures that the learnt feature extractors produce the strongest response to a **spatially local input pattern**.

CNN Convolutional Layer: Shared Weights

- We show 3 hidden neurons belonging to the same feature map (the layer right above the input layer).



- Weights of the same color are shared—constrained to be identical.
- Gradient descent can still be used to learn such shared parameters, with only a small change to the original algorithm.
- Replicating neurons in this way allows for features to be detected regardless of their position in the input (spatial invariance).
- Additionally, weight sharing increases learning efficiency by greatly **reducing the number of free parameters** being learnt.

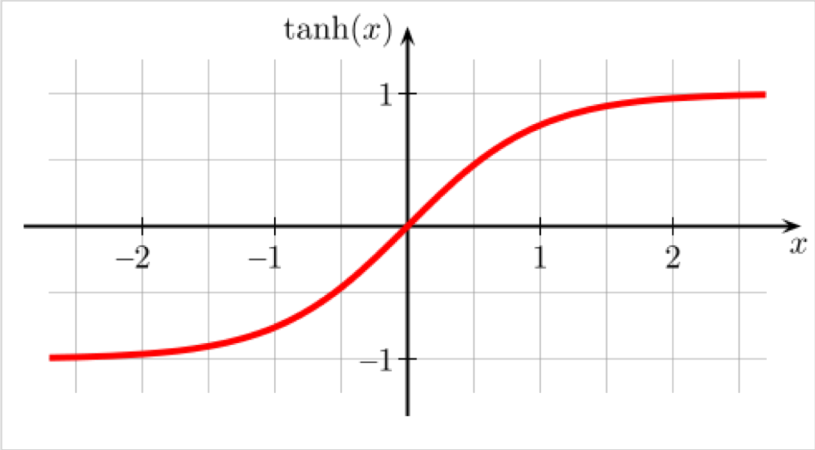
CNN Architecture: Non-linear Layer

- Intuition: Increase the nonlinearity of the entire architecture without affecting the receptive fields of the convolution layer

- A layer of neurons that applies the non-linear activation function, such as,
  - $f(x) = \max(0, x)$ (ReLU)
  - $f(x) = \tanh x$
  - $f(x) = |\tanh x|$
  - $f(x) = (1 + e^{-x})^{-1}$

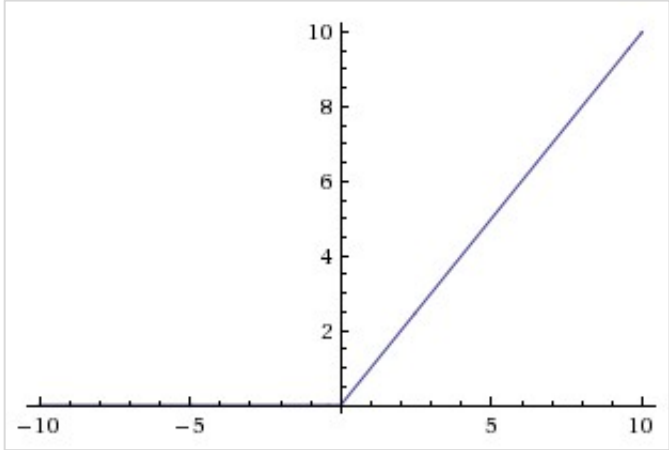
Non-linearity

- Tanh(x)



$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

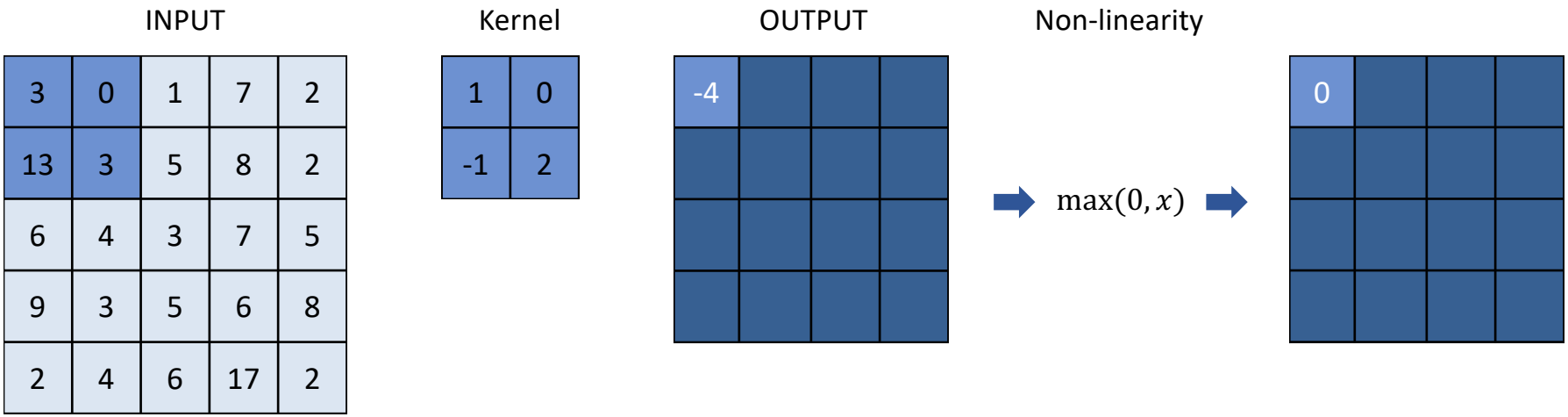
- ReLU



$$f(x) = \max(0, x)$$

Convolutional Neural Networks (CNNs)

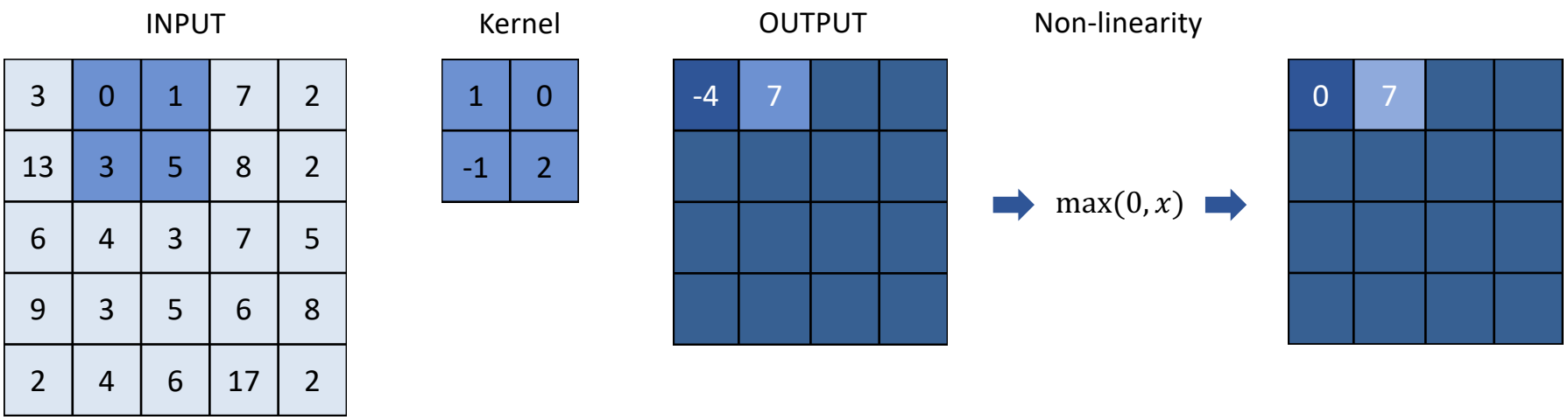
Example: 4x4 convolution with ReLU non-linearity



$$3 * 1 + 0 * 0 + 13 * (-1) + 3 * 2 = -4$$

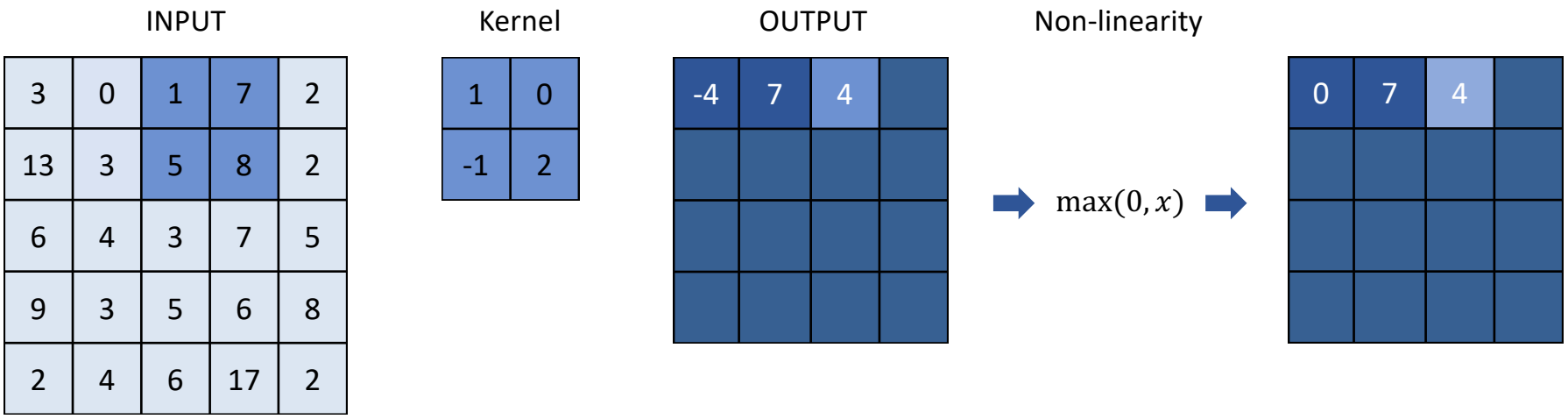
Convolutional Neural Networks (CNNs)

Example: 4x4 convolution with ReLU non-linearity



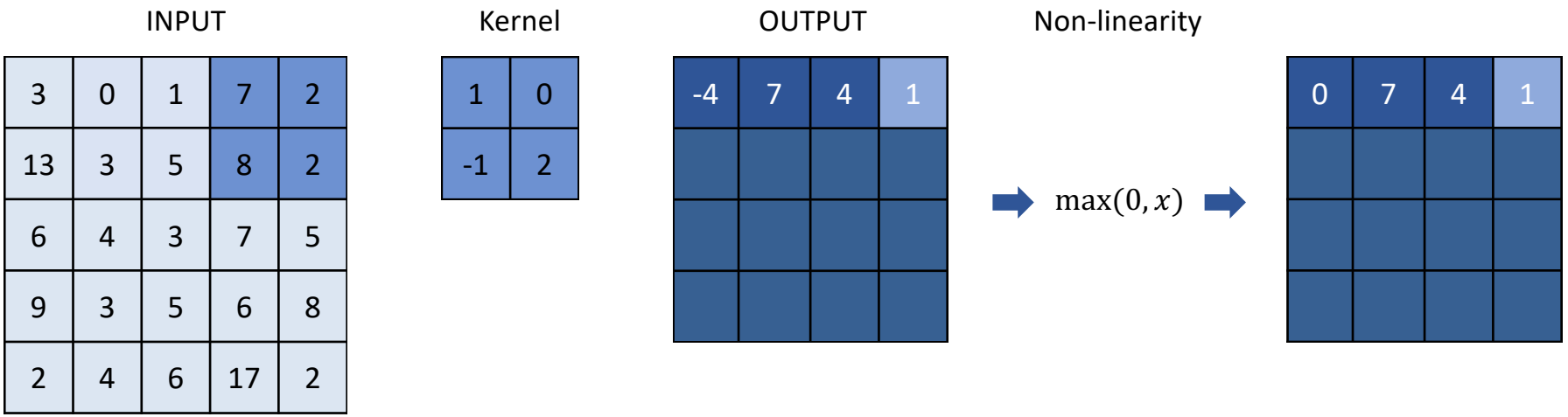
Convolutional Neural Networks (CNNs)

Example: 4x4 convolution with ReLU non-linearity



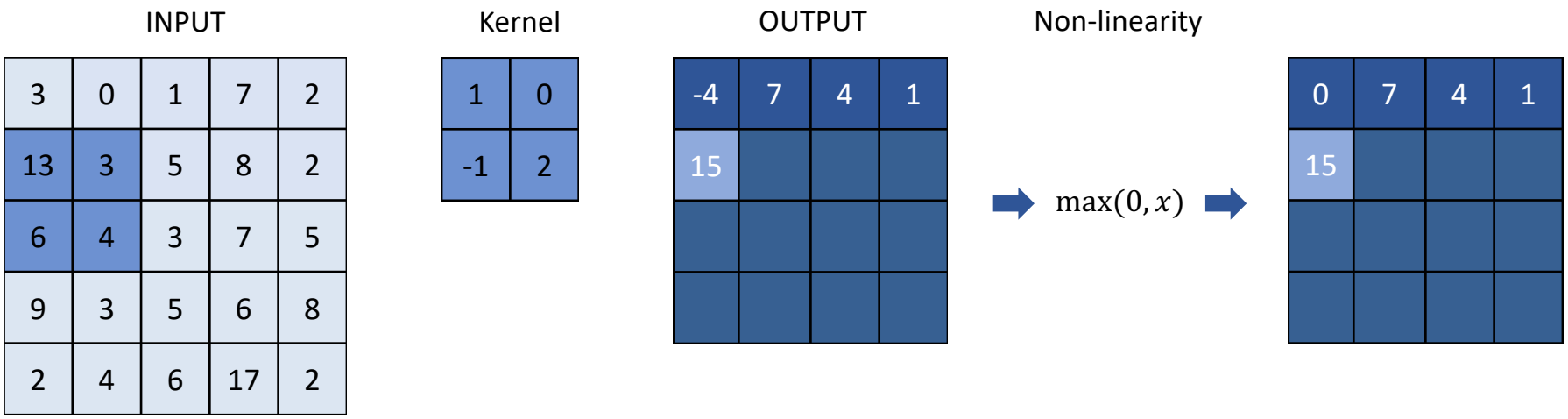
Convolutional Neural Networks (CNNs)

Example: 4x4 convolution with ReLU non-linearity



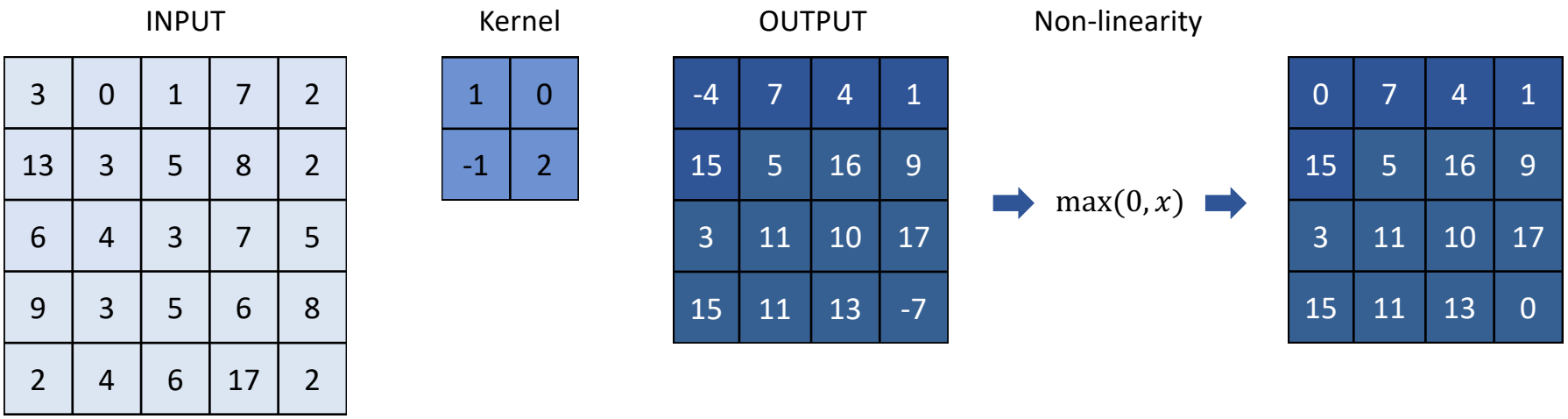
Convolutional Neural Networks (CNNs)

Example: 4x4 convolution with ReLU non-linearity



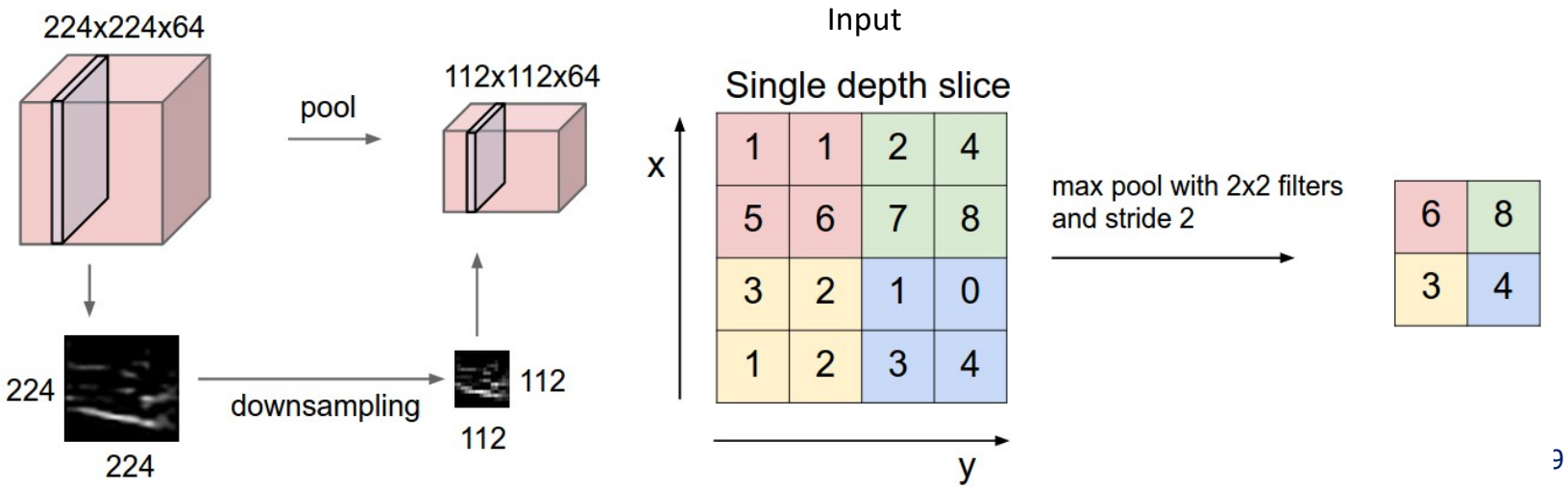
Convolutional Neural Networks (CNNs)

Example: 4x4 convolution with ReLU non-linearity



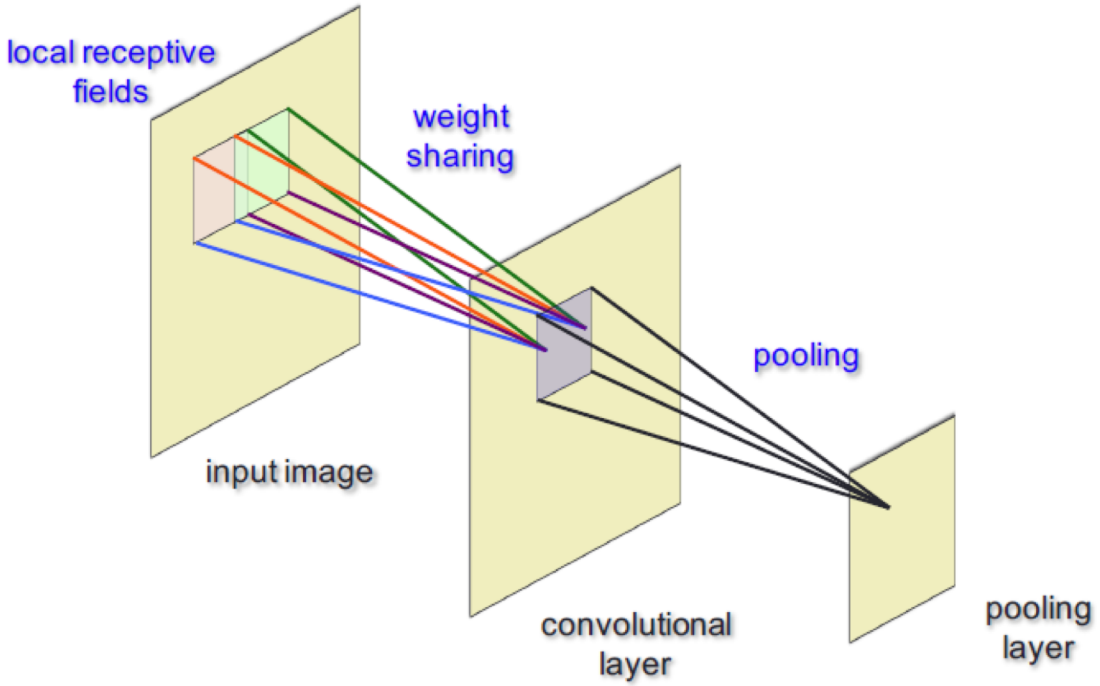
CNN Architecture: Pooling Layer

- Intuition: to progressively reduce the spatial size of the representation to **reduce the amount of parameters** and computation in the network, and hence to also **control overfitting**
- Pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value of the features in that region.

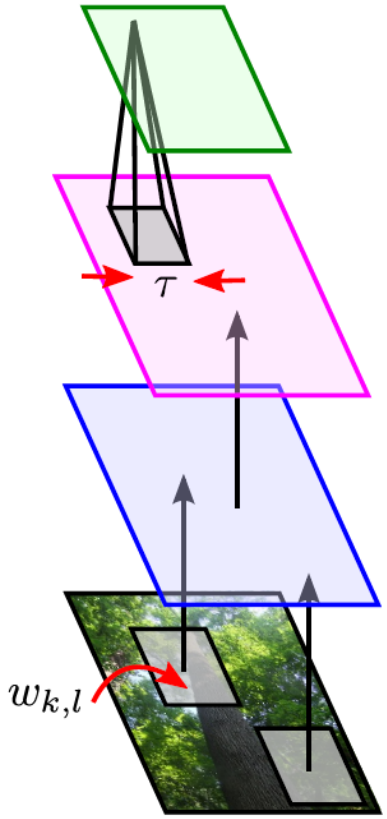


Pooling

- Common pooling operations:
  - Max pooling: reports the maximum output within a rectangular neighborhood.
  - Average pooling: reports the average output of a rectangular neighborhood (possibly weighted by the distance from the central pixel).



Building-blocks for CNN



$$x_{i,j} = \max_{|k|<\tau,|l|<\tau} y_{i-k,j-l}$$

mean or subsample also used

pooling stage

Feature maps of a larger region are combined.

$$y_{i,j} = f(a_{i,j})$$

e.g. $f(a) = [a]_+$

$f(a) = \text{sigmoid}(a)$

non-linear stage

Feature maps are trained with neurons.

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

Shared weights

convolutional stage

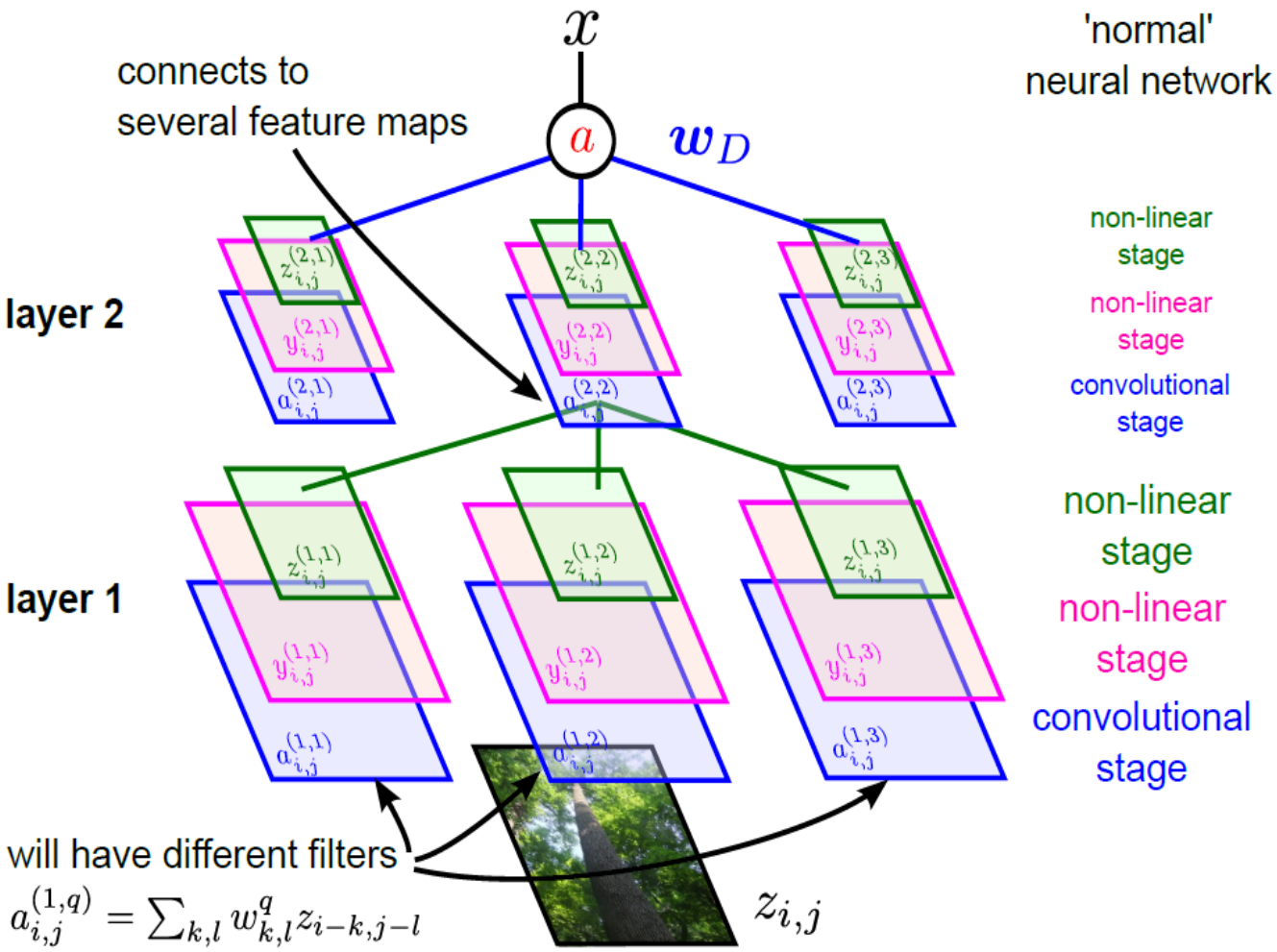
Each sub-region yields a feature map, representing its feature.

$z_{i,j}$

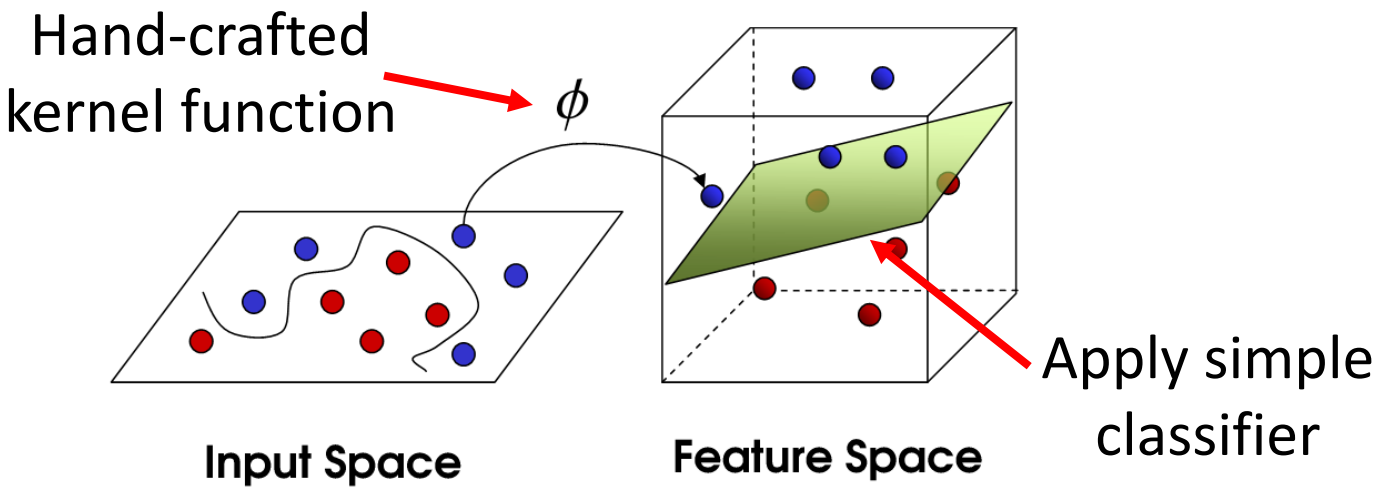
Images are segmented into sub-regions.

input image

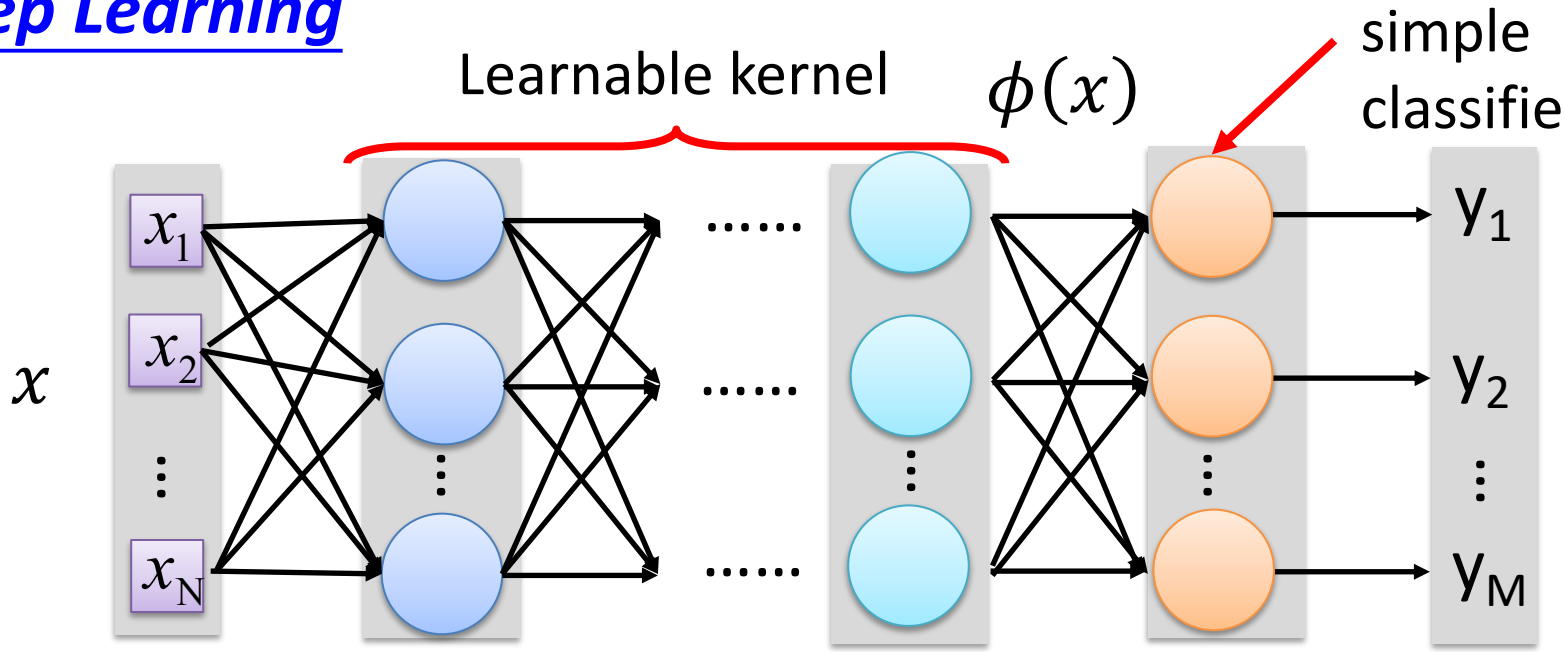
Full CNN


x
connects to
several feature maps
a
w_D
'normal'
neural network
layer 2
layer 1
non-linear
stage
non-linear
stage
convolutional
stage
non-linear
stage
non-linear
stage
convolutional
stage
will have different filters
$a_{i,j}^{(1,q)} = \sum_{k,l} w_{k,l}^{q} z_{i-k,j-l}$
$z_{i,j}$

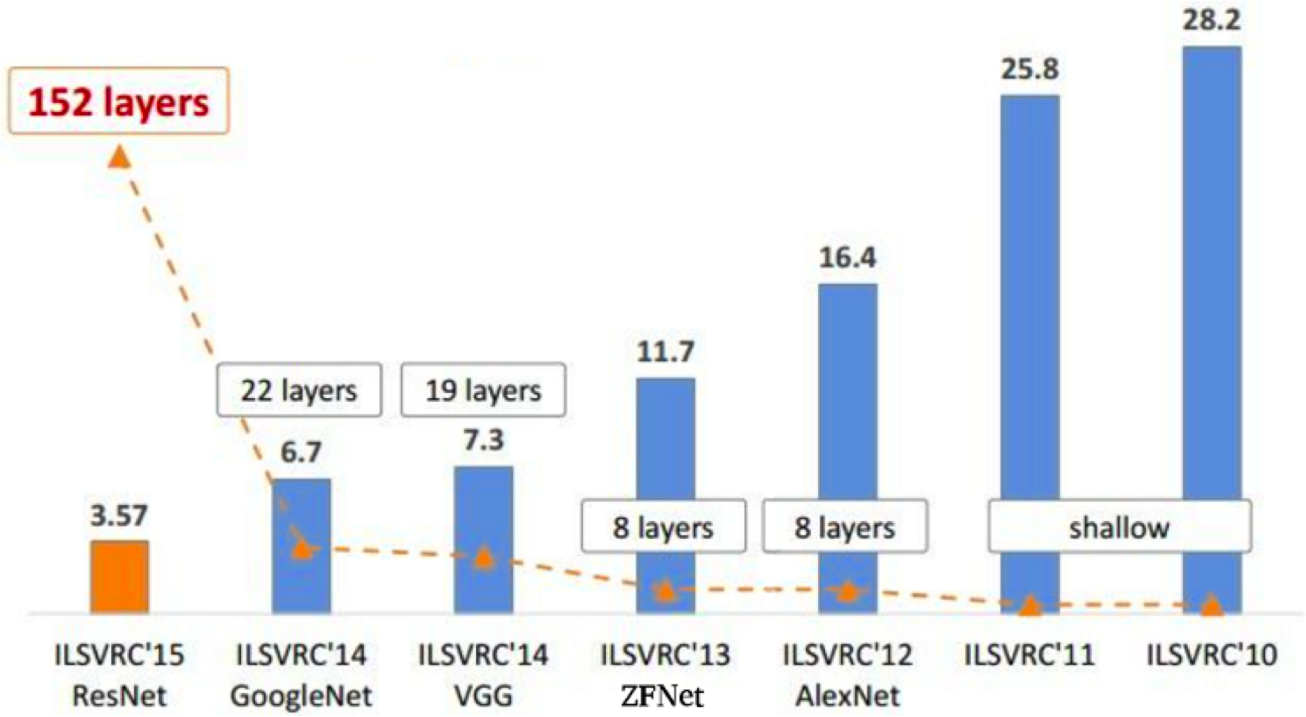
SVM



Deep Learning



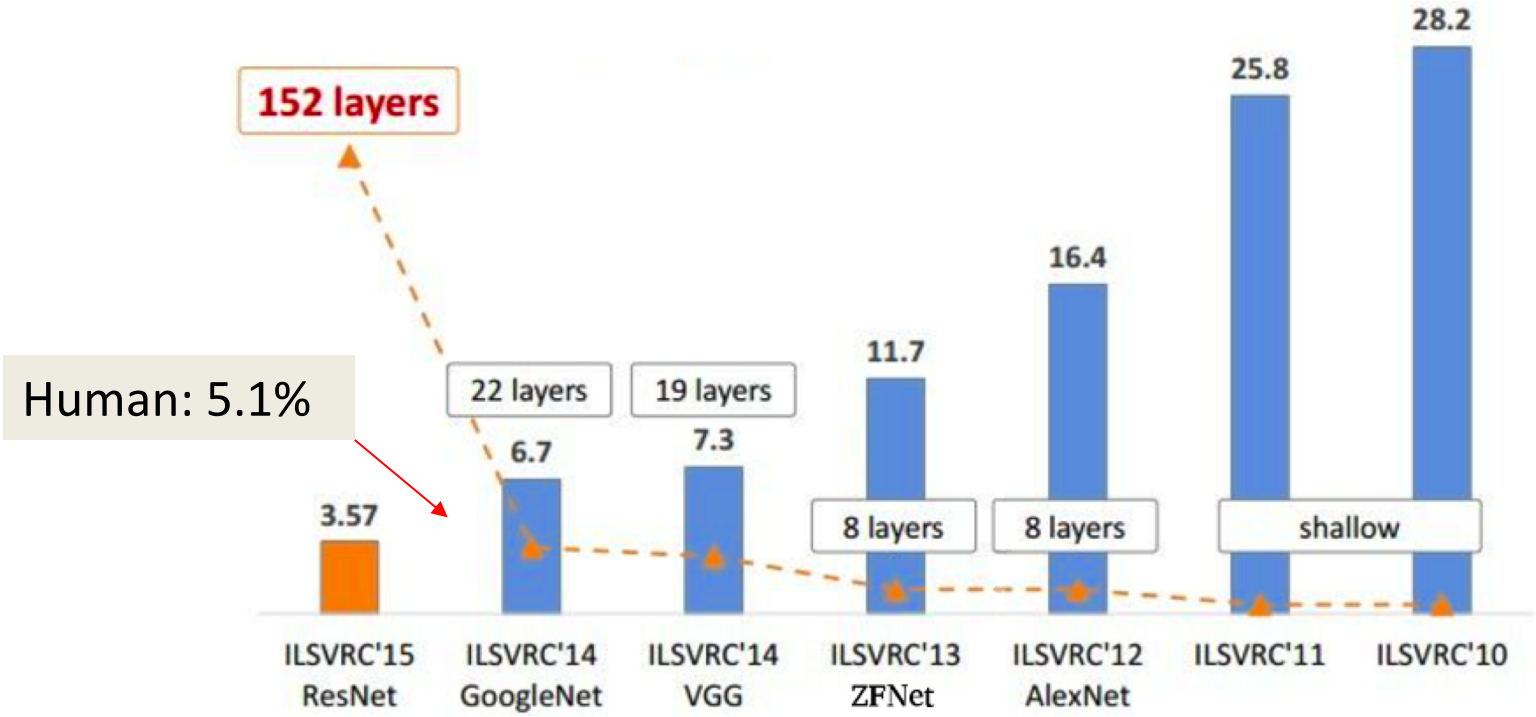
What is happening to PR field?



ImageNet Large Scale Visual Recognition Competition: top-5 error (%)

Figure source: Kaiming He

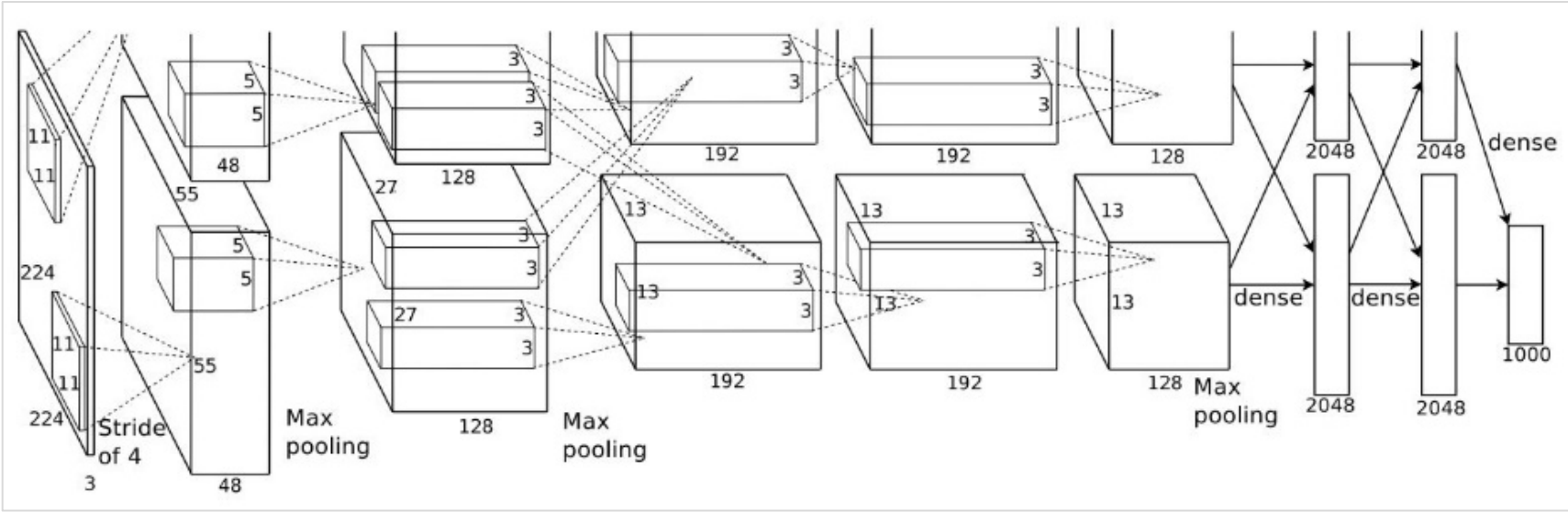
What is happening to PR field?



ImageNet Large Scale Visual Recognition Competition: top-5 error (%)

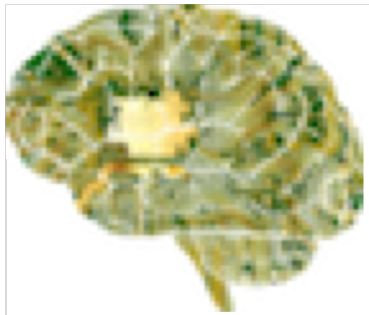
Figure source: Kaiming He

Deep CNN: ImageNet 2012 winner



- Multiple feature maps per convolutional layer.
- Multiple convolutional layers for extracting features at different levels.
- Higher-level layers take the feature maps in lower-level layers as input.

Tools for Deep Learning



Deep Learning

… moving beyond shallow machine learning since 2006!

http://deeplearning.net/software_links/

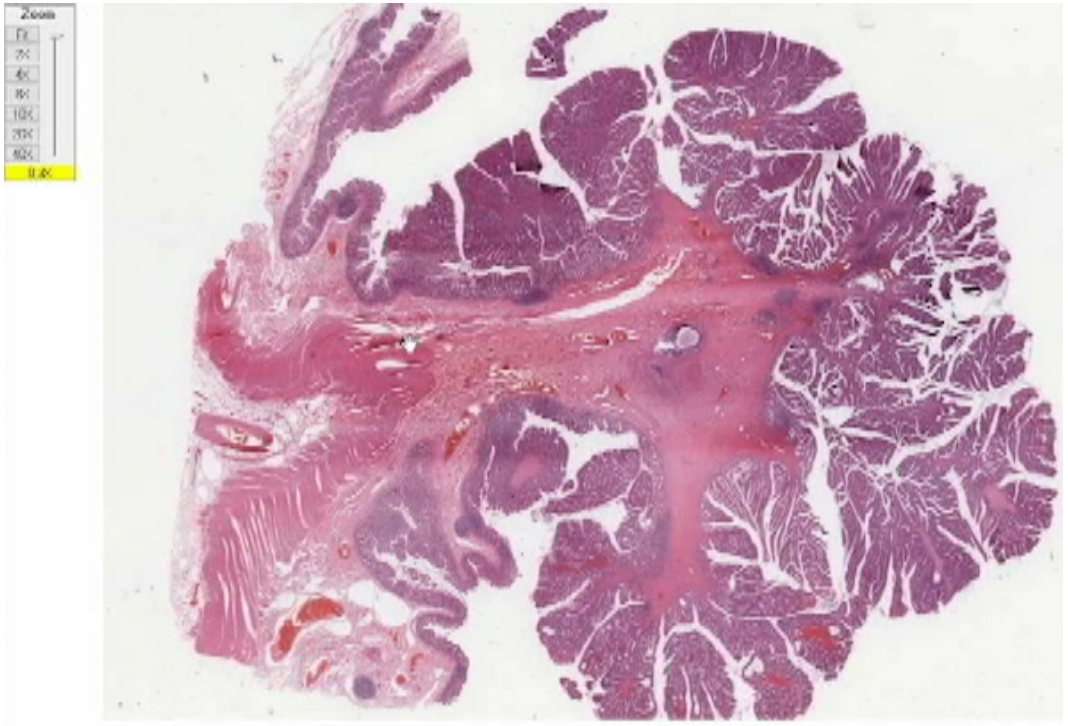
- Caffe (Python, Matlab)
- Theano (Python)
- Torch (Lua)
- TensorFlow (Python, C++)
- … many others

A practical example…

Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, ITALY

Histological assessment of cancer

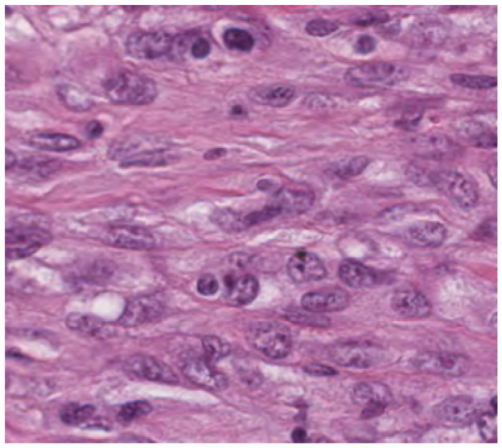
- Examination of the morphological characteristics of tissues under a microscope, the specimen having been sectioned, stained, and mounted on a slide.
- Unlike conventional microscopy, specimens are now digitalized by scanners under the form of **Whole Slide Images** (WSIs), **large multi-resolution images** (~Gigapixels), and then visually assessed by the pathologists

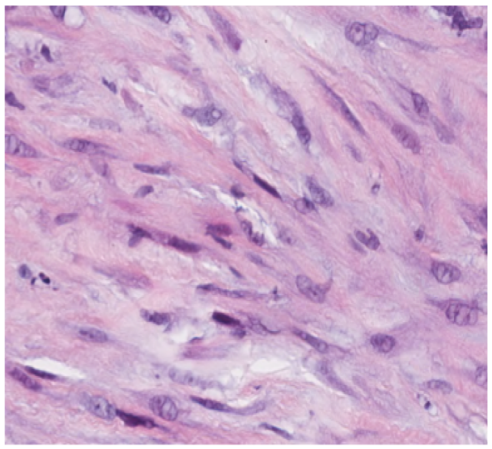


Histological assessment of cancer

- Gold standard for the primary diagnosis and differentiation of many cancers
- Histological examination can be seen as the assessment of image textures at different levels of detail
  - Tissue level: spatial arrangement of the cells in a tissue
  - Cell level: spatial arrangement of subcellular components

pleural mesothelioma





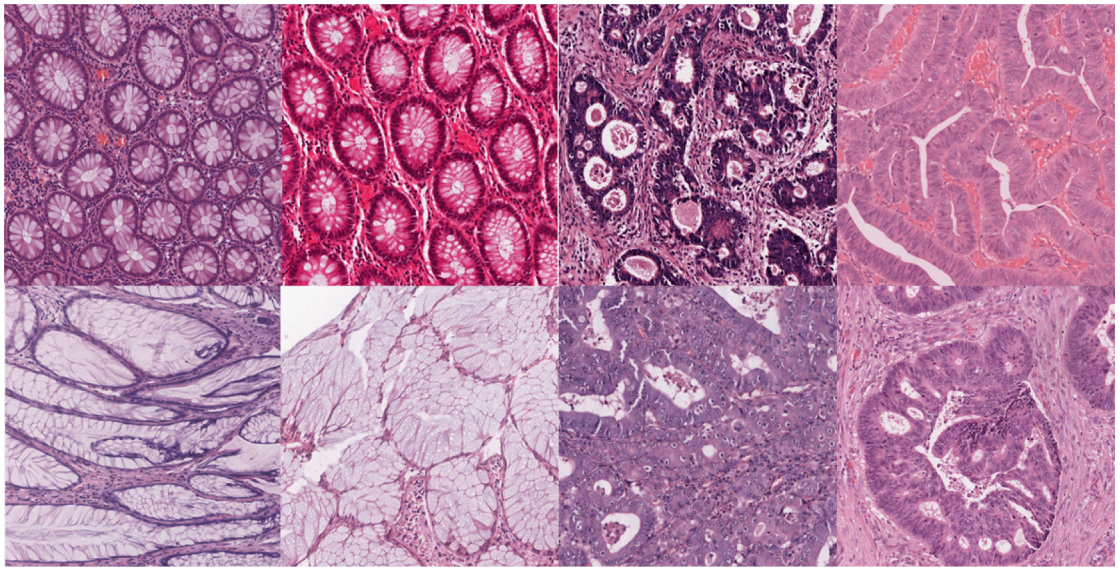
active pulmonary fibrosis

Histological assessment of cancer

- Visual assessment has two major drawbacks:
  - Extremely time-consuming
  - Highly subjective and affected by variability, both inter and intra observer
- Growing efforts to develop computer-aided diagnostic techniques for **automated histological assessment**
- The main challenge is the **variability** of the image characteristics
  1. Variability of tissue/cell morphology
  2. Inter-patients variability
  3. Variability of staining/slicing/acquisition process

Histological assessment of cancer

- The main challenge is the **variability** of the image characteristics
  - Variability of tissue/cell morphology
  - Inter-patient variability
  - Variability of staining/slicing/acquisition process

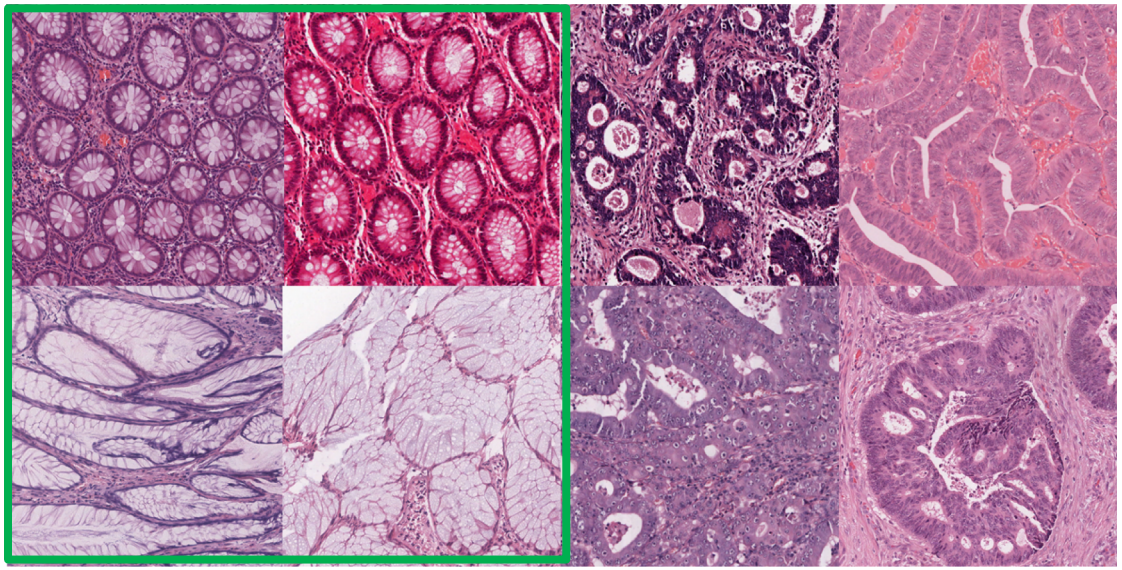


→ intra-class variability

Histological assessment of cancer

- The main challenge is the **variability** of the image characteristics
  - Variability of tissue/cell morphology
  - Inter-patient variability
  - Variability of staining/slicing/acquisition process

Healthy

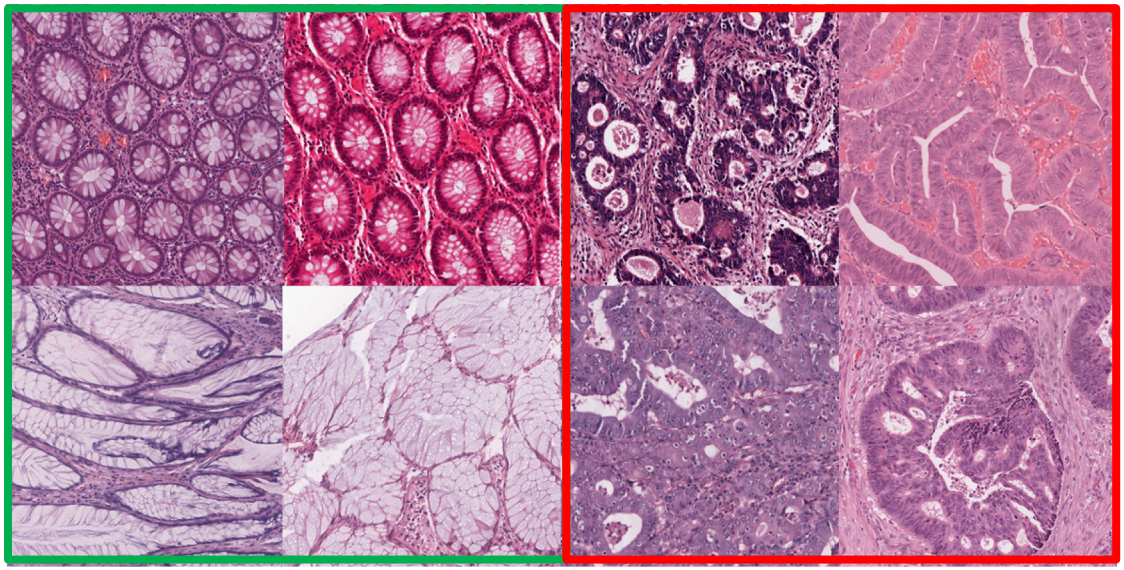


→ intra-class variability

Histological assessment of cancer

- The main challenge is the **variability** of the image characteristics
  - Variability of tissue/cell morphology
  - Inter-patient variability
  - Variability of staining/slicing/acquisition process

Healthy

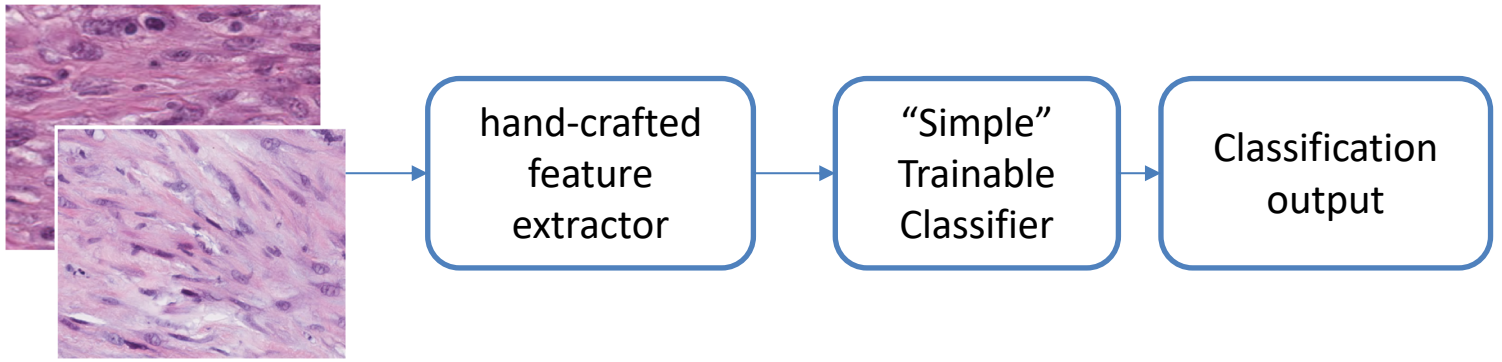


Cancer

→ intra-class variability

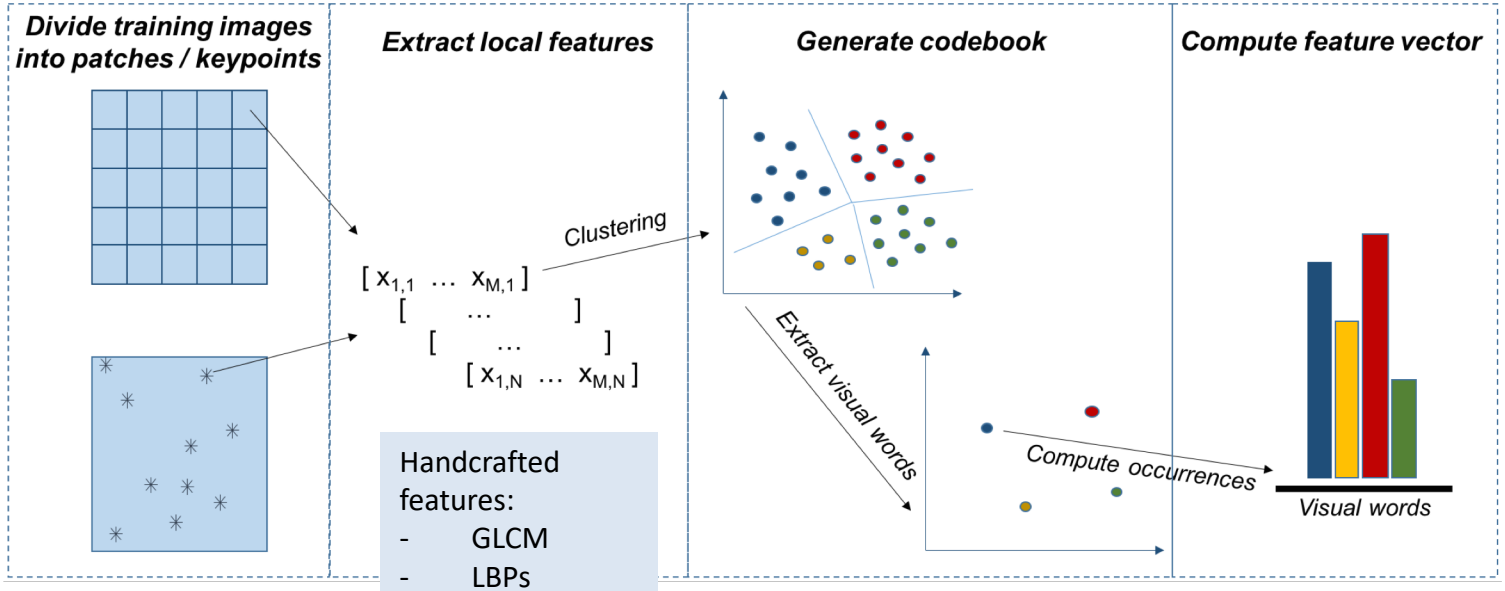
Traditional PR model

- Traditional pattern recognition models use hand-crafted features and relatively simple trainable classifier:



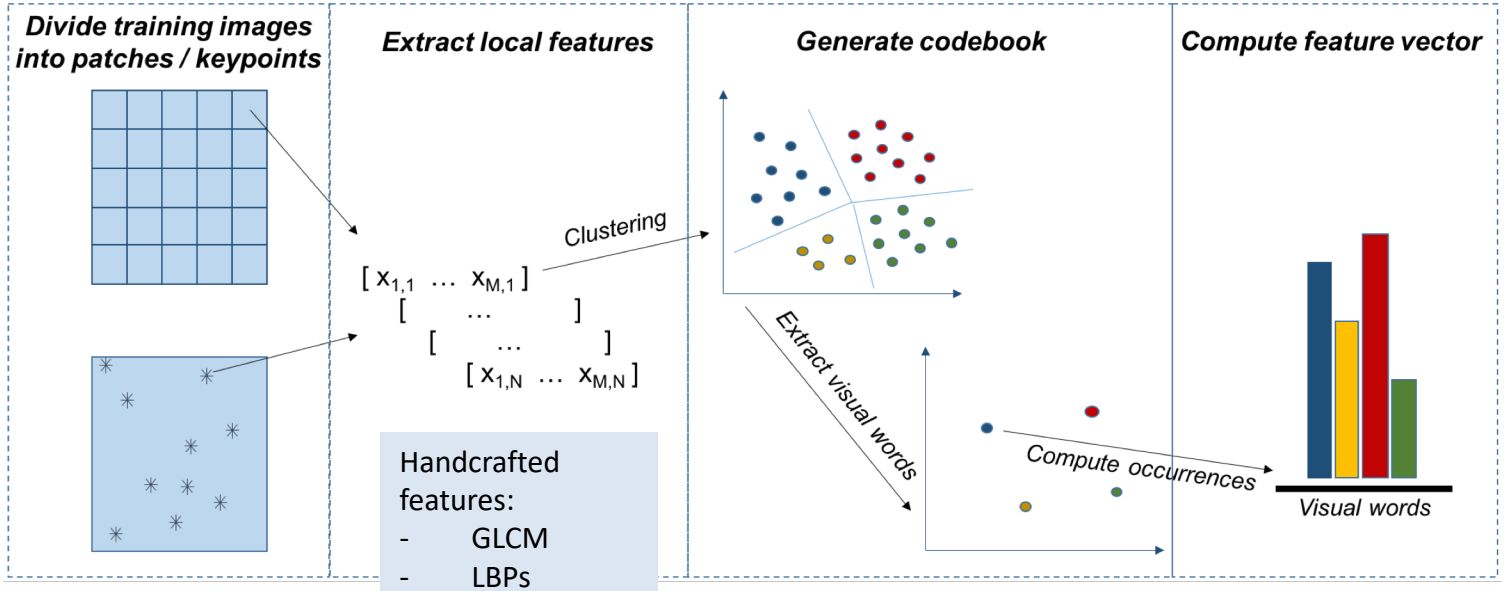
Hand-crafted feature extraction

- State of the art BoW paradigm:



Hand-crafted feature extraction

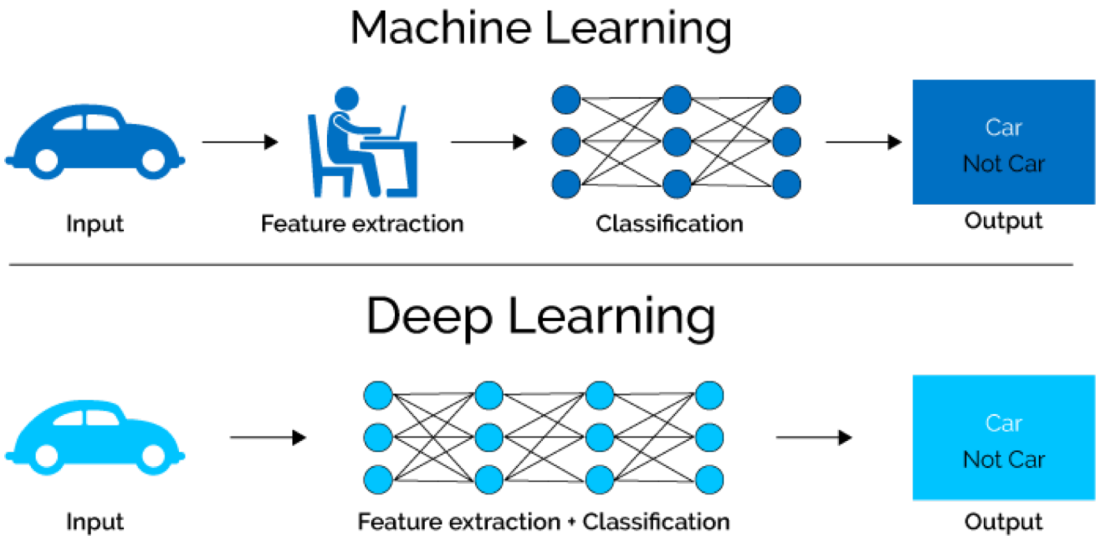
- State of the art BoW paradigm:



PROBLEM: hand-crafted features tend to be application-specific and cannot be easily generalized to different contexts

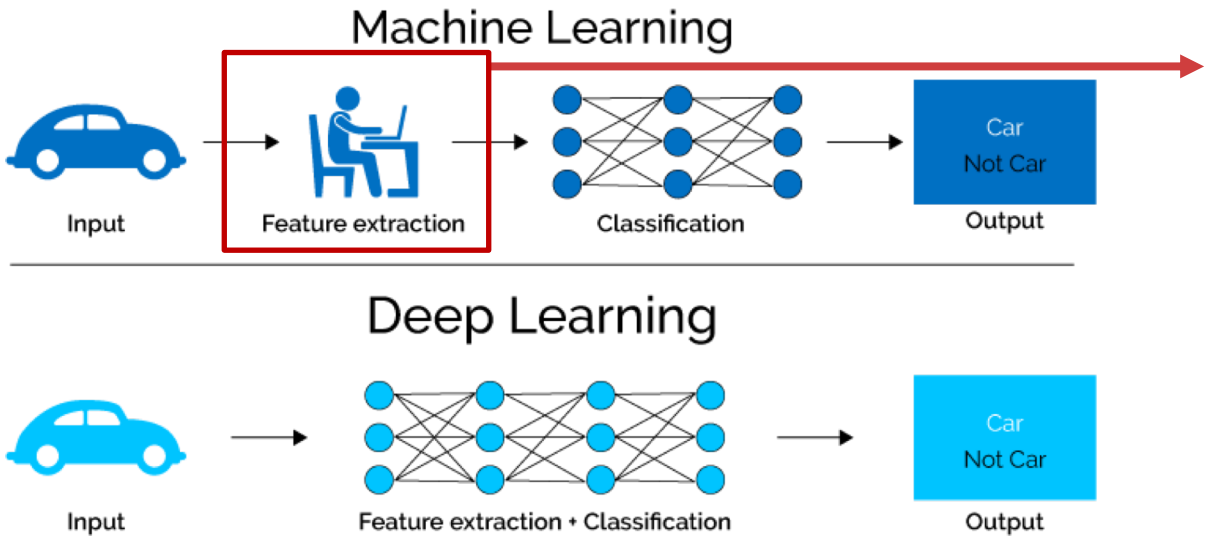
Alternative to feature design: deep learning approach

- DL architectures are composed by **multiple linear** and **non-linear transformations of the data**, with the goal of yielding more abstract and more useful representation
- DL **automatically finds out the features** important for classification, after consecutive hierarchical identification of complex concepts



Alternative to feature design: deep learning approach

- DL architectures are composed by **multiple linear** and **non-linear transformations of the data**, with the goal of yielding more abstract and more useful representation
- DL **automatically finds out the features** important for classification, after consecutive hierarchical identification of complex concepts

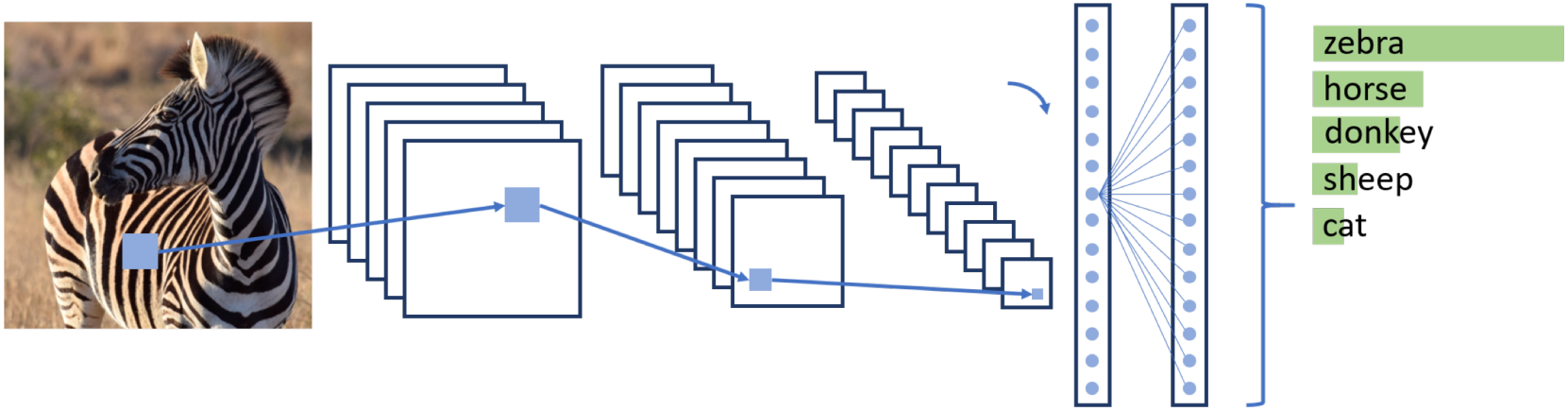


No need for features engineering

DL **derives suitable features** space directly **from training data**

Convolutional Neural Networks (CNNs)

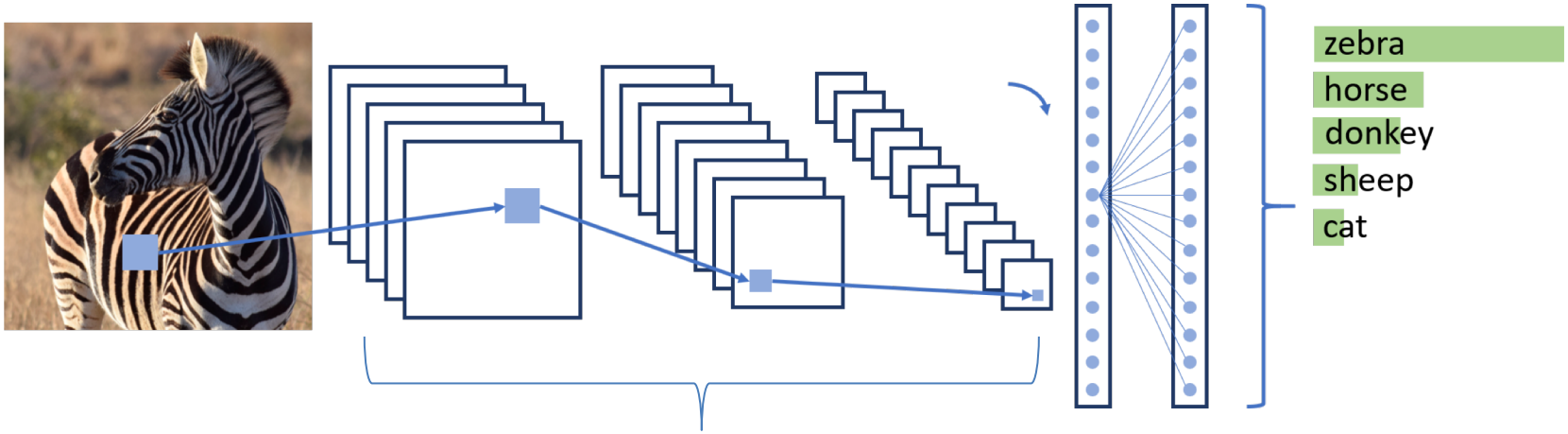
- Deep, feed-forward artificial neural networks, successfully applied to analyze and classify natural images.
- CNN architecture is inspired by connectivity pattern between neurons of the animal visual cortex



- It is made up of multiple locally connected trainable stages, piled one after the other, with two or more fully-connected layers as the last step

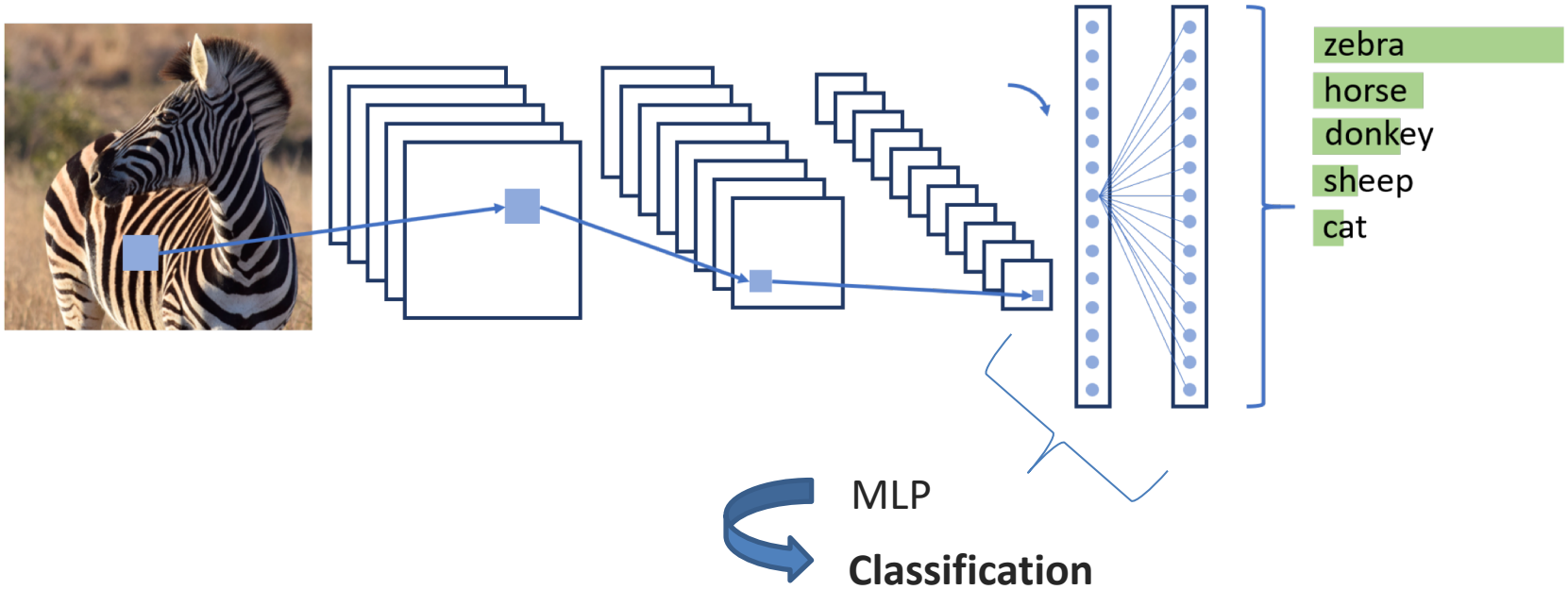
Convolutional Neural Networks (CNNs)

- The **first part** of the network is devoted to **learning the image representation**. Successive layers learn features at an increasing level of abstraction.



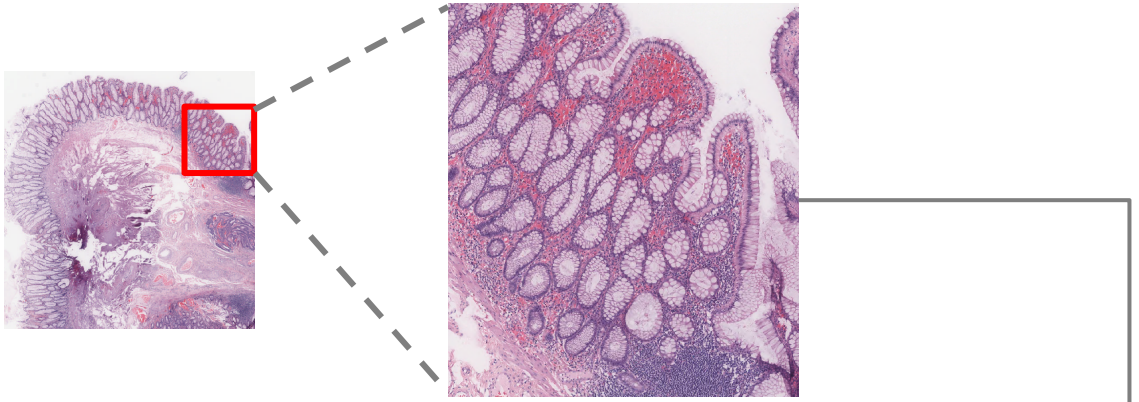
Convolutional Neural Networks (CNNs)

- The **last fully connected part** is devoted to **classification** and acts like a traditional multilayer perceptron (MLP).

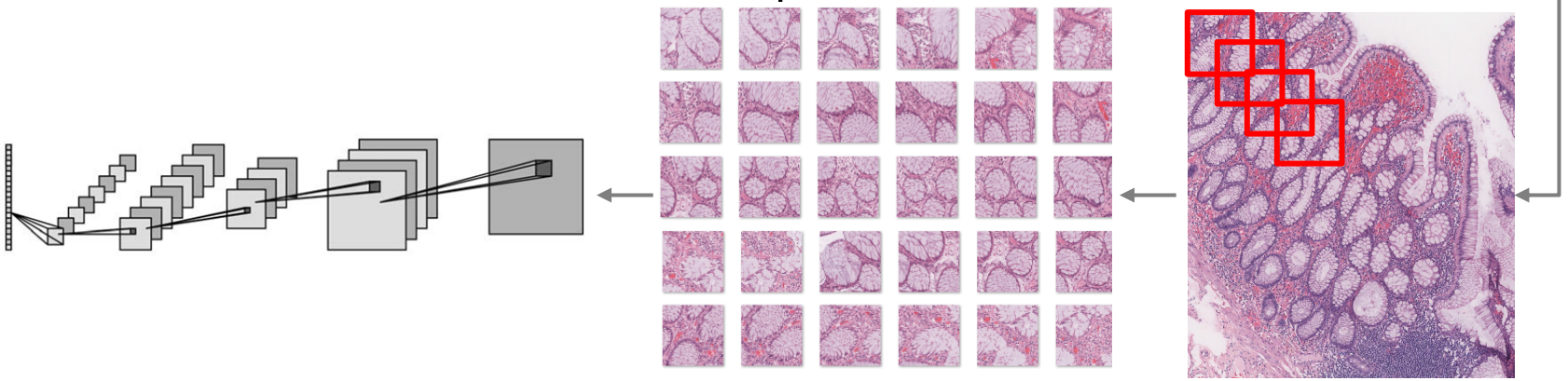


CNNs applied to WSIs

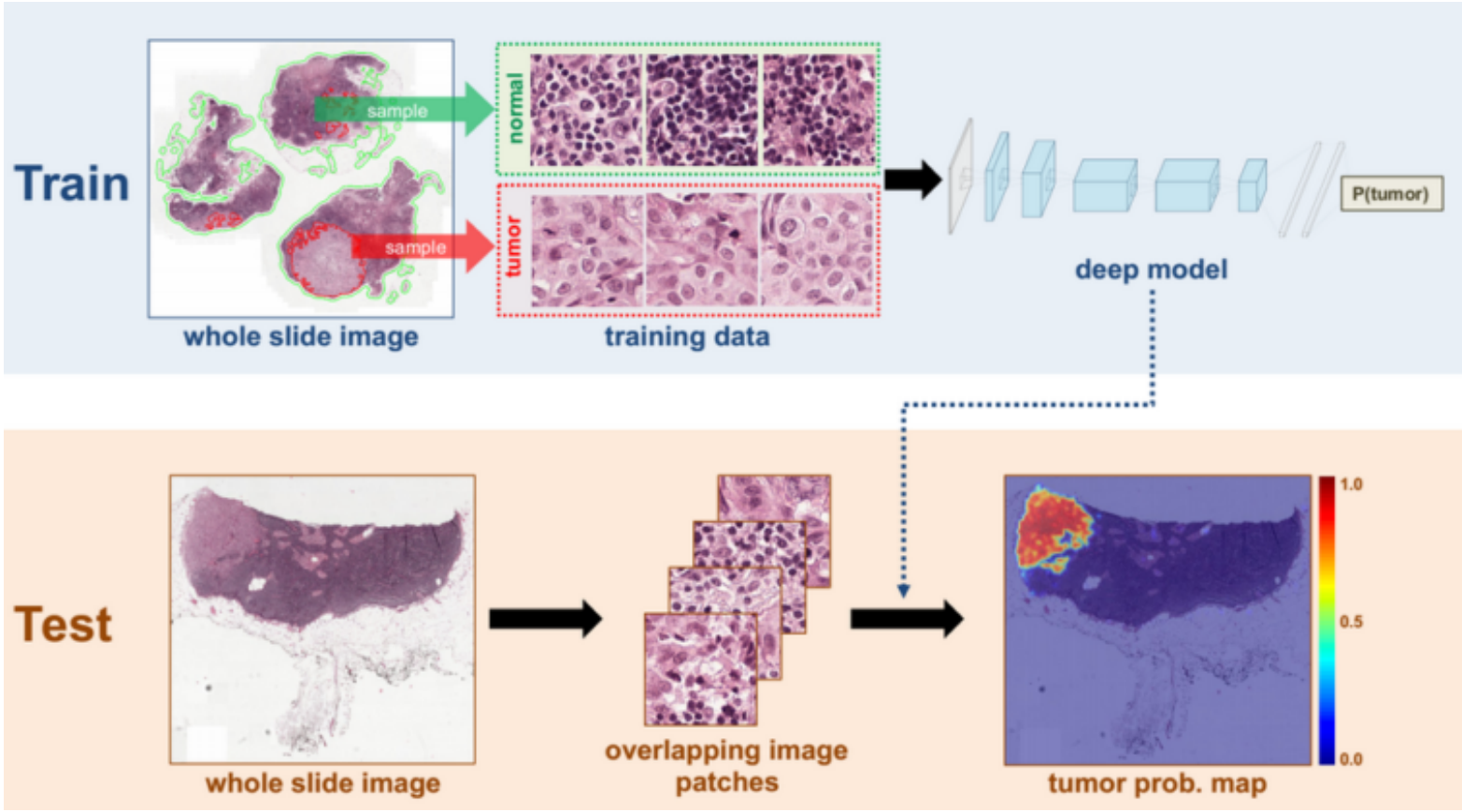
- Cropping of ROIs (based on the tissue of interest) on each WSI



- Each ROIs is divided into small patches



CNNs applied to WSIs



Are CNNs suitable for Digital Pathology?

Pros:

- ✓ DL extract **hierarchical knowledge from the data itself**.
- ✓ No need for handcrafted features design
- ✓ Potentially useful for knowledge discovery

Cons:

- x Need computational resources
- x Need huge **annotated** datasets for training

Are CNNs suitable for Digital Pathology?

Pros:

- ✓ DL extract **hierarchical knowledge from the data itself**.
- ✓ No need for handcrafted features design
- ✓ Potentially useful for knowledge discovery

Cons:

- x Need computational resources
- x Need huge **annotated** datasets for training

GPUs

Are CNNs suitable for Digital Pathology?

Pros:

- ✓ DL extract **hierarchical knowledge from the data itself**.
- ✓ No need for handcrafted features design
- ✓ Potentially useful for knowledge discovery

Cons:

- x Need computational resources
- x Need huge **annotated** datasets for training

Not so easy to find in every-day clinic
Not always reliable

Are CNNs suitable for Digital Pathology?

Pros:

- ✓ DL extract **hierarchical knowledge from the data itself**.
- ✓ No need for handcrafted features design
- ✓ Potentially useful for knowledge discovery

Cons:

- x Need computational resources
- x Need huge **annotated** datasets for training

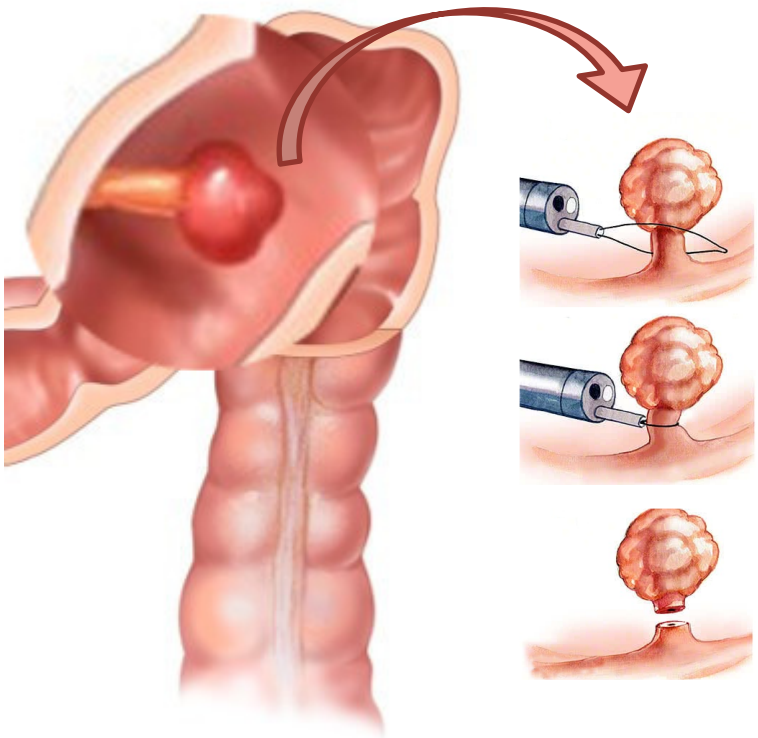
Not so easy to find in every-day clinic
Not always reliable

TRANSFER LEARNING?

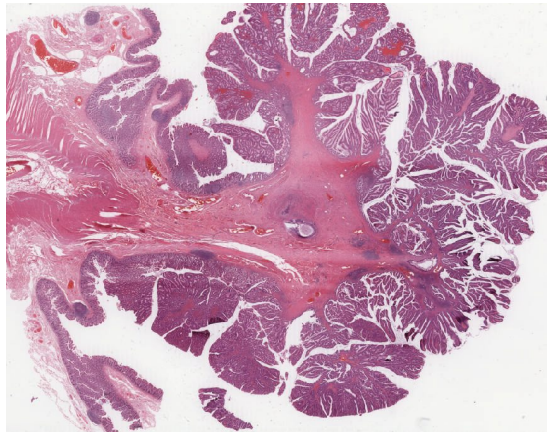
At least the features learnt by the lower level layers should be generalizable to different applications → We can try to use/adapt CNNs pre-trained on different applications

Case study: assessment of colorectal biopsies

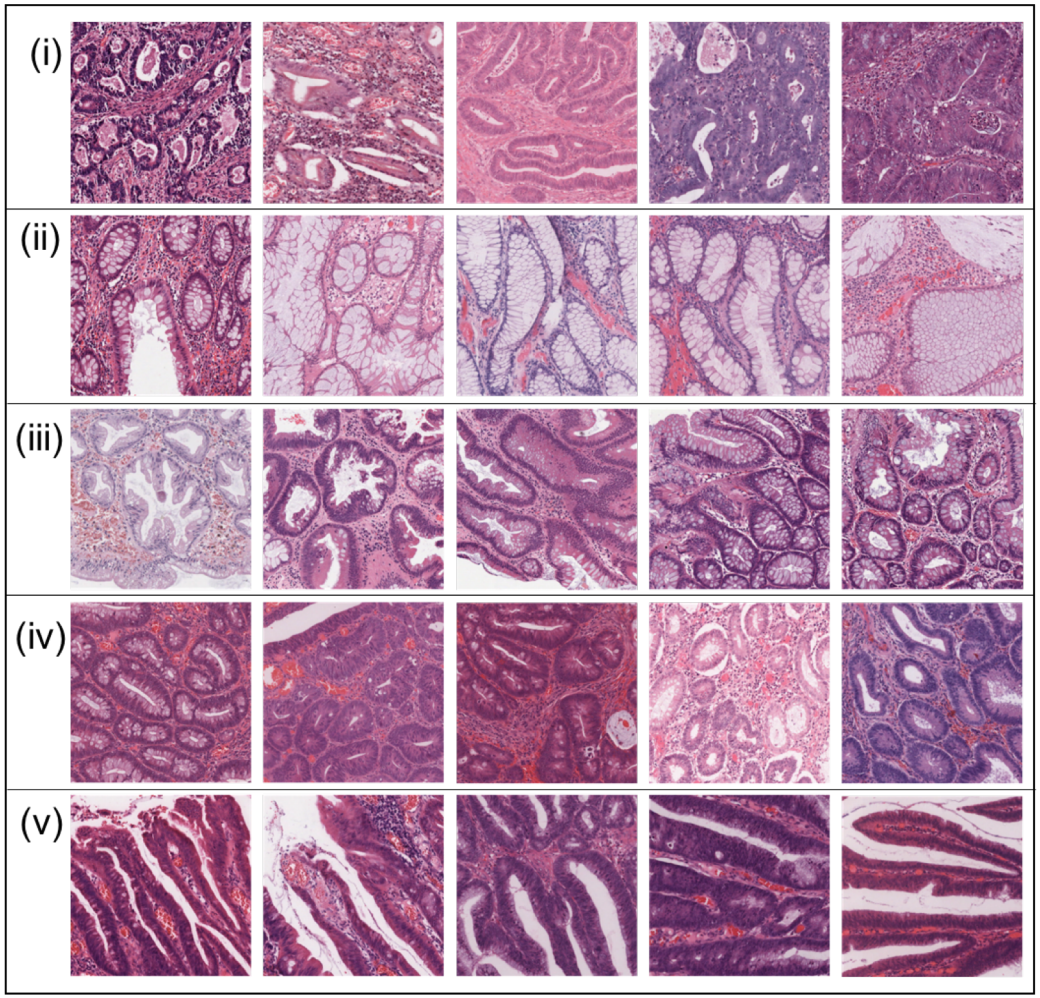
- CRC **originates** from the most internal layer of the intestinal wall **as an abnormal tissue growth** called **polyp or adenoma** whose neoplastic cells tend to infiltrate the surrounding tissues



- The lesion is removed during colonoscopy
- Tissue sample is
  i. Sectioned via microtome
  ii. Fixed in formalin
  iii. Stained (H&E)



Case study: assessment of colorectal biopsies



So far, 5 classes of interest:

(i) Adenocarcinoma, AC;

(ii) Hyperplastic polyp, H;

(iii) Serrated adenoma, S;

(iv) Tubular adenoma, T;

(v) Villous adenoma, V

10k+ patches, but only 40 independent WSIs

Full training

patch-wise classification accuracy

per-patient patch score

depth

MODEL	PATCH SCORE	PATIENT SCORE
SimpleNet	0.6	0.63 +- 0.43
LeNet	0.74	0.79 +- 0.28
AlexNet	0.74	0.79 +- 0.27
Vgg16	0.69	0.76 +- 0.37
Inception	0.67	0.70 +- 0.4
ResNet	0.68	0.75 +- 0.34

Full training

patch-wise classification accuracy

per-patient patch score

depth

MODEL	PATCH SCORE	PATIENT SCORE
SimpleNet	0.6	0.63 +- 0.43
LeNet	0.74	0.79 +- 0.28
AlexNet	0.74	0.79 +- 0.27
Vgg16	0.69	0.76 +- 0.37
Inception	0.67	0.70 +- 0.4
ResNet	0.68	0.75 +- 0.34

BoW + SVM	0.83	0.75 +- 0.14

Full training

patch-wise classification accuracy

per-patient patch score

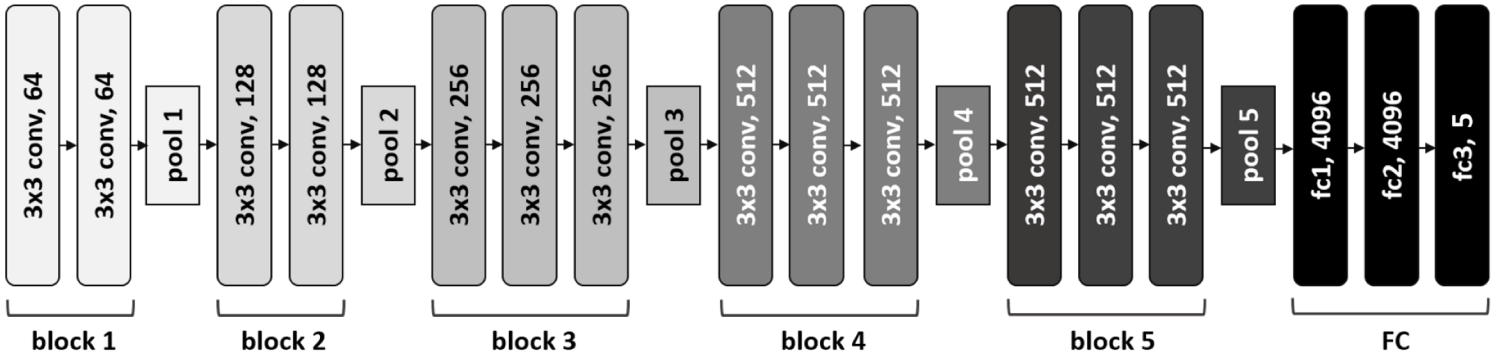
depth

MODEL	PATCH SCORE	PATIENT SCORE
SimpleNet	0.6	0.63 +- 0.43
LeNet	0.74	0.79 +- 0.28
AlexNet	0.74	0.79 +- 0.27
Vgg16	0.69	0.76 +- 0.37
Inception	0.67	0.70 +- 0.4
ResNet	0.68	0.75 +- 0.34

BoW + SVM	0.83	0.75 +- 0.14

Transfer learning

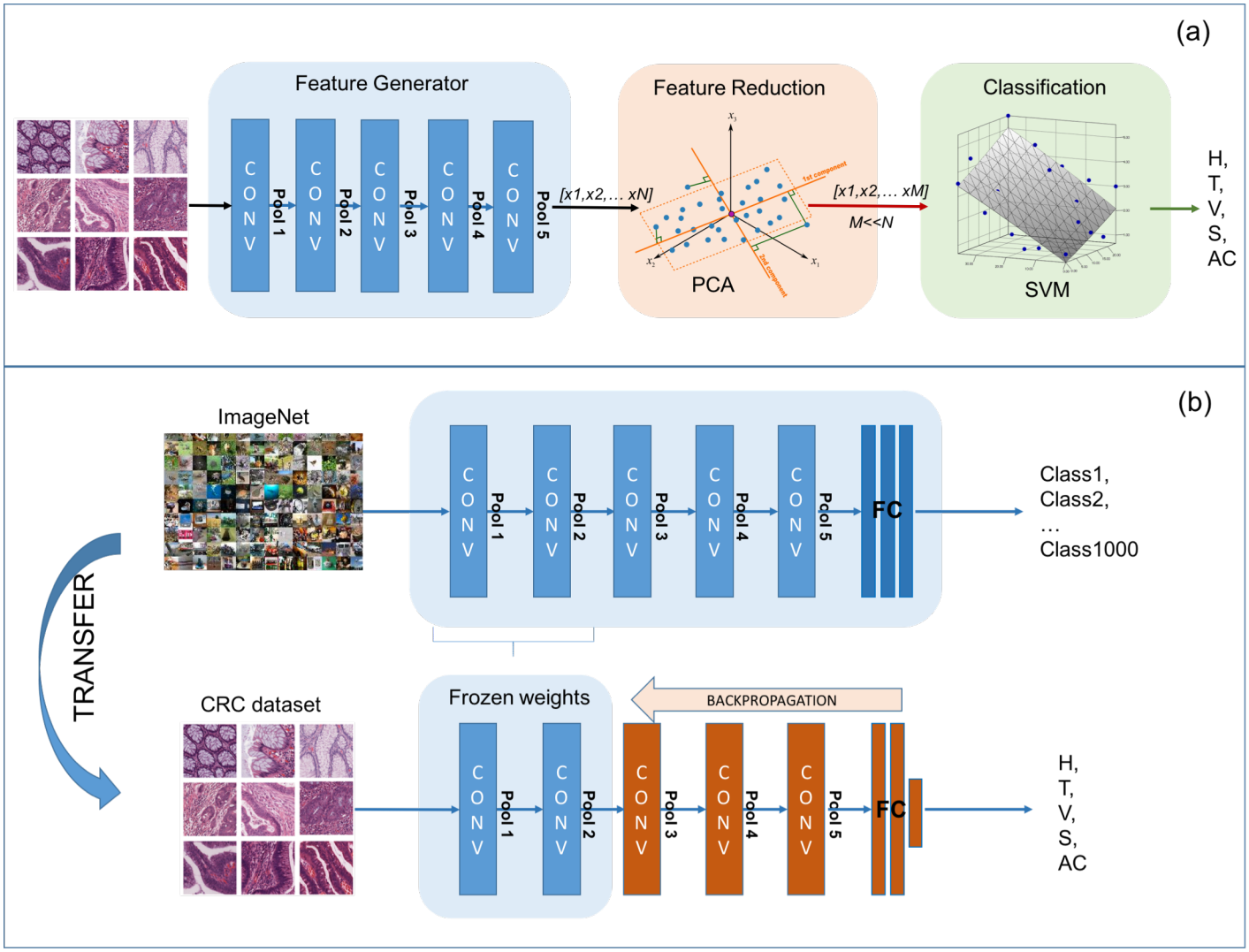
- VGG16 model:



- Pre-trained on the **ImageNet 2012 dataset** (1.2 million photographs from1000 different categories of natural objects).

Content and characteristics are completely different from our target!

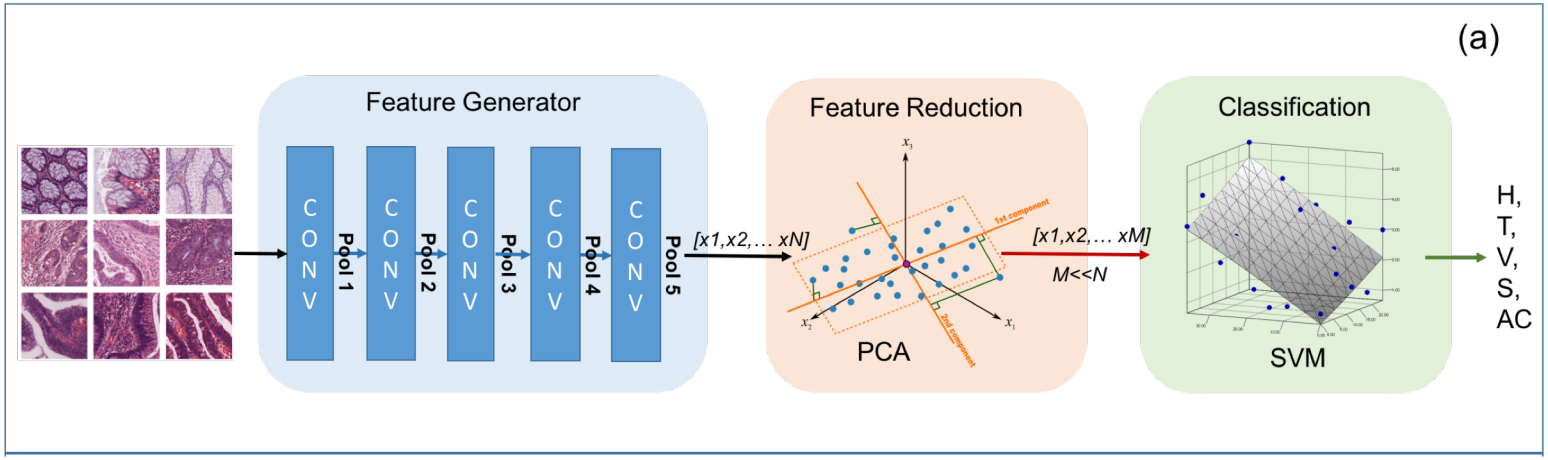
Transfer learning



CNN as features generator

CNN fine tuning

CNN as a feature generator



- Optimization of the block used as feature generator
- PCA features reduction
- SVM classification (RBF kernel)

CNN fine tuning

COMPLETE fine-tuning

- The network is initialized with weights learnt on the ImageNet, then re-trained on the CRC dataset

CNN fine tuning

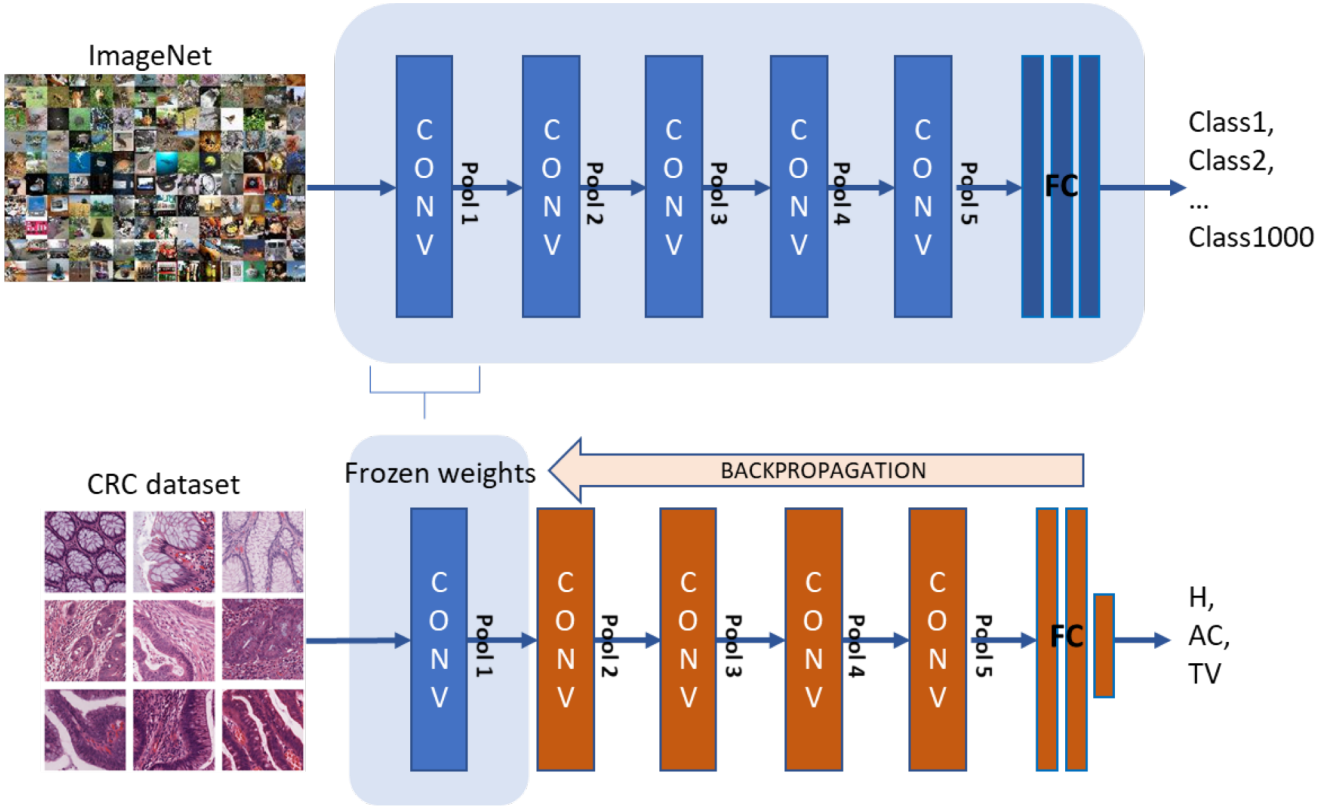
COMPLETE fine-tuning

- The network is initialized with weights learnt on the ImageNet, then re-trained on the CRC dataset

PARTIAL fine-tuning

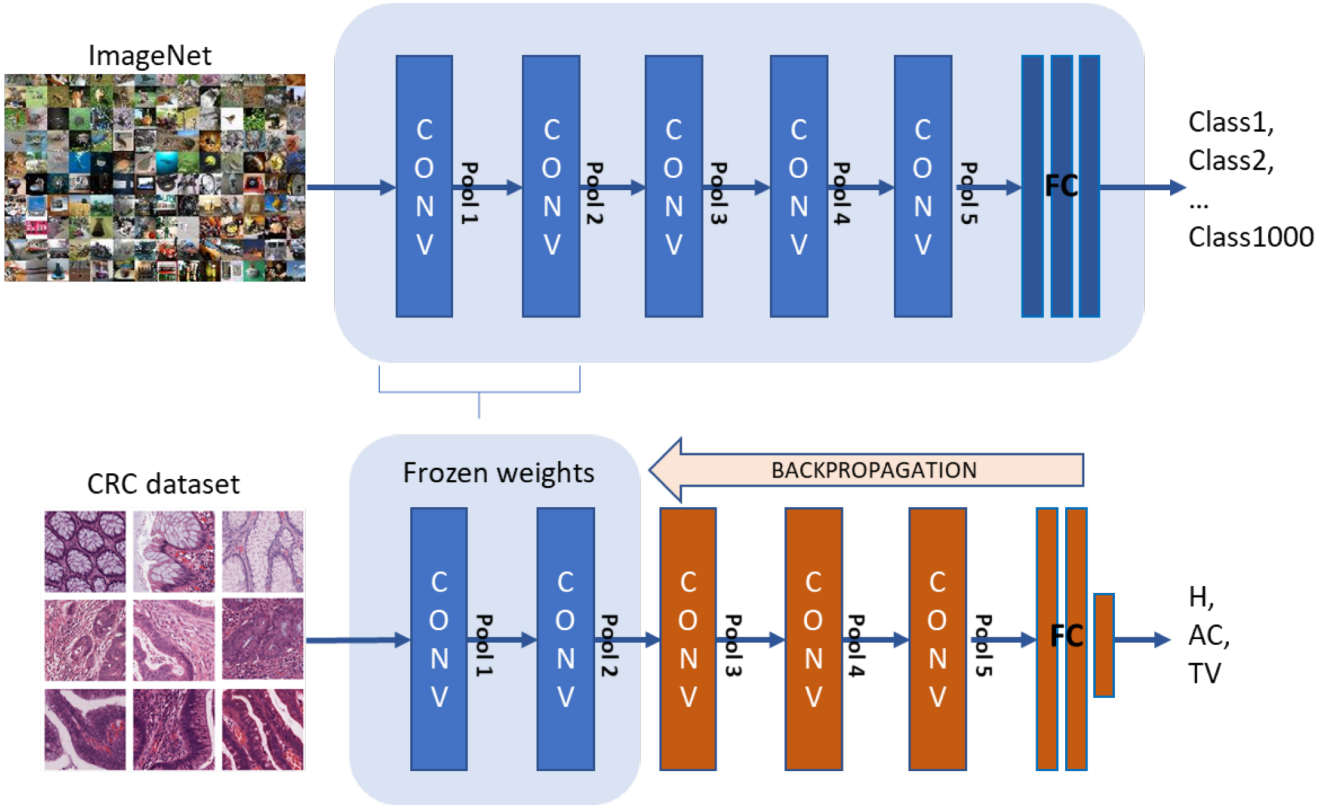
- The network is initialized with weights learnt on the ImageNet
- The network is re-trained on the CRC dataset, keeping the weights of the first layers frozen to their initial values

CNN fine tuning



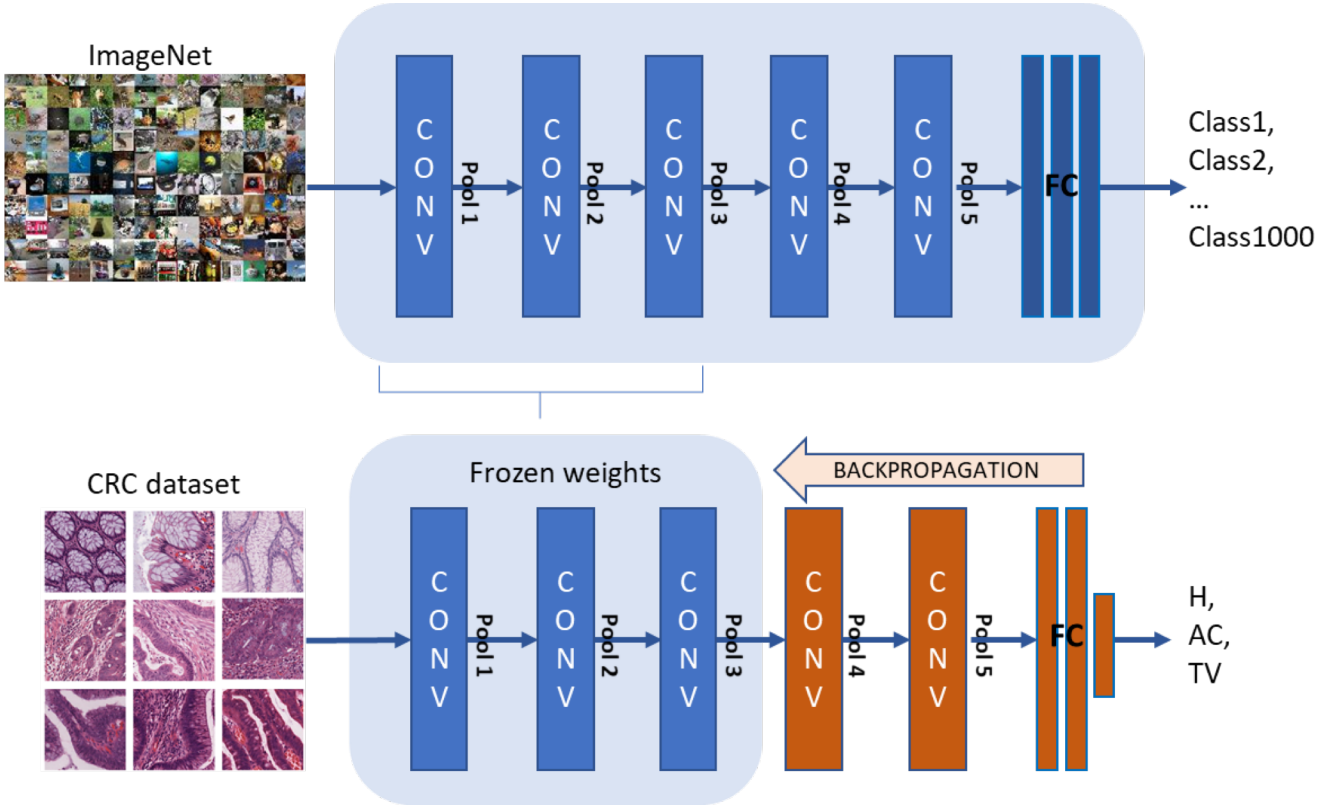
- Different configuration for the fine tuning by changing the starting block for the backpropagation algorithm

CNN fine tuning



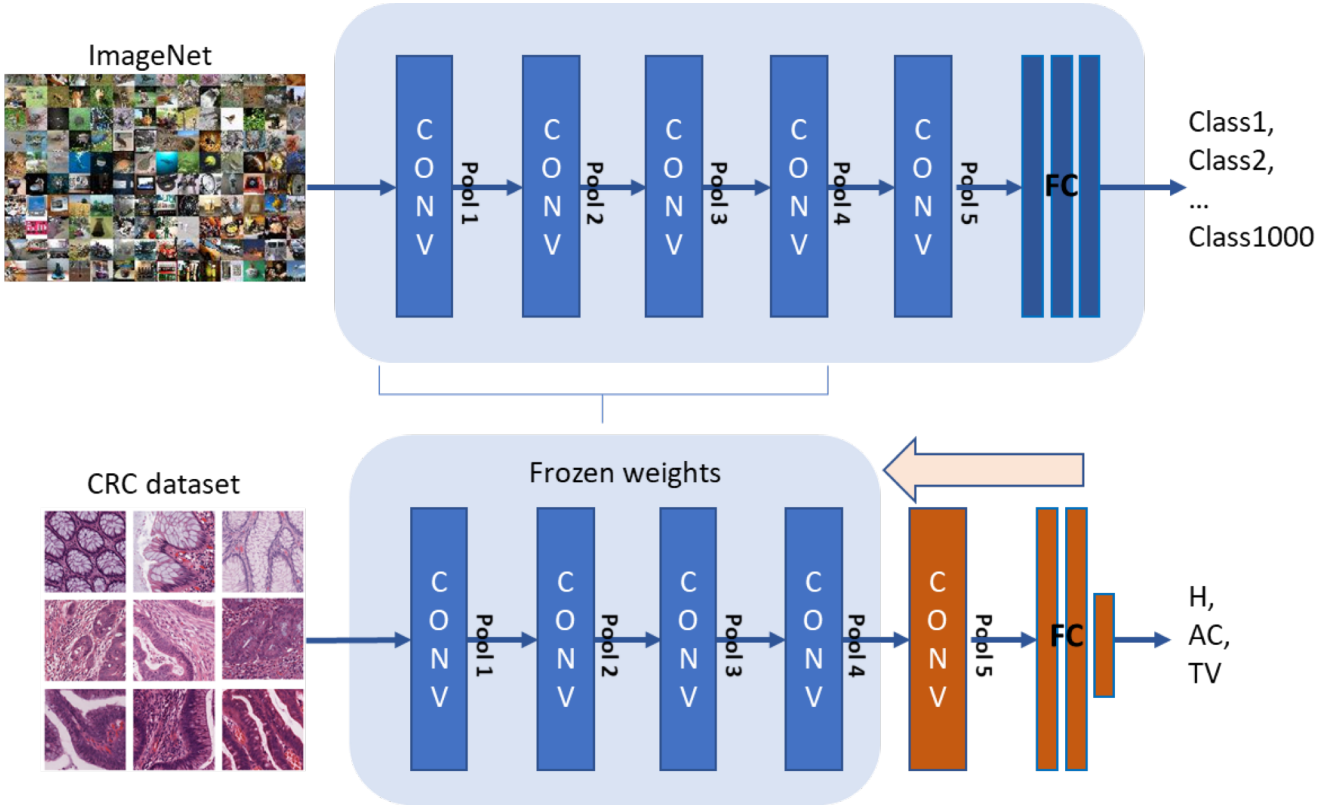
- Different configuration for the fine tuning by changing the starting block for the backpropagation algorithm

CNN fine tuning



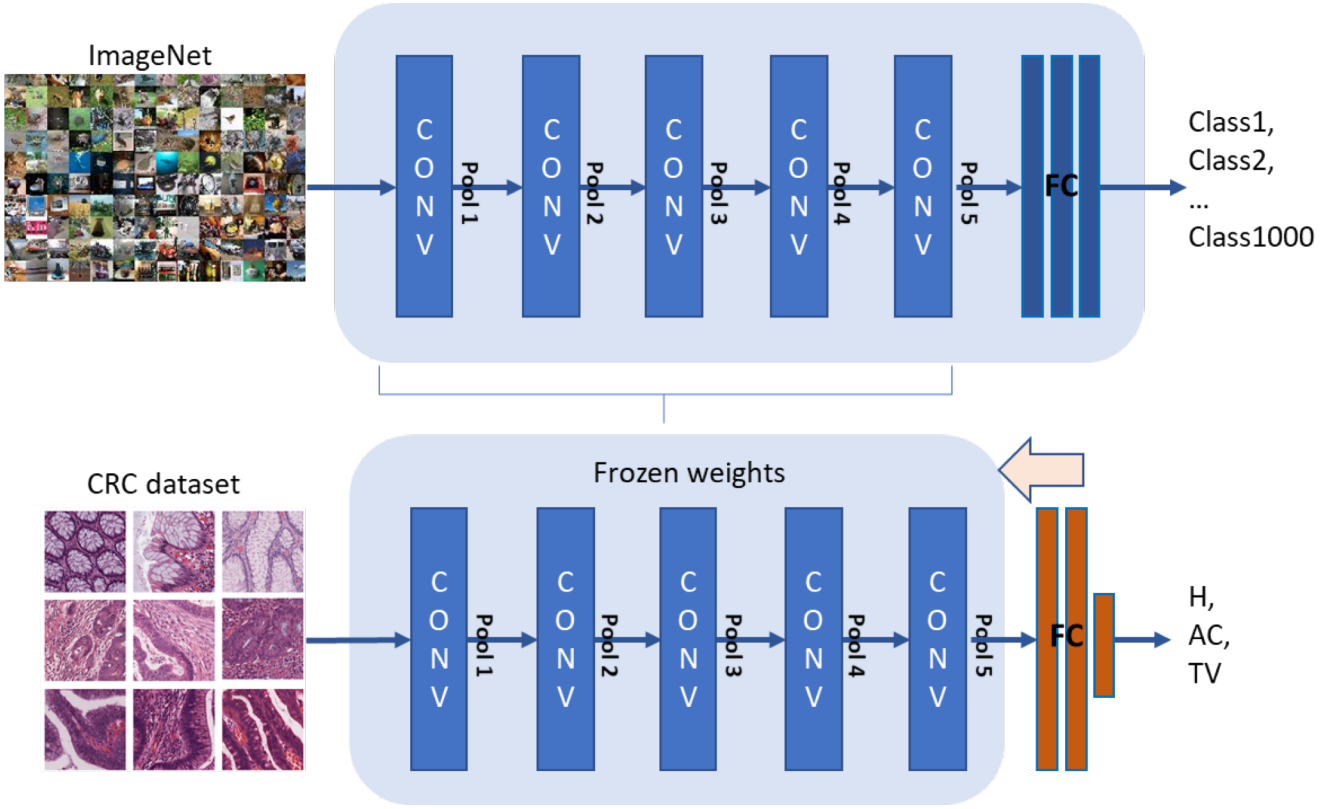
- Different configuration for the fine tuning by changing the starting block for the backpropagation algorithm

CNN fine tuning



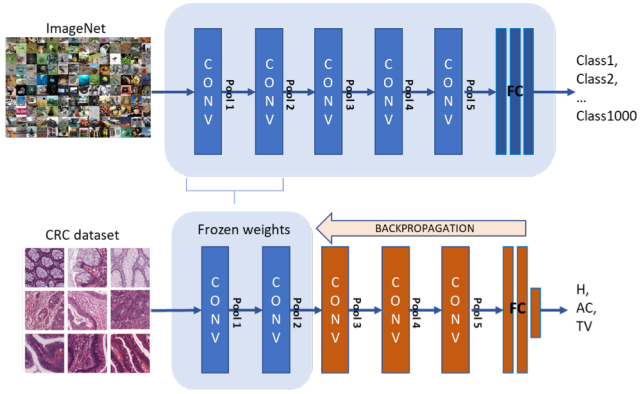
- Different configuration for the fine tuning by changing the starting block for the backpropagation algorithm

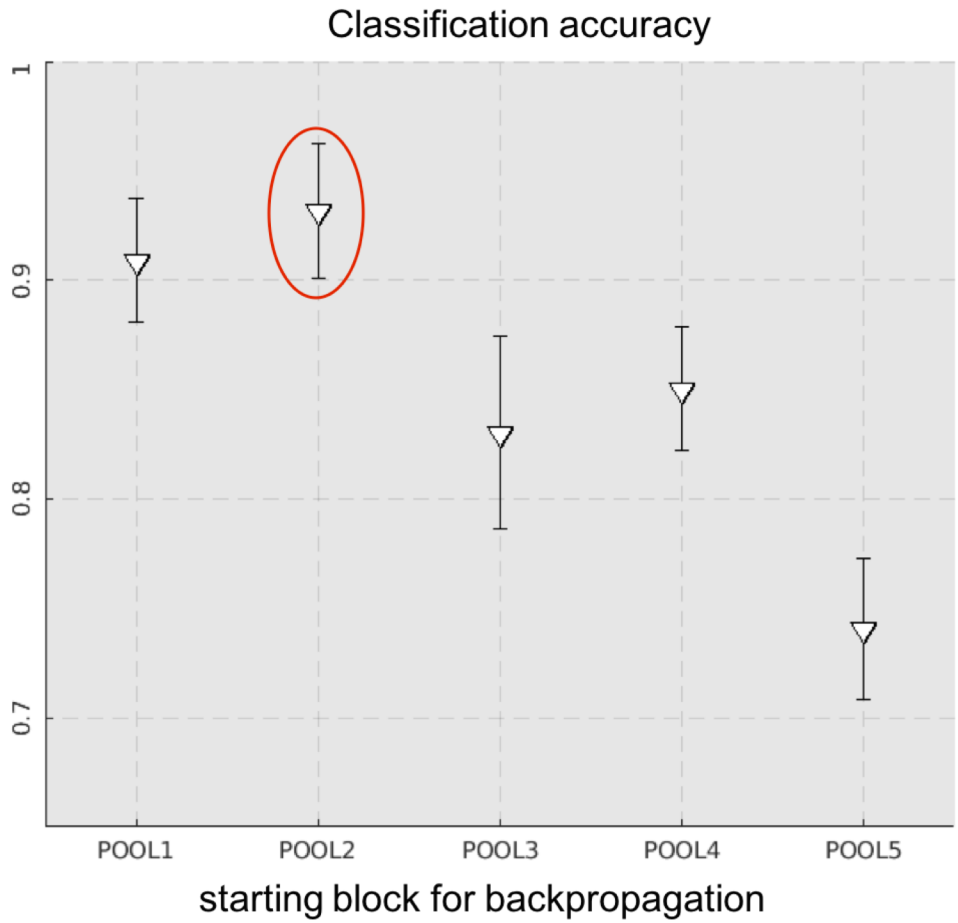
CNN fine tuning



- Different configuration for the fine tuning by changing the starting block for the backpropagation algorithm

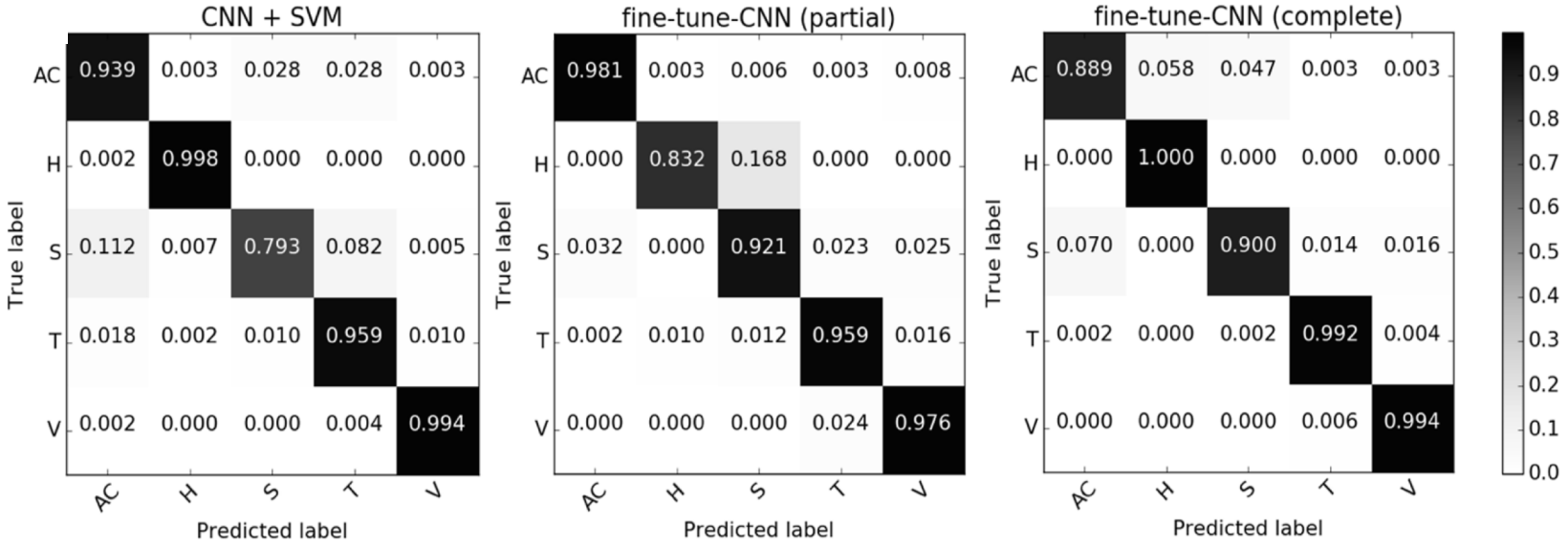
CNN fine-tuning





CNN transfer learning

Model	PATCH SCORE	PATIENT SCORE
CNN + SVM	0.93	0.95 +- 0.07
Fine-tuning (partial)	0.93	0.93 +- 0.08
Fine-tuning (complete)	0.96	0.96 +- 0.08



CNN transfer learning

Both transfer learning approaches overcome full training. This has two implications:

- **Low-level features** learnt by the CNN **can be successfully transferred** to a **totally different domain**
- TL is a good option to overcome the paucity of labelled data in everyday clinic, while considerably reducing the computational efforts

Ongoing- future works:

- Building general-purpose CNN architecture for histopathology
- Pleural mesothelioma: rare asbestos-related form of cancer
- Studying the the feature maps to have deeper insights on features that are relevant to detect cancer