Medical Image Processing for Interventional Applications

Rosenblatt's Perceptron

Online Course – Unit 25
Andreas Maier, Dieter Hahn, Frank Schebesch
Pattern Recognition Lab (CS 5)

virtuelle hochschule bayern

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Topics

Objective Function

Assumption: Classes are linearly separable.
Goal: We want to compute a linear decision boundary.

Therfore, we compute a hyperplane that minimizes the distance of misclassified feature vectors to the decision boundary:

- Class labels are $y \in \{-1, +1\}$.
- The decision boundary is a linear function:

$$y^* = \operatorname{sgn}(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{x} + \alpha_0).$$

- Parameters α_0 and $\boldsymbol{\alpha}$ are chosen according to the optimization problem:

$$\text{minimize} \quad D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot \left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{x}_i + \alpha_0\right),$$

  where $\mathcal{M}$ includes the misclassified feature vectors.

Objective Function: Remarks

- The elements of the sum in the objective function depend on the set of misclassified feature vectors $\mathcal{M}$.
- In each iteration step the cardinality of $\mathcal{M}$ might change.
- The cardinality of $\mathcal{M}$ is a discrete variable.
- We have competing variables:
  - continuous parameters of linear decision boundary,
  - discrete cardinality of $\mathcal{M}$.

Minimization of Objective Function

The gradient of the objective function

$$D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot (\boldsymbol{\alpha}^\mathsf{T} \boldsymbol{x}_i + \alpha_0)$$

is given by the partial derivatives:

$$\frac{\partial}{\partial \alpha_0} D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i,$$

$$\frac{\partial}{\partial \boldsymbol{\alpha}} D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot \boldsymbol{x}_i.$$

Minimization of Objective Function

We want to take an update step right after having visited each misclassified observation.

The update rule in the $(k+1)$-st iteration step is:

$$\begin{pmatrix} \alpha_0^{(k+1)} \\ \boldsymbol{\alpha}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}.$$

Here λ is the learning rate which can be set to 1 without loss of generality.

Minimization of Objective Function

Algorithm 1: Perceptron learning algorithm

Input : training data $S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$

Output: $\alpha_0^{(k)}$ and $\boldsymbol{\alpha}^{(k)}$

initialize $k = 0$, $\alpha_0^{(0)} = 0$ and $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$;

repeat

  select pair $(\boldsymbol{x}_i, y_i)$ from training set;

  if $y_i \cdot \left(\boldsymbol{x}_i^\mathsf{T} \boldsymbol{\alpha}^{(k)} + \alpha_0^{(k)} \right) \leq 0$ **then**

$$\begin{pmatrix} \alpha_0^{(k+1)} \\ \boldsymbol{\alpha}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix};$$

    $k \leftarrow k + 1$;

  end

until $y_i \cdot \left(\boldsymbol{x}_i^\mathsf{T} \boldsymbol{\alpha}^{(k)} + \alpha_0^{(k)} \right) > 0 \quad \forall\, i$;

Remarks on Perceptron Learning

- The update rule is extremely simple.
- Nothing happens if we classify all $\boldsymbol{x}_i$ correctly using the given linear decision boundary.
- The parameter $\boldsymbol{\alpha}$ of the decision boundary is a linear combination of feature vectors.
- Thus, the decision boundary is:

$$F(\boldsymbol{x}) = \left(\sum_{i \in \mathcal{E}} y_i \cdot \boldsymbol{x}_i \right)^{\mathsf{T}} \boldsymbol{x} + \sum_{i \in \mathcal{E}} y_i$$

$$= \sum_{i \in \mathcal{E}} y_i \cdot \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + \sum_{i \in \mathcal{E}} y_i,$$

where $\mathcal{E}$ is the set of indices that required an update.

Remarks on Perceptron Learning

- The final linear decision boundary depends on the initialization, i.e. $\alpha_0^{(0)}$ and $\boldsymbol{\alpha}^{(0)}$.
- The number of iterations can be rather large.
- If data are not linearly separable, the proposed learning algorithm will not converge. The algorithm will end up in hard to detect cycles.

Convergence of Learning Algorithm

Theorem (Convergence Theorem of Rosenblatt and Novikoff)

Assume that for all $i = 1, 2, \ldots, m$

$$y_i \left(\boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{\alpha}^* + \alpha_0^* \right) \geq \rho,$$

where $\rho > 0$ *and* $\| \boldsymbol{\alpha}^* \| = 1$.
Let $M = \max_i \| \boldsymbol{x}_i \|_2$. *Then it can be shown that the perceptron learning algorithm converges to a linear decision boundary after* k *iterations, where* k *is bounded by*

$$k \leq \frac{(\alpha_0^{*2} + 1)(1 + M^2)}{\rho^2}.$$

Topics

Take Home Messages

- The perceptron learning algorithm is based on a discrete optimization problem whose objective function changes in each iteration step.
- It is a very simple learning rule.
- The number of iterations does **not** depend on the dimension of feature vectors!

Further Readings

Rosenblatt's original work on the perceptron:

Frank Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6 (Nov. 1958), pp. 386–408. DOI: 10.1037/h0042519

These books provide a good start into machine learning and neural networks:

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. Springer, Feb. 2009
- Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. DOI: 10.1017/CBO9780511812651

Medical Image Processing for Interventional Applications

Neural Networks

Online Course – Unit 26

Andreas Maier, Vincent Christlein, Dieter Hahn, Frank Schebesch
Pattern Recognition Lab (CS 5)

virtuelle hochschule bayern

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Topics

Physiological Motivation

Synaptic transmission from a presynaptic neuron to a postsynaptic cell:

1. **Action potential** traveling along the membrane of the presynaptic cell, until it reaches the synapse

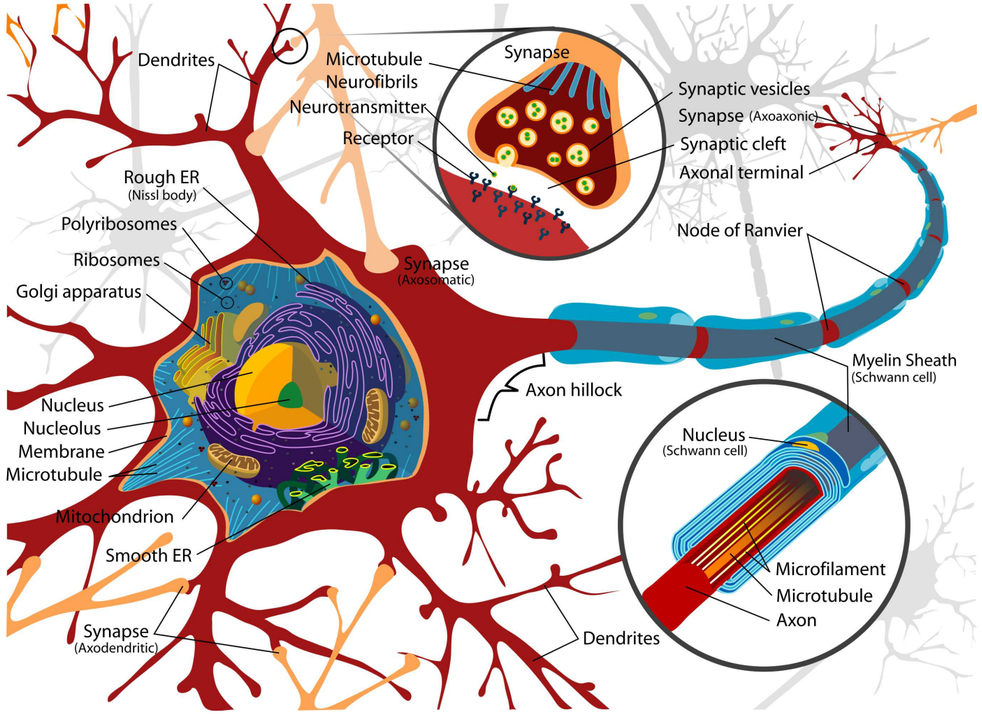


Figure 1: Illustration of a neuron cell (Wiki Commons)

Physiological Motivation

Synaptic transmission from a presynaptic neuron to a postsynaptic cell:

1. **Action potential** traveling along the membrane of the presynaptic cell, until it reaches the synapse
2. **Electrical depolarization** of the membrane at the synapse

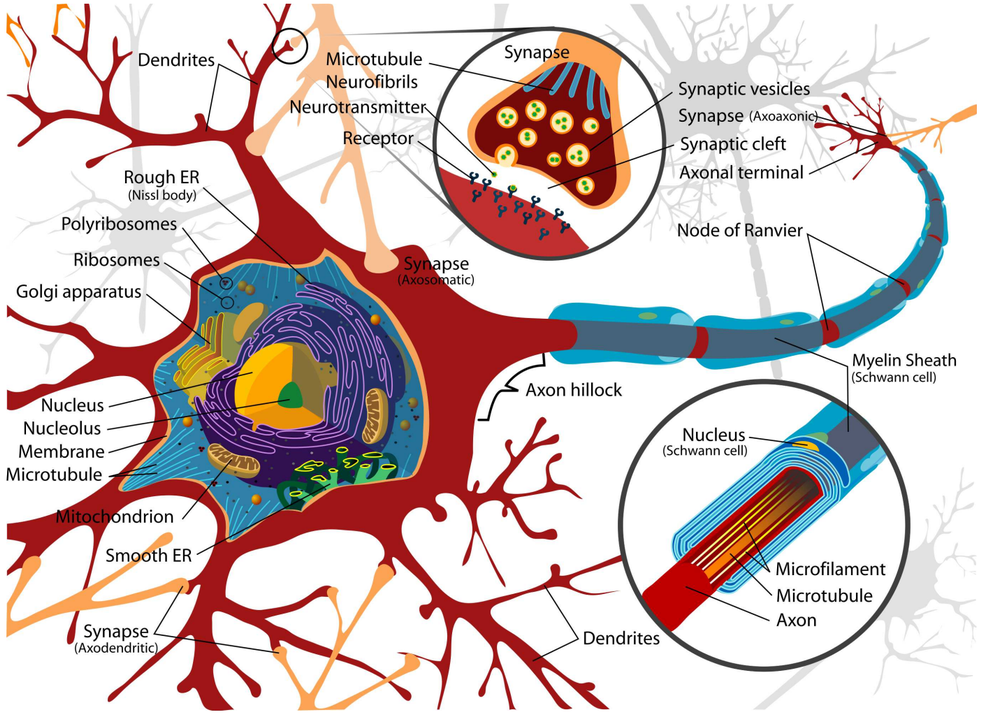


Figure 1: Illustration of a neuron cell (Wiki Commons)

Physiological Motivation

Synaptic transmission from a presynaptic neuron to a postsynaptic cell:

1. **Action potential** traveling along the membrane of the presynaptic cell, until it reaches the synapse
2. **Electrical depolarization** of the membrane at the synapse
3. **Neurotransmitter** binding to chemical receptor molecules located on the membrane of the postsynaptic cell.

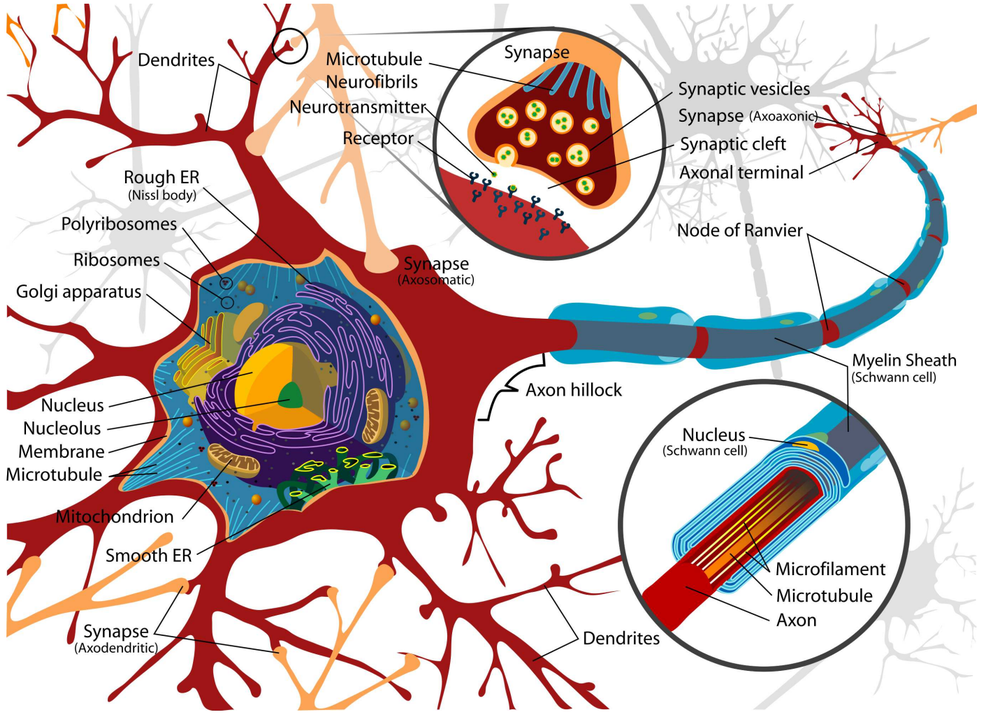


Figure 1: Illustration of a neuron cell (Wiki Commons)

Physiological Motivation

Synaptic transmission from a presynaptic neuron to a postsynaptic cell:

1. **Action potential** traveling along the membrane of the presynaptic cell, until it reaches the synapse
2. **Electrical depolarization** of the membrane at the synapse
3. **Neurotransmitter** binding to chemical receptor molecules located on the membrane of the postsynaptic cell.
4. Opening of ion channels in the postsynaptic cell membrane, causing ions to enter or exit the cell and change the local transmembrane potential: **excitatory** or **inhibitory postsynaptic potential (EPSP/IPSP)**.

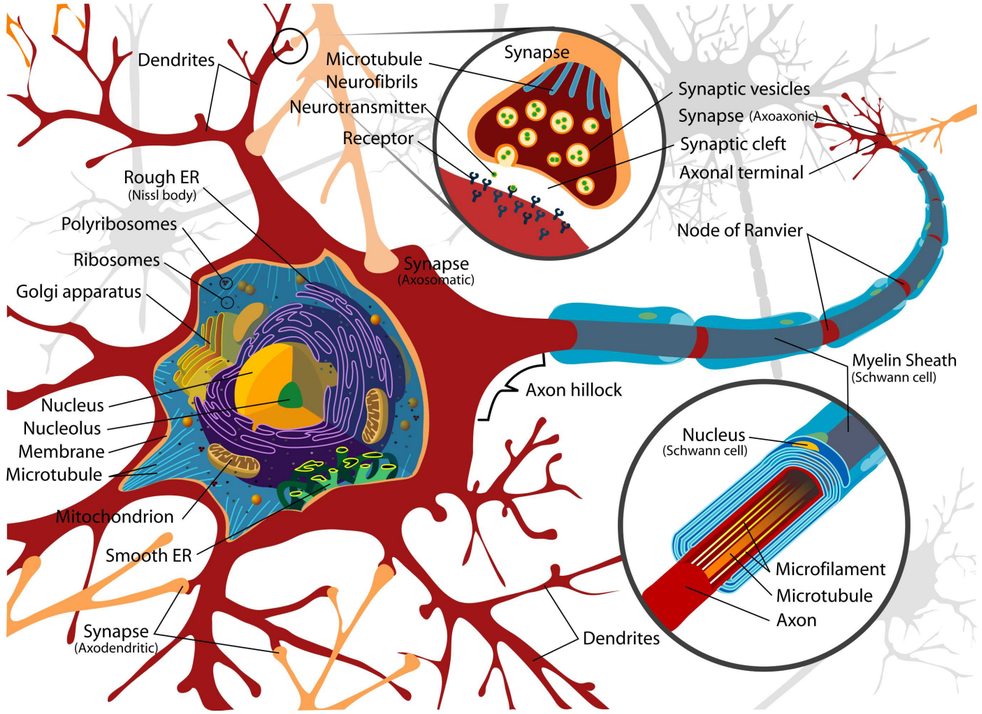


Figure 1: Illustration of a neuron cell (Wiki Commons)

Physiological Motivation

Synaptic transmission from a presynaptic neuron to a postsynaptic cell:

1. **Action potential** traveling along the membrane of the presynaptic cell, until it reaches the synapse
2. **Electrical depolarization** of the membrane at the synapse
3. **Neurotransmitter** binding to chemical receptor molecules located on the membrane of the postsynaptic cell.
4. Opening of ion channels in the postsynaptic cell membrane, causing ions to enter or exit the cell and change the local transmembrane potential: **excitatory** or **inhibitory postsynaptic potential (EPSP/IPSP)**.
5. Due to thermal shaking, neurotransmitter molecules eventually break loose from the receptors and drift away.

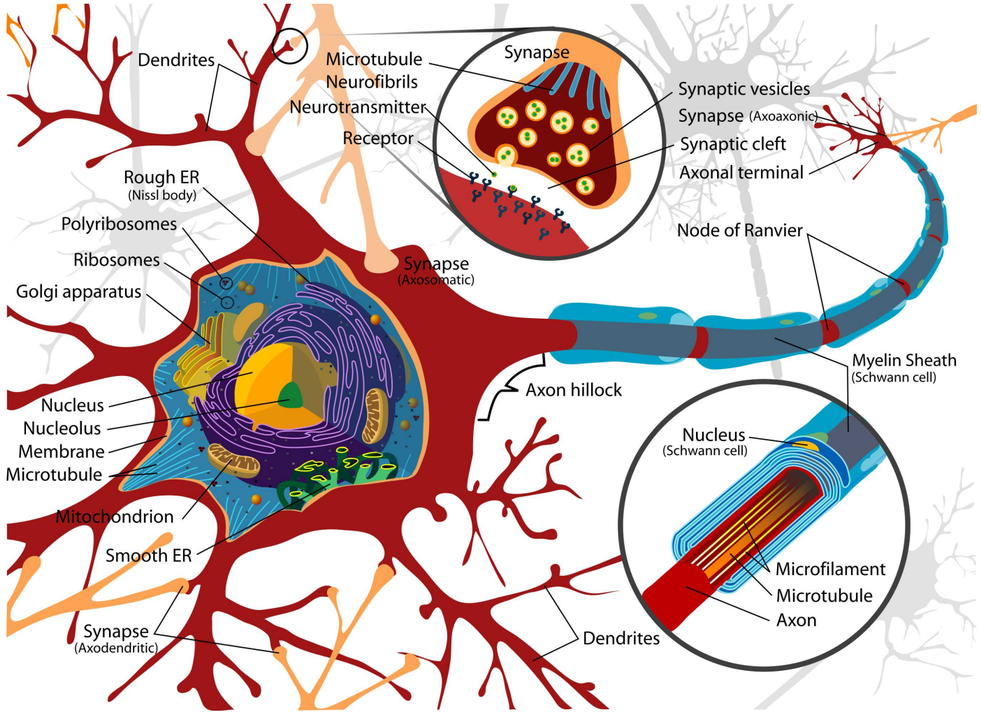


Figure 1: Illustration of a neuron cell (Wiki Commons)

Physiological Motivation

Synaptic transmission from a presynaptic neuron to a postsynaptic cell:

1. **Action potential** traveling along the membrane of the presynaptic cell, until it reaches the synapse
2. **Electrical depolarization** of the membrane at the synapse
3. **Neurotransmitter** binding to chemical receptor molecules located on the membrane of the postsynaptic cell.
4. Opening of ion channels in the postsynaptic cell membrane, causing ions to enter or exit the cell and change the local transmembrane potential: **excitatory** or **inhibitory postsynaptic potential (EPSP/IPSP)**.
5. Due to thermal shaking, neurotransmitter molecules eventually break loose from the receptors and drift away.
6. The neurotransmitter is either reabsorbed by the presynaptic cell, and then repackaged for future release, or else it is broken down metabolically.

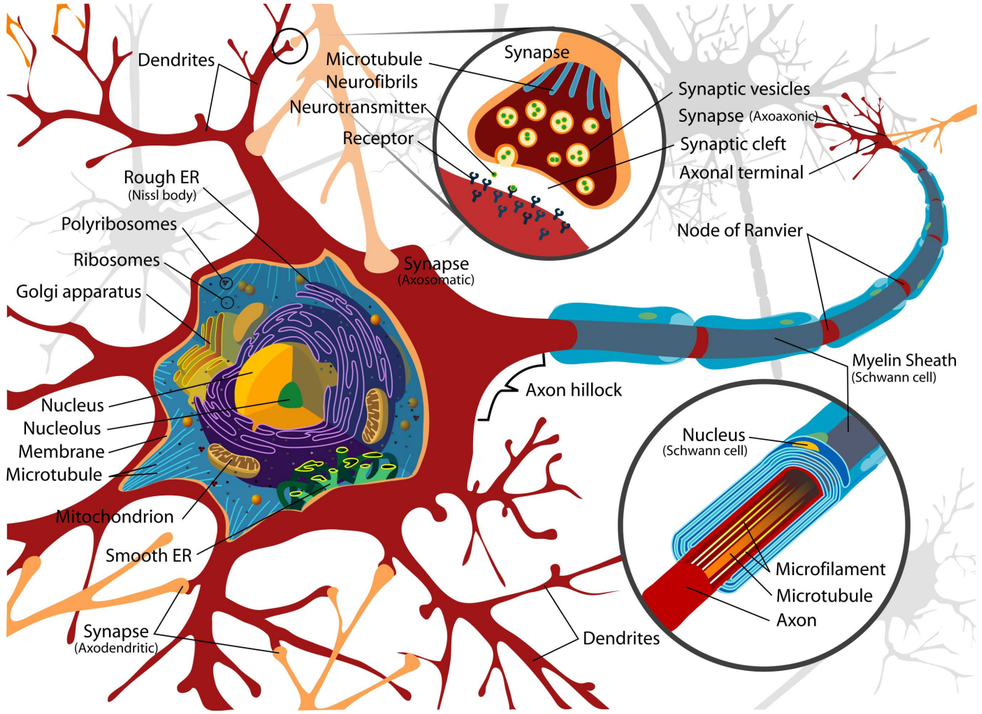


Figure 1: Illustration of a neuron cell (Wiki Commons)

New Action Potentials

- EPSPs resulting from transmitter release at a single synapse are generally far too small to trigger a spike in the postsynaptic neuron.

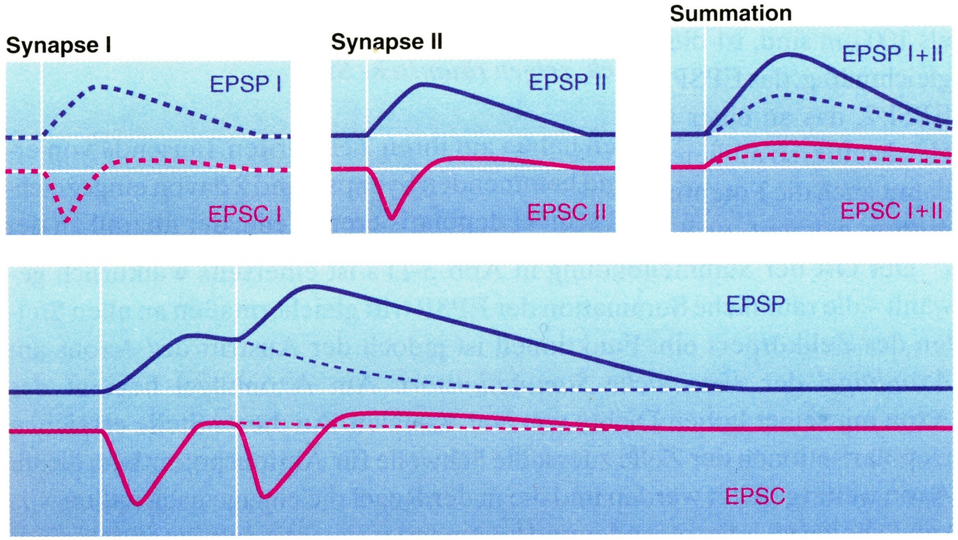


Figure 2: Summation process

New Action Potentials

- EPSPs resulting from transmitter release at a single synapse are generally far too small to trigger a spike in the postsynaptic neuron.
- A neuron may receive inputs from thousands of other neurons.

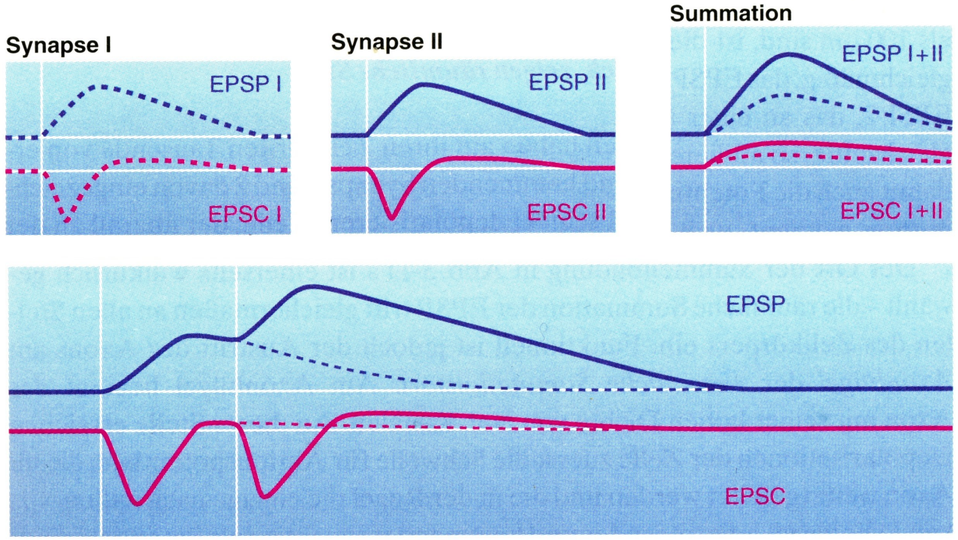


Figure 2: Summation process

New Action Potentials

- EPSPs resulting from transmitter release at a single synapse are generally far too small to trigger a spike in the postsynaptic neuron.
- A neuron may receive inputs from thousands of other neurons.
- If the postsynaptic cell is sufficiently depolarized, an **action potential** will occur.

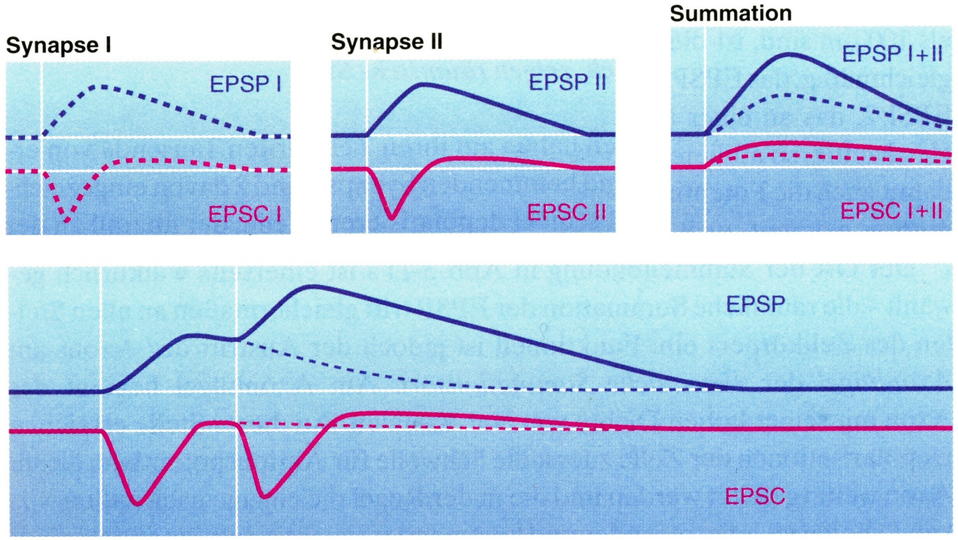


Figure 2: Summation process

New Action Potentials

- EPSPs resulting from transmitter release at a single synapse are generally far too small to trigger a spike in the postsynaptic neuron.
- A neuron may receive inputs from thousands of other neurons.
- If the postsynaptic cell is sufficiently depolarized, an **action potential** will occur.
- Action potentials are not graded; they are all-or-none responses.

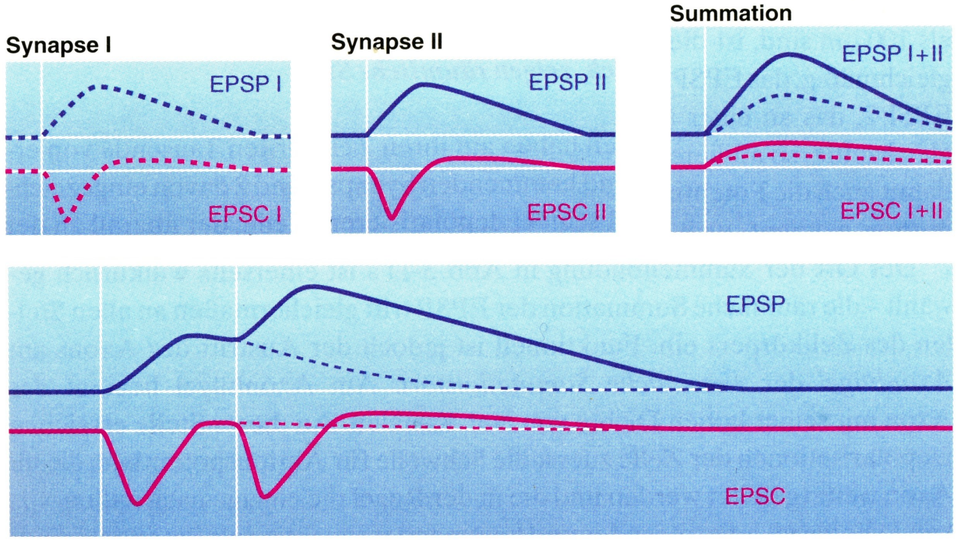


Figure 2: Summation process

New Action Potentials

- EPSPs resulting from transmitter release at a single synapse are generally far too small to trigger a spike in the postsynaptic neuron.
- A neuron may receive inputs from thousands of other neurons.
- If the postsynaptic cell is sufficiently depolarized, an **action potential** will occur.
- Action potentials are not graded; they are all-or-none responses.
- Postsynaptic potentials are subject to **spatial** and **temporal** summation.

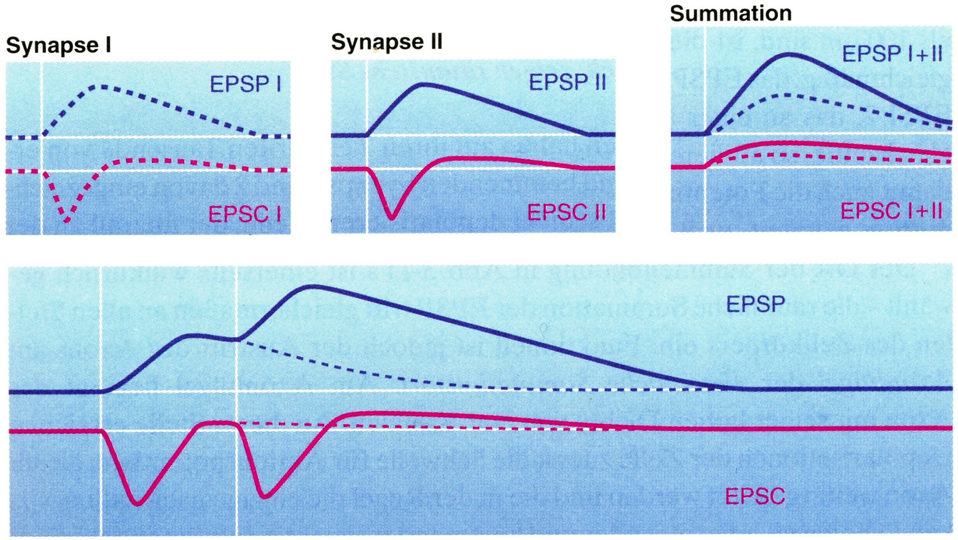


Figure 2: Summation process

Multi-Layer Perceptrons

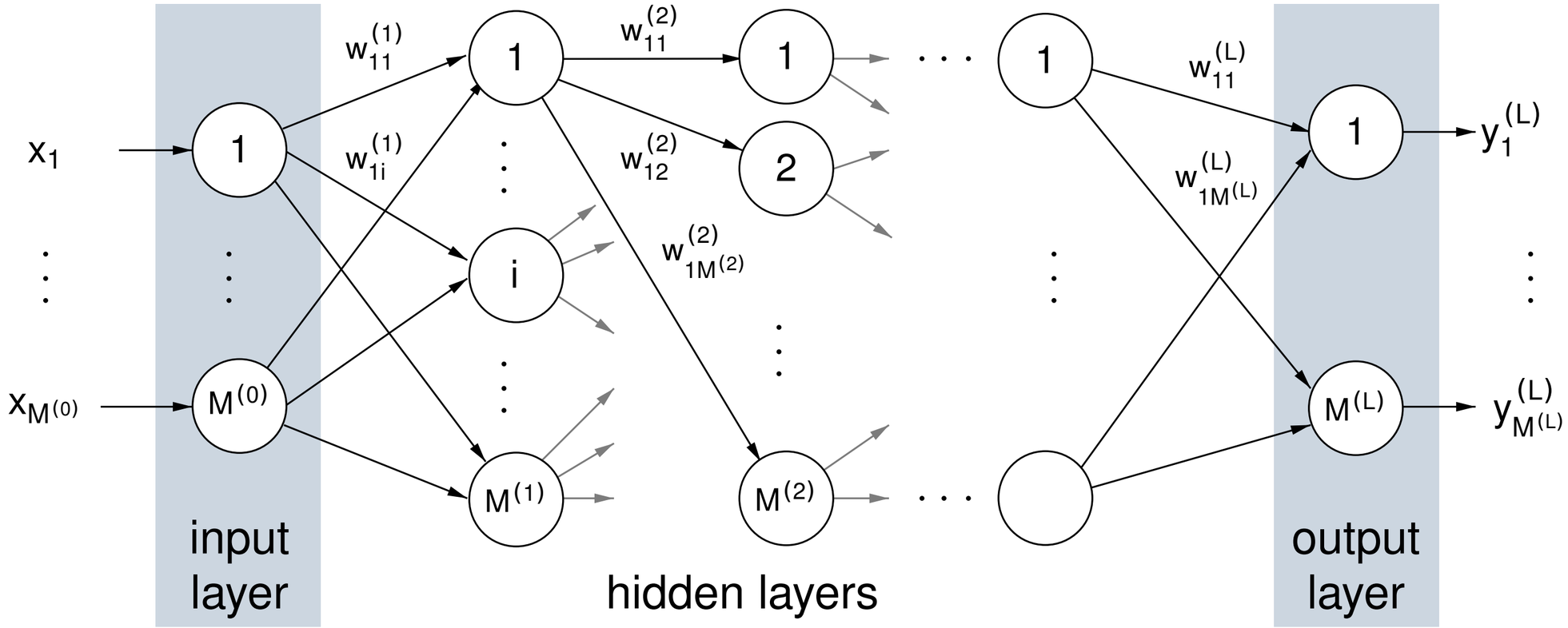
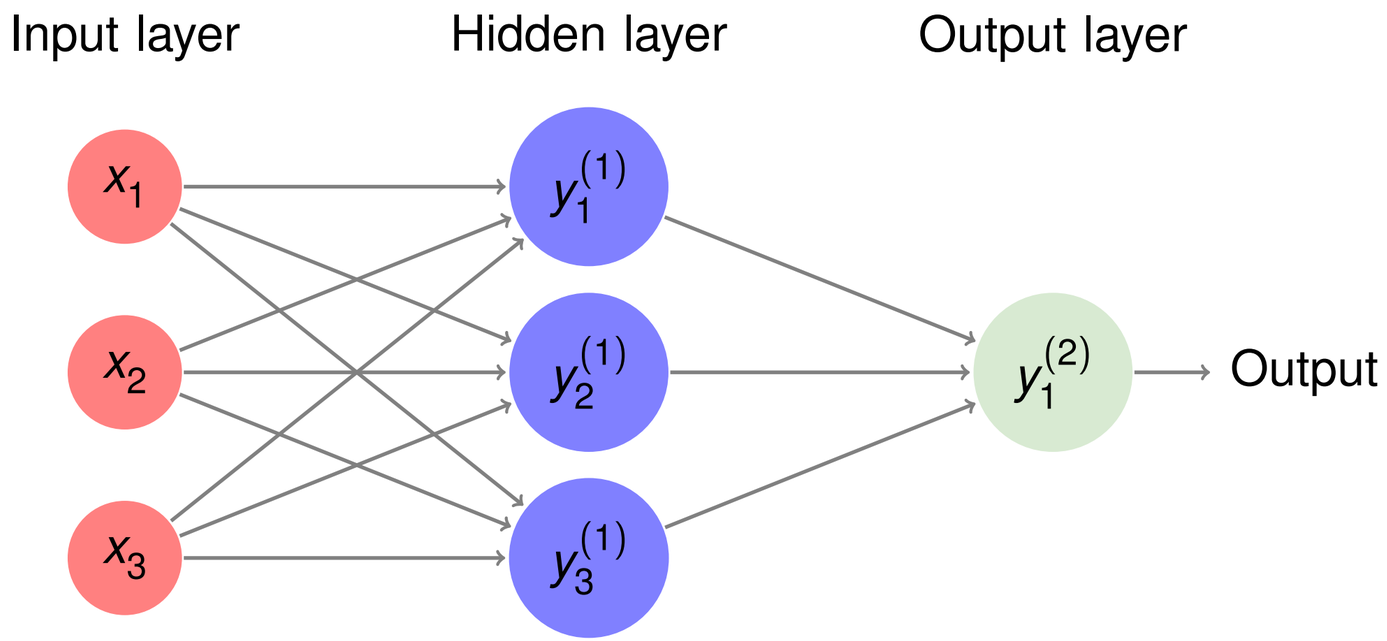


Figure 3: Topology

Multi-Layer Perceptrons



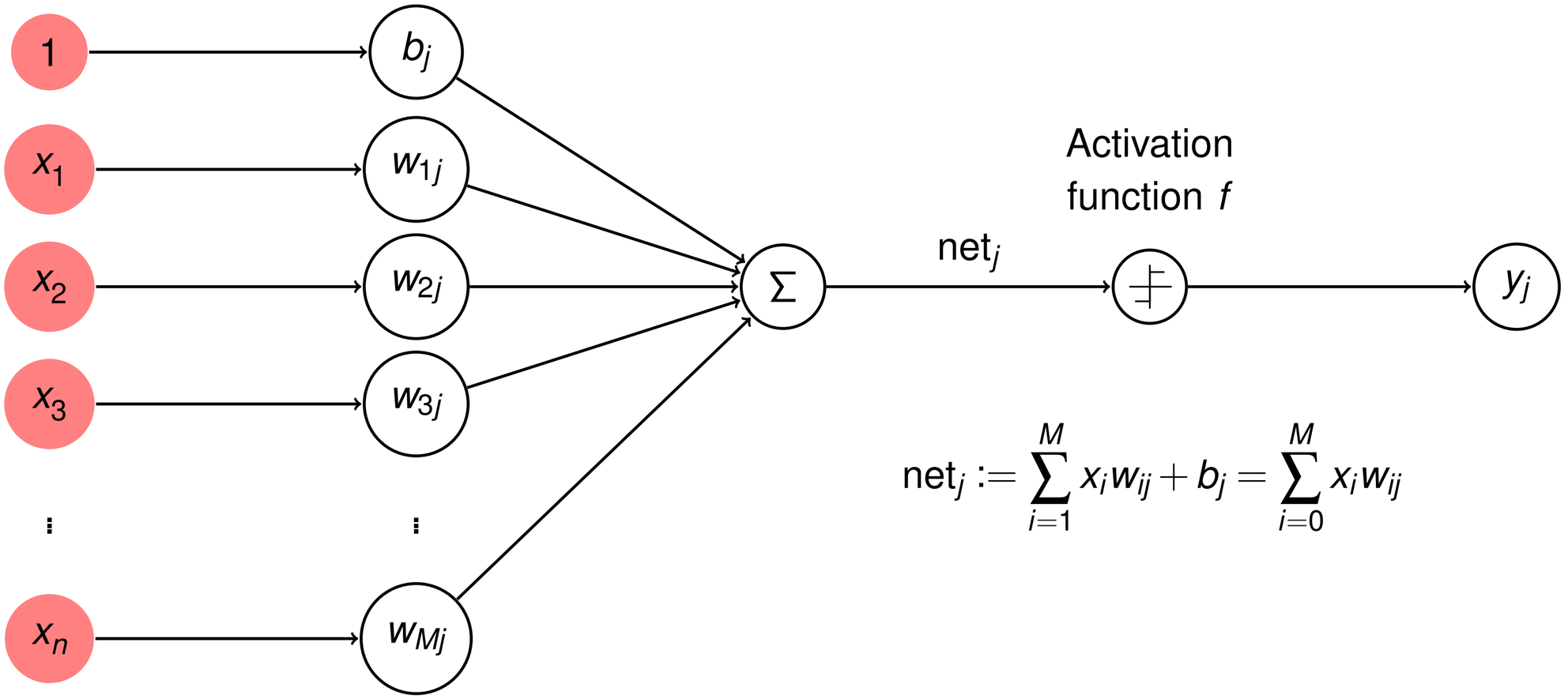
Network architecture

input units $\rightarrow$ dimensions of features

output units $\rightarrow$ typically number of classes

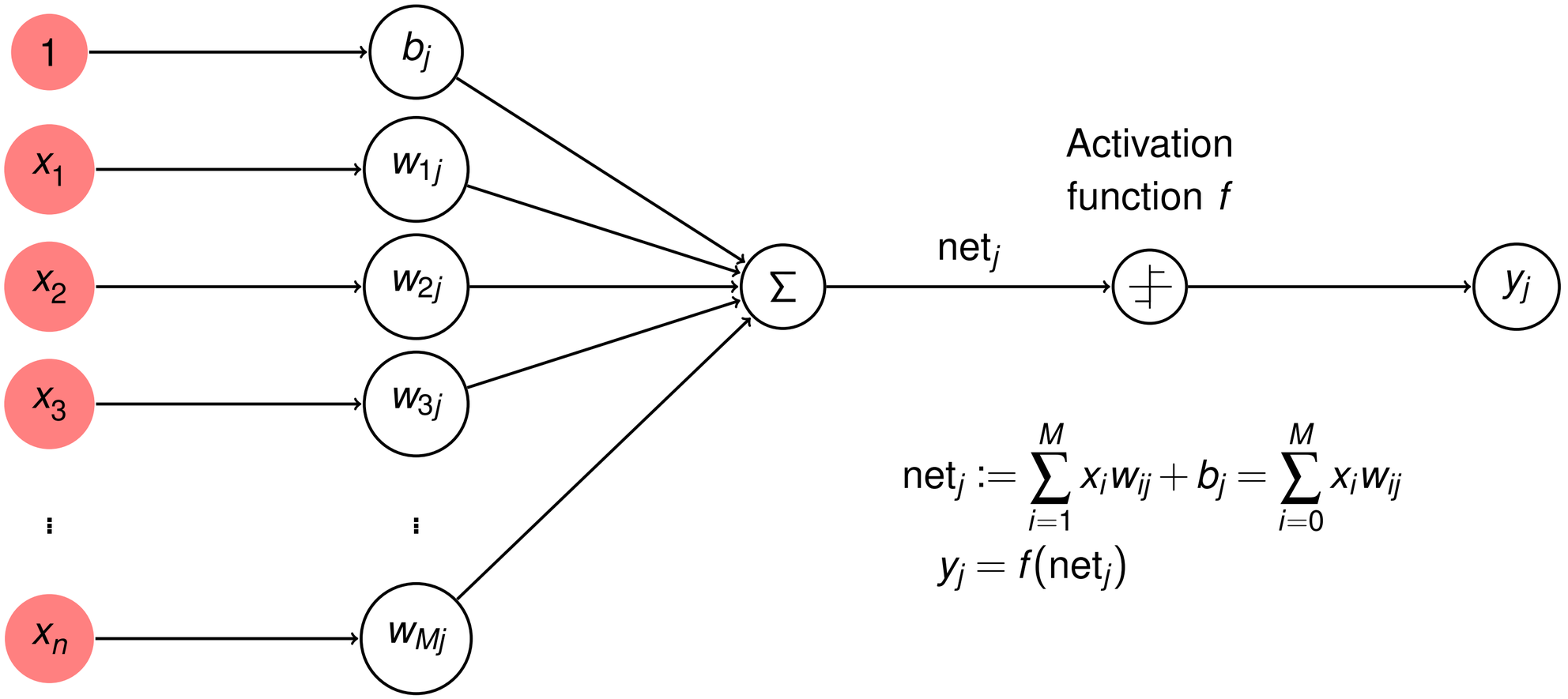
hidden units $\rightarrow$ design issue (underfitting, overfitting)

Single Layer



$$\text{net}_j := \sum_{i=1}^{M} x_i w_{ij} + b_j = \sum_{i=0}^{M} x_i w_{ij}$$

Single Layer



$$\text{net}_j := \sum_{i=1}^{M} x_i w_{ij} + b_j = \sum_{i=0}^{M} x_i w_{ij}$$

$$y_j = f(\text{net}_j)$$

Examples for Activation Functions

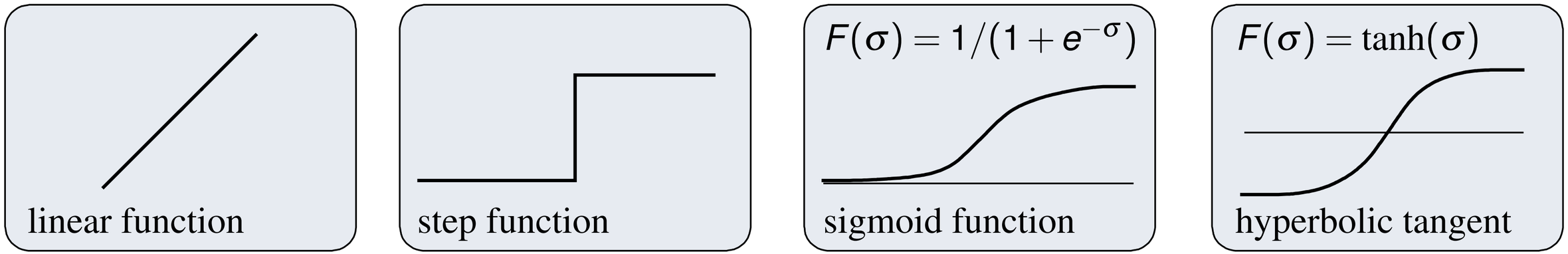
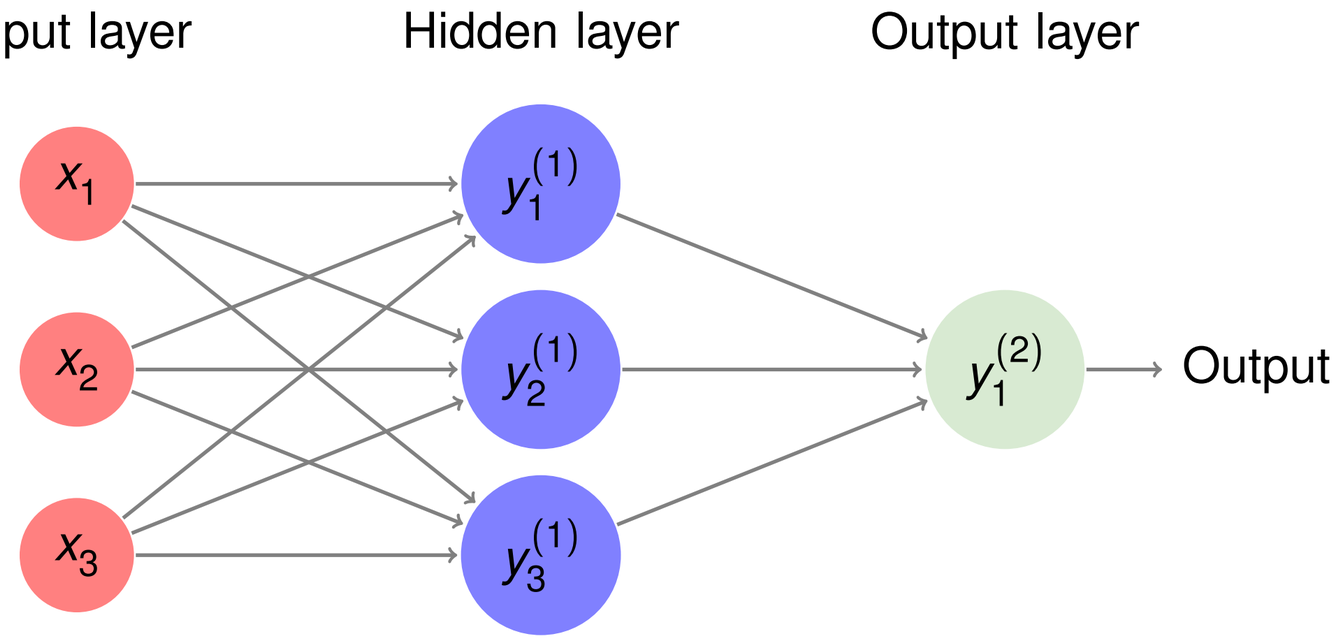


Figure 6: Traditional activation functions

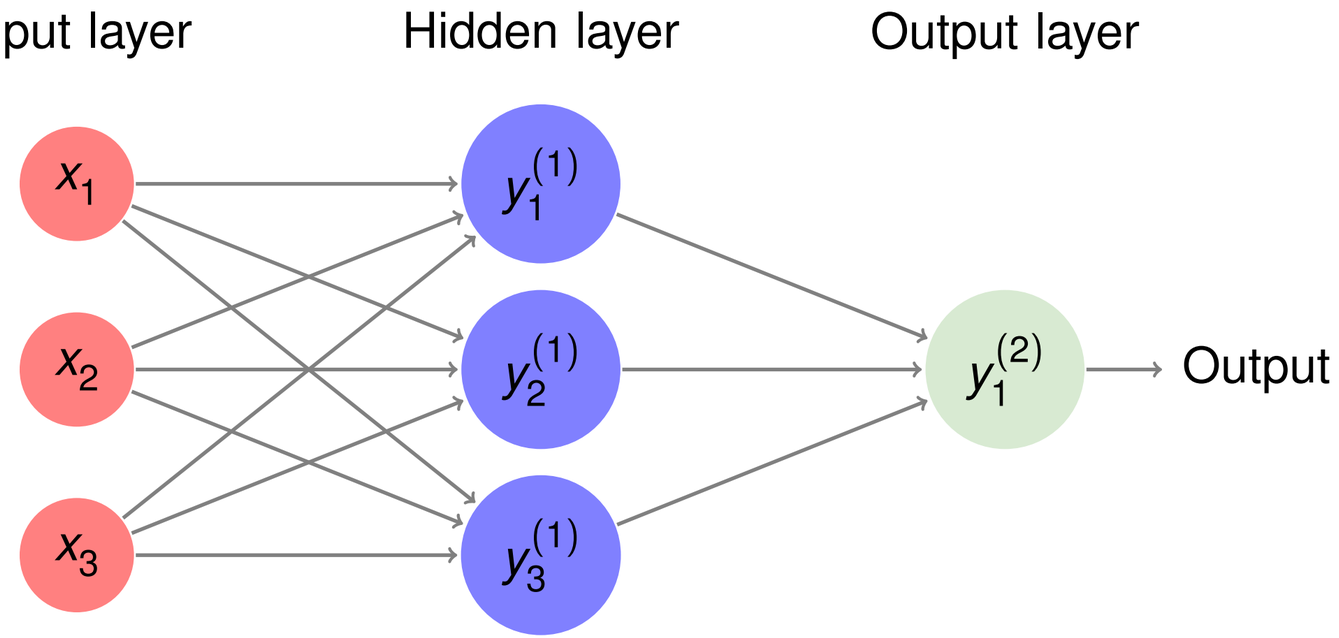
Multi-Layer Perceptrons



Activation of Layer 1:

$$y_1^{(1)} = f\left(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3\right)$$
$$y_2^{(1)} = f\left(w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3\right)$$
$$y_3^{(1)} = f\left(w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3\right)$$

Multi-Layer Perceptrons



Layer 2:

$$y_1^{(2)} = f\left(w_{11}^{(2)} y_1^{(1)} + w_{12}^{(2)} y_2^{(1)} + w_{13}^{(2)} y_3^{(1)}\right)$$

$\rightarrow$ Inputs are propagated through the network to the output.

$\rightarrow$ Feed forward neural network

Topics

Multi-Layer Perceptrons
- Physiological Motivation
- Topology and Activation Functions

Backpropagation

Summary
- Take Home Messages
- Further Readings

Backpropagation Algorithm

Supervised Learning Algorithm

- Gradient descent to adjust the weights reducing the training error ε:

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial \varepsilon}{\partial w_{ij}^{(l)}}$$

- Typical error function $\rightarrow$ **mean squared error**:

$$\varepsilon_{\text{MSE}}(\boldsymbol{w}) = \frac{1}{2} \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)})^2$$

Backpropagation Algorithm

Adjusting the weights $w_{jk}^{(L)}$ of the output layer:

$$\frac{\partial \varepsilon_{\text{MSE}}}{\partial w_{jk}^{(L)}} = \frac{\partial \varepsilon_{\text{MSE}}}{\partial \text{net}_k^{(L)}} \cdot \frac{\partial \text{net}_k^{(L)}}{\partial w_{jk}^{(L)}} = -\delta_k^{(L)} \cdot y_j^{(L-1)}$$

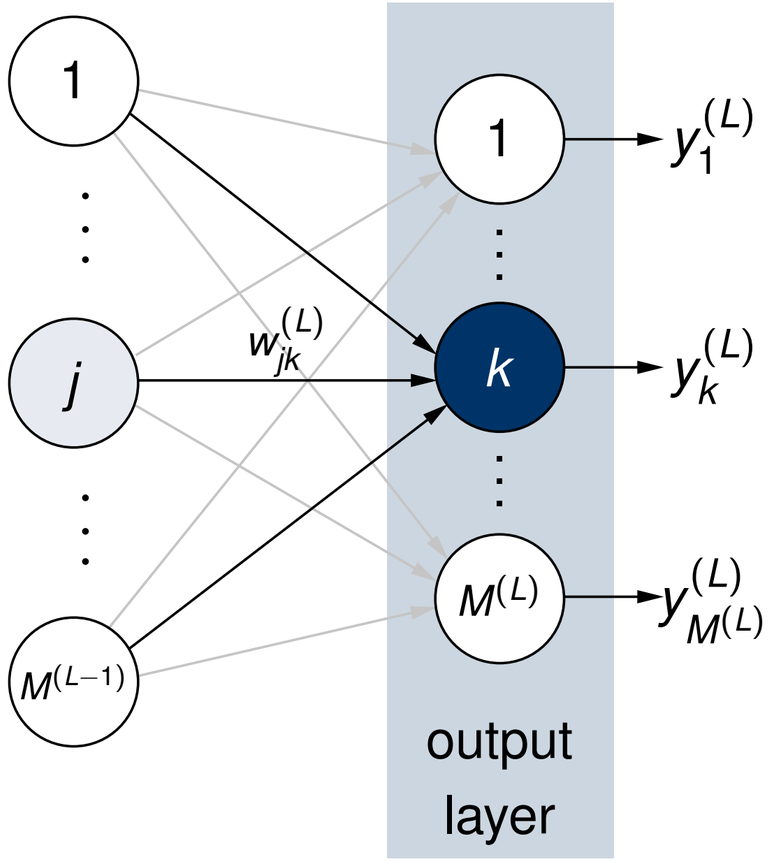


Figure 8: Error propagation at the output layer

Backpropagation Algorithm

Adjusting the weights $w_{jk}^{(L)}$ of the output layer:

$$\frac{\partial \varepsilon_{\text{MSE}}}{\partial w_{jk}^{(L)}} = \frac{\partial \varepsilon_{\text{MSE}}}{\partial \text{net}_k^{(L)}} \cdot \frac{\partial \text{net}_k^{(L)}}{\partial w_{jk}^{(L)}} = -\delta_k^{(L)} \cdot y_j^{(L-1)}$$

The **sensitivity** $\delta_k^{(L)}$:

$$\begin{aligned}\delta_k^{(L)} &= -\frac{\partial \varepsilon_{\text{MSE}}}{\partial \text{net}_k^{(L)}} = -\frac{\partial \varepsilon_{\text{MSE}}}{\partial y_k^{(L)}} \cdot \frac{\partial y_k^{(L)}}{\partial \text{net}_k^{(L)}} \\ &= \left(t_k - y_k^{(L)}\right) f'\left(\text{net}_k^{(L)}\right)\end{aligned}$$

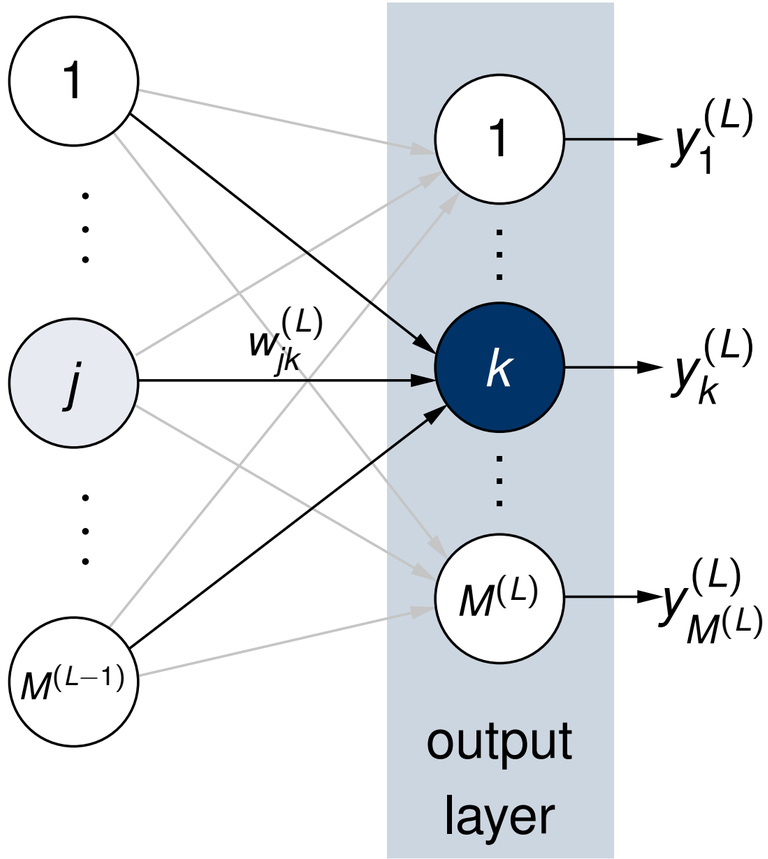


Figure 8: Error propagation at the output layer

Backpropagation Algorithm

Adjusting the weights $w_{jk}^{(l)}$ of the hidden layers:

- Desired output values for the hidden layers are not known.

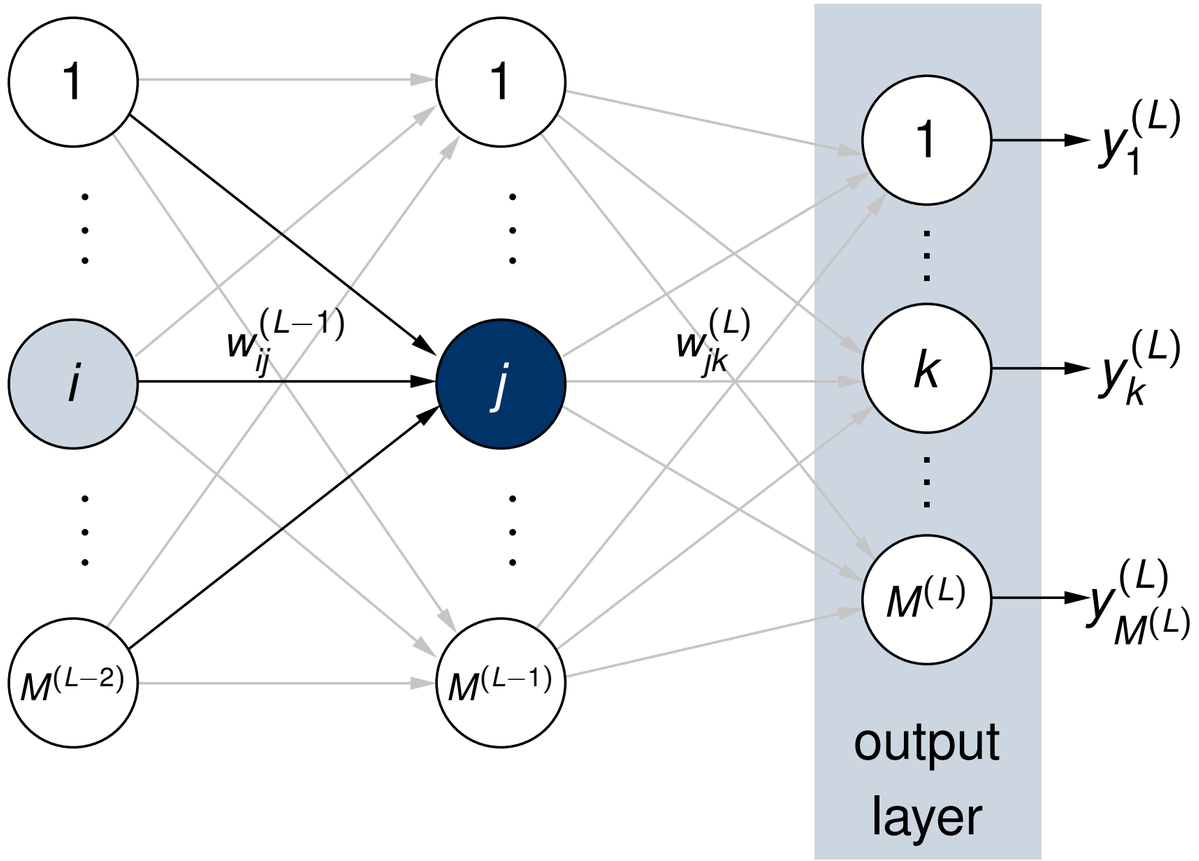


Figure 9: Error propagation at a hidden layer

Backpropagation Algorithm

Adjusting the weights $w_{jk}^{(l)}$ of the hidden layers:

- Desired output values for the hidden layers are not known.
- For the weights $w_{ij}^{(L-1)}$ of the last hidden layer:

$$\frac{\partial \varepsilon_{\text{MSE}}}{\partial w_{ij}^{(L-1)}} = \frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} \cdot \frac{\partial y_j^{(L-1)}}{\partial \text{net}_j^{(L-1)}} \cdot \frac{\partial \text{net}_j^{(L-1)}}{\partial w_{ij}^{(L-1)}}$$
$$= \frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} \cdot f'(\text{net}_j^{(L-1)}) \cdot y_i^{(L-2)}$$

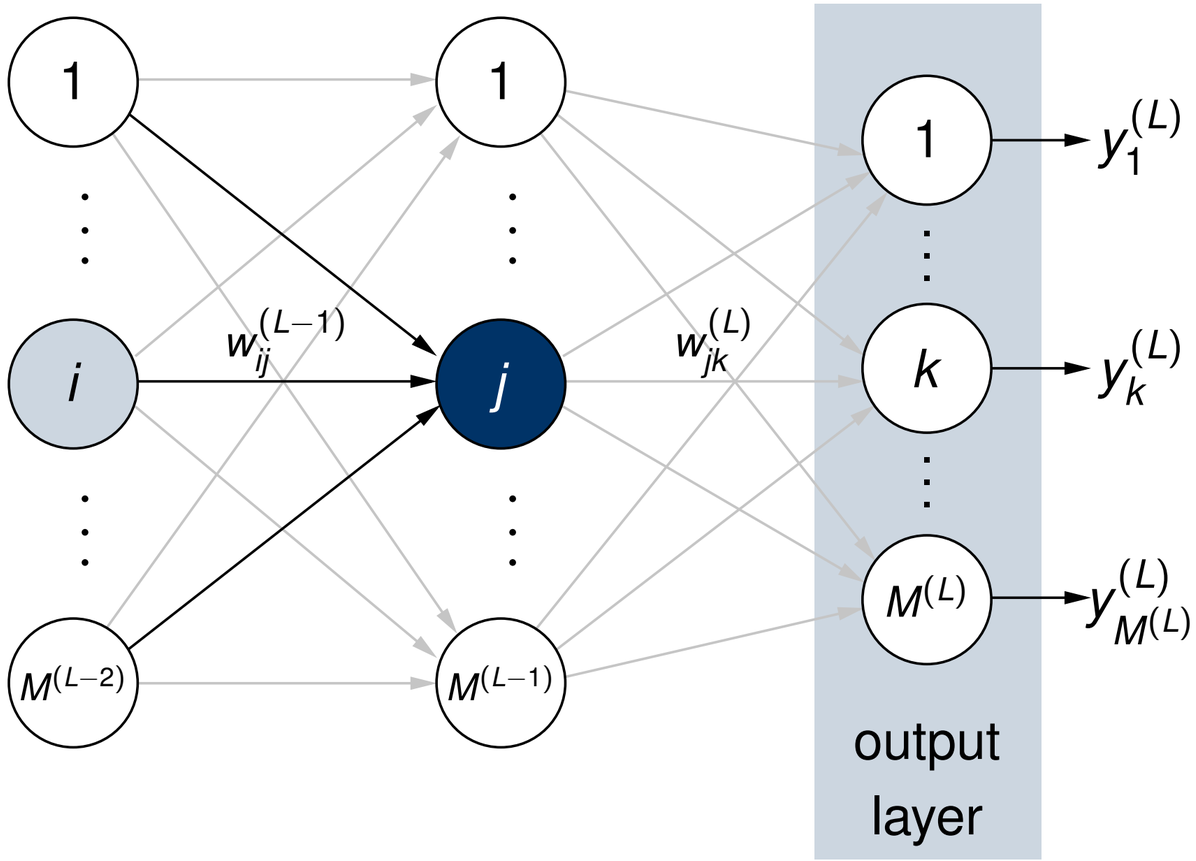


Figure 9: Error propagation at a hidden layer

Backpropagation Algorithm

$$\frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} = \frac{\partial}{\partial y_j^{(L-1)}} \left[\frac{1}{2} \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)})^2 \right]$$

$$= - \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)}) \frac{\partial y_k^{(L)}}{\partial y_j^{(L-1)}}$$

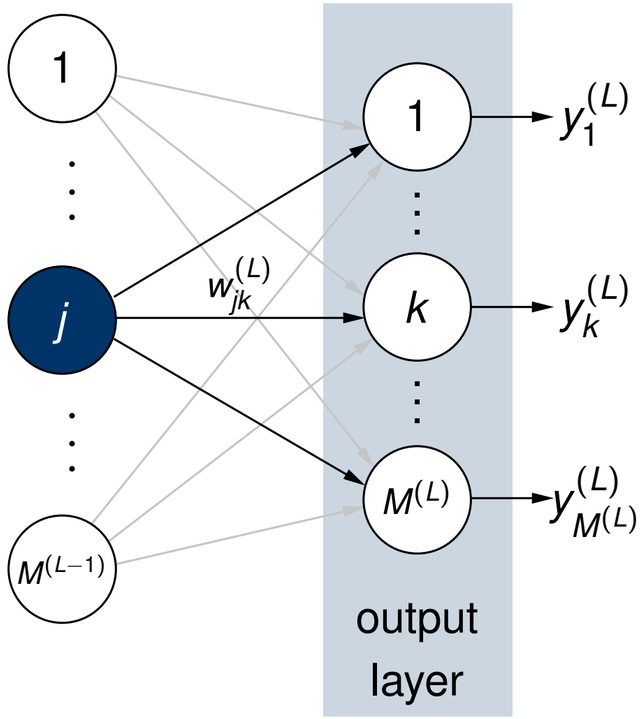


Figure 10: Error propagation at the output layer

Backpropagation Algorithm

$$\frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} = \frac{\partial}{\partial y_j^{(L-1)}} \left[\frac{1}{2} \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)})^2 \right]$$

$$= - \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)}) \frac{\partial y_k^{(L)}}{\partial y_j^{(L-1)}}$$

$$= - \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)}) \frac{\partial y_k^{(L)}}{\partial \text{net}_k^{(L)}} \cdot \frac{\partial \text{net}_k^{(L)}}{\partial y_j^{(L-1)}}$$

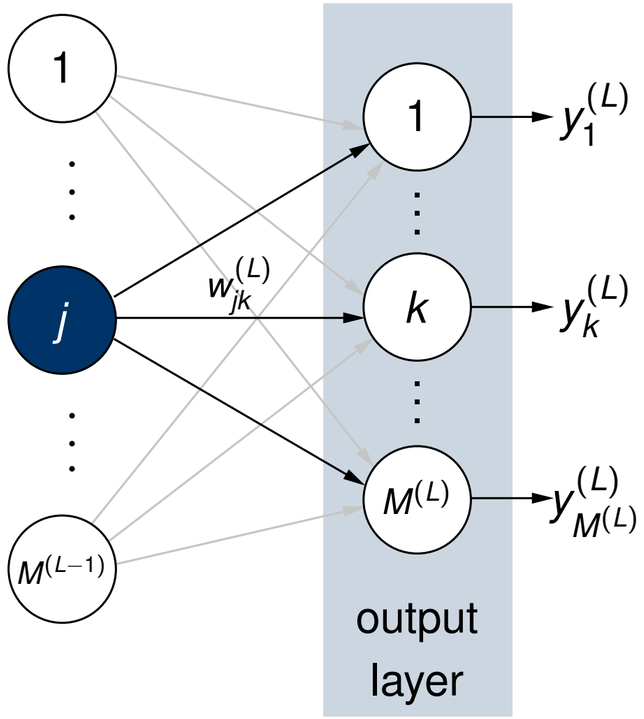


Figure 10: Error propagation at the output layer

Backpropagation Algorithm

$$
\begin{aligned}
\frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} &= \frac{\partial}{\partial y_j^{(L-1)}} \left[\frac{1}{2} \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right)^2 \right] \\
&= - \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right) \frac{\partial y_k^{(L)}}{\partial y_j^{(L-1)}} \\
&= - \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right) \frac{\partial y_k^{(L)}}{\partial \text{net}_k^{(L)}} \cdot \frac{\partial \text{net}_k^{(L)}}{\partial y_j^{(L-1)}} \\
&= - \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right) f'\left(\text{net}_k^{(L)} \right) w_{jk}^{(L)}
\end{aligned}
$$

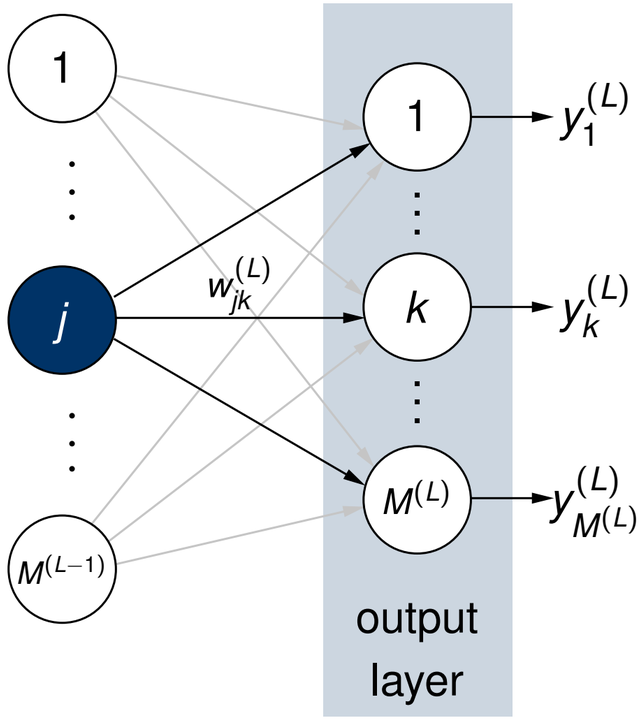


Figure 10: Error propagation at the output layer

Backpropagation Algorithm

$$
\begin{aligned}
\frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} &= \frac{\partial}{\partial y_j^{(L-1)}} \left[\frac{1}{2} \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right)^2 \right] \\
&= - \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right) \frac{\partial y_k^{(L)}}{\partial y_j^{(L-1)}} \\
&= - \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right) \frac{\partial y_k^{(L)}}{\partial \text{net}_k^{(L)}} \cdot \frac{\partial \text{net}_k^{(L)}}{\partial y_j^{(L-1)}} \\
&= - \sum_{k=1}^{M^{(L)}} \left(t_k - y_k^{(L)} \right) f'(\text{net}_k^{(L)}) \, w_{jk}^{(L)} \\
&= - \sum_{k=1}^{M^{(L)}} \delta_k^{(L)} w_{jk}^{(L)}
\end{aligned}
$$

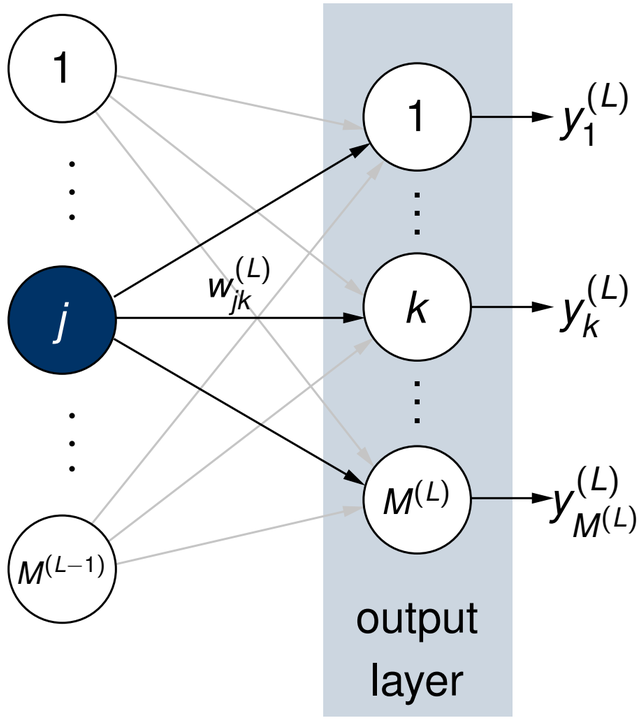


Figure 10: Error propagation at the output layer

Backpropagation Algorithm

Sensitivity $\delta_j^{(l)}$ for any hidden layer l, $0 < l < L$:

$$\delta_j^{(l)} = f'(\text{net}_j^{(l)}) \sum_{k=1}^{M^{(l+1)}} w_{jk}^{(l+1)} \, \delta_k^{(l+1)}$$

Backpropagation Algorithm

Sensitivity $\delta_j^{(l)}$ for any hidden layer l, $0 < l < L$:

$$\delta_j^{(l)} = f'(\text{net}_j^{(l)}) \sum_{k=1}^{M^{(l+1)}} w_{jk}^{(l+1)} \, \delta_k^{(l+1)}$$

Update rule:

$$\Delta w_{ij}^{(l)} = \eta \, \delta_j^{(l)} \, y_i^{(l-1)}$$

Topics

Take Home Messages

- We talked about the physiological background (neurons, synapses, action potentials, EPSP/IPSP, spatial and temporal summation) which motivates neural networks in general.
- The topology of multi-layer perceptrons consists of an input, some hidden, and an output layer.
- Each neuron is connected with an activation function.
- The backpropagation algorithm is a special case of a gradient descent method.

Further Readings

For those interested in more physiological details, this (German) book might be helpful:

Robert F. Schmidt, Florian Lang, and Manfred Heckmann, eds. *Physiologie des Menschen: Mit Pathophysiologie*. 31st ed. Springer Medizin Verlag Heidelberg, 2010. DOI: 10.1007/978-3-642-01651-6

Use these references for further information on machine learning, neural networks, and backpropagation:

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. Springer, Feb. 2009
- Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. DOI: 10.1017/CBO9780511812651
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-propagating Errors”. In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0

Medical Image Processing for Interventional Applications

Deep Learning

Online Course – Unit 27
Andreas Maier, Vincent Christlein, Tobias Würfl, Frank Schebesch
Pattern Recognition Lab (CS 5)

virtuelle hochschule bayern

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Topics

Upswing of Neural Networks

Convolutional Neural Networks
- Concept
- Optimization

Examples

Summary
- Take Home Messages
- Further Readings

Progress in Research

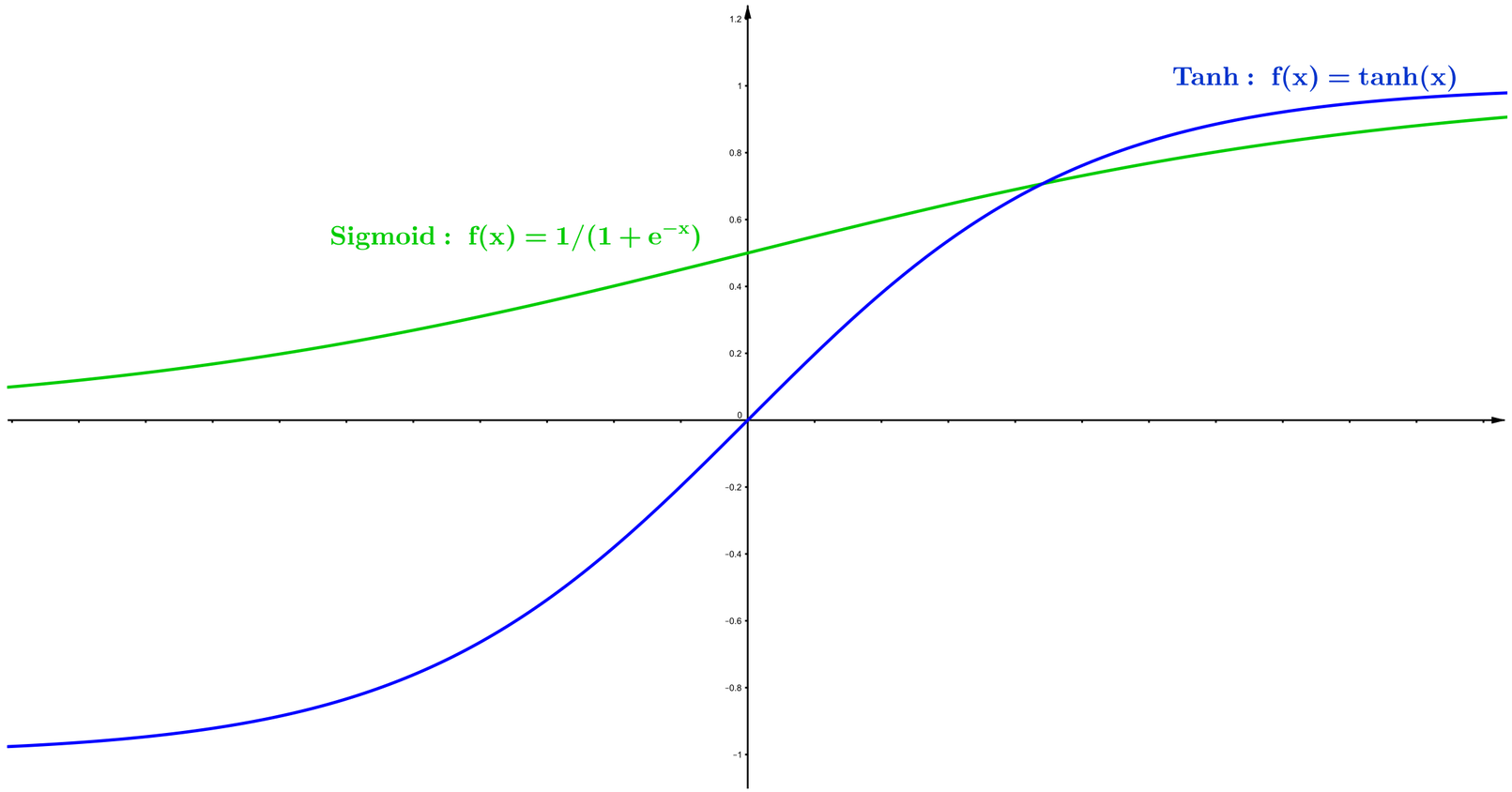


Figure 1: The use of rectified linear units (ReLU) lead to improved results in neural network learning.

Progress in Research

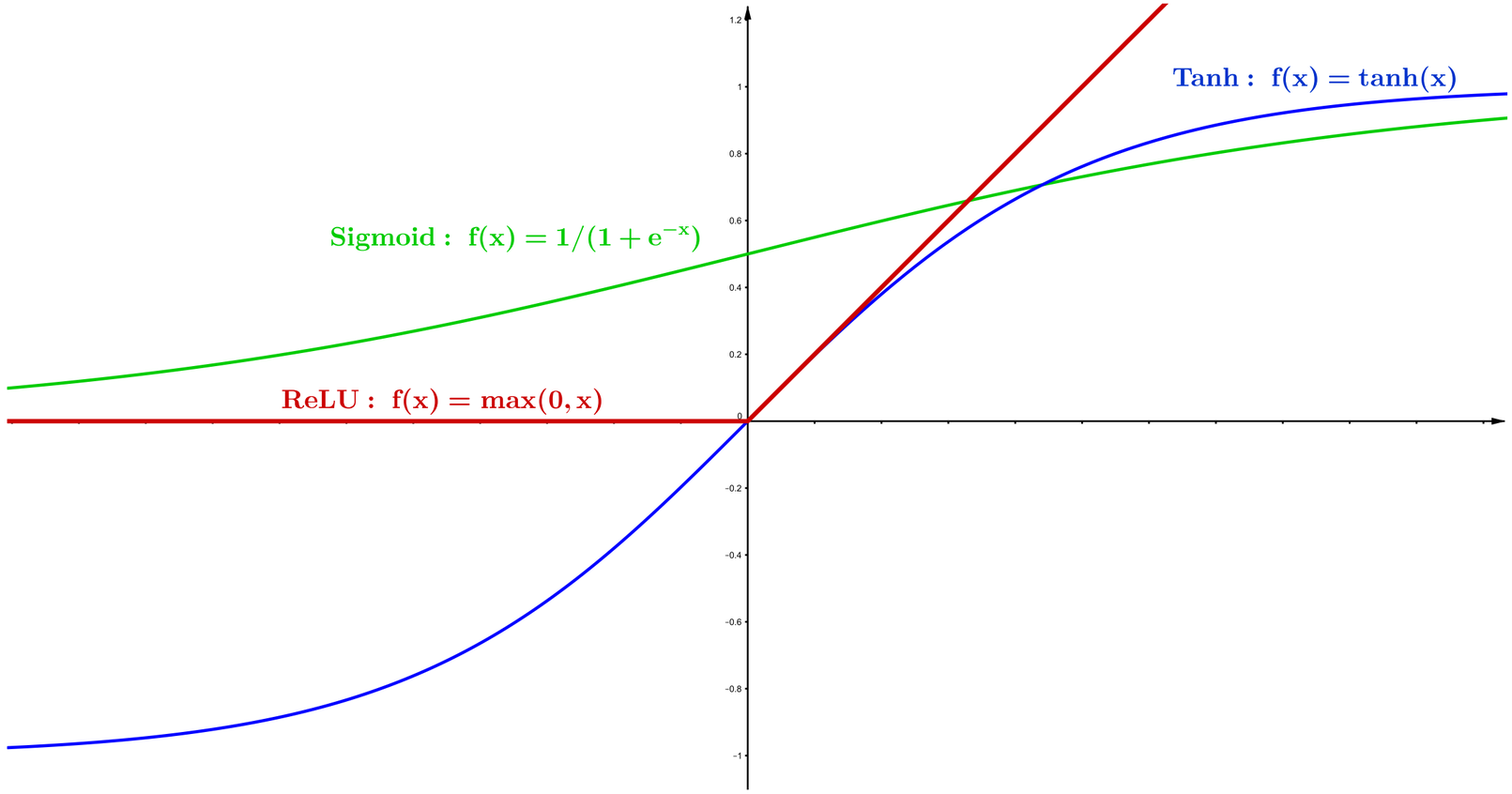


Figure 1: The use of rectified linear units (ReLU) lead to improved results in neural network learning.

Progress in Research

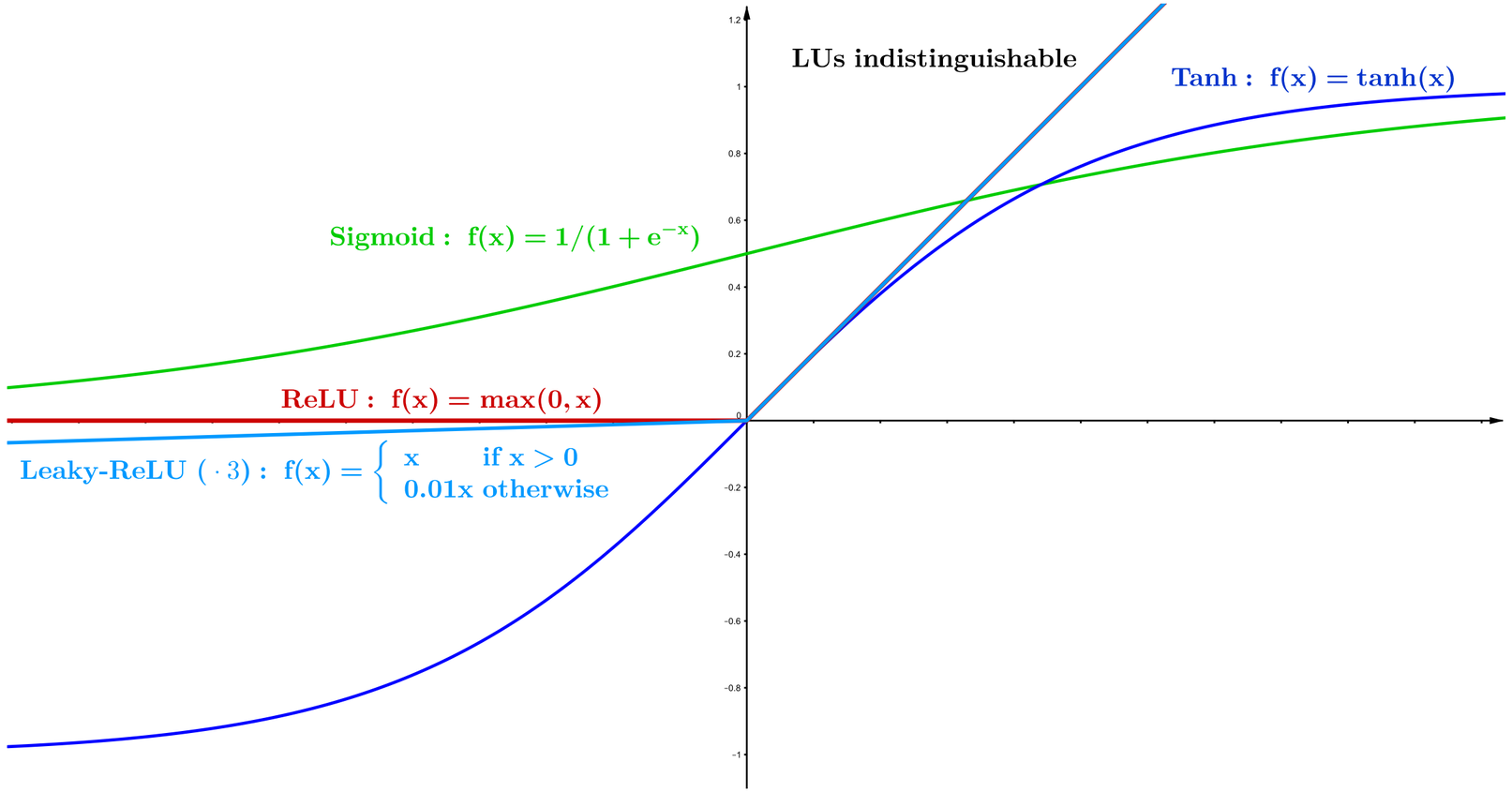


Figure 1: The use of rectified linear units (ReLU) lead to improved results in neural network learning.

Progress in Research

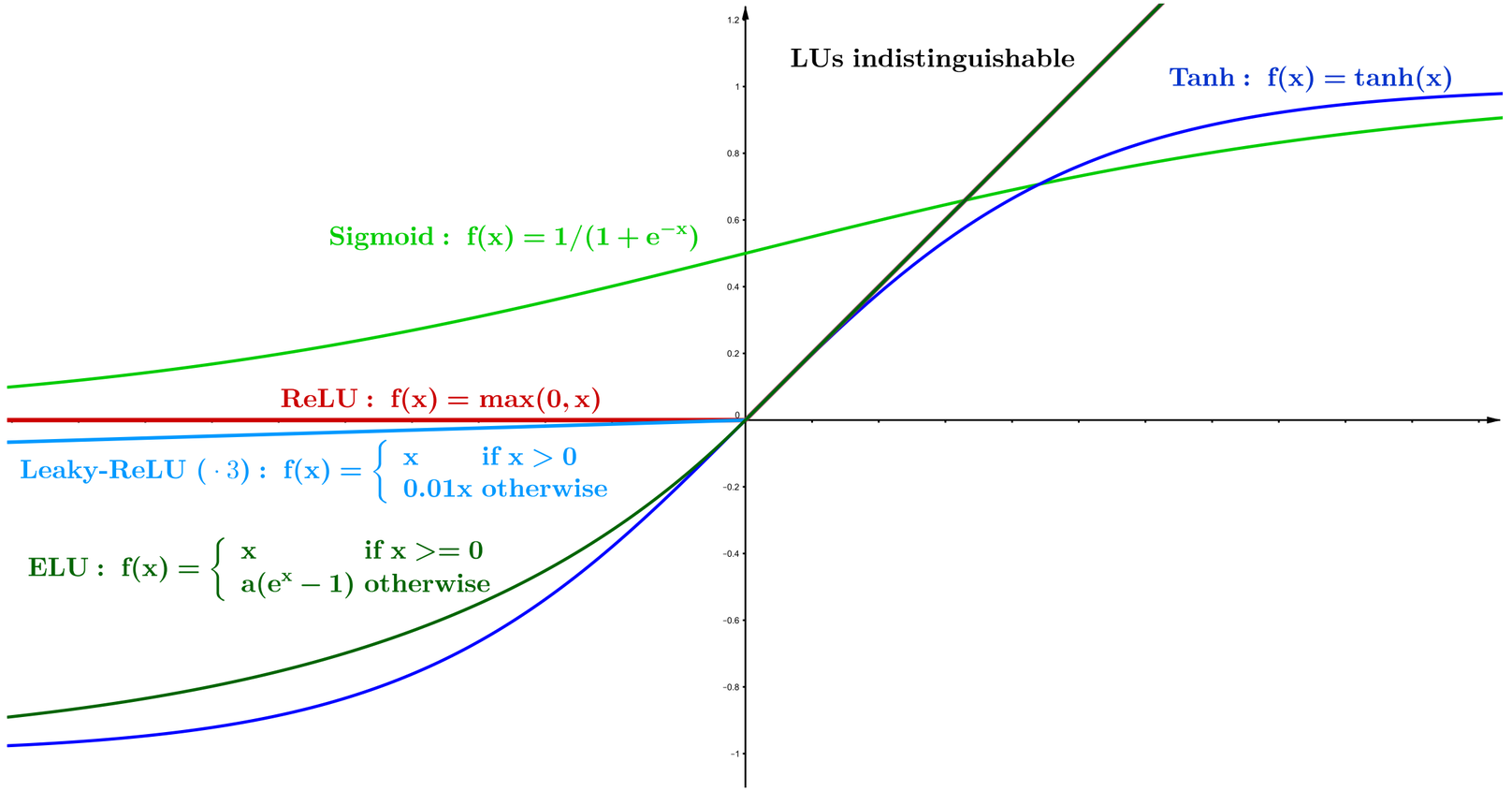
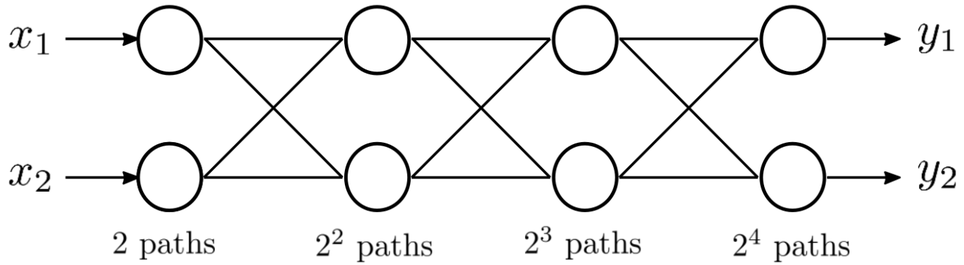


Figure 1: The use of rectified linear units (ReLU) lead to improved results in neural network learning.

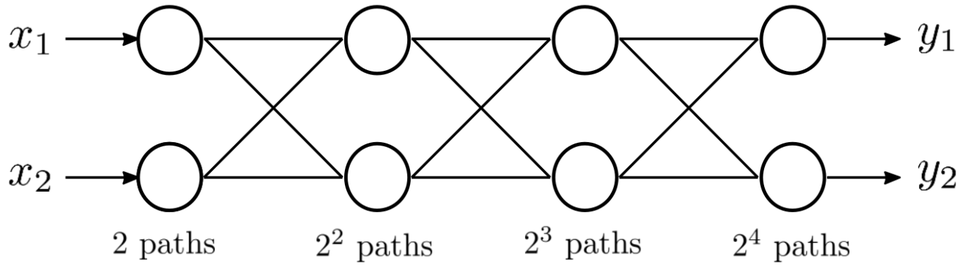
Advantages of Deeper Networks

- Exponential feature reuse:

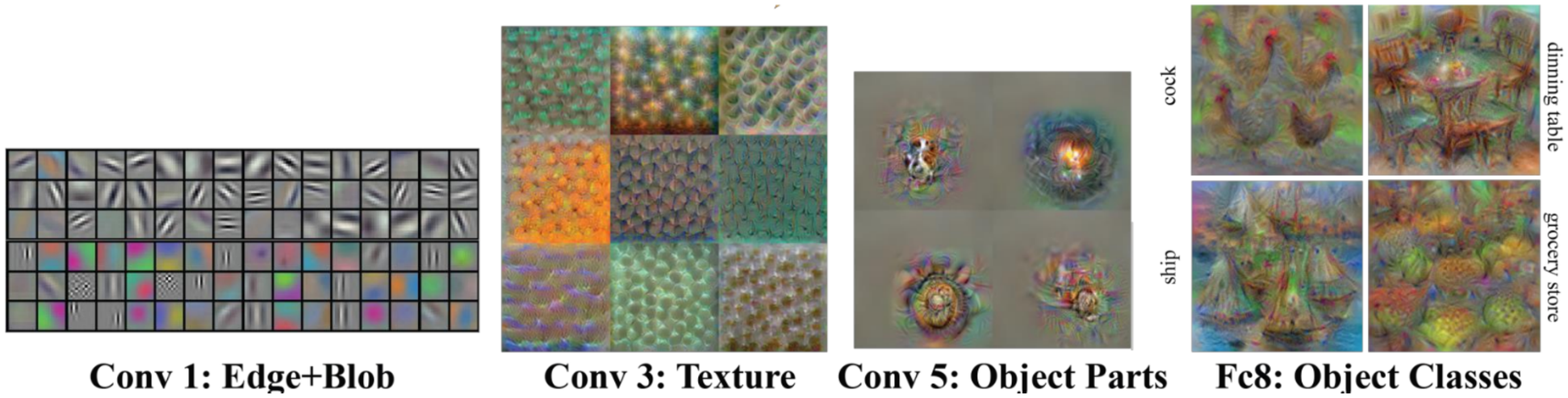


Advantages of Deeper Networks

- Exponential feature reuse:

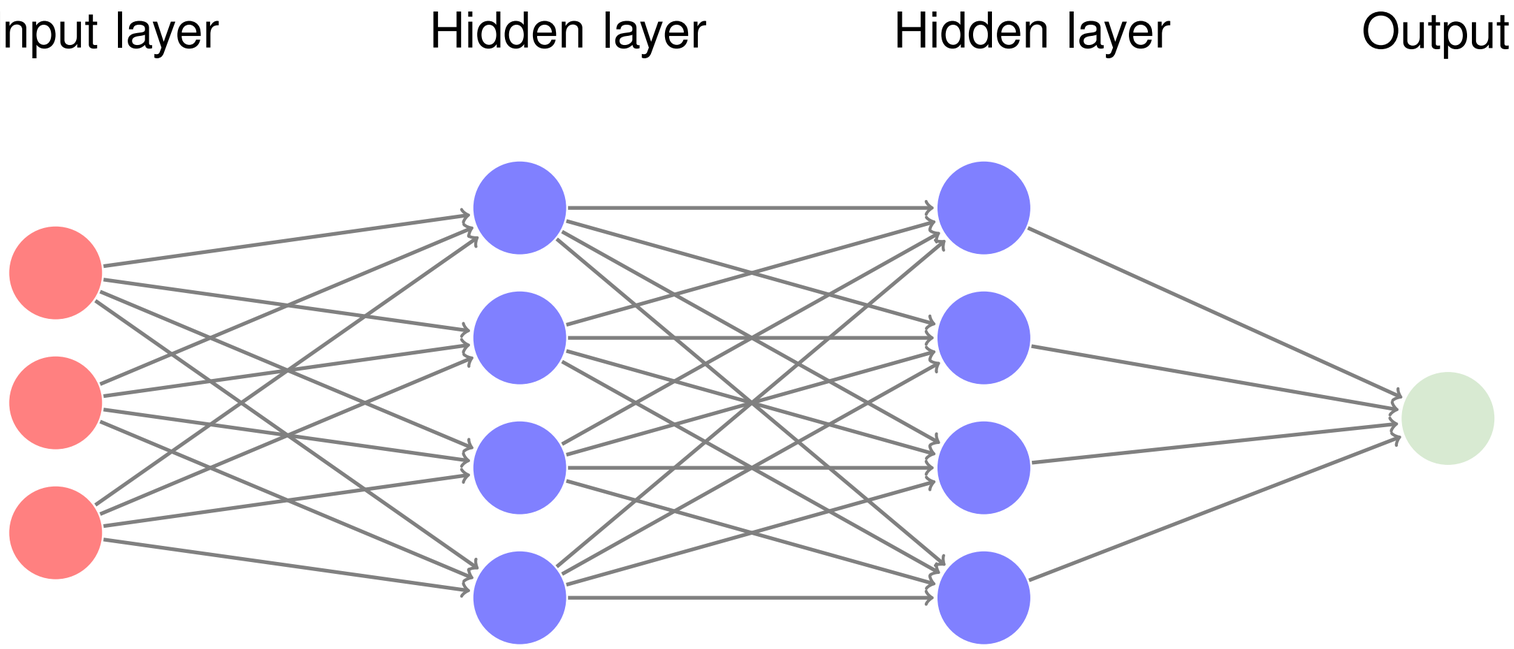


- Increasingly abstract features (Wei et al.):



Topics

Fully-Connected Layer

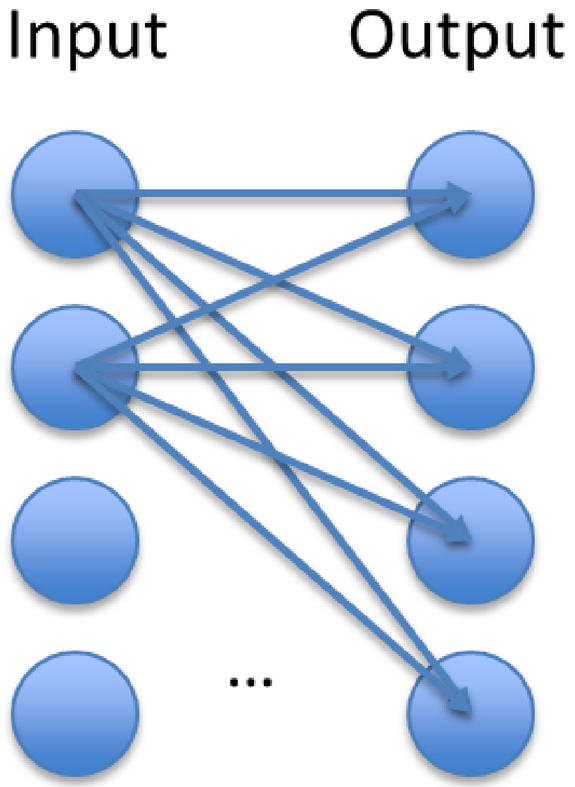


- Fully pairwise connections
- Expensive to train (e. g., $200 \times 200 \times 3$ image $\rightarrow 120,000$ weights in each neuron of the first hidden layer)
- Ignores spatial structure of images
- Still often used as last layer (more common nowadays: average pooling)

Special Layers

Full connectivity corresponds to a matrix multiplication:

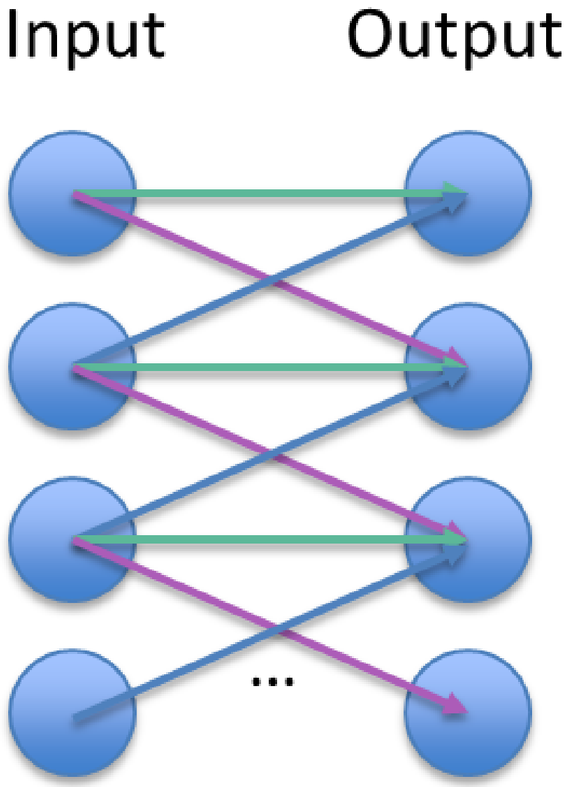
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$



Convolutional Layers

Convolution by constraining the weights:

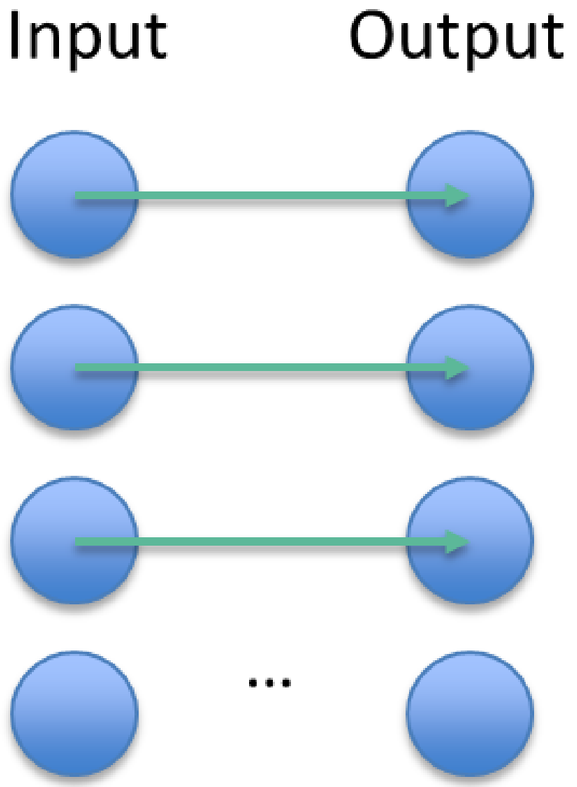
$$y = a * x$$



Multiplicative Layers

Simple weighting:

$$y = a \cdot x$$



Deep Network

- Input: 3-D volume
- Output: 3-D volume
- Convolution along **width** and **height**
  $\rightarrow$ full connectivity along depth!
- ***Receptive field*** = filter size
- Learn set of filter kernels
- # kernels = output volume depth
  = # activation maps
- One slice of the cube: ***activation map***
  (a. k. a. feature map)

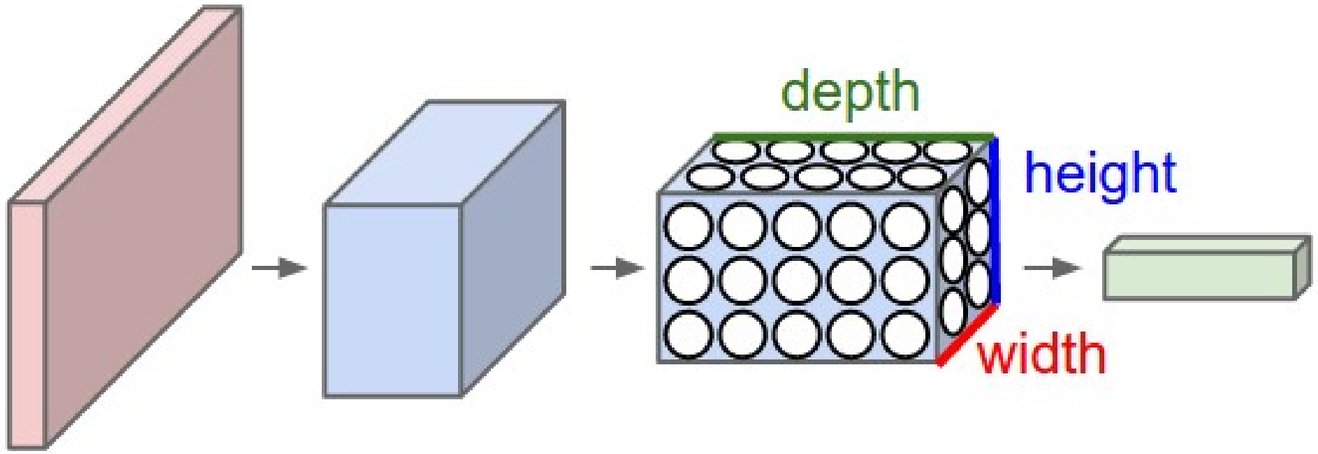


Figure 8: Schematic of convolutional layers (CS231n CNNs for Visual Recognition)

Output Volume Size

Example

- Input volume size: $32 \times 32 \times 3$
- Filter size: 7×7
  $\rightarrow 7 \times 7 \times 3$ weights (+1 for bias)
- 5 filters $\rightarrow$ 5 output maps
- Output volume: $26 \times 26 \times 5$ (padding=0, stride=1)

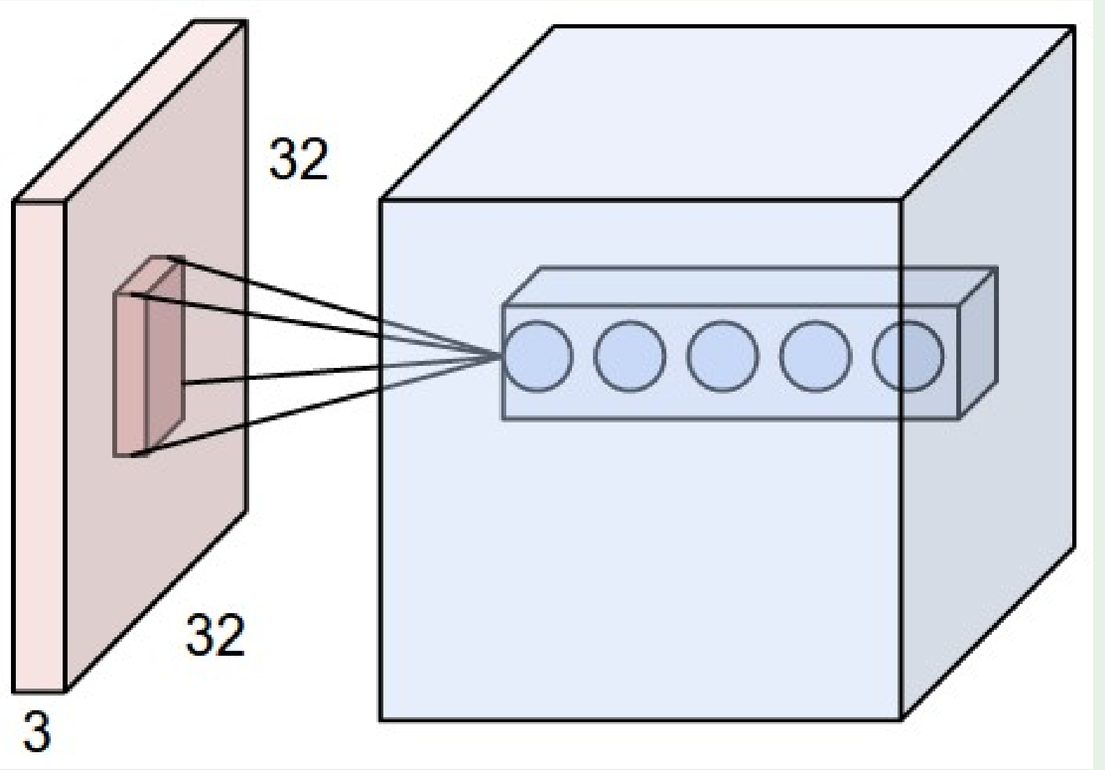


Figure 9: Example with input layer of size $32 \times 32 \times 3$ (CS231n CNNs for Visual Recognition)

Pooling Layers

- Max pooling
- Fractional max pooling
- Mean pooling
- L_p pooling
- Stochastic pooling
- Spatial pyramid pooling
- Generalized pooling
- ...

$\rightarrow$ Instead of pooling: convolution with stride 2

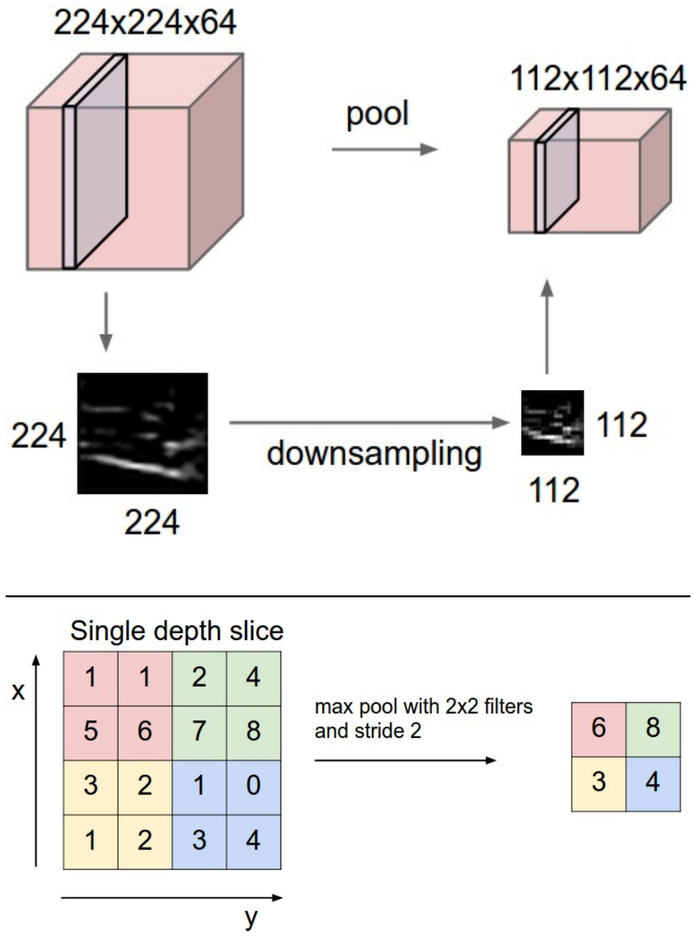


Figure 10: How pooling works (CS231n CNNs for Visual Recognition)

Loss Function

A common choice is the (multinomial) **cross-entropy**:

$$L_{\boldsymbol{y}'}(\boldsymbol{y}) = -\sum_{i=1}^{K} y_i' \log(y_i),$$

with the so-called **softmax** activation:

$$y_i := f(z) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}.$$

Notation:

K	number of classes
$\boldsymbol{z} := \text{net}^L, \boldsymbol{z} \in \mathbb{R}^K$	output layer
$\boldsymbol{y} \in \mathbb{R}^K$	estimated outputs
$\boldsymbol{y}' \in \mathbb{R}^K$	true class – encoded as one-hot vector
y_i	estimated output for class i
y_i'	true label output for class i

Optimization Methods

Gradient descent:

- Let θ_t denote the weights $\{w_{ij}\}$ at specific iteration t:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t).$$

- Iteratively update weights θ in direction of negative gradient of loss function L (steepest descent).
- Step size is defined by the learning rate η.

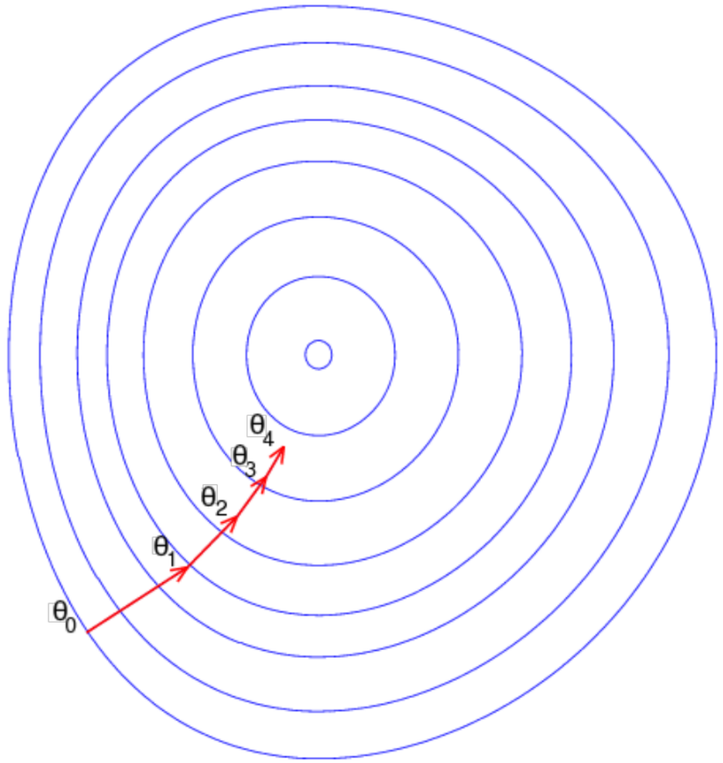


Figure 11: Search optimum in direction of steepest descent.

Optimization Methods

Gradient descent:

- Let θ_t denote the weights $\{w_{ij}\}$ at specific iteration t:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t).$$

- Iteratively update weights θ in direction of negative gradient of loss function L (steepest descent).
- Step size is defined by the learning rate η.

Other optimization methods:

- Stochastic gradient descent (SGD)
- Conjugate descent
- (Quasi-)Newton methods

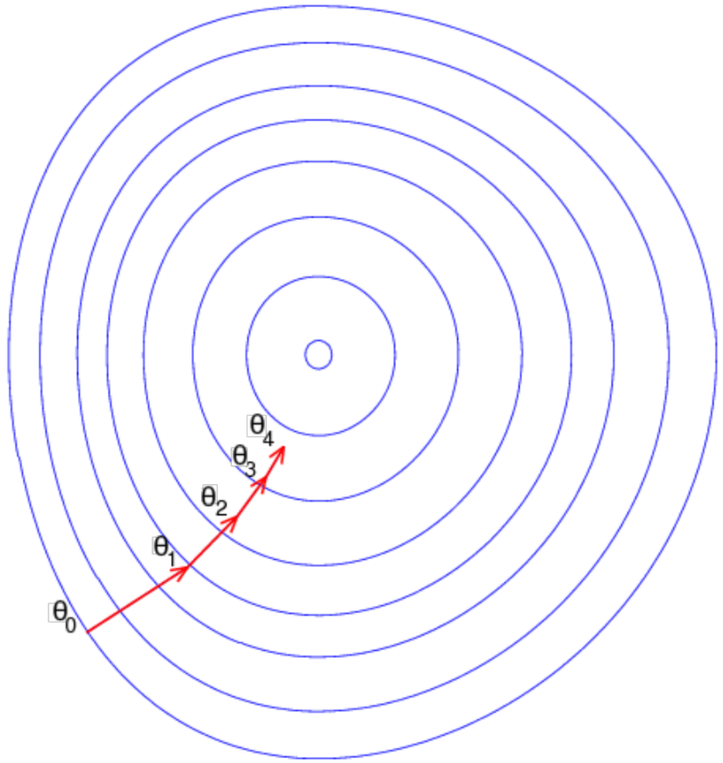


Figure 11: Search optimum in direction of steepest descent.

Optimization: Why Does It Work?

Problems:

- Optimization problem not convex
- Exponential number of local minima

Optimization: Why Does It Work?

Problems:

- Optimization problem not convex
- Exponential number of local minima

Possible Answers (Choromanska et al., 2015, Dauphin et al., 2014)

- Very high dimensional function

$\rightarrow$ Critical points are saddle points

$\rightarrow$ Local minima exist but very close to global minima

- Local minimum might be better than global minima (overfitting!).

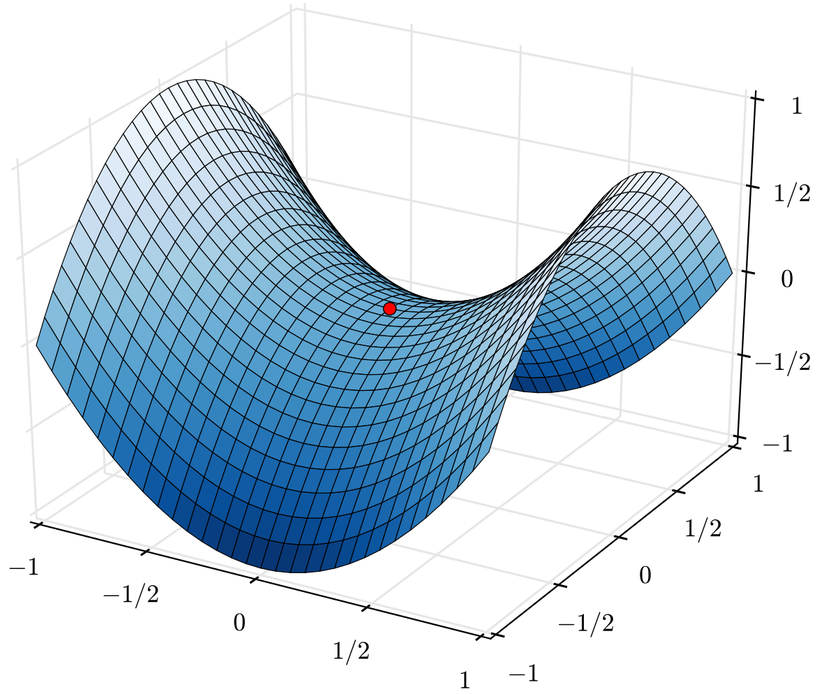


Figure 12: Saddle point of a 2-D function (Wiki Commons)

Topics

Upswing of Neural Networks

Convolutional Neural Networks
- Concept
- Optimization

Examples

Summary
- Take Home Messages
- Further Readings

LeNet-5 (LeCun et al., 1998)

Sequence:

1. Convolution
2. Pooling
3. Non-linearity

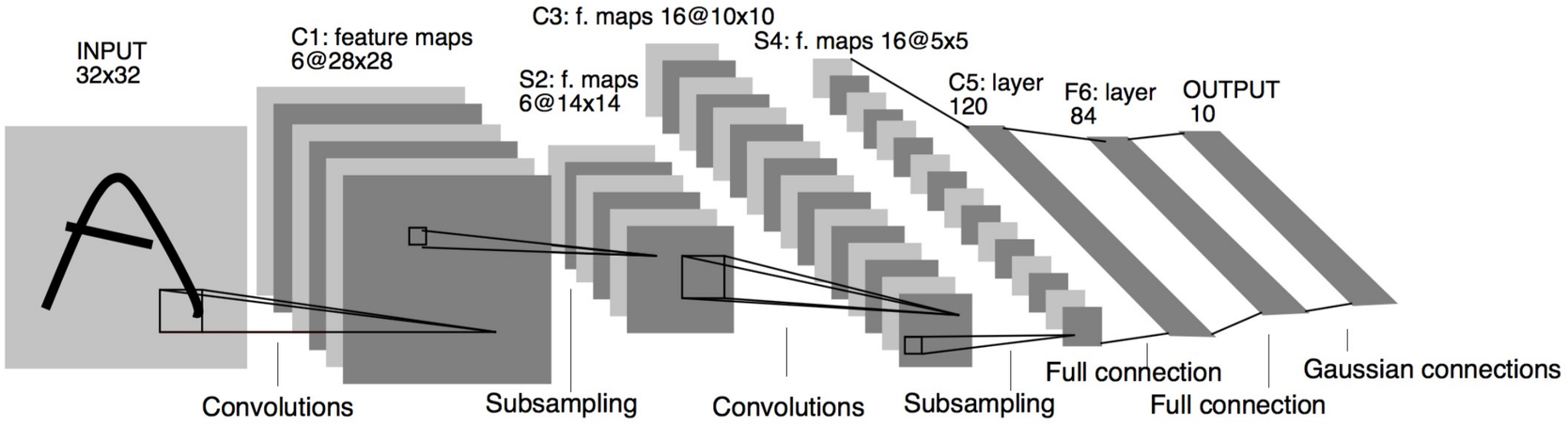


Figure 13: Structure of LeNet (Source: LeCun et al., 1998)

VGG Network (Simonyan & Zisserman, 2014)

- Smaller kernel sizes (3×3)
- Multiple smaller kernels can emulate larger receptive fields
- 16 / 19 Layers

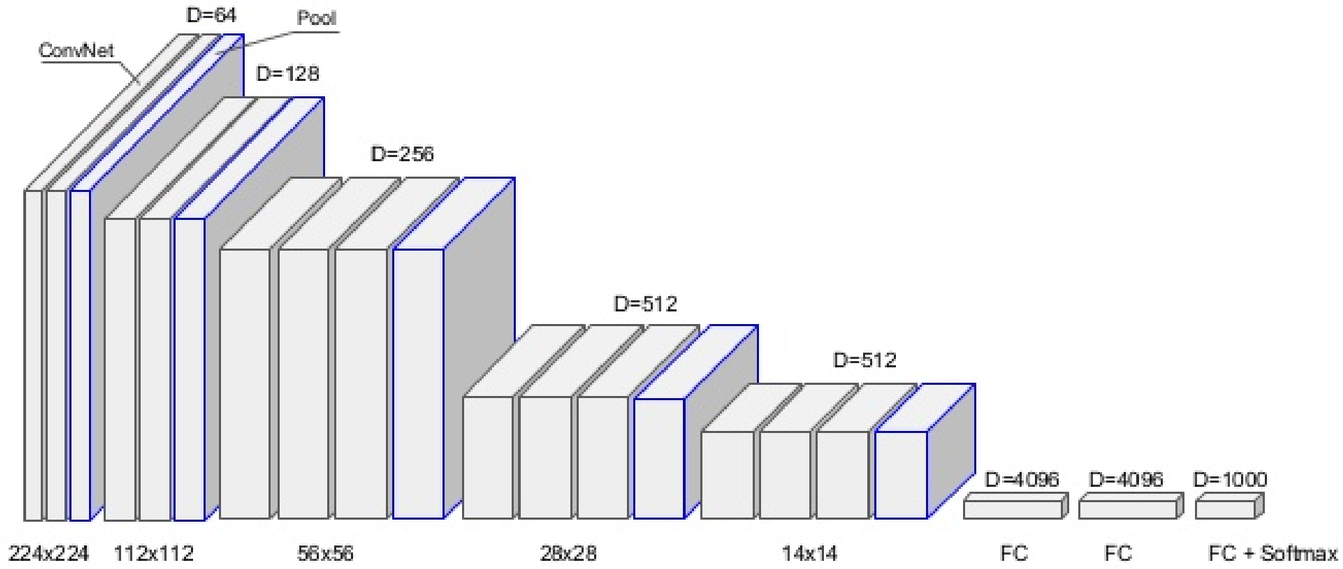


Figure 14: Structure of VGGNet (Source: Louis Monier, Holberton School)

Topics

Take Home Messages

- Around 2015 neural networks regained a lot of attention in the machine learning community. Its success reached public attention as well.
- The choice of deep network structures and certain activation functions made deep learning a powerful tool for a lot of tasks, including object recognition and segmentation.
- Through the success of deep learning, a lot of publications, libraries and pre-trained networks are available now (often open access / open source).

Further Readings

- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539
- Anna Choromanska et al. “The Loss Surface of Multilayer Networks”. In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Guy Lebanon and S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. San Diego, California, USA, May 2015, pp. 192–204
- Yann N. Dauphin et al. “Identifying and Attacking the Saddle Point Problem in High-dimensional Non-convex Optimization”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. 2014, pp. 2933–2941
- Y. LeCun et al. “Gradient-based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. DOI: 10.1109/5.726791
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. 2012, pp. 1097–1105
- Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *Computing Research Repository* abs/1409.1556 (2014). Accessed: 25. May 2017. URL: https://arxiv.org/abs/1409.1556

Medical Image Processing for Interventional Applications

Image Denoising with Deep Learning

Online Course – Unit 28

Andreas Maier, Katrin Mentl, Frank Schebesch
Pattern Recognition Lab (CS 5)

Topics

Image Denoising in X-ray Imaging

Denoising using Sparse Coding Networks

Summary

- Take Home Messages
- Further Readings

Image Denoising

Importance in X-ray imaging: Reduction of harmful patient radiation exposure results in low-dose images which suffer from noise ➔ impedes accurate diagnosis.

Goal: Reconstruct initial image $\boldsymbol{I}_0 \in \mathbb{R}^M$ from its noisy low-dose measurement $\boldsymbol{I}_{\boldsymbol{n}} \in \mathbb{R}^M$, which is usually assumed to be corrupted by *a*dditive zero-mean white Gaussian noise $\boldsymbol{n} \in \mathbb{R}^M$ as:

$$\boldsymbol{I}_{\boldsymbol{n}} = \boldsymbol{I}_0 + \boldsymbol{n}.$$

Remark: Depending on the application, image noise can be described by different noise models or assumed to be unknown.

Application Example: X-ray Fluoroscopy

Acquisition of multiple images per session:

➔ monitor dynamic processes during interventional procedures

➔ further decrease in radiation dose per frame

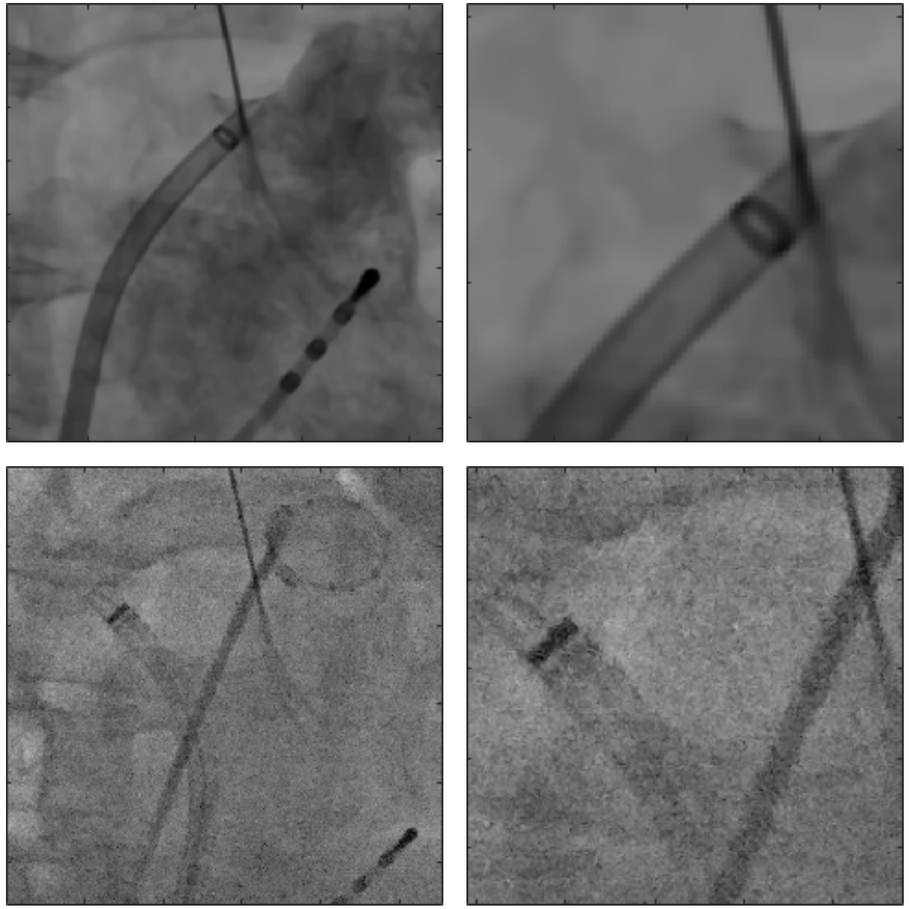


Figure 1: High and low-dose X-ray fluoroscopy images

Application Example: X-ray Fluoroscopy

Acquisition of multiple images per session:

➔ monitor dynamic processes during interventional procedures

➔ further decrease in radiation dose per frame

Requirements for denoising algorithm:

- Reconstruction of sharp edges (organ boundaries, medical tools)
- No introduction of artifacts
- Real-time processing (e. g., during interventional procedures)

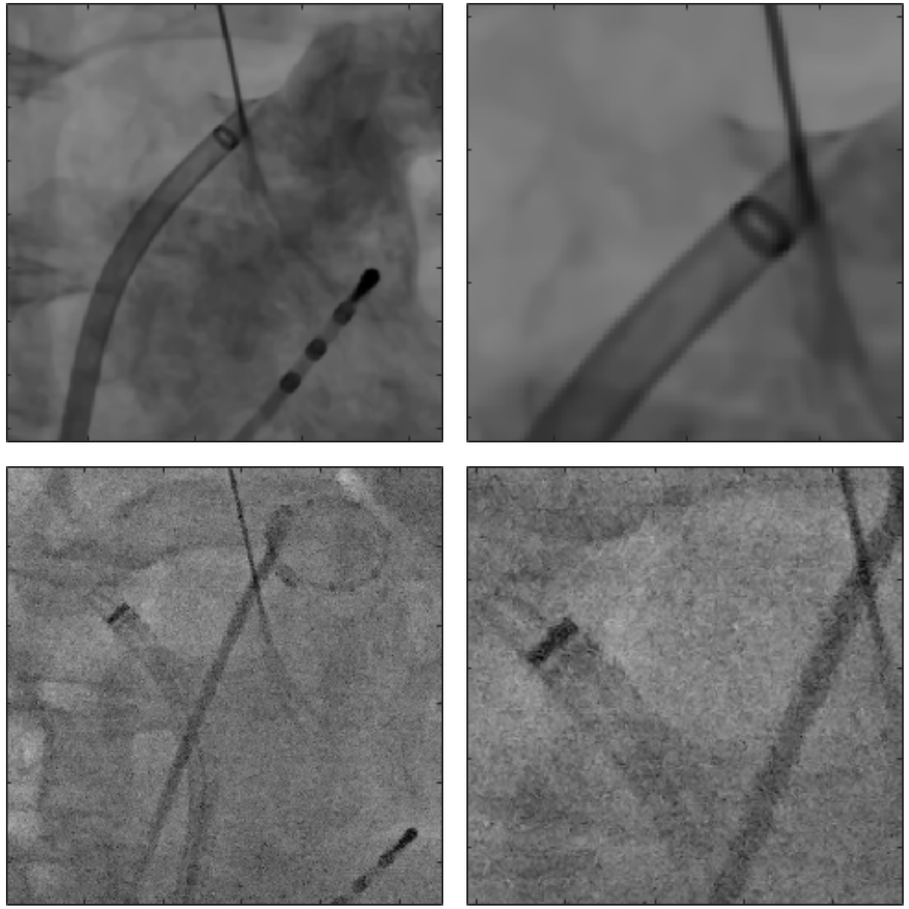


Figure 1: High and low-dose X-ray fluoroscopy images

Sparse Coding Approach to Image Denoising

Assumption: A noiseless signal can be represented as a ***sparse*** linear combination of basis vectors.

➔ A noisy signal will **violate** this sparsity assumption.

Sparse Coding Approach to Image Denoising

Assumption: A noiseless signal can be represented as a ***sparse*** linear combination of basis vectors.
➔ A noisy signal will **violate** this sparsity assumption.

Formulation of the denoising problem:

- Find the sparsest representation $\boldsymbol{\alpha} \in \mathbb{R}^L$ (coefficients) of a corrupted image $\boldsymbol{I_n}$ in a basis $\boldsymbol{D} \in \mathbb{R}^{M \times L}$ by:

$$\left(\hat{\boldsymbol{D}}, \hat{\boldsymbol{\alpha}}\right) = \arg\min_{\boldsymbol{D},\boldsymbol{\alpha}} \phi\left(\boldsymbol{I_n}, \boldsymbol{D\alpha}\right) + \lambda\, \psi\left(\boldsymbol{\alpha}\right),$$

  where ϕ denotes a distance metric between the noisy image and the found solution, and ψ denotes a sparsity enforcement penalty.

Sparse Coding Approach to Image Denoising

Assumption: A noiseless signal can be represented as a ***sparse*** linear combination of basis vectors.
➔ A noisy signal will **violate** this sparsity assumption.

Formulation of the denoising problem:

- Find the sparsest representation $\boldsymbol{\alpha} \in \mathbb{R}^L$ (coefficients) of a corrupted image $\boldsymbol{I_n}$ in a basis $\boldsymbol{D} \in \mathbb{R}^{M \times L}$ by:

$$\left(\hat{\boldsymbol{D}}, \hat{\boldsymbol{\alpha}}\right) = \arg\min_{\boldsymbol{D},\boldsymbol{\alpha}} \phi\left(\boldsymbol{I_n}, \boldsymbol{D\alpha}\right) + \lambda\, \psi\left(\boldsymbol{\alpha}\right),$$

  where ϕ denotes a distance metric between the noisy image and the found solution, and ψ denotes a sparsity enforcement penalty.
- Estimate of the denoised image:

$$\boldsymbol{I}_r = \hat{\boldsymbol{D}}\hat{\boldsymbol{\alpha}}$$

Sparse Coding Approach to Image Denoising

Assumption: A noiseless signal can be represented as a ***sparse*** linear combination of basis vectors.
➔ A noisy signal will **violate** this sparsity assumption.

Formulation of the denoising problem:

- Find the sparsest representation $\boldsymbol{\alpha} \in \mathbb{R}^L$ (coefficients) of a corrupted image $\boldsymbol{I_n}$ in a basis $\boldsymbol{D} \in \mathbb{R}^{M \times L}$ by:

$$\left(\hat{\boldsymbol{D}}, \hat{\boldsymbol{\alpha}}\right) = \arg\min_{\boldsymbol{D},\boldsymbol{\alpha}} \phi\left(\boldsymbol{I_n}, \boldsymbol{D\alpha}\right) + \lambda\, \psi\left(\boldsymbol{\alpha}\right),$$

  where ϕ denotes a distance metric between the noisy image and the found solution, and ψ denotes a sparsity enforcement penalty.
- Estimate of the denoised image:

$$\boldsymbol{I}_r = \hat{\boldsymbol{D}}\hat{\boldsymbol{\alpha}}$$

➔ Difficult non-convex optimization problem

Iterative Shrinkage Algorithms

Assumption: A suitable representation basis ***D*** is given.
Observation: Noise spreads almost uniformly across the transform domain:

- low-magnitude coefficients mainly due to noise,
- essential image features captured by few high-magnitude coefficients.

Iterative Shrinkage Algorithms

Assumption: A suitable representation basis $\boldsymbol{D}$ is given.
Observation: Noise spreads almost uniformly across the transform domain:

- low-magnitude coefficients mainly due to noise,
- essential image features captured by few high-magnitude coefficients.

Idea: Enforce sparsity by iteratively applying a shrinkage function h_θ to the representation coefficients:

$$\boldsymbol{\alpha}_{i+1} = h_\theta\left(\boldsymbol{\alpha}_i - \mu \boldsymbol{D}^\mathsf{T}\left(\boldsymbol{D}\boldsymbol{\alpha}_i - \boldsymbol{I_n}\right)\right), \quad \mu > 0.$$

h_θ depends on the sparsity constraint.
Problem: This algorithm can be slow to converge.

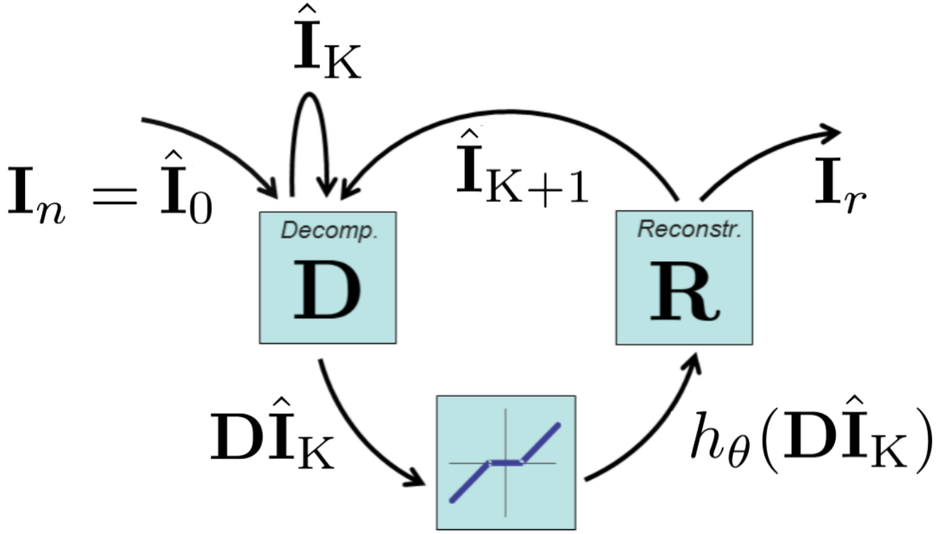


Figure 2: Scheme of an iterative shrinkage thresholding algorithm (ISTA)

Topics

Image Denoising in X-ray Imaging

Denoising using Sparse Coding Networks

Summary

- Take Home Messages
- Further Readings

Denoising using Sparse Coding Networks

Question: Can we train a neural network to do the denoising likewise?

➔ Unfold each iteration into a layer of a **neural network**, trained as a ***denoising autoencoder***.

Denoising Autoencoder

- The noisy inputs $\boldsymbol{I_n}$ are mapped to a **hidden representation** $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha} = h_\theta(\boldsymbol{b} + \boldsymbol{D}\boldsymbol{I_n}).$$

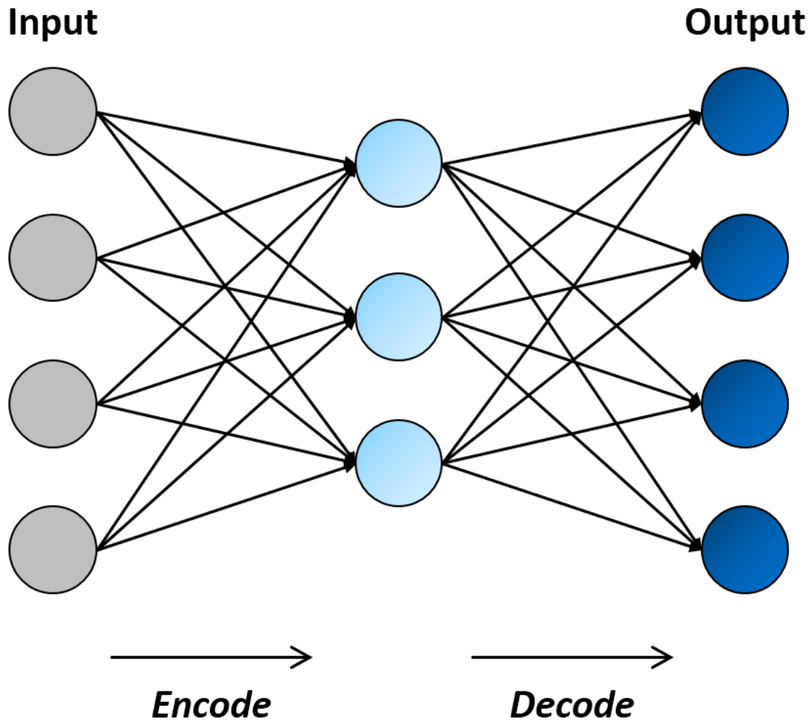


Figure 3: Autoencoder scheme

Denoising Autoencoder

- The noisy inputs $\boldsymbol{I_n}$ are mapped to a **hidden representation** $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha} = h_\theta(\boldsymbol{b} + \boldsymbol{D}\boldsymbol{I_n}).$$

- The hidden representation is mapped linearly back into a **denoised estimate** $\boldsymbol{I_r}$ of the same shape as the input by:

$$\boldsymbol{I_r} = \boldsymbol{b'} + \boldsymbol{R}\boldsymbol{\alpha},$$

where $\boldsymbol{R}$, $\boldsymbol{b'}$ denote the operator for the image reconstruction and the according bias, respectively.

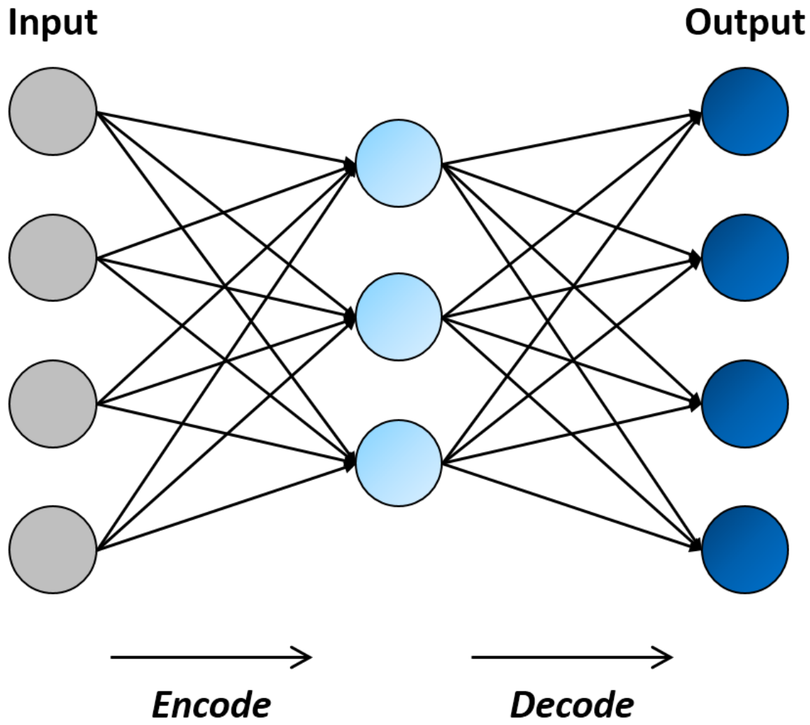


Figure 3: Autoencoder scheme

Denoising Autoencoder

- The noisy inputs $\boldsymbol{I_n}$ are mapped to a **hidden representation** $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha} = h_\theta(\boldsymbol{b} + \boldsymbol{D}\boldsymbol{I_n}).$$

- The hidden representation is mapped linearly back into a **denoised estimate** $\boldsymbol{I_r}$ of the same shape as the input by:

$$\boldsymbol{I_r} = \boldsymbol{b'} + \boldsymbol{R}\boldsymbol{\alpha},$$

  where $\boldsymbol{R}$, $\boldsymbol{b'}$ denote the operator for the image reconstruction and the according bias, respectively.
- Parameters $\boldsymbol{D}$, $\boldsymbol{R}$, $\boldsymbol{b}$, $\boldsymbol{b'}$ are updated to minimize the loss $L(\boldsymbol{I_r}, \boldsymbol{I_0})$ using **backpropagation**.
- ***Sparse*** denoising autoencoder ➔ sparsity-enforcing activation h_θ
- **Convolutional** layout: $\boldsymbol{D}$, $\boldsymbol{R}$ consist of a set of learnable filters.

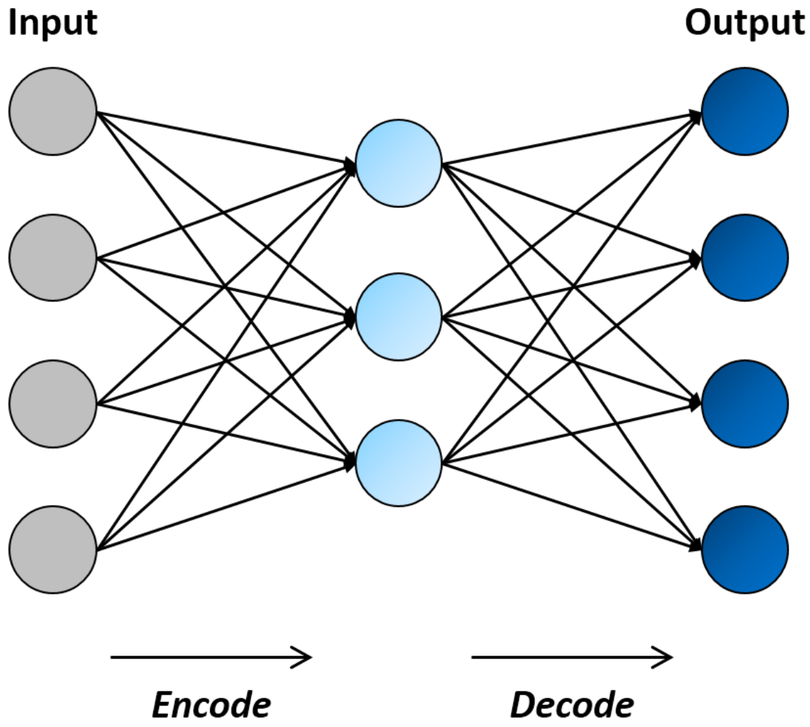


Figure 3: Autoencoder scheme

Activation Function

- Threshold each representation coefficient:

$$\hat{\alpha}_j = h_\theta(\alpha_j) = \frac{(\alpha_j^2 - k\sigma^2)_+}{\alpha_j}.$$

- Optimal threshold is proportional to the noise standard deviation σ.
- Scale factor k is adjusted during training.

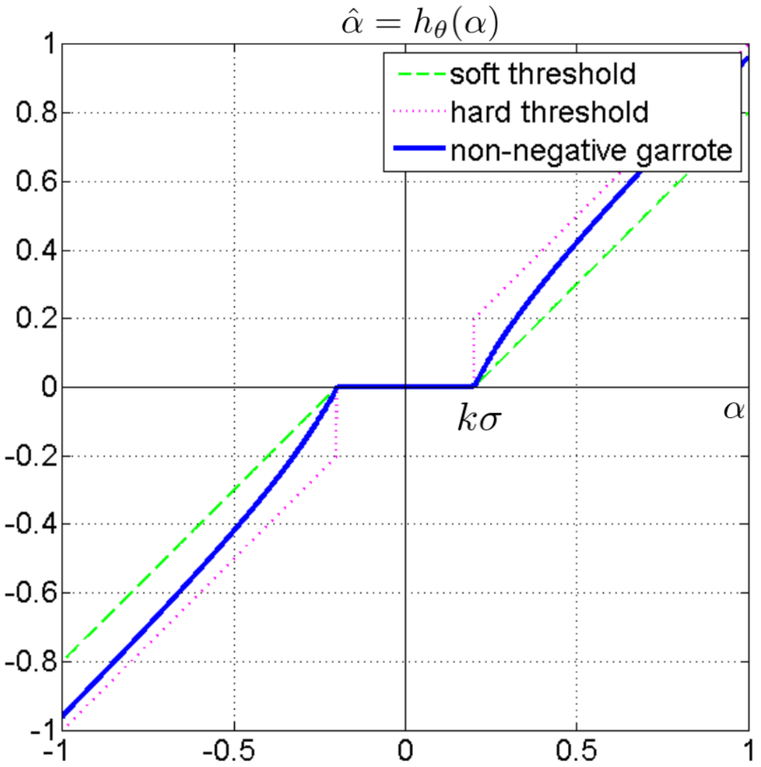


Figure 4: Non-negative garrote function

Single-scale Sparse Coding Network

The network is trained as ***sparse convolutional denoising autoencoder***:

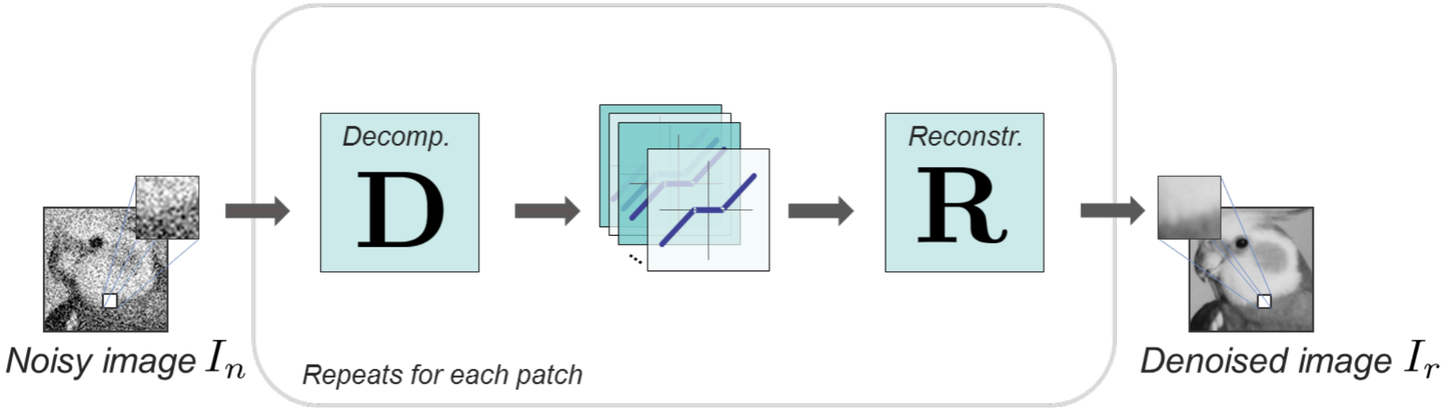


Figure 5: Diagram depicting the workflow in the single-scale network

Single-scale Sparse Coding Network

The network is trained as ***sparse convolutional denoising autoencoder***:

1. **decomposition** of noisy input signal to hidden representation coefficients using a convolutional layer,

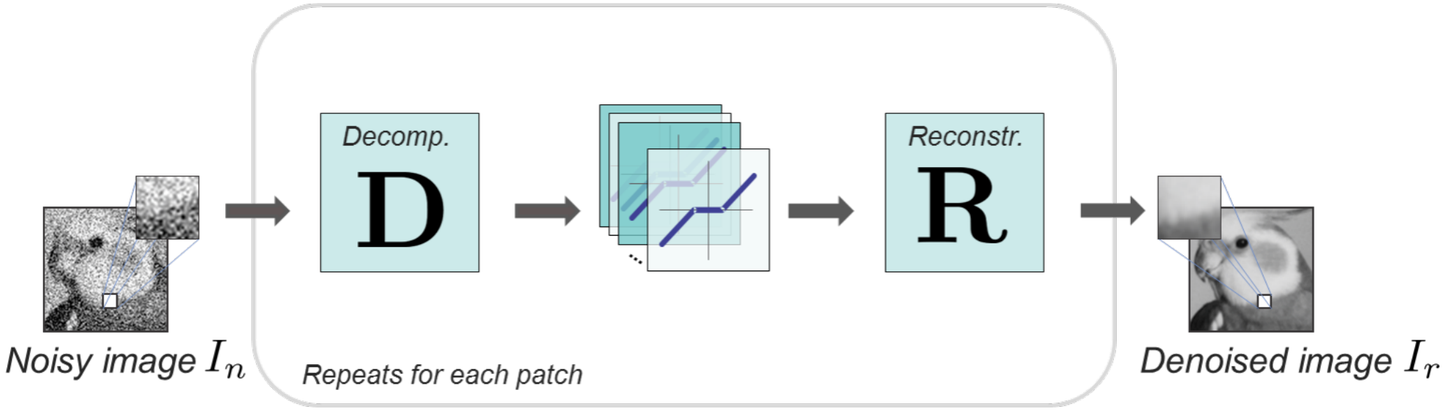


Figure 5: Diagram depicting the workflow in the single-scale network

Single-scale Sparse Coding Network

The network is trained as ***sparse convolutional denoising autoencoder***:

1. **decomposition** of noisy input signal to hidden representation coefficients using a convolutional layer,
2. **thresholding** of the coefficients using a shrinkage function,

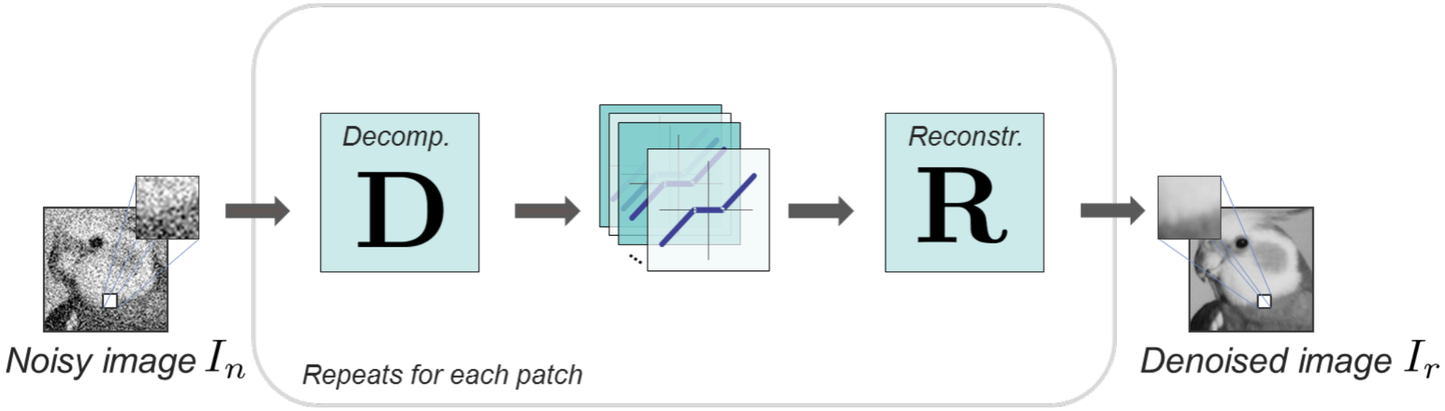


Figure 5: Diagram depicting the workflow in the single-scale network

Single-scale Sparse Coding Network

The network is trained as ***sparse convolutional denoising autoencoder***:

1. **decomposition** of noisy input signal to hidden representation coefficients using a convolutional layer,
2. **thresholding** of the coefficients using a shrinkage function,
3. **reconstruction** to the same shape as the input signal using a convolutional transpose layer.

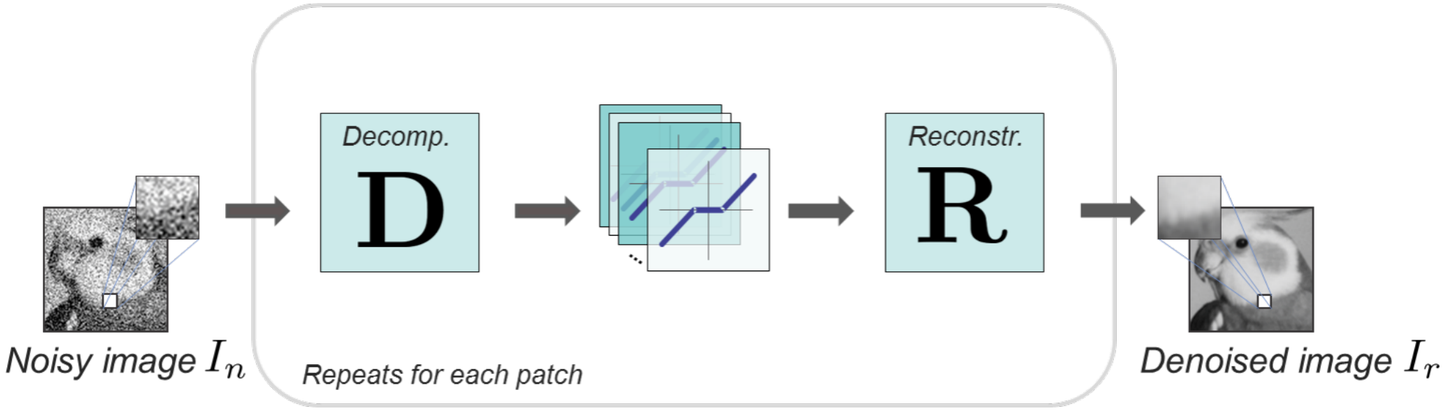


Figure 5: Diagram depicting the workflow in the single-scale network

Multiscale Sparse Coding Network

Goal: Capture important image features on various scales and effectively reduce noise at the same time.

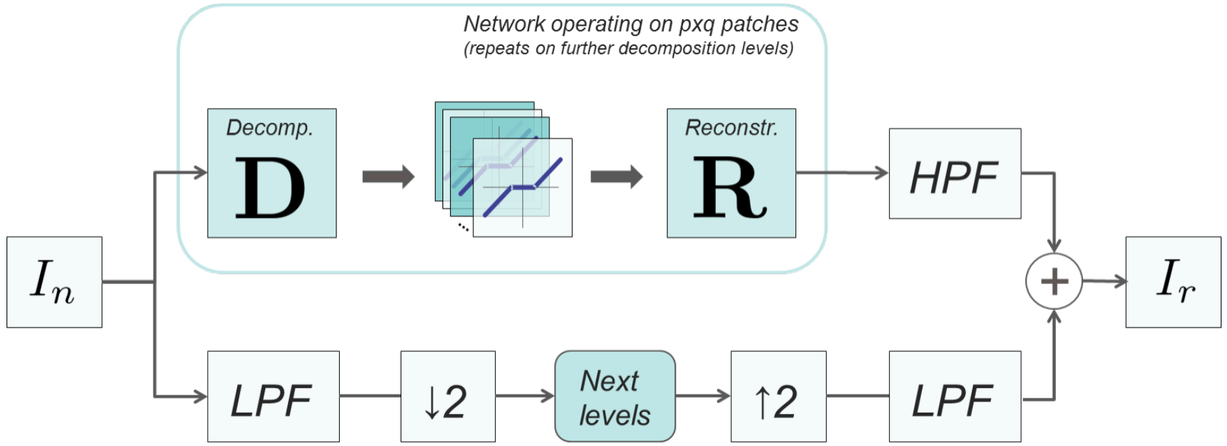


Figure 6: Diagram depicting the workflow in the multi-scale network

Multiscale Sparse Coding Network

Goal: Capture important image features on various scales and effectively reduce noise at the same time.

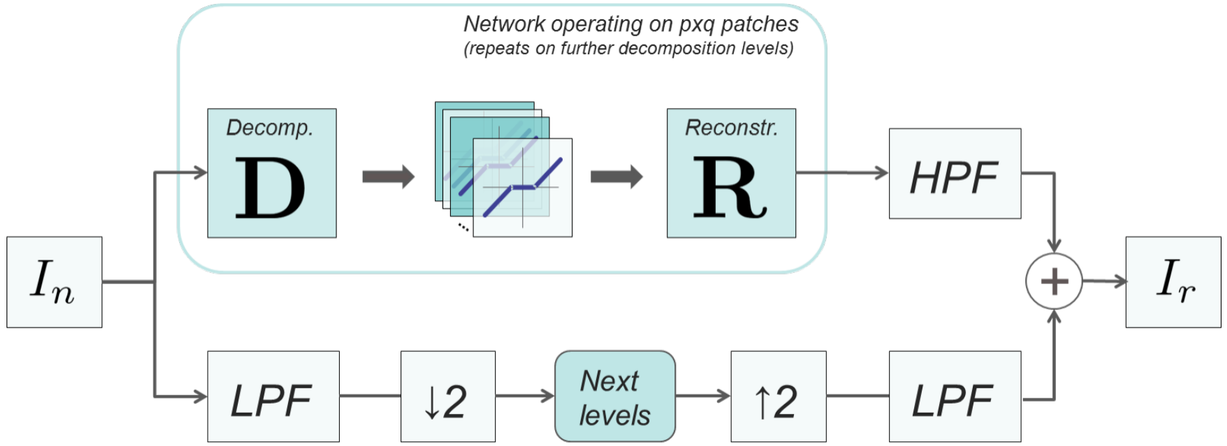


Figure 6: Diagram depicting the workflow in the multi-scale network

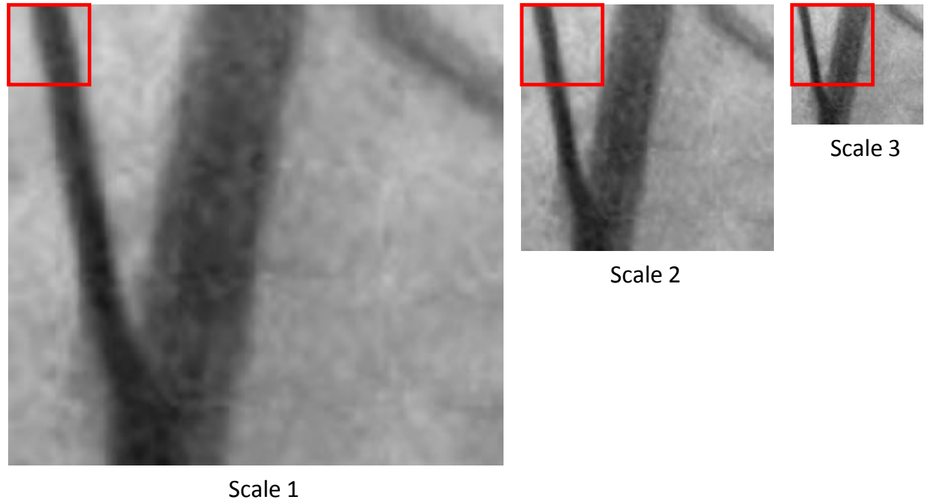


Figure 7: Downsampled versions of the input image (top) yield different representation filter kernels on each scale (bottom).

From Single-scale to Multiscale Networks

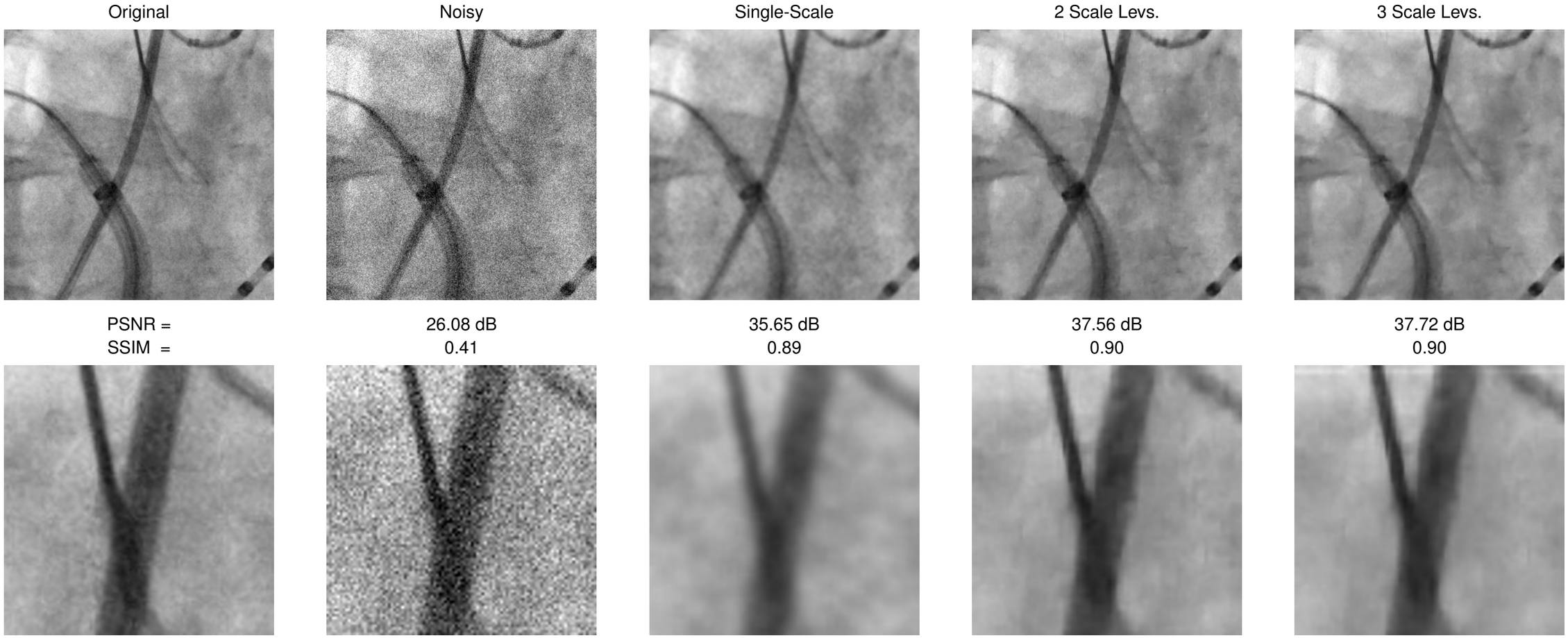


Figure 8: Results of image denoising with shallow sparse coding networks with increasing levels of decomposition. The single-scale network produces the result for $L = 1$ scale level and the multiscale network is used for scale levels $L = 2$ and $L = 3$.

Deep Multiscale Sparse Coding Network

- **Cascade** multiple shallow multiscale sparse coding layers to obtain a deep network structure.
- **Each layer** corresponds to an ***iteration*** of the iterative shrinkage algorithm.
- **Train** the recombination weights ω_{k1}, ω_{k2}, ω_{k3}.

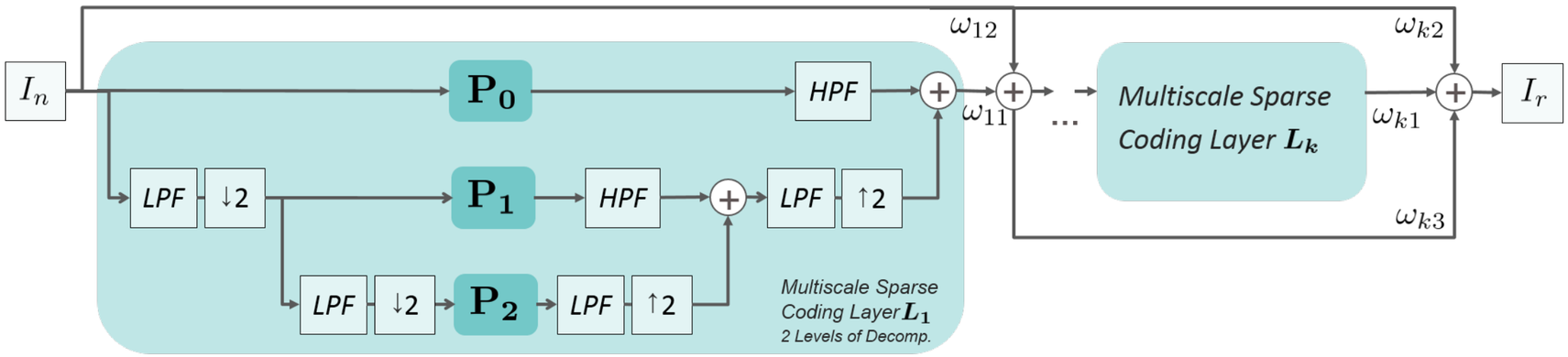


Figure 9: Diagram depicting the workflow in the deep network

From Shallow to Deep Networks

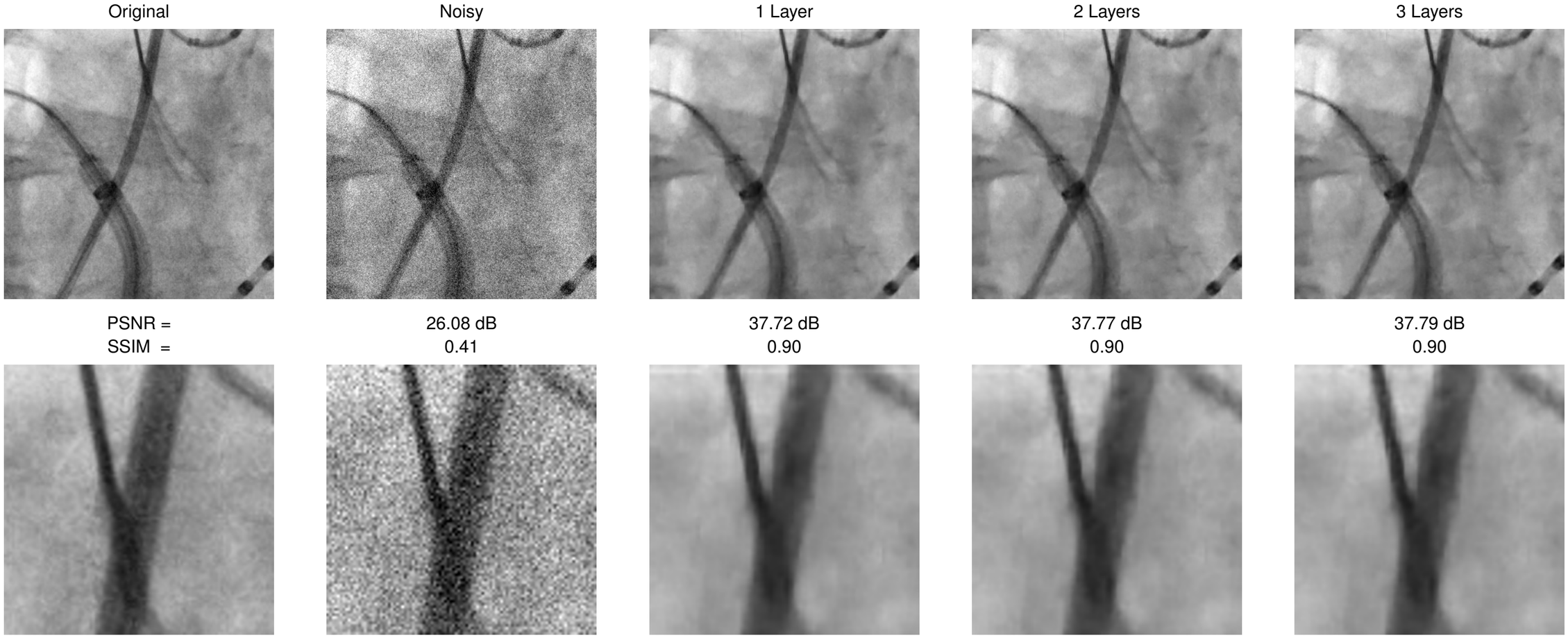


Figure 10: Results of image denoising with a deep network built by cascading an increasing number (from $K = 1$ to $K = 3$) of multiscale networks operating on $L = 3$ scale levels.

Comparison Deep Network Output to Original Image

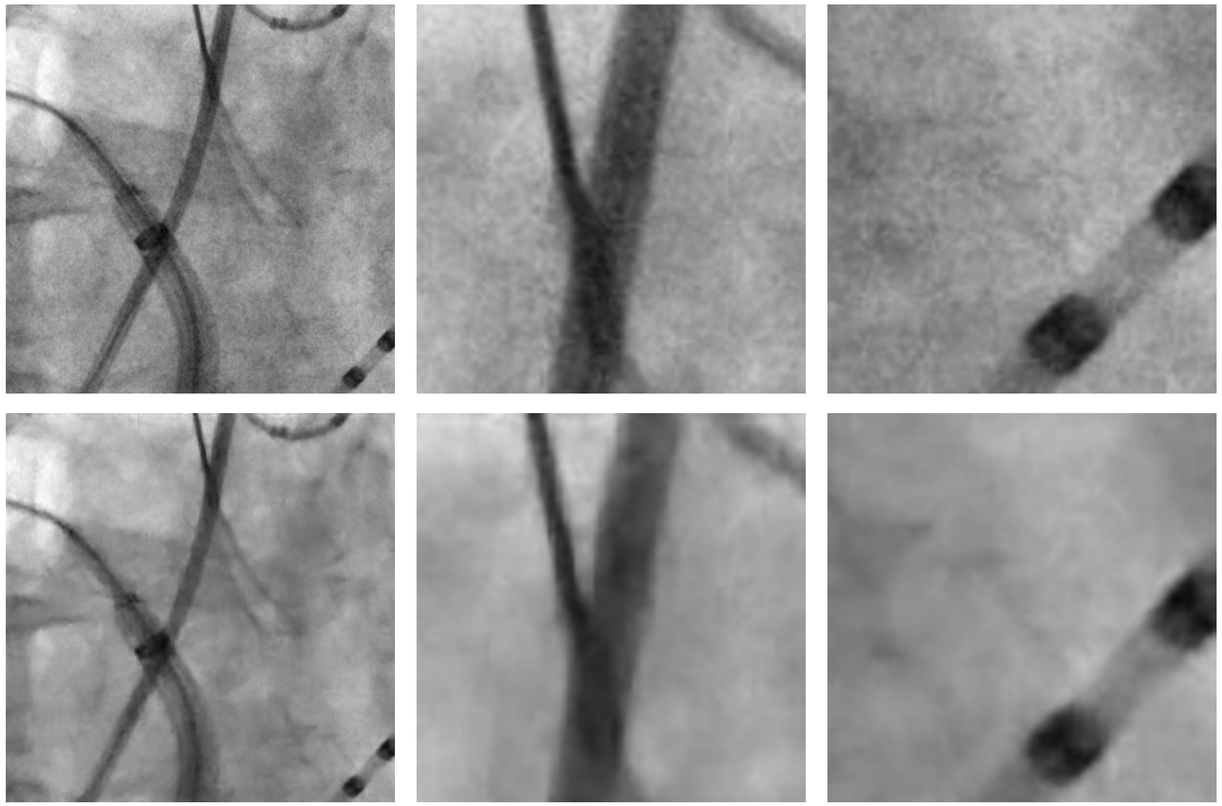
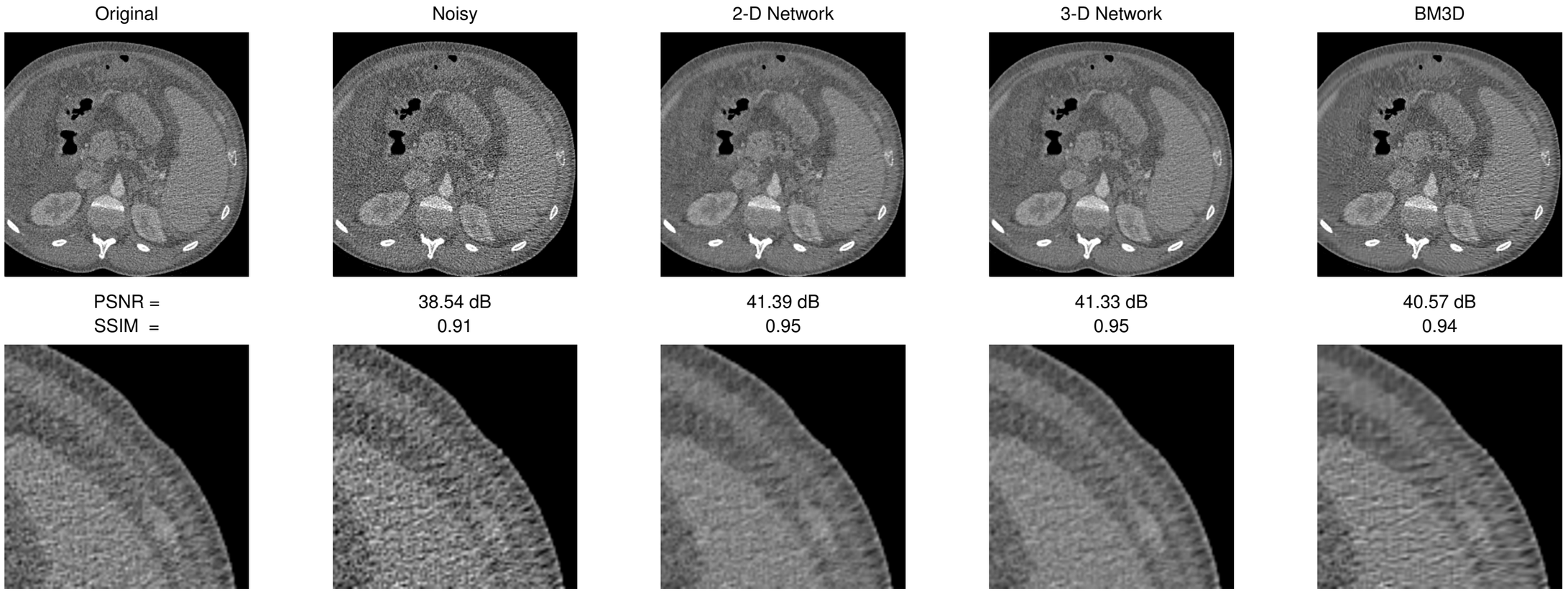


Figure 11: Comparison of the original image (top left) and two different regions (top middle/right) to the respective deep network output images (bottom)

2-D/3-D Computed Tomography (CT) Experiments



- High-dose datasets from 10 patients and corresponding synthetically corrupted datasets are provided.
- Noise level and underlying noise model are unknown.
- Results are compared to the state-of-the-art in image denoising: *block-matching and 3-D filtering* (BM3D).

Topics

Image Denoising in X-ray Imaging

Denoising using Sparse Coding Networks

Summary

- Take Home Messages
- Further Readings

Take Home Messages

- The denoising networks are inspired by sparsity-based denoising algorithms and are based on learning a sparse representation basis for 2-D/3-D data.
- The multiscale network shows superior results over the single-scale network, and deeper networks can improve the results further.
- The state-of-the-art in image denoising BM3D introduces artifacts into denoised images, while the shown networks produce comparable or even superior results without generating any noticable artifacts.

Further Readings

- Amir Beck and Marc Teboulle. “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems”. In: *SIAM Journal on Imaging Sciences* 2.1 (Mar. 2009), pp. 183–202. DOI: 10.1137/080716542
- Leo Breiman. “Better Subset Regression Using the Nonnegative Garrote”. In: *Technometrics* 37.4 (Nov. 1995), pp. 373–384. DOI: 10.2307/1269730
- Kostadin Dabov et al. “Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering”. In: *IEEE Transactions on Image Processing* 16.8 (Aug. 2007), pp. 2080–2095. DOI: 10.1109/TIP.2007.901238
- Ingrid Daubechies, Michel Defrise, and Christine De Mol. “An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint”. In: *Communications on Pure and Applied Mathematics* 57.11 (Aug. 2004), pp. 1413–1457. DOI: 10.1002/cpa.20042
- Karol Gregor and Yann LeCun. “Learning Fast Approximations of Sparse Coding”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Ed. by Johannes Fürnkranz and Thorsten Joachims. Haifa, Israel: Omnipress, June 2010, pp. 399–406
- Yevgen Matviychuk et al. “Learning a Multiscale Patch-based Representation for Image Denoising in X-ray Fluoroscopy”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2016, pp. 2330–2334. DOI: 10.1109/ICIP.2016.7532775

Medical Image Processing for Interventional Applications

Inpainting with Deep Learning

Online Course – Unit 29

Andreas Maier, Mathias Unberath, Jonas Hajek, Frank Schebesch
Pattern Recognition Lab (CS 5)

virtuelle hochschule bayern

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Topics

Digital Subtraction Angiography

Inpainting Using U-Net

Summary

- Take Home Messages
- Further Readings

Digital Subtraction Angiography

- **Angiography:** imaging of blood vessels, e. g., to detect stenoses

Figure 1: Scan showing a stenosis (image courtesy of Department of Cardiology, FAU)

Digital Subtraction Angiography

- **Angiography:** imaging of blood vessels, e. g., to detect stenoses
- **Digital subtraction angiography (DSA):**
  - Injection of contrast agent into the blood flow (e. g., through femoral artery for coronary angiography)
  - Acquisition of X-ray images without (mask scan) and with contrast agent (fill scan) in the blood
  - Digital subtraction of background (mask) $\rightarrow$ improved visualization of contrasted arteries

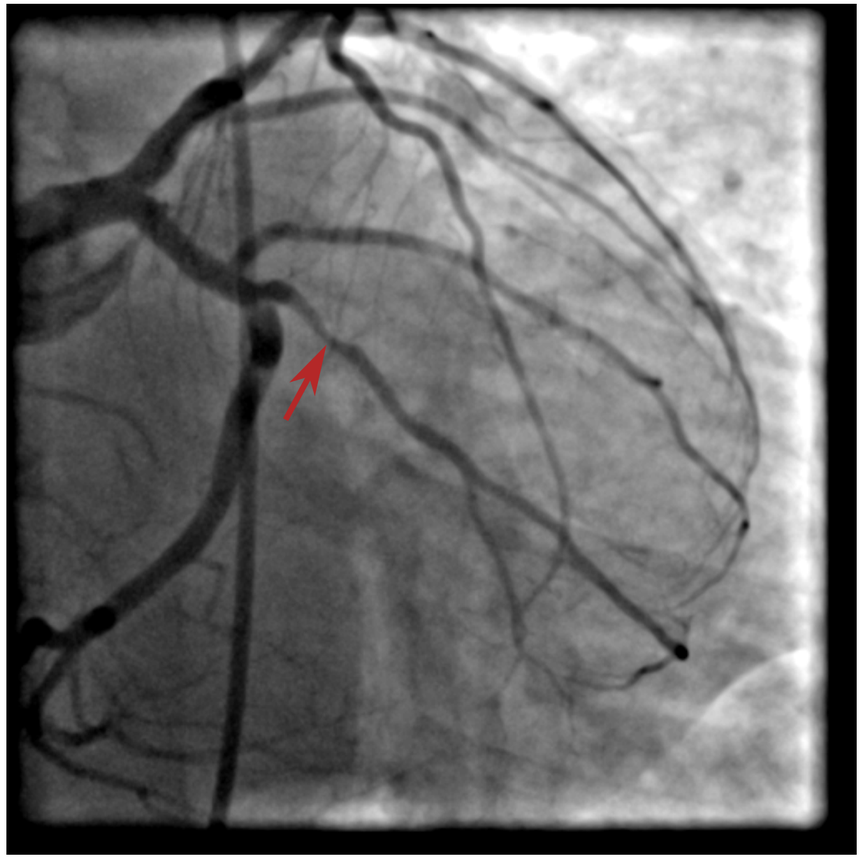


Figure 1: Scan showing a stenosis (image courtesy of Department of Cardiology, FAU)

Virtual DSA

Problem: Motion between the acquisition of mask and fill scan introduces misalignment artifacts that deteriorate diagnostic value.

Virtual DSA

Problem: Motion between the acquisition of mask and fill scan introduces misalignment artifacts that deteriorate diagnostic value.

Virtual DSA (vDSA):

- Directly estimate mask images from fill scans with prior segmentation of the arteries.
- Virtual single-frame background subtraction enables accurate motion compensation for 3-D reconstructions of the arteries (cf. Unberath et al., 2016).

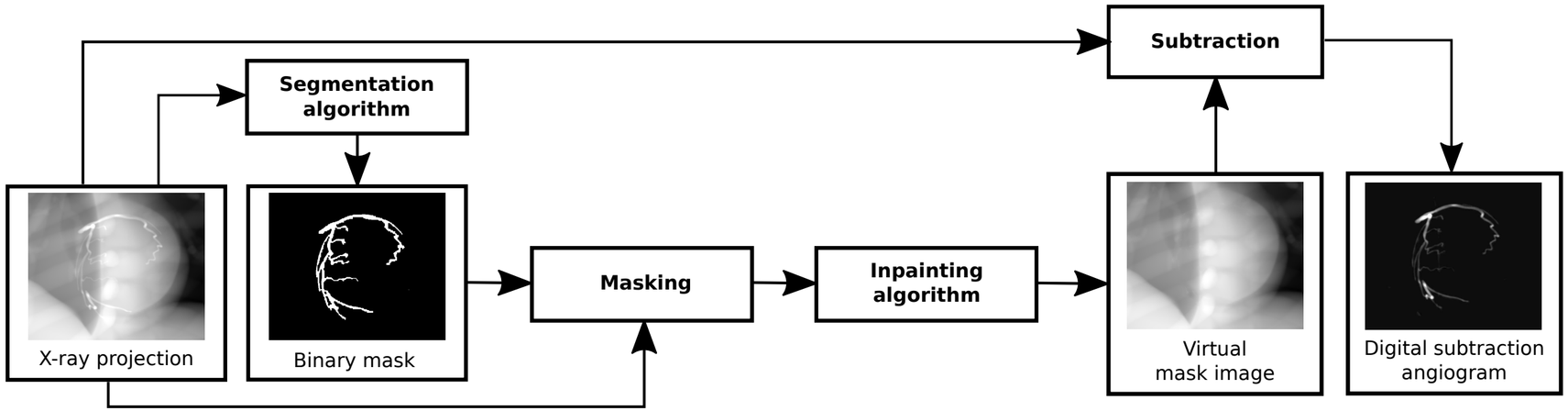


Figure 2: Scheme of the virtual DSA processing pipeline

Vessel Segmentation

- Vessel mask images are generated by vessel segmentation similar to what was introduced in an earlier unit.
- It is based on:
  - ***blobness*** $B = \frac{\nu_2}{\nu_1}$,
  - ***structureness*** $S = \sqrt{\nu_1^2 + \nu_2^2}$,
  - and scale-space image gradients,

  where $|\nu_1| \geq |\nu_2|$ denote the eigenvalues of the local Hessian matrices in scale-space.

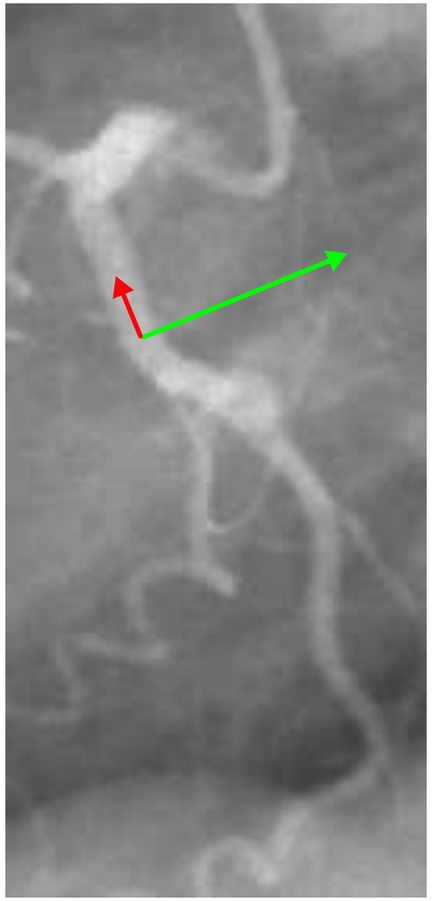


Figure 3: Local principal directions (symbolically scaled by curvature)

Topics

Digital Subtraction Angiography

Inpainting Using U-Net

Summary

- Take Home Messages
- Further Readings

Inpainting

- The segmentation can be used to remove the blood vessels from the fill scan. In order to estimate the mask image, the vessel area has to be filled properly.

Inpainting

- The segmentation can be used to remove the blood vessels from the fill scan. In order to estimate the mask image, the vessel area has to be filled properly.
- Inpainting can be achieved by interpolation techniques like defect pixel interpolation, which most often work well for small regions only.

Inpainting

- The segmentation can be used to remove the blood vessels from the fill scan. In order to estimate the mask image, the vessel area has to be filled properly.
- Inpainting can be achieved by interpolation techniques like defect pixel interpolation, which most often work well for small regions only.
- Several image inpainting methods based on denoising autoencoders and convolutional neural networks (CNNs) have been successfully applied on natural images (Xie, Xu, and Chen, 2012; Cai et al., 2017).

Inpainting

- The segmentation can be used to remove the blood vessels from the fill scan. In order to estimate the mask image, the vessel area has to be filled properly.
- Inpainting can be achieved by interpolation techniques like defect pixel interpolation, which most often work well for small regions only.
- Several image inpainting methods based on denoising autoencoders and convolutional neural networks (CNNs) have been successfully applied on natural images (Xie, Xu, and Chen, 2012; Cai et al., 2017).

Question: Can we use a deep learning network for inpainting in medical images?

U-Net

Contracting path:

- Downsampling layers decrease the feature sizes with increasing depth of the network.
- Each level consists of two convolution layers followed by a max pooling downsampling layer.
- Features lose spatial but gain contextual information.

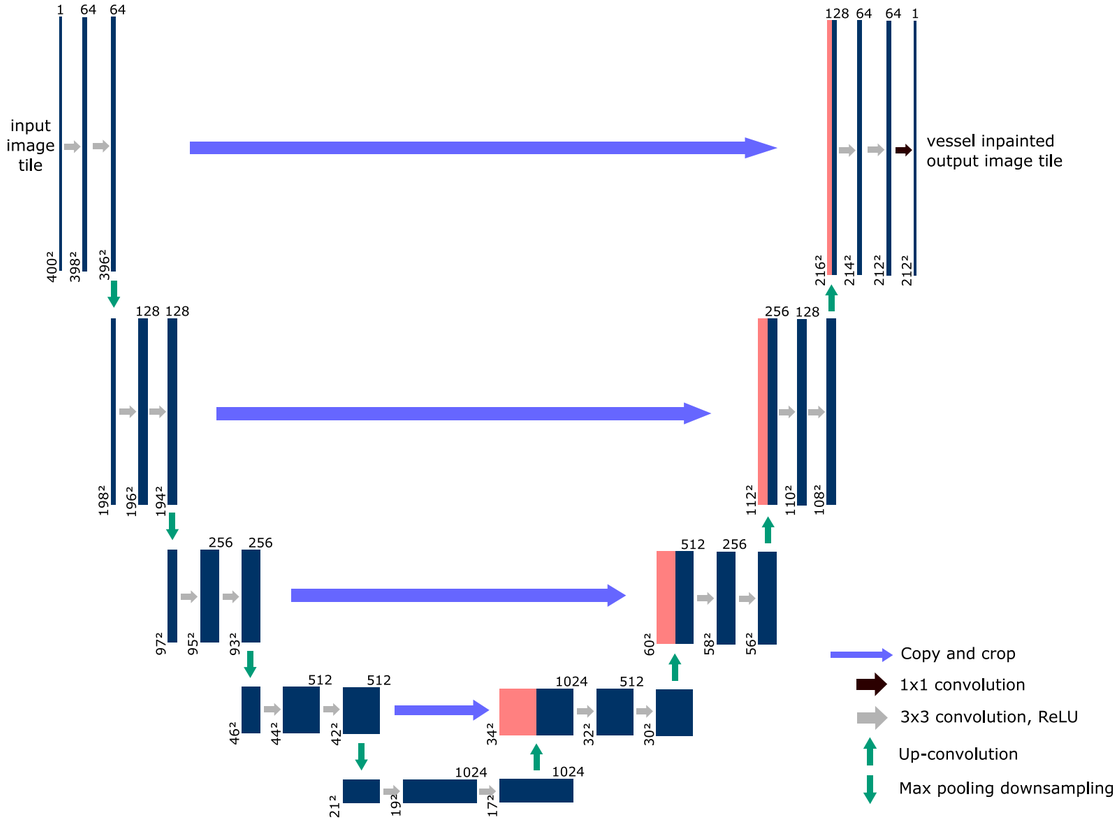


Figure 4: Example of a U-net architecture

U-Net

Contracting path:

- Downsampling layers decrease the feature sizes with increasing depth of the network.
- Each level consists of two convolution layers followed by a max pooling downsampling layer.
- Features lose spatial but gain contextual information.

Expanding path:

- Upsampling layers propagate contextual information to higher resolution levels.
- The output of these layers is combined with features from the corresponding downsampling layer.

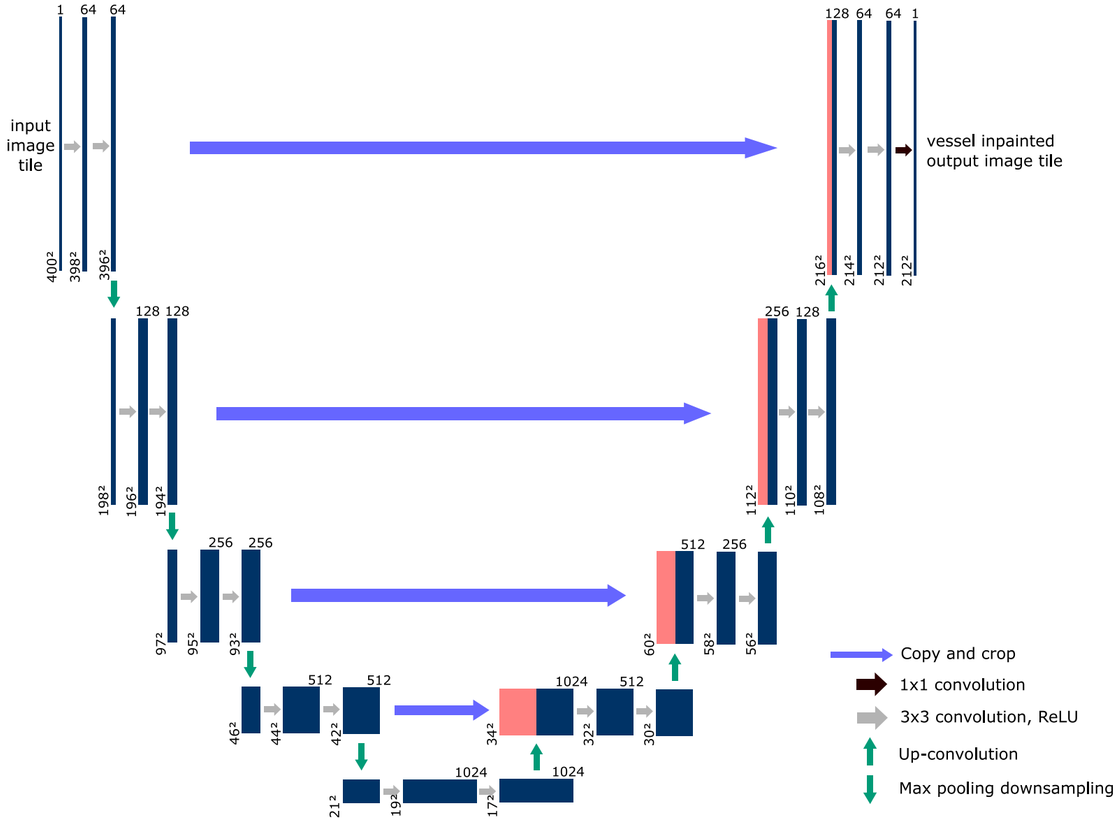


Figure 4: Example of a U-net architecture

Training

- Images and corresponding segmentation masks are split into training, test, and validation sets.
- All images are then partitioned into overlapping tiles of 400×400 pixels.

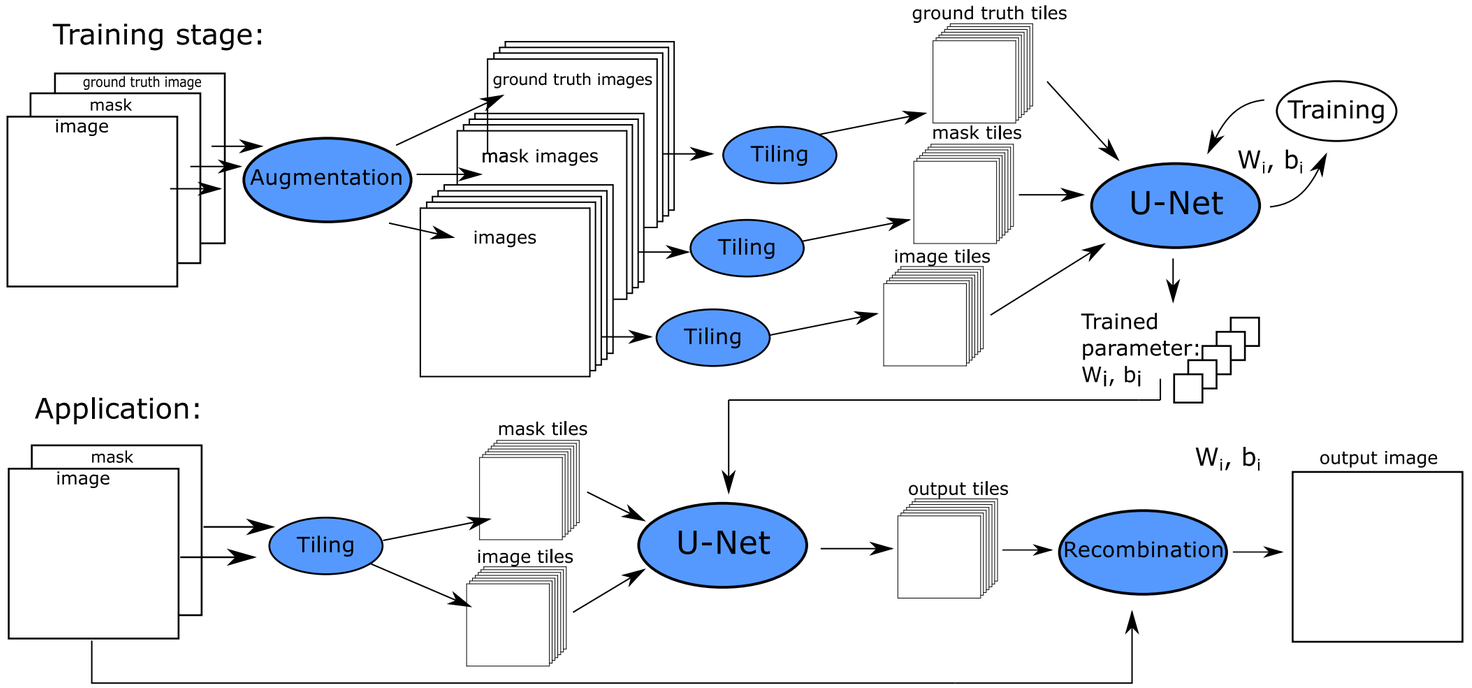


Figure 5: Scheme of training the network and applying it to new data

Inpainting Strategy

Two different U-nets are trained:

Inpainting Strategy

Two different U-nets are trained:

- **Mask-guided:** Use dilated versions M_i^d of the segmentation masks for images $i = 1, \ldots, N$ for very localized inpainting. The loss function used here is:

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{|M_i^d|} \sum_{\boldsymbol{p} \in \Omega} M_i^d(\boldsymbol{p}) \left\| Y_i(\boldsymbol{p}) - [X_i(\boldsymbol{\theta})](\boldsymbol{p}) \right\|_2 \right),$$

  where Ω is the image domain, Y_i denote no-contrast ground truth images, X the output of the net, and $\boldsymbol{\theta}$ the net parameters.

Inpainting Strategy

Two different U-nets are trained:

- **Mask-guided:** Use dilated versions M_i^d of the segmentation masks for images $i = 1, \ldots, N$ for very localized inpainting. The loss function used here is:

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{|M_i^d|} \sum_{\boldsymbol{p} \in \Omega} M_i^d(\boldsymbol{p}) \left\| Y_i(\boldsymbol{p}) - [X_i(\boldsymbol{\theta})](\boldsymbol{p}) \right\|_2 \right),$$

  where Ω is the image domain, Y_i denote no-contrast ground truth images, X the output of the net, and $\boldsymbol{\theta}$ the net parameters.
- **Semi-blind:** Refinement of the U-net above: Estimate intensities for **every** pixel instead of only the masked ones (i. e., remove mask weights in the loss stated above). Here, masks are only needed to generate defect images for training.
  This approach is similar to *blind* inpainting, but still needs the intermediate segmentation step for the masks.

Inpainting Strategy

Two different U-nets are trained:

- **Mask-guided:** Use dilated versions M_i^d of the segmentation masks for images $i = 1, \ldots, N$ for very localized inpainting. The loss function used here is:

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{|M_i^d|} \sum_{\boldsymbol{p} \in \Omega} M_i^d(\boldsymbol{p}) \left\| Y_i(\boldsymbol{p}) - [X_i(\boldsymbol{\theta})](\boldsymbol{p}) \right\|_2 \right),$$

  where Ω is the image domain, Y_i denote no-contrast ground truth images, X the output of the net, and $\boldsymbol{\theta}$ the net parameters.
- **Semi-blind:** Refinement of the U-net above: Estimate intensities for **every** pixel instead of only the masked ones (i. e., remove mask weights in the loss stated above). Here, masks are only needed to generate defect images for training.
  This approach is similar to *blind* inpainting, but still needs the intermediate segmentation step for the masks.

Both networks are optimized by stochastic gradient descent and back-propagation.

XCAT and Clinical Projections

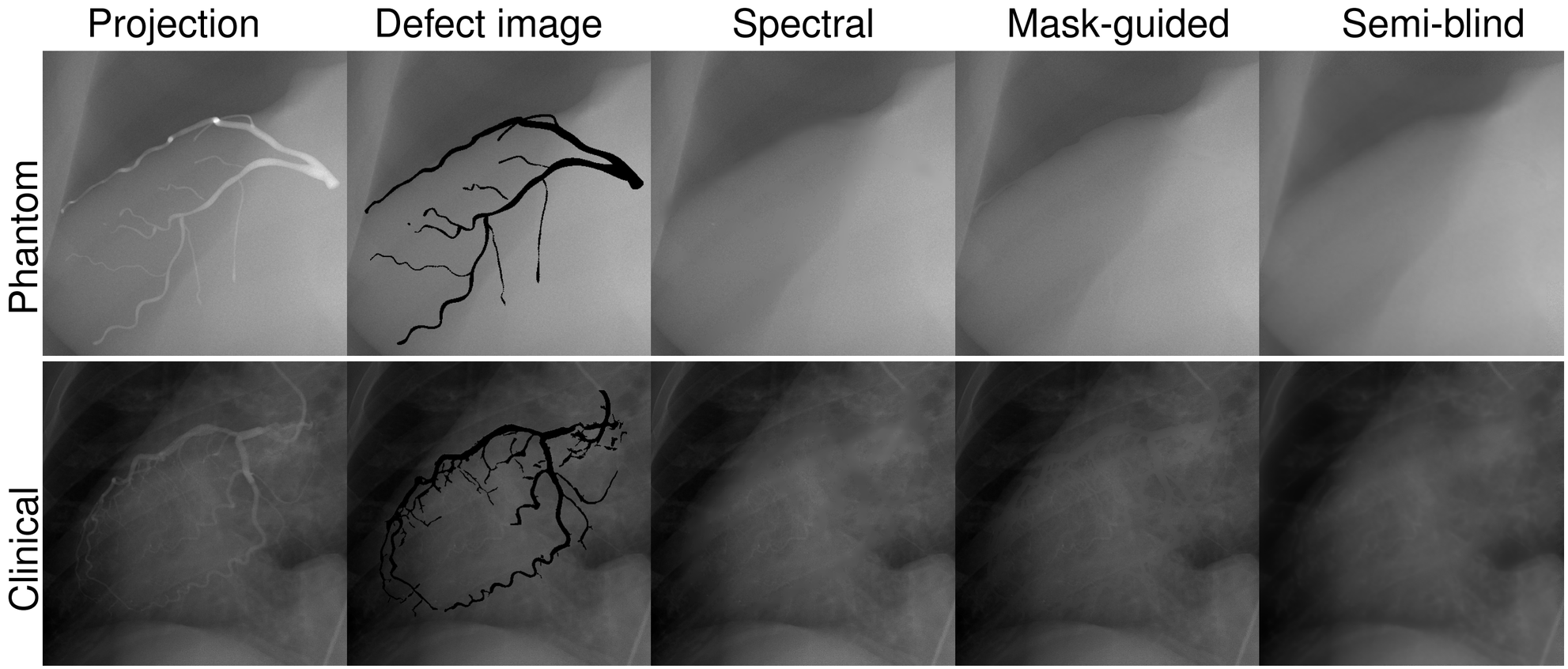


Figure 6: From left to right, the columns show the input image, the defect image, and inpainting results achieved with spectral deconvolution as in Unberath et al., 2016 versus the *mask-guided* and *semi-blind* U-net.

Evaluation

- The performance of the semi-blind U-net is:
  - superior to the mask-guided net and spectral deconvolution in the high noise case,
  - comparable to spectral deconvolution in the low noise realizations.
- Approach is ongoing research:
  - net was trained on phantom data **only**, yet
  - performance on clinical data is promising.

	Spectral	Mask-guided	Semi-blind
High noise	0.74	0.69	0.78
Low noise	0.97	0.95	0.96

Table 1: Average SSIM over all projections

Topics

Digital Subtraction Angiography

Inpainting Using U-Net

Summary

- Take Home Messages
- Further Readings

Take Home Messages

- Inpainting is an important tool for digital subtraction angiography to estimate the background.
- Segmentation of vessels allows virtual DSA.
- Deep learning can be used to perform the inpainting globally on the image. However, so far, it still needs segmentations for both training and application.

Further Readings

- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III*. ed. by Nassir Navab et al. Vol. 9351. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28
- Mathias Unberath et al. “Consistency-based Respiratory Motion Estimation in Rotational Angiography”. In: *Medical Physics* (2016). (in press). DOI: 10.1002/mp.12021
- Junyuan Xie, Linli Xu, and Enhong Chen. “Image Denoising and Inpainting with Deep Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. 2012, pp. 341–349
- Nian Cai et al. “Blind Inpainting Using the Fully Convolutional Neural Network”. In: *The Visual Computer* 33.2 (Feb. 2017), pp. 249–261. DOI: 10.1007/s00371-015-1190-z