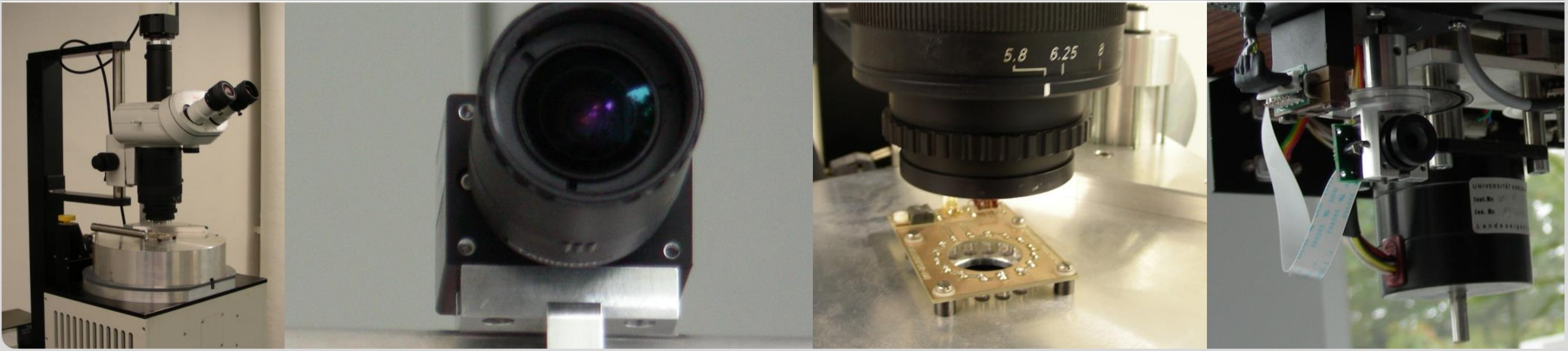
Assignment 07 – Deep Learning

Christian Kinzig

INSTITUTE OF MEASUREMENT AND CONTROL SYSTEMS, DEPARTMENT OF MECHANICAL ENGINEERING



Multi-Layer Perceptrons (MLP)

- MLPs are highly parameterized, non-linear functions

$$f_{MLP}^{\vec{w}} : \mathbb{R}^n \to \mathbb{R}^q$$

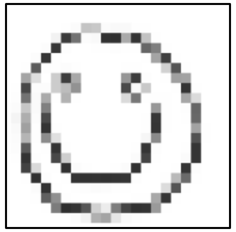
$$f_{MLP}^{\vec{w}} : \vec{x} \mapsto \vec{y}$$

parameter vector, weight vector

(input-) pattern

output

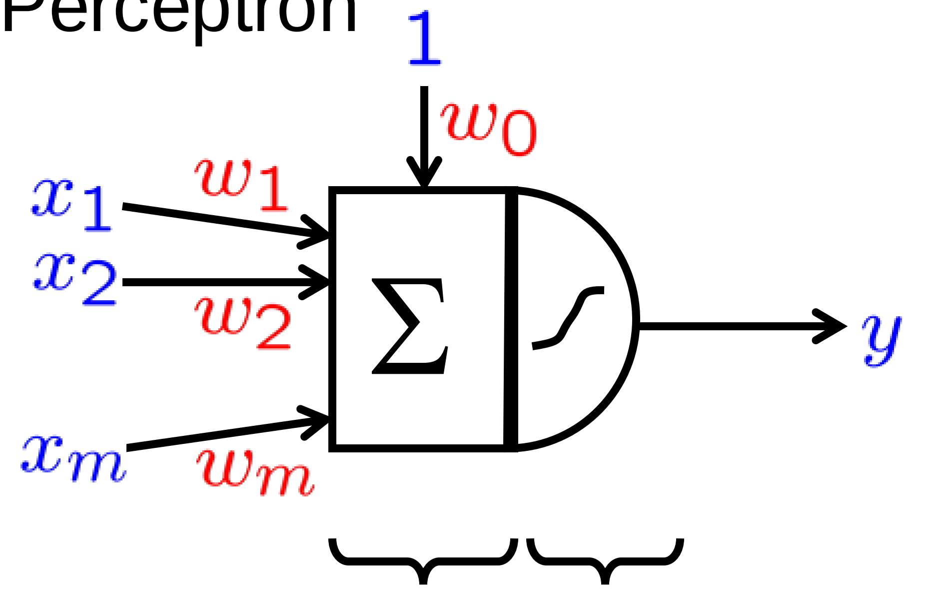
- Example: classification of images
  - $\vec{x}$: feature vector, e.g. vector of all gray values in image
  - $\vec{y}$: 1-of-q-vector that models probabilities for each of q possible categories, e.g. smiley is happy/sad/frustrated



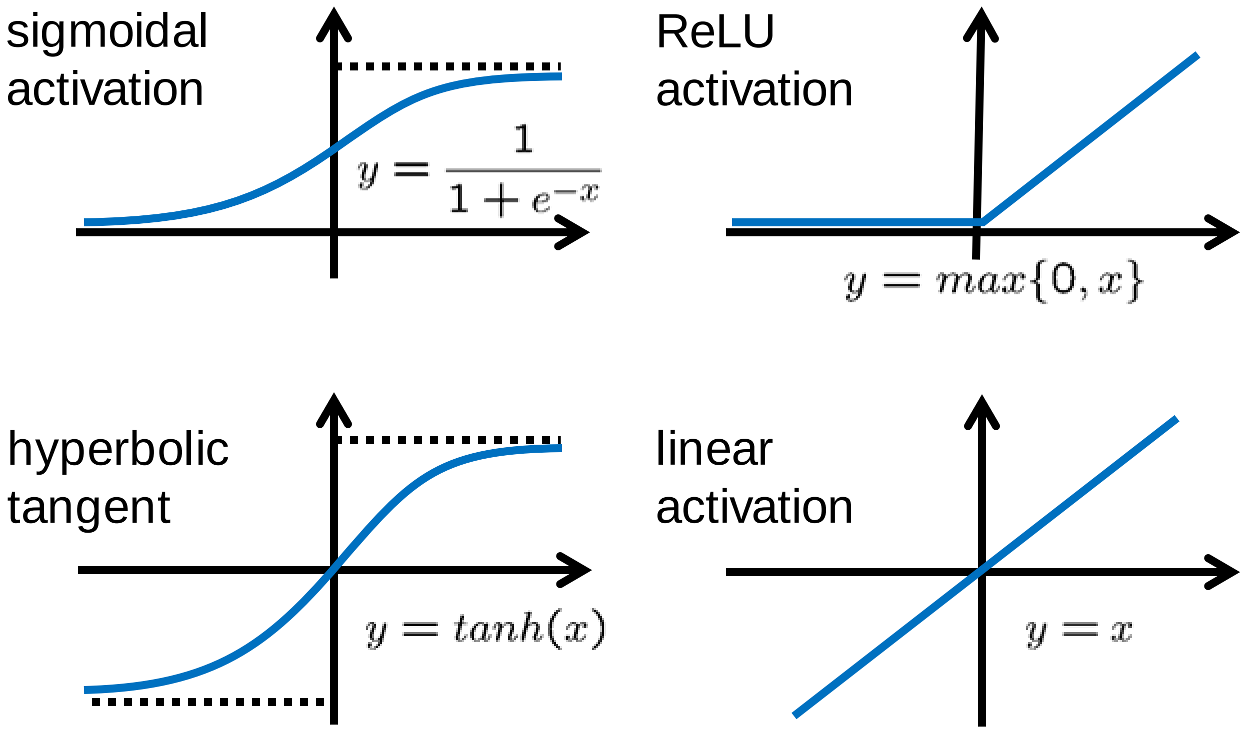
Institute of Measurement and Control Systems, Department of Mechanical Engineering

Internal Structure of MLPs

- Perceptron



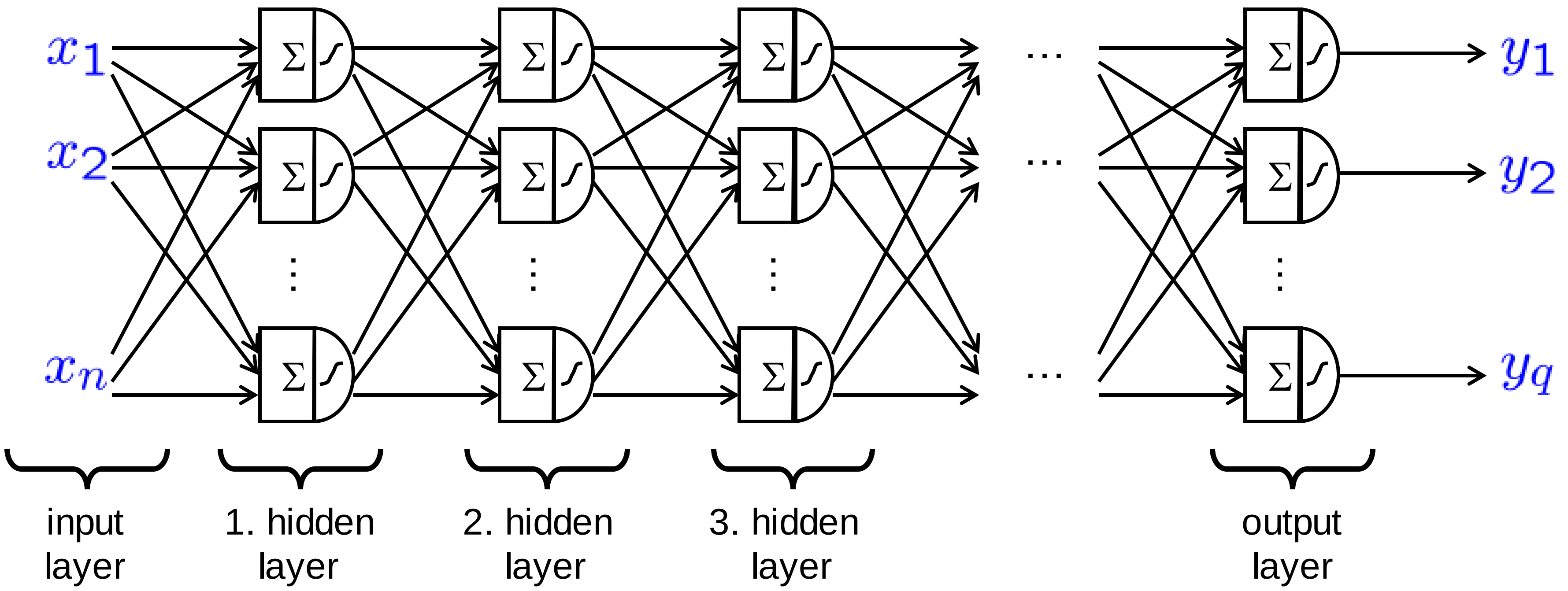
$$y = f_{act}\left(w_0 + \sum_{i=1}^{m} w_i x_i\right)$$



Institute of Measurement and Control Systems, Department of Mechanical Engineering

Internal Structure of MLPs

- Layered arrangement of many perceptrons



- Network structure creates set of highly nonlinear function
- Many weights
- Deep architectures: typically >5 hidden layers

Institute of Measurement and Control Systems, Department of Mechanical Engineering

Training of MLPs

- How do we determine weights of MLP?
  - Basic idea: minimize error for training examples

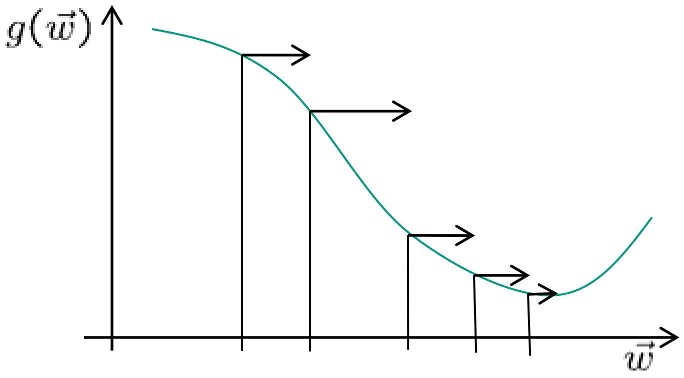
$$\left.\begin{array}{cc} \vec{x}^{(1)}, & \vec{t}^{(1)} \\ \vdots & \vdots \\ \underbrace{\vec{x}^{(p)}}_{\substack{\text{input} \\ \text{pattern}}}, & \underbrace{\vec{t}^{(p)}}_{\substack{\text{desired output} \\ \text{(ground truth)}}} \end{array}\right\} \text{training set}$$

  - Solve $\underset{\vec{w}}{minimize} \sum_{j=1}^{p} err\left(f_{MLP}^{\vec{w}}(\vec{x}^{(j)}), \vec{t}^{(j)}\right)$ for appropriate error measure err
  - Algorithm: gradient descent (backpropagation)

Gradient Descent (Backpropagation)

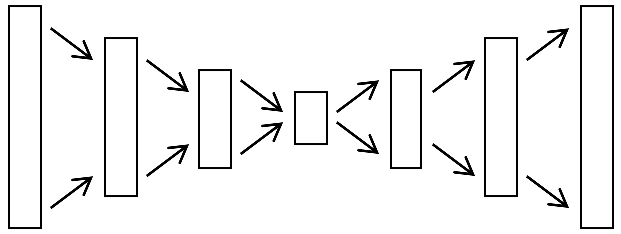
- Goal: $\underset{\vec{w}}{minimize}\; g(\vec{w})$ with $g(\vec{w}) := \sum_{j=1}^{p} err\left(f_{MLP}^{\vec{w}}(\vec{x}^{(j)}), \vec{t}^{(j)}\right)$

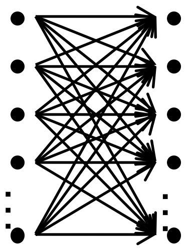
- Algorithm:
  1. Initialize weights $\vec{w}$ randomly with small numbers
  2. Calculate gradient $\frac{\partial g(\vec{w})}{\partial \vec{w}}$
  3. Update weights $\vec{w} \leftarrow \vec{w} - \varepsilon \frac{\partial g(\vec{w})}{\partial \vec{w}}$ with small learning rate $\varepsilon > 0$
  4. GoTo 2. until stopping criterion reached

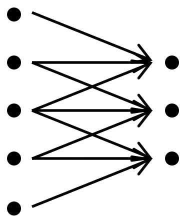


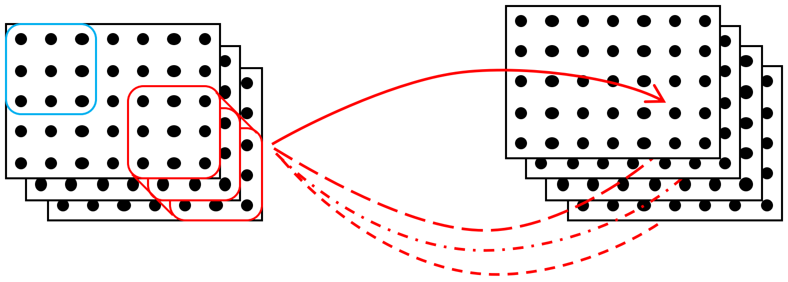
Deep Learning

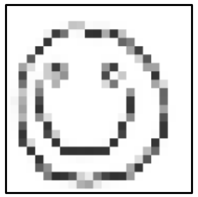
- Larger training sets (millions instead of hundreds)
- More powerful computers, parallel implementations on multi-core CPUs and GPUs
- Special network structures
  - Autoencoders
  - Convolutional networks
  - …
- Weight sharing
- Layer-wise learning
- Learning from unlabeled examples
- Learning of useful features







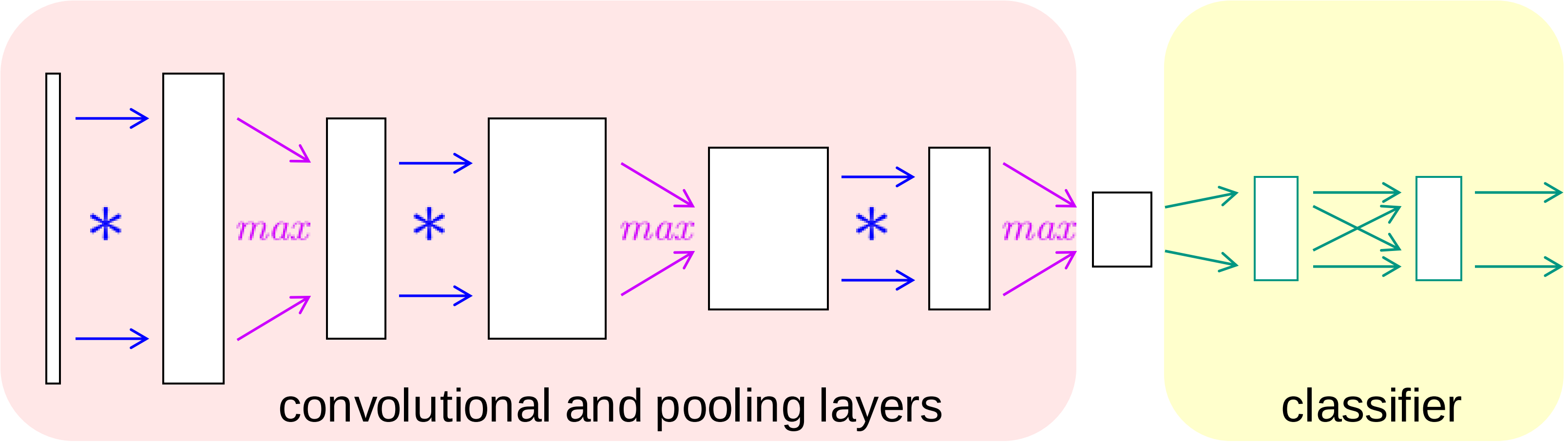




Institute of Measurement and Control Systems, Department of Mechanical Engineering

Convolutional Networks

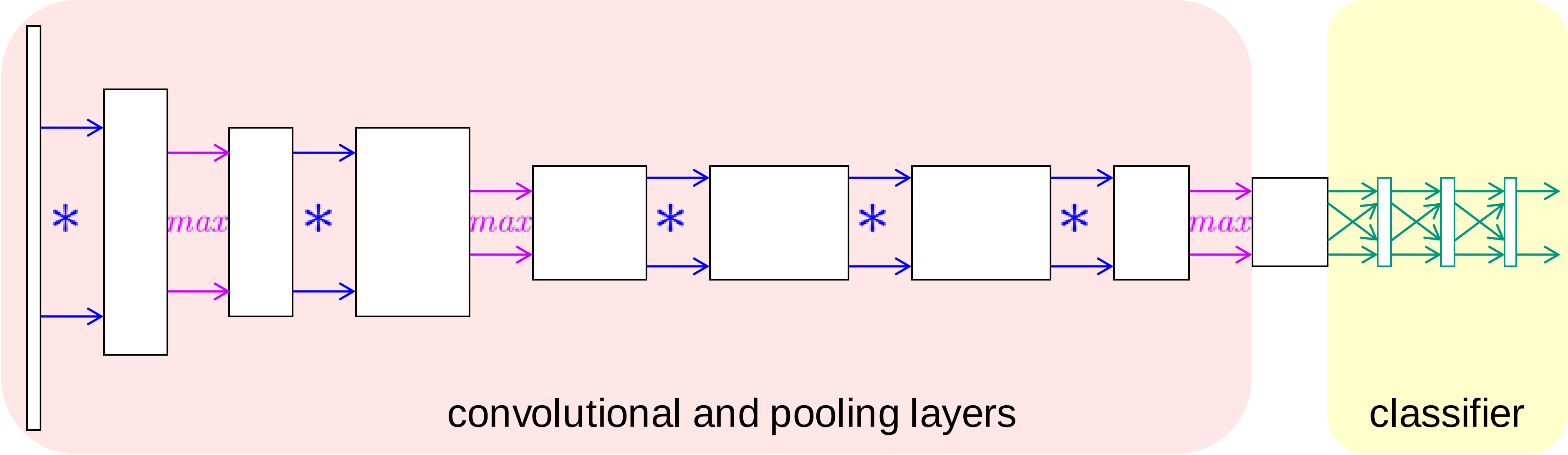
- Convolutional Networks (CNNs) combine
  - Convolutional layers
  - Pooling layers
  - Fully connected classifier network



Institute of Measurement and Control Systems, Department of Mechanical Engineering

Example: AlexNet

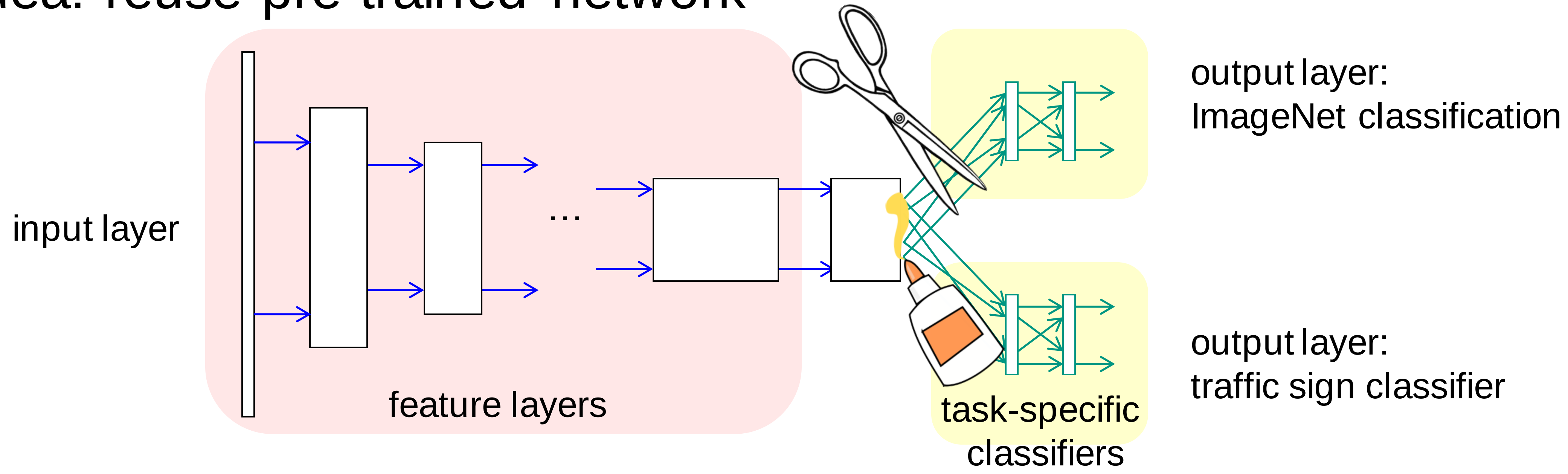
- A. Krizhevsky, I. Sutskever, G. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks“, NIPS, 2012
  - Classification of images, 1000 categories
  - Data set: 1,2 millions of images
  - Apporach: convolutional network, 60 millions of weights



Usage of Pre-Trained Feature Networks

Karlsruhe Institute of Technology

- Idea: reuse pre-trained network

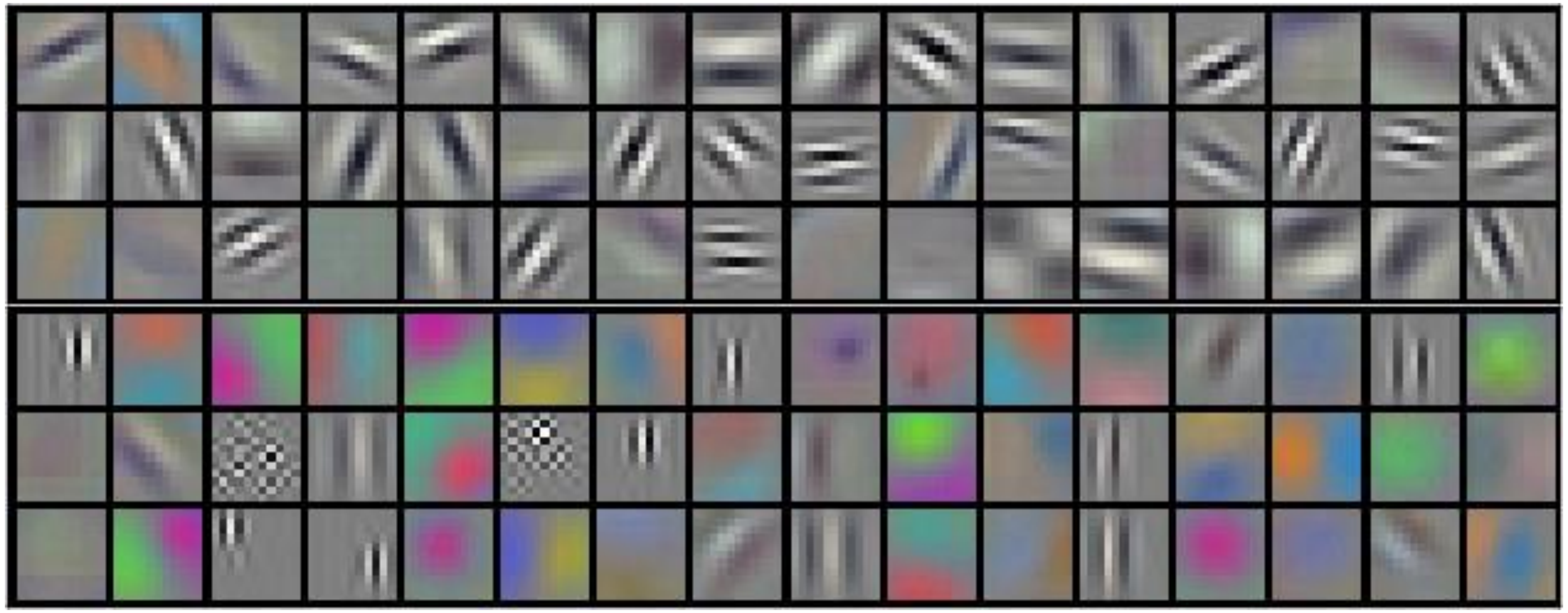


1. Train other task with large training set
2. Throw away classification layers of other task
3. Create new classification layers for new task
4. Train weights of new classification layer while preserving feature layers

Institute of Measurement and Control Systems,
Department of Mechanical Engineering

Convolutional Networks

- Which features are learned in hidden layers?
  - 1. layer: gray level edges, color edges, blobs

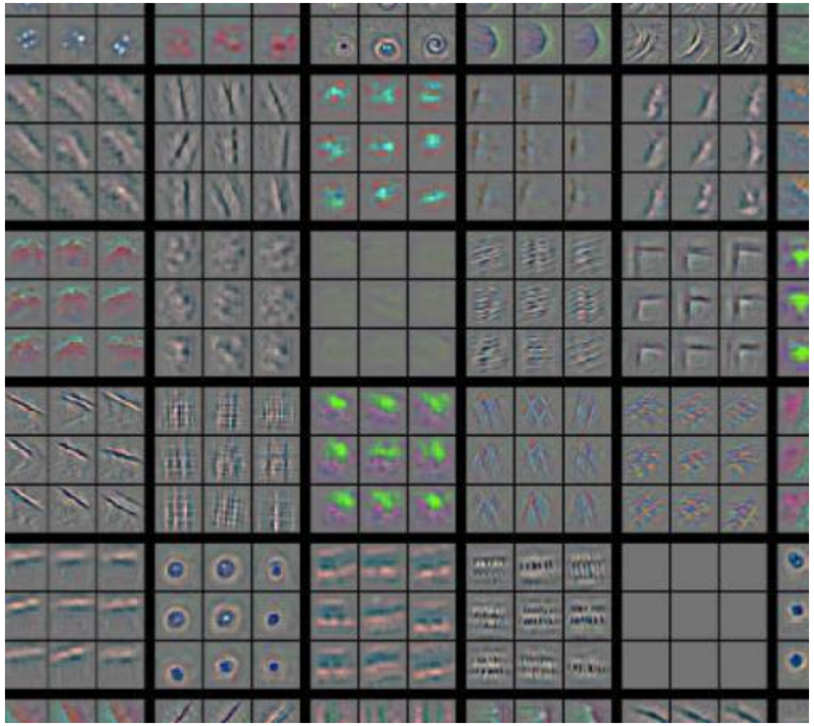


Taken from:
http://image-net.org/challenges/LSVRC/2012/supervision.pdf

Institute of Measurement and Control Systems,
Department of Mechanical Engineering

Convolutional Networks

- Which features are learned in hidden layers?
  - 2. layer: corners, round structures

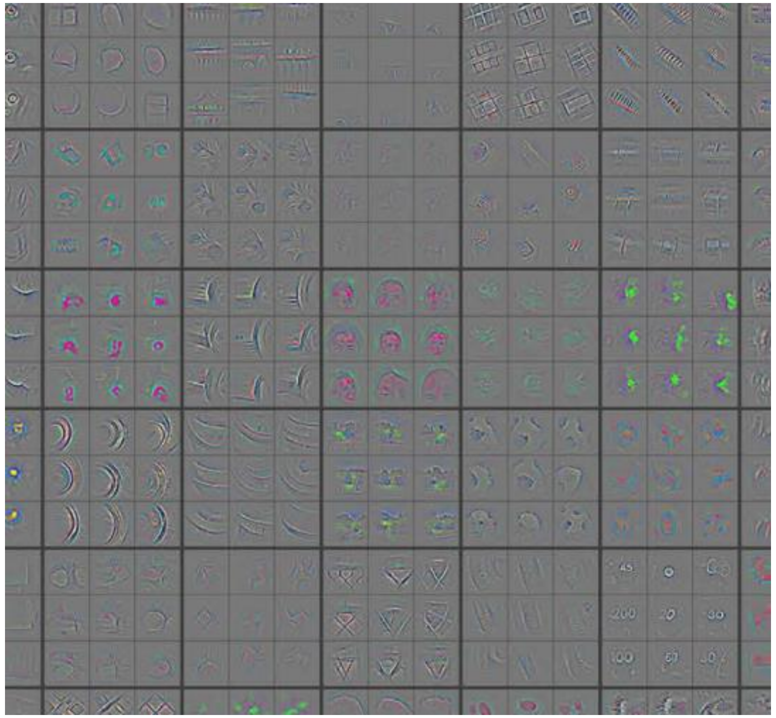


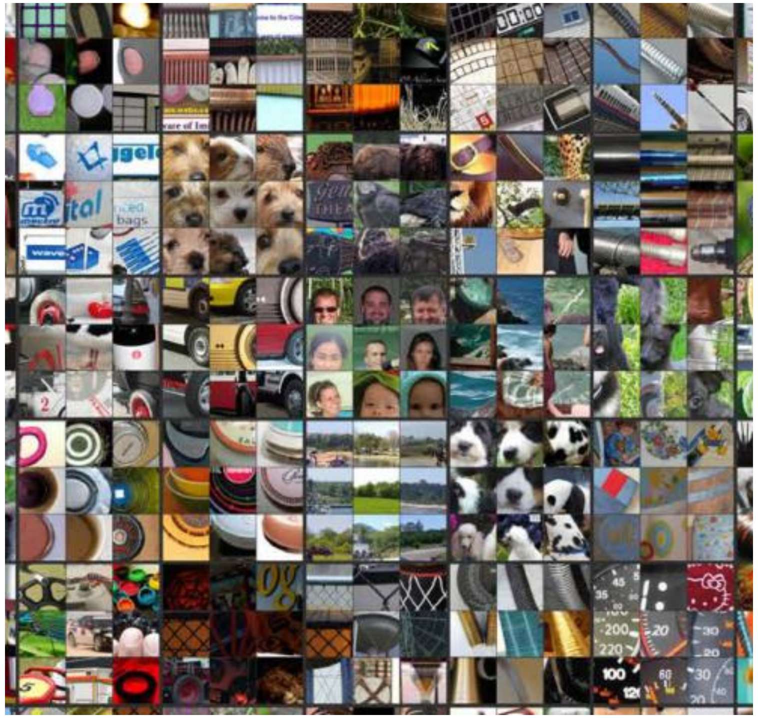
Taken from:
M.D. Zeller, R. Fergus, „Visualizing and Understanding Convolutional Neural Networks“, arXiv:1311.2901 v2, 13. Nov. 2013

Institute of Measurement and Control Systems, Department of Mechanical Engineering

Convolutional Networks

- Which features are learned in hidden layers?
  - 3. layer: shapes, gratings



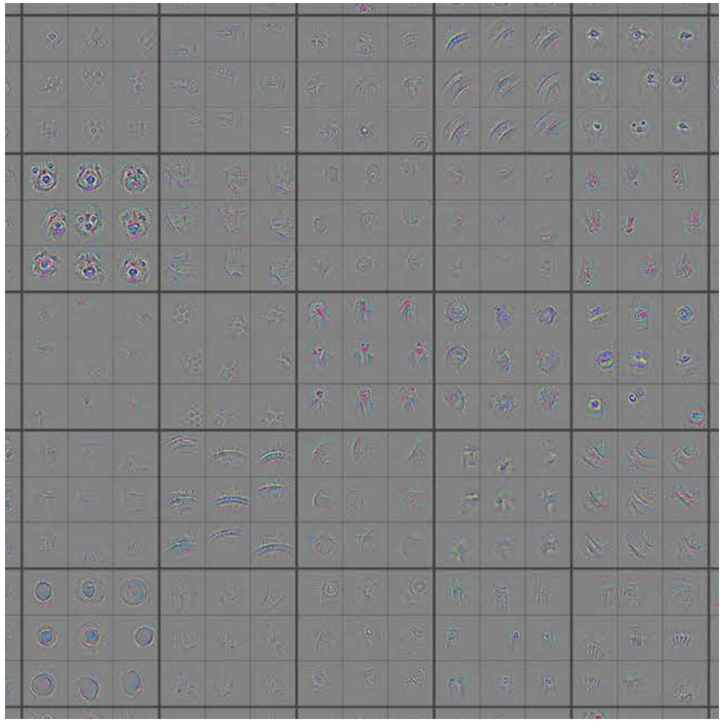


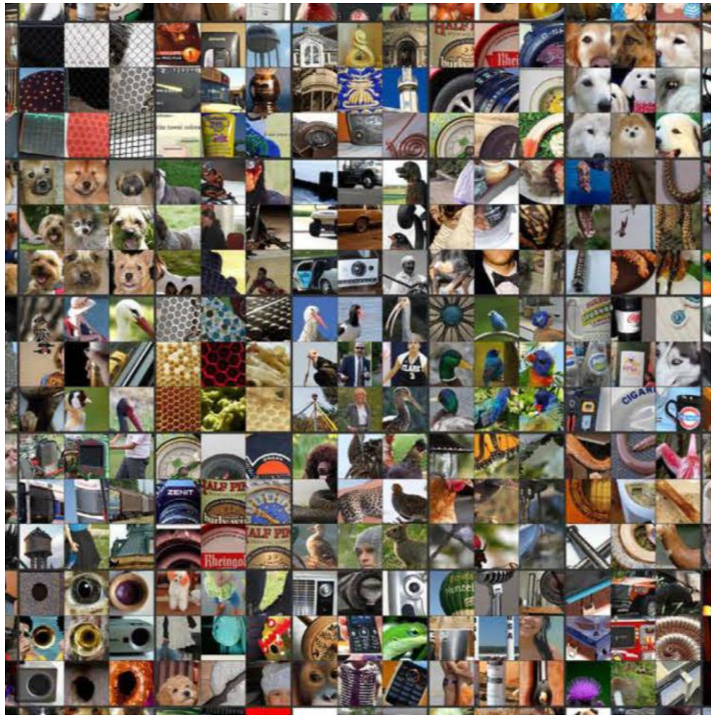
Taken from:
M.D. Zeller, R. Fergus, „Visualizing and Understanding Convolutional Neural Networks“, arXiv:1311.2901 v2, 13. Nov. 2013

Institute of Measurement and Control Systems, Department of Mechanical Engineering

Convolutional Networks

- Which features are learned in hidden layers?
  - 4. layer: textured geometries

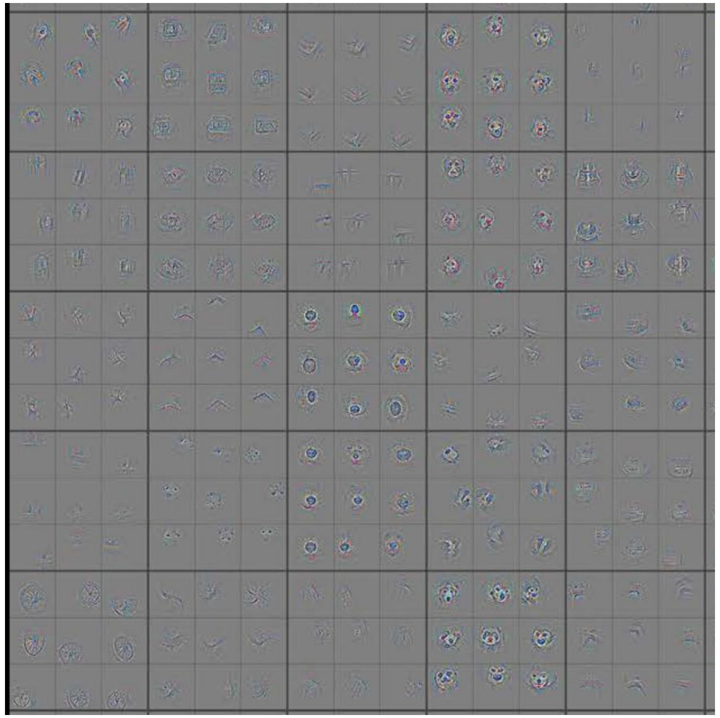




Taken from:
M.D. Zeller, R. Fergus, „Visualizing and Understanding Convolutional Neural Networks“, arXiv:1311.2901 v2, 13. Nov. 2013

Convolutional Networks

- Which features are learned in hidden layers?
  - 5. layer: objects



Taken from:
M.D. Zeller, R. Fergus, „Visualizing and Understanding Convolutional Neural Networks“, arXiv:1311.2901 v2, 13. Nov. 2013

Institute of Measurement and Control Systems, Department of Mechanical Engineering

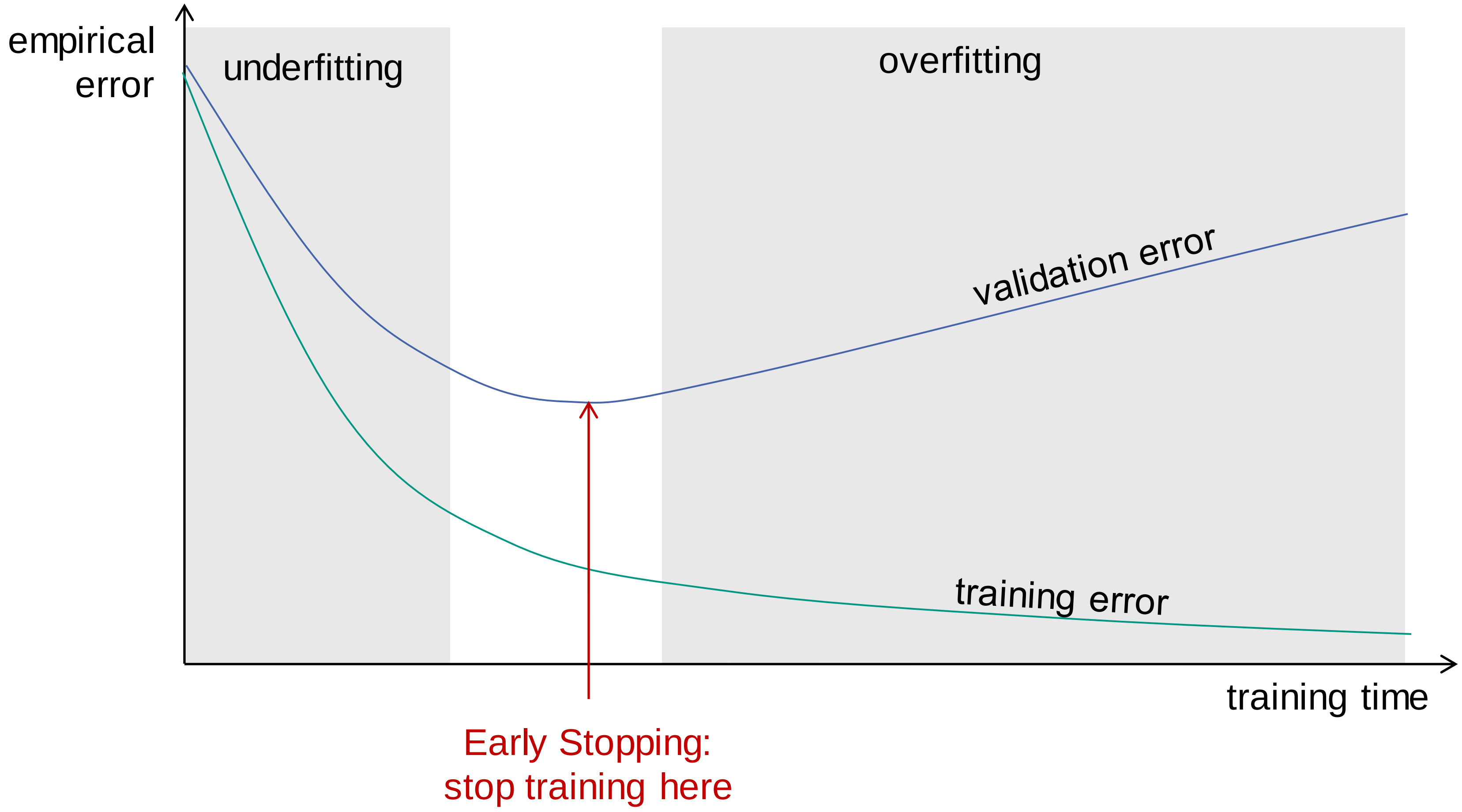
Convolutional Networks

- From layer to layer…
  - Features become more and more geometrically complex
  - Features become more and more independent of position
  - Features become more and more inpendent of pattern size
  - Features become more and more specific

Principles for Training MLPs

- There‘s no data like more data!
  - Remember data tuning
- Rigorous validation of training process
- Regularisation of training process
  - Early stopping, Weight decay/L2 regularisation, Dropout, Stochastic gradient descent, Multi task learning, Use pretrained networks
- Reuse of practical knowledge (of others)
  - Successful network structures
  - Successful training processes

Typical Progression of Error during Training



Institute of Measurement and Control Systems,
Department of Mechanical Engineering

Modifications of Gradient Descent

- Stochastic gradient descent

$$\vec{w} \leftarrow \vec{w} - \varepsilon \cdot \frac{\partial}{\partial \vec{w}} \sum_{j=1}^{p} err\left(f_{MLP}^{\vec{w}}(\vec{x}^{(j)}), \vec{t}^{(j)}\right)$$

Calculate gradient from all training examples.

$$\vec{w} \leftarrow \vec{w} - \varepsilon \cdot \frac{\partial}{\partial \vec{w}} \sum_{j \in S} err\left(f_{MLP}^{\vec{w}}(\vec{x}^{(j)}), \vec{t}^{(j)}\right)$$

with $S \subseteq \{1, \ldots, p\}$

Calculate gradient from subset of all training examples. Subsets typically cycle through all examples.

Advantages:

- Speed up
- A little bit less overfitting

Institute of Measurement and Control Systems, Department of Mechanical Engineering