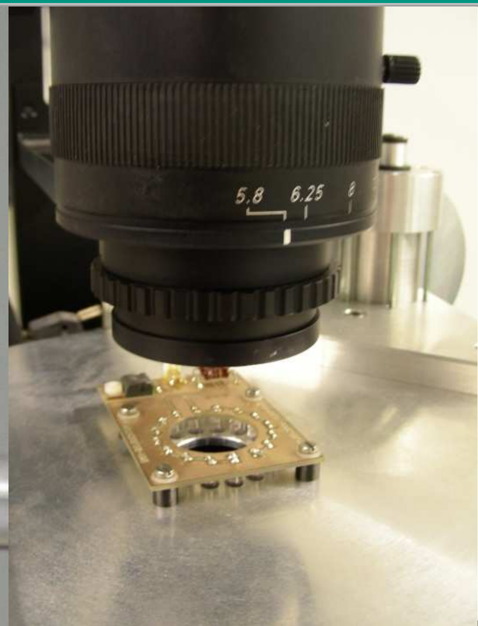
Machine Vision

Chapter 12: Deep Learning

Dr. Martin Lauer

Institut für Mess- und Regelungstechnik





MULTI-LAYER PERCEPTRONS

(AS ONE KIND OF ARTIFICIAL NEURAL NETWORKS)

Multi-Layer Perceptrons (MLP)

- MLPs are highly parameterized, non-linear functions

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

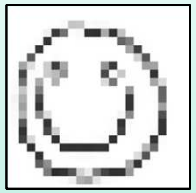
$$f^{\vec{w}}_{MLP} : \vec{x} \mapsto \vec{y}$$

parameter vector,
weight vector

(input-)
pattern

output

example: classification of images

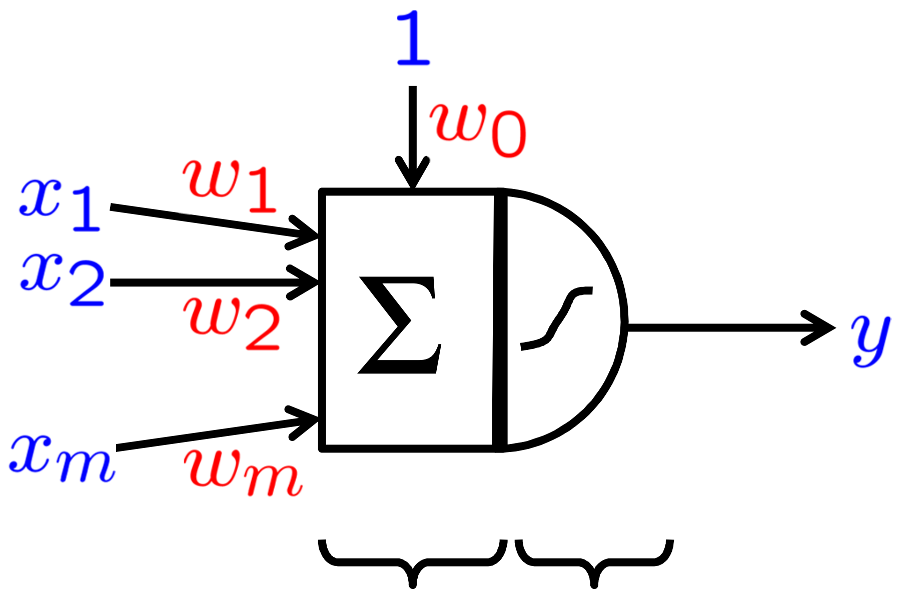


$\vec{x}$ feature vector,
e.g. vector of all gray values in image

$\vec{y}$ 1-of-q-vector that models probabilities for each of q possible categories, e.g. smiley is happy/sad/frustrated

Internal Structure of MLPs

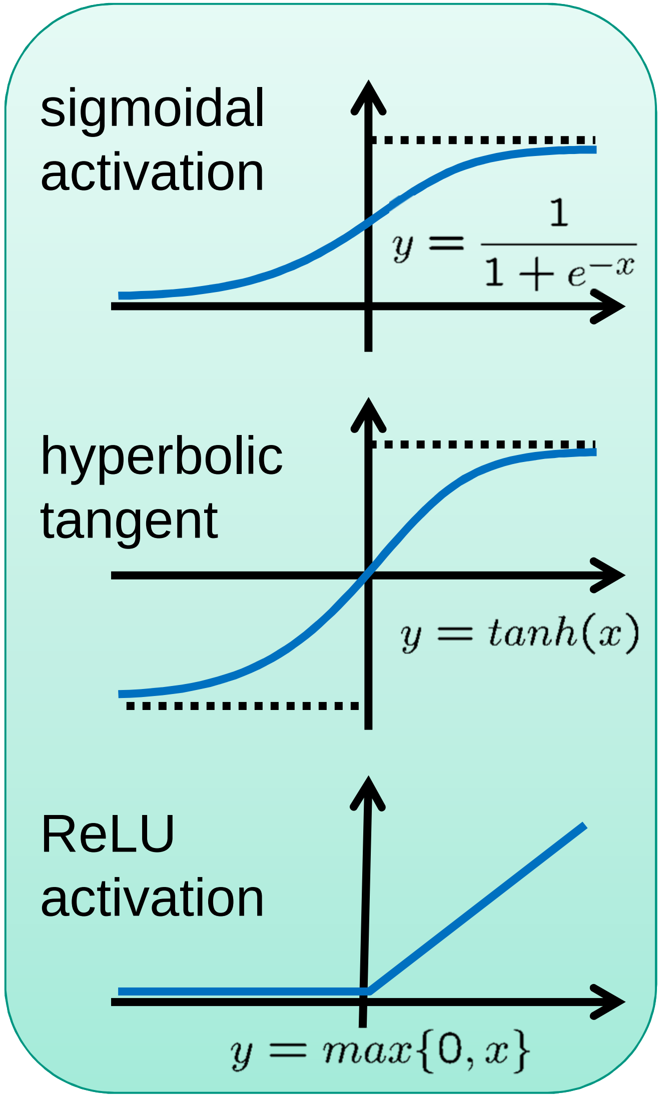
- perceptron



linear combination of inputs and weights

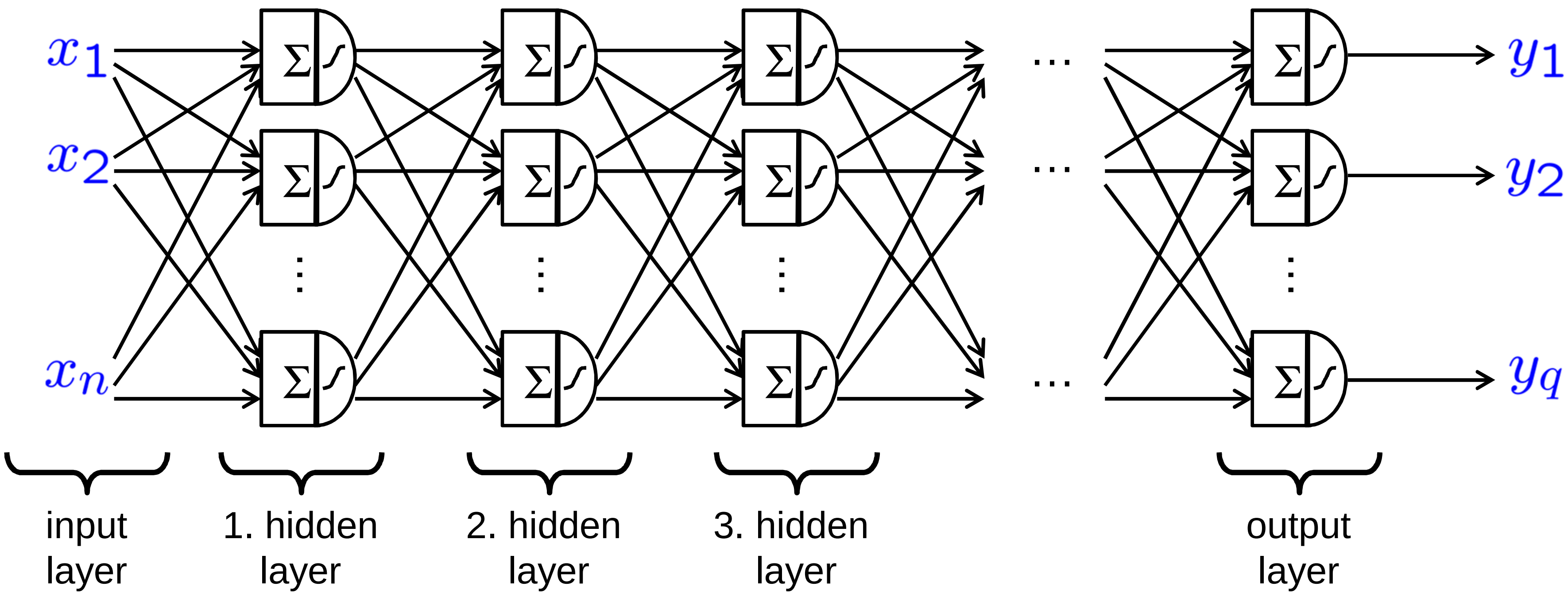
non linear activation function

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



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

Training of MLPs

- how do we determine weights of MLP?
  - basic idea: minimize error for training examples

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

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

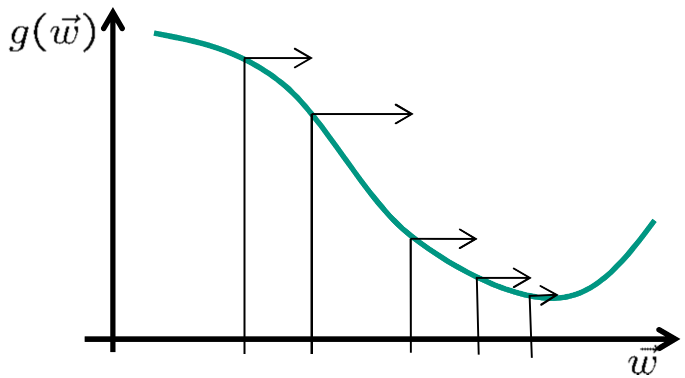
    for appropriate error measure err
  - algorithm: gradient descent (backpropagation)

Gradient Descent (Backpropagation)

- goal:

$$\underset{\vec{w}}{minimize}\; g(\vec{w}) \quad \text{with } g(\vec{w}) := \sum_{j=1}^{p} err\Big(f_{MLP}^{\vec{w}}(\vec{x}^{(j)}), \vec{t}^{(j)}\Big)$$

- 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



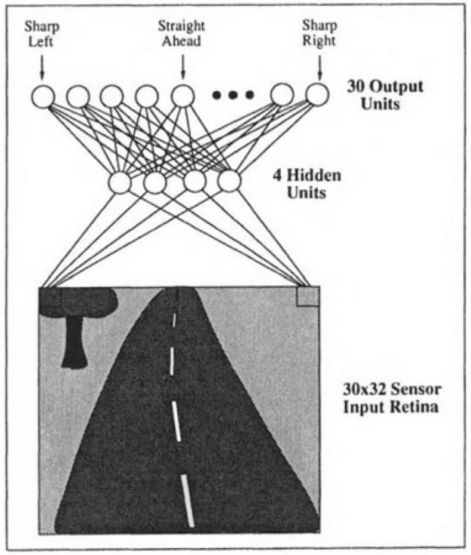
- improvements:
  – discussed later

Training MLPs (traditional methods)

- problems with traditional training methods
  - too many weight, too few training examples
  - too slow
  - numerical problems, local minima

  ➡ overfitting, underfitting, insufficient generalization

- traditional techniques to overcome problems
  - regularization (e.g. early stopping, weight decay, Bayesian learning)
  - preprocessing of patterns, feature extraction, reduction of dimensionality
  - choose smaller MLPs, less layers, less hidden neurons, network pruning
  - replace neural networks by other methods (e.g. SVMs, boosting, etc.)



ALVINN-architecture
taken from: D. A. Pomerleau, „Neural network perception for mobile robot guidance", 1993

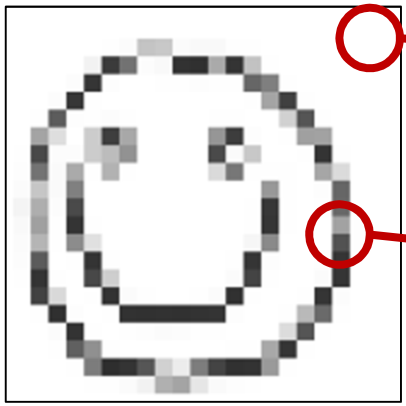
Deep Learning

- what is different in 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
    - recurrent networks/LSTM
    - (deep belief networks/restricted Boltzmann machines)
    - …
  - weight sharing
  - layer-wise learning
  - dropout
  - learning of useful features
  - learning from unlabeled examples

Learning of Features

- observation:
  - many pixels do not provide much information
  - neighboring pixels are highly correlated

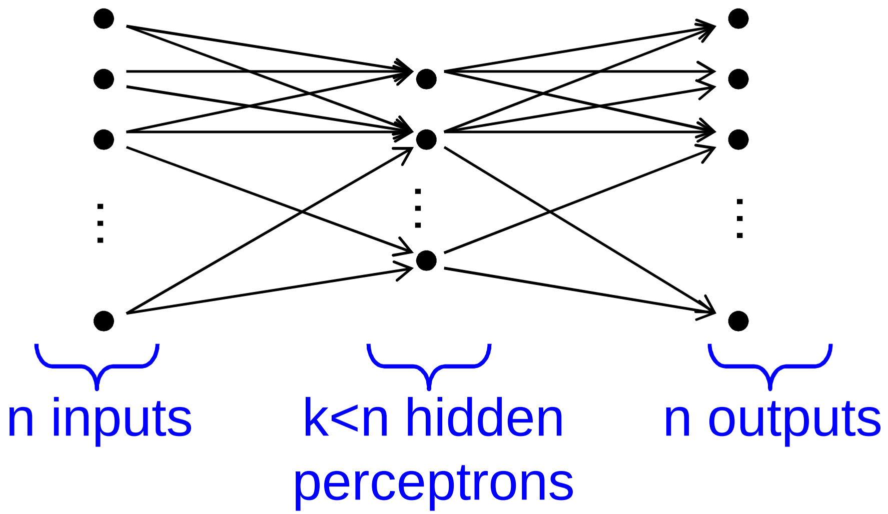
- example: smileys



- how can we seperate relevant information form irrelevant information?

Autoencoder

- MLP with such a structure



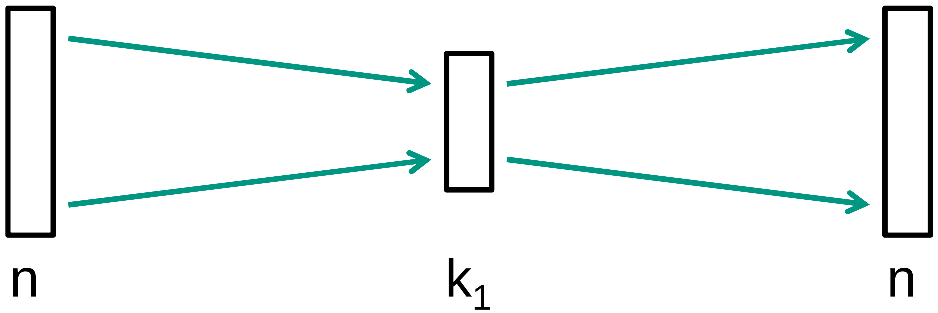
- learn identity function $$\underset{\vec{w}}{minimize} \sum_{j=1}^{p} \left(f_{MLP}^{\vec{w}}(\vec{x}^{(j)}) - \vec{x}^{(j)} \right)^2$$

→ hidden layer must compress image content

kind of neural principal component analysis

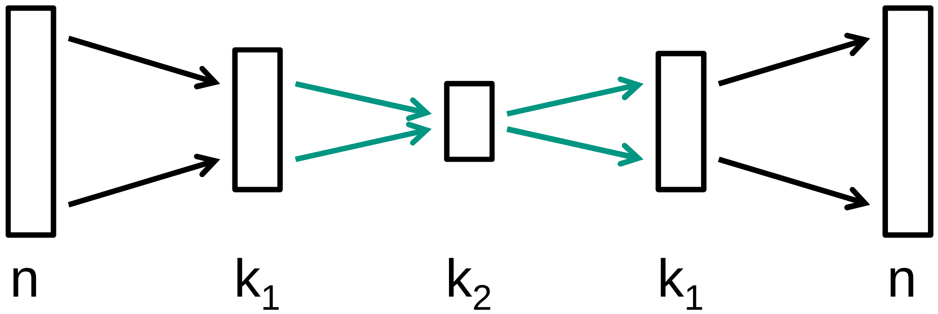
Stacked Autoencoders

- incremental training of multi-layered autoencoders
  1. train autoencoder with single hidden layer



train weights of
these connections

  2. extend autoencoder by additional hidden layer



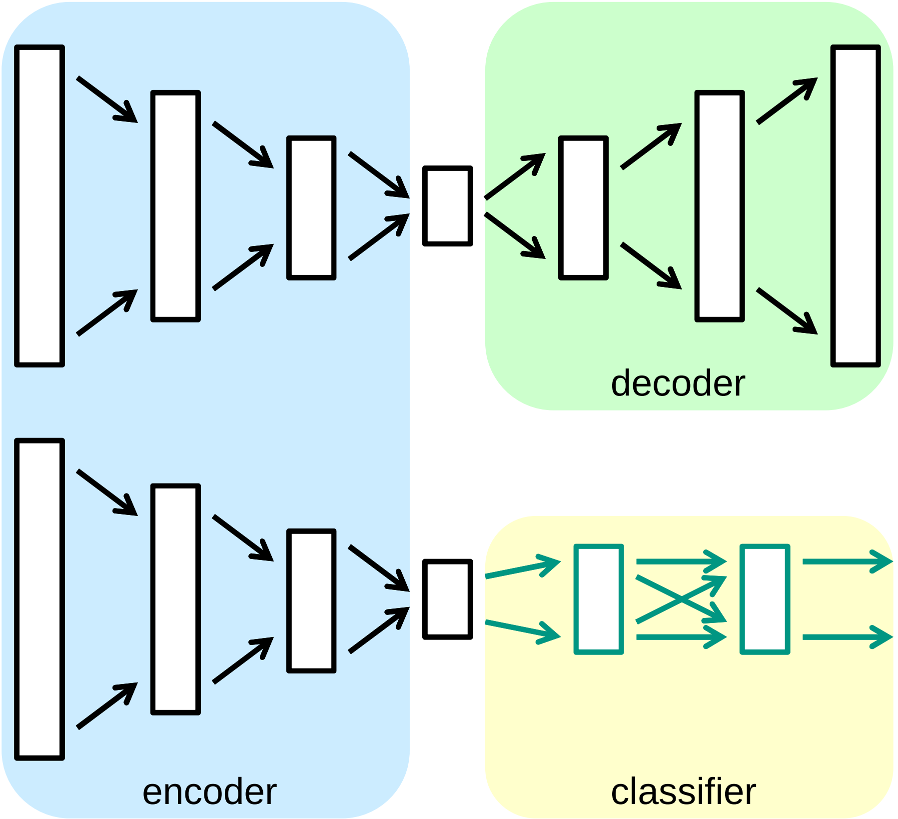
keep weights of
these connections
unchanged from
previous step

  3. repeat process analogously to add further hidden layers

→ compression of information increases from layer to layer
non-linear, multi-layered principal component analysis

Stacked Autoencoders for Classification

1. train stacked autoencoder
2. replace decoder network by fully-connected classifier network
3. train classifier network
4. train all weights of encoder and classifier network for a few iterations

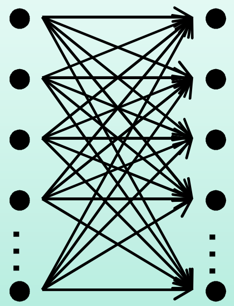


<u>advantages</u>:

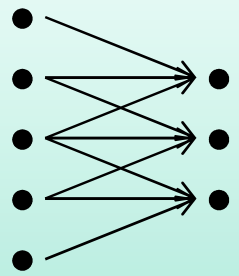
- stacked autoencoder can be trained with unlabeled examples
- incremental training achieves better results

Local Receptive Fields

- local receiptive fields for structured data



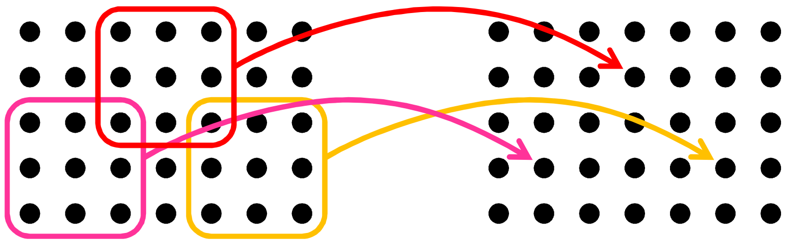
fully connected layer



local receptive fields

Local receptive fields force the network to process information locally.

example: local features for images

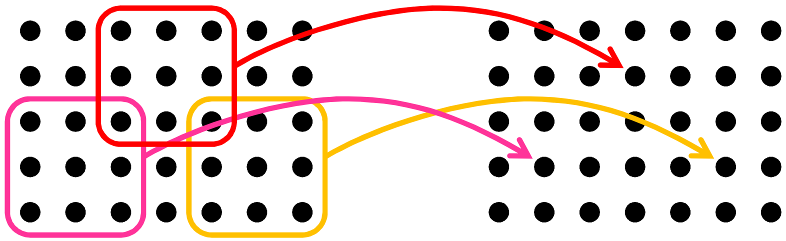


input layer:
gray values for
each pixel

feature layer:
e.g. edge feature
for each pixel

Weight Sharing

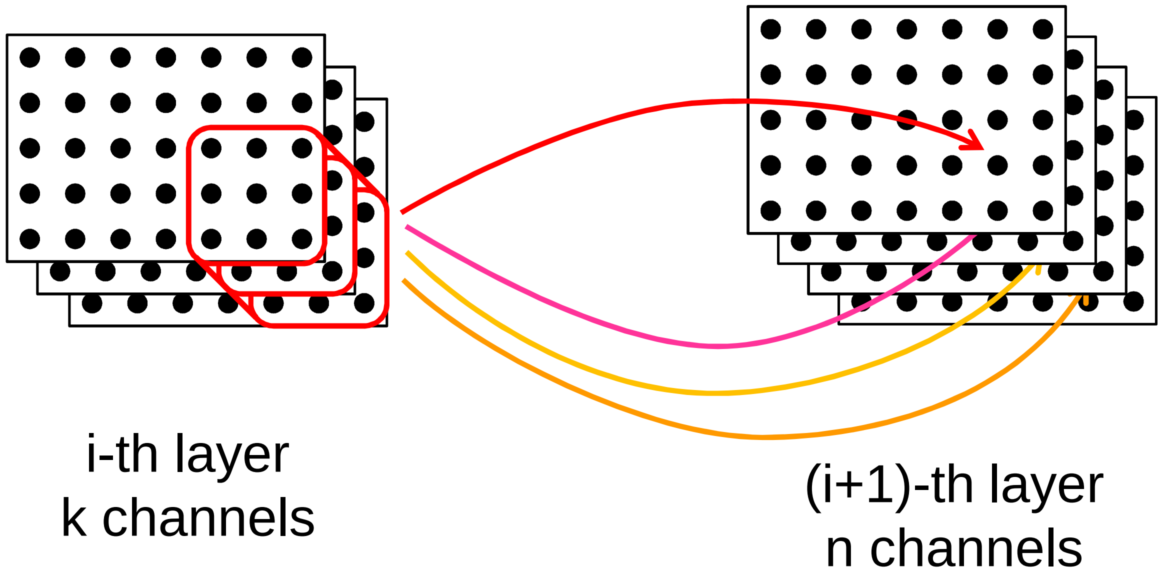
- can we generate the same local features for all pixels?
  - <u>weight sharing</u>: binding the weights of different perceptrons
  - <u>convolutional layers</u>: binding the weights of all perceptrons of one layer



$$L_{i+1} = f_{act}\left(L_i * \begin{array}{|c|c|c|}\hline w_1 & w_2 & w_3 \\ \hline w_4 & w_5 & w_6 \\ \hline w_7 & w_8 & w_9 \\ \hline\end{array} + \boxed{w_0}\right)$$

Multi-Channel Feature Layers

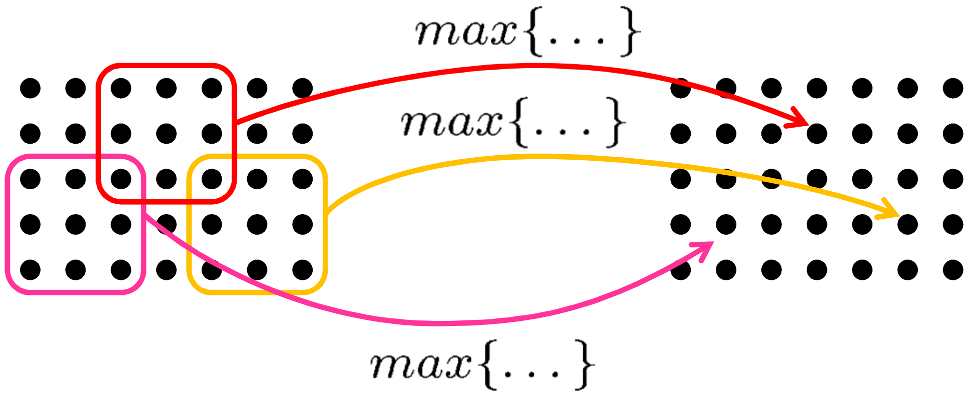
- in each hidden layer one would like to compute several different features for each pixel → <u>mulit-channel layers</u>



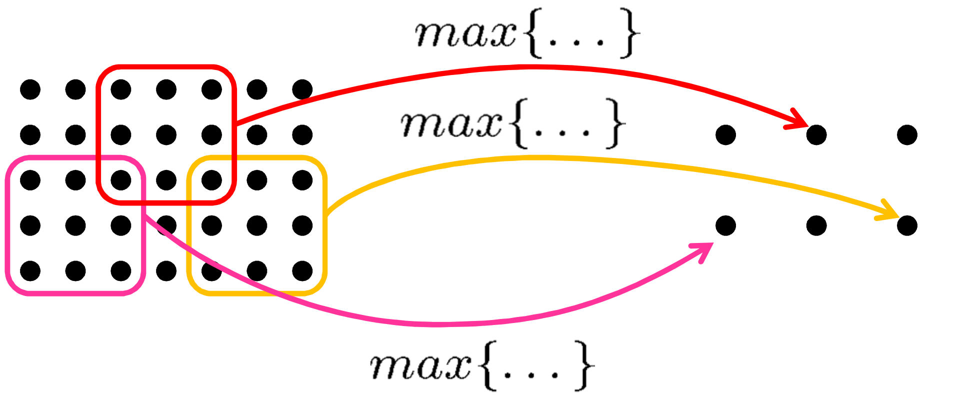
convolution kernels are tensors of size $h \times w \times k$

Max-Pooling

- pooling layers are designed to aggregate information spatially
- <u>Max-Pooling</u>: calculate maximum from local receptive field

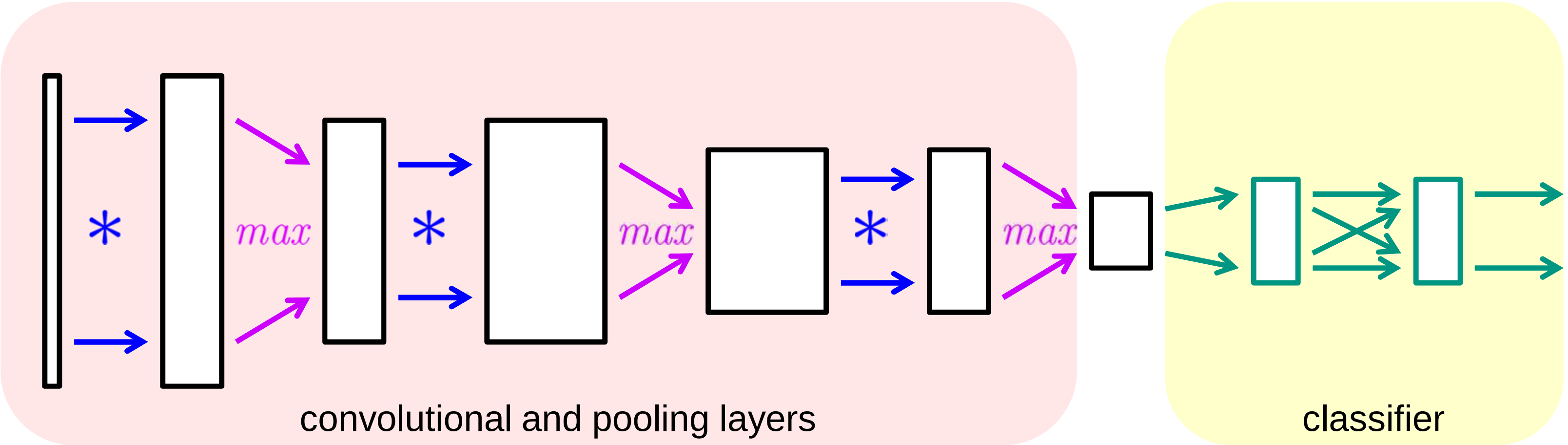


- pooling is often combined with shrinking of layers



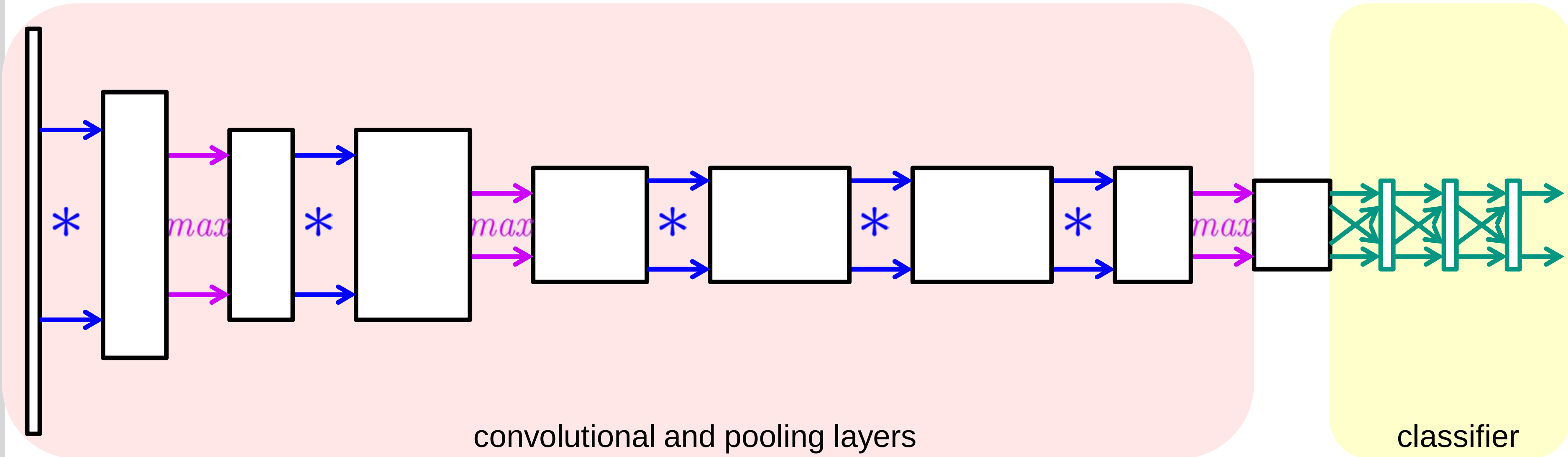
Convolutional Networks

- <u>Convolutional</u> Networks (CNNs) combine
  - convolutional layers
  - pooling layers
  - fully connected classifier network



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



Example: AlexNet

- results on test set
  - top-1-error: 37%
  - top-5-error: 17%

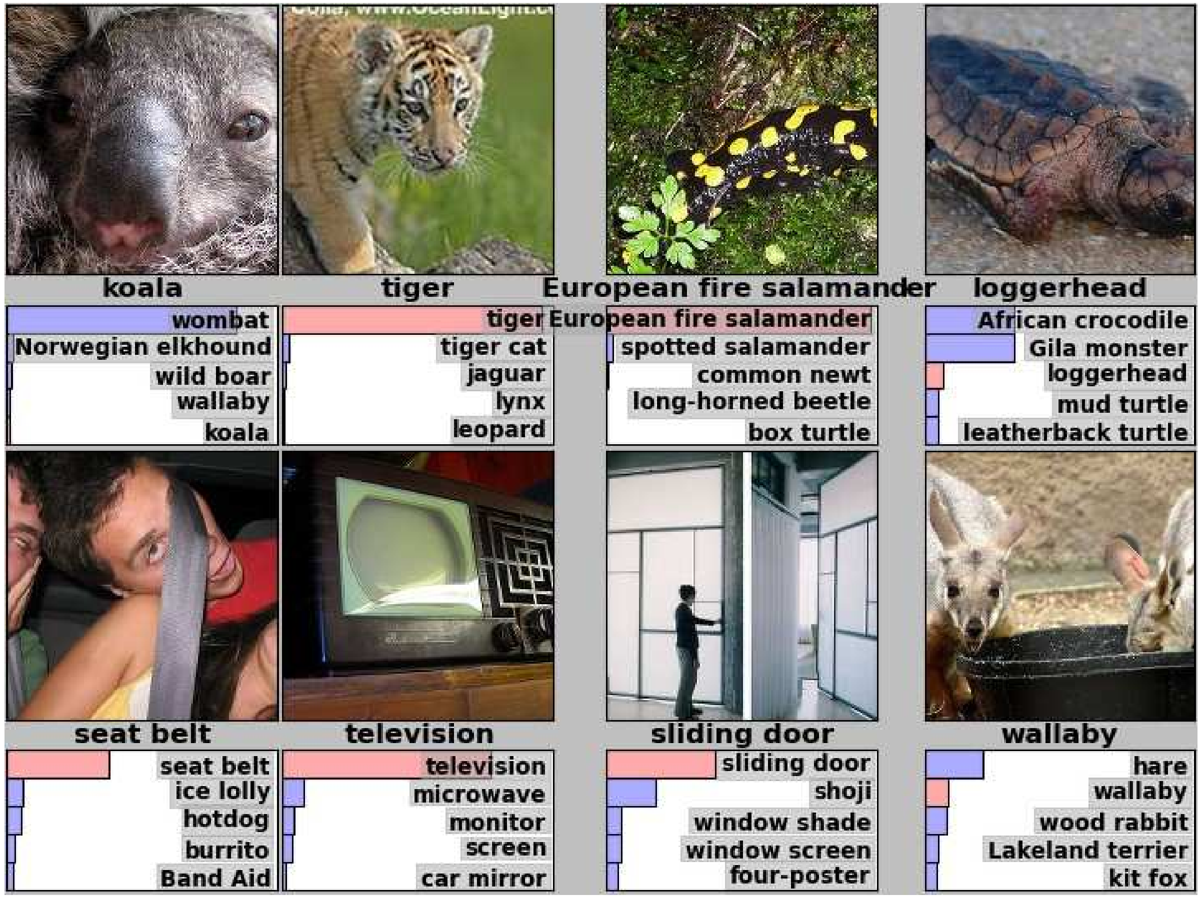
- newer approaches:
  - top-5-error: <5%
    (better than humans)

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

Example: AlexNet

- results on test set
  - top-1-error: 37%
  - top-5-error: 17%

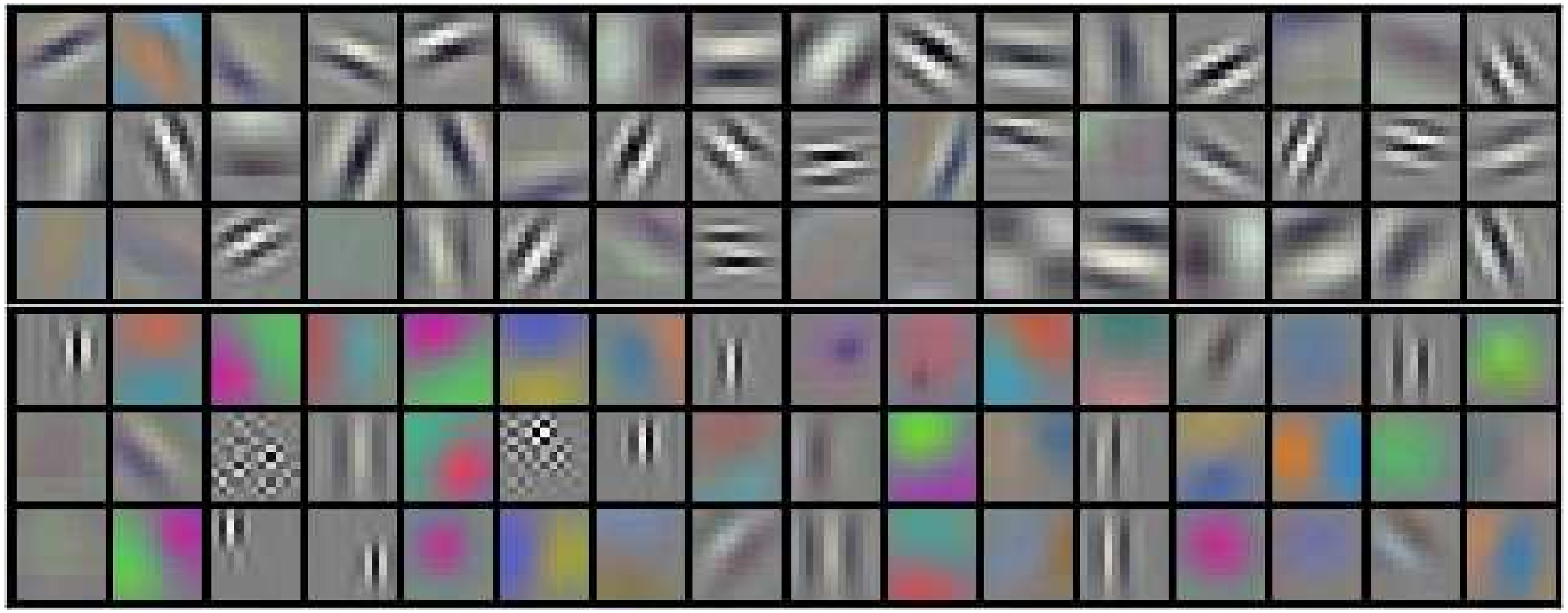
- newer approaches:
  - top-5-error: <5%
    (better than humans)



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

Convolutional Networks

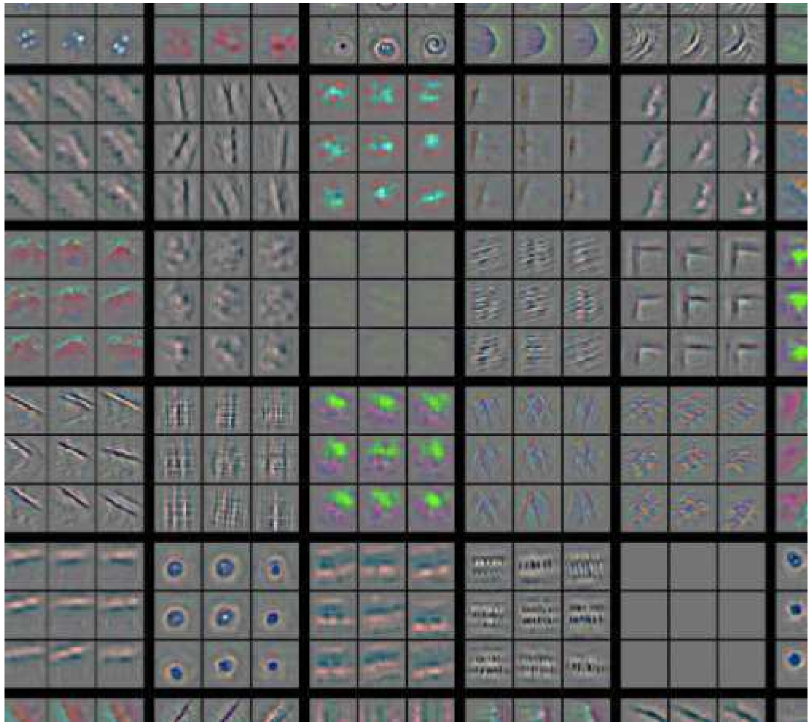
- which features are learned in hidden layers?
  - 1. layer: gray level edges, color edges, blobs



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

Convolutional Networks

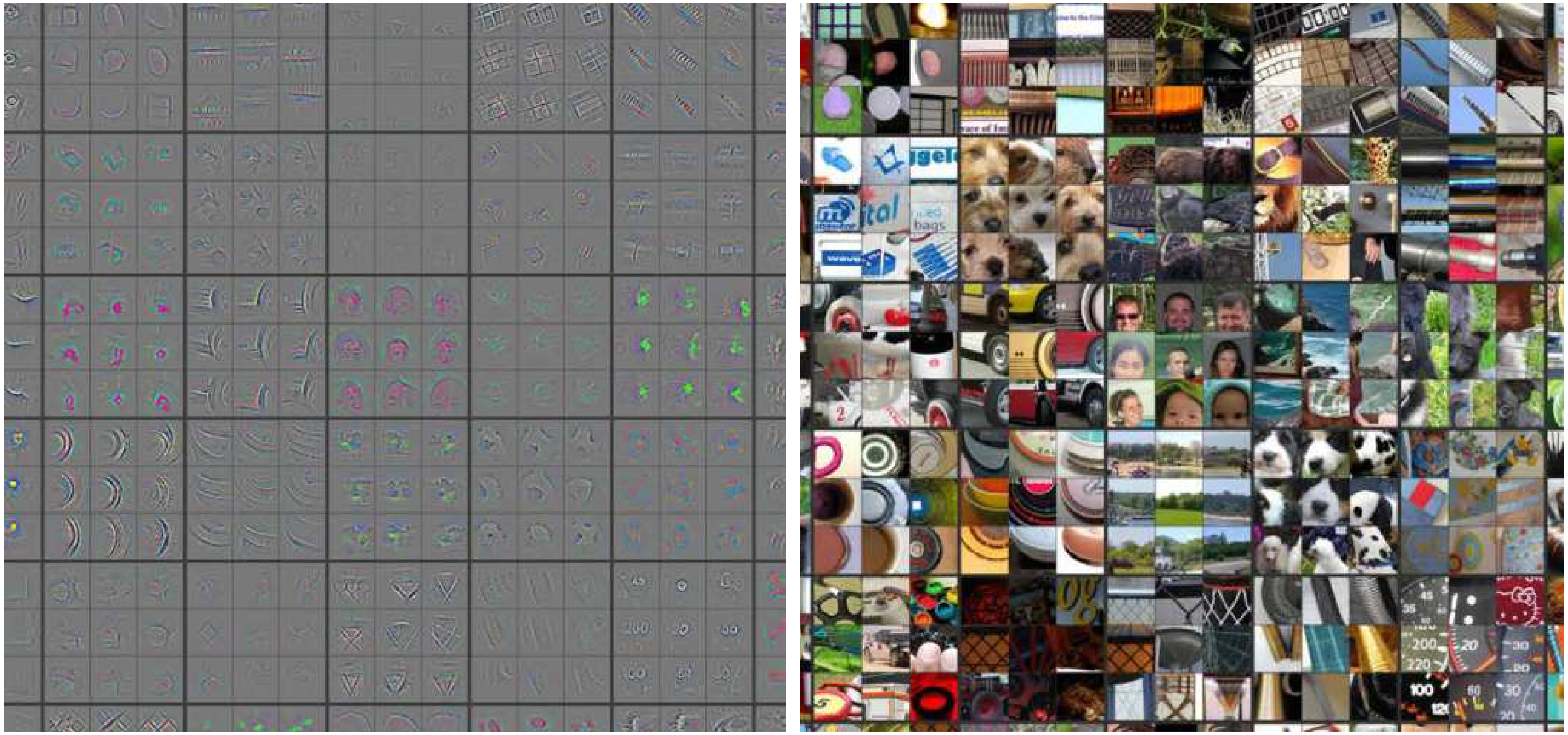
- which features are learned in hidden layers?
  - 2. layer: corners, round structures



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

Convolutional Networks

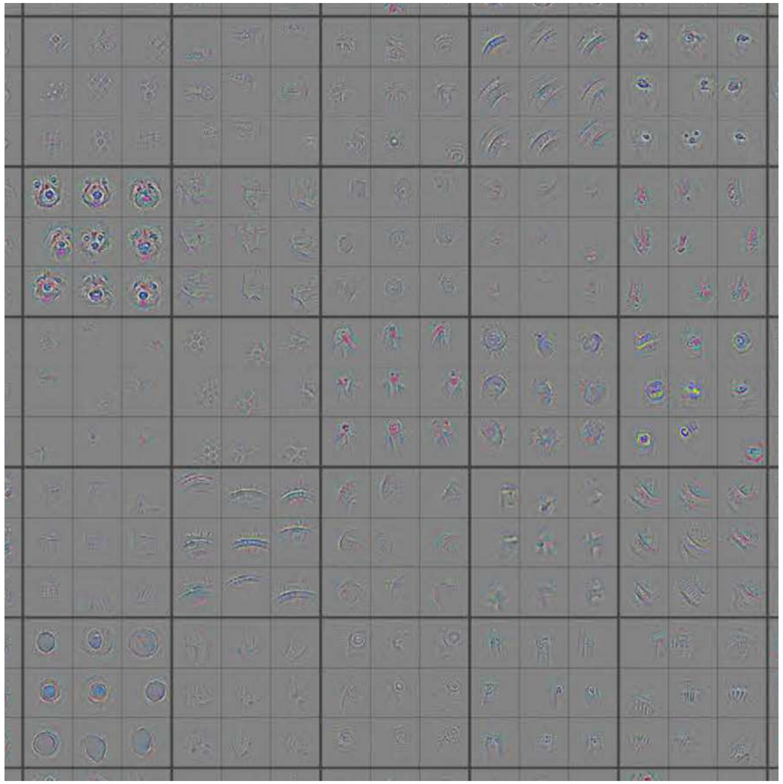
- which features are learned in hidden layers?
  - 3. layer: shapes, gratings

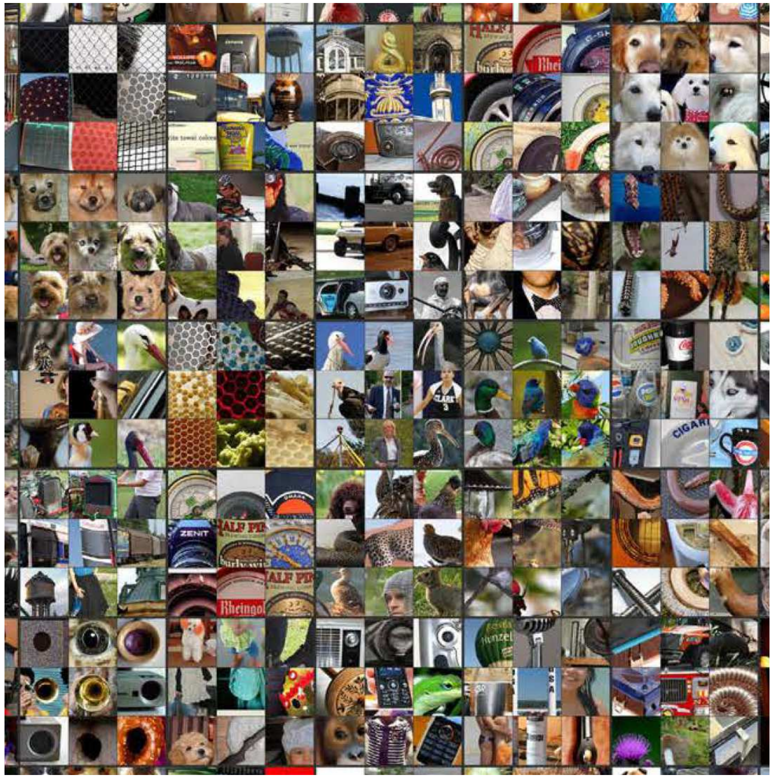


Taken from:
M.D. Zeller, R. Fergus, „Visualizing and Understanding Convolutional Neural Networks“, arXiv:1311.2901 v2, 13. Nov. 2013

Convolutional Networks

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

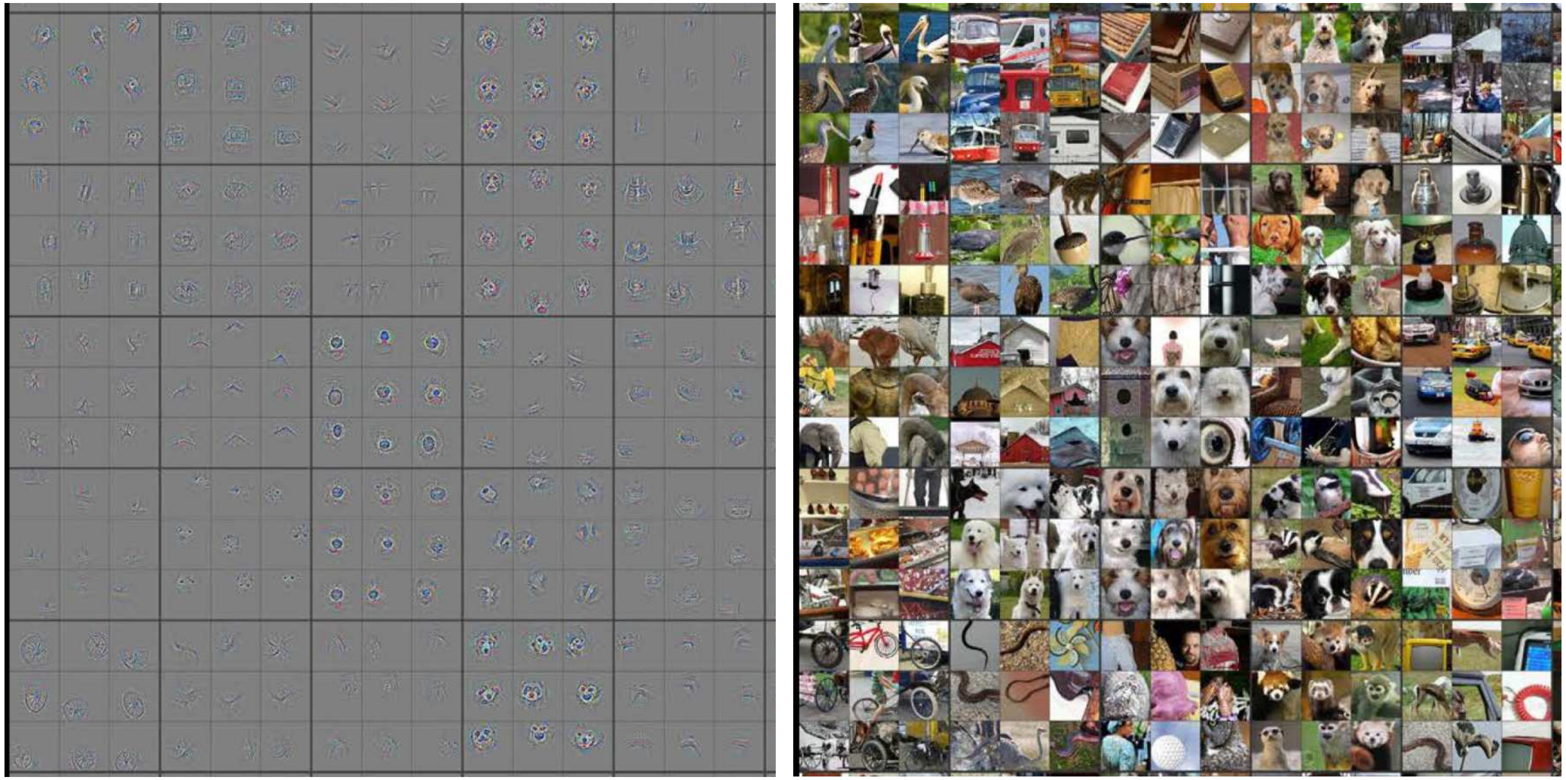




Taken from:
M.D. Zeller, R. Fergus, „Visualizing and Understanding Convolutional Neural Networks“, arXiv:1311.2901 v2, 13. Nov. 2013

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

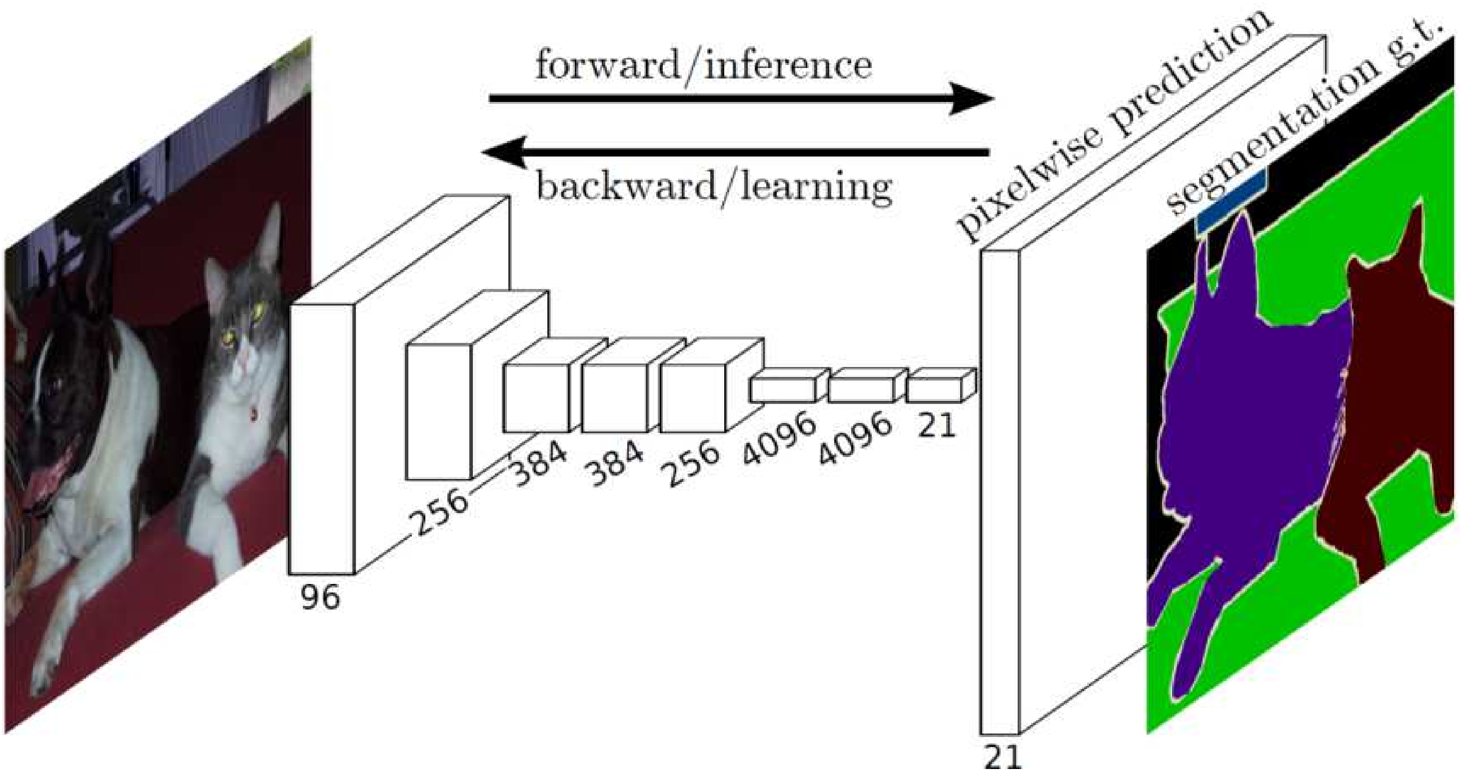
Convolutional Networks

- 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

APPLICATION EXAMPLES

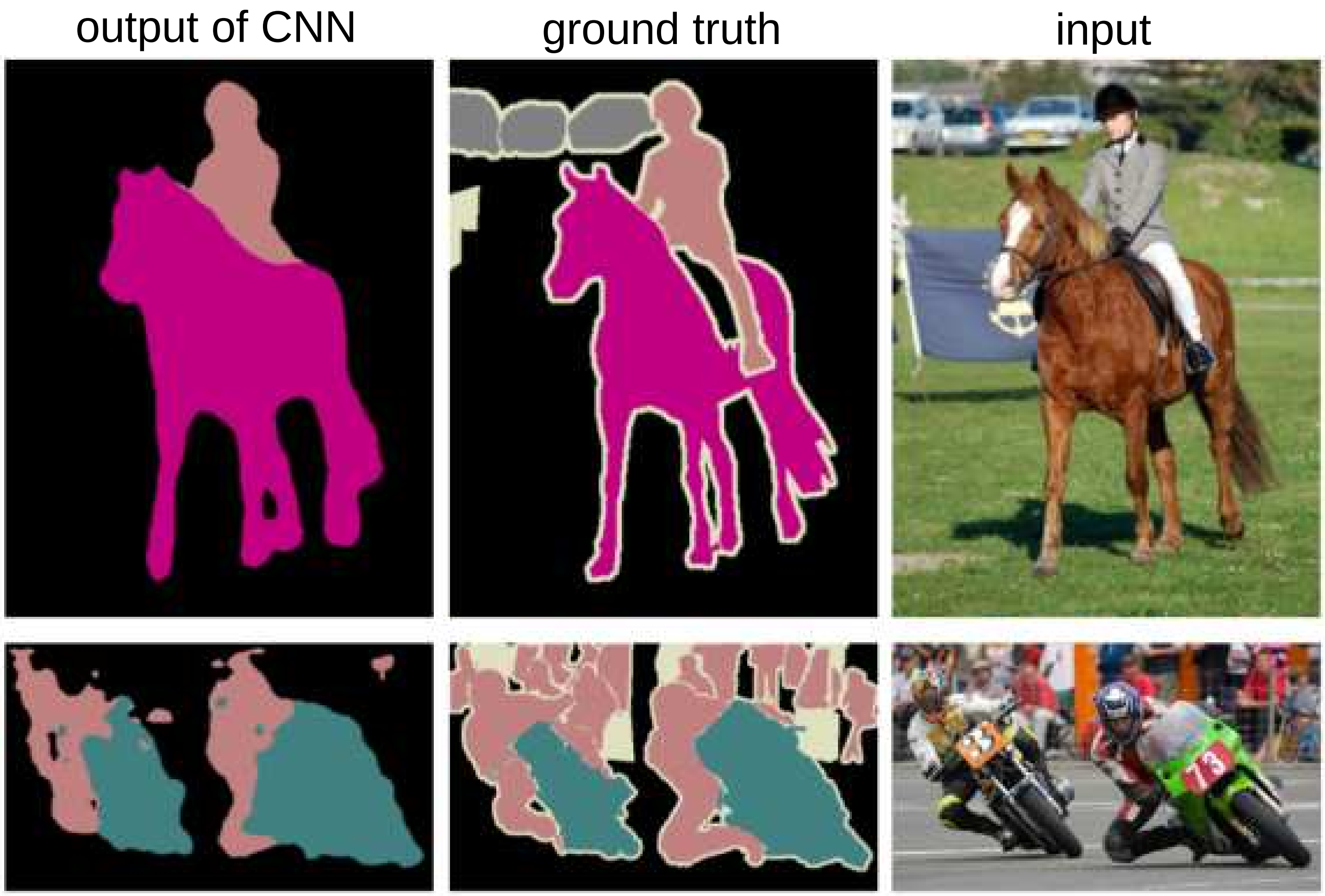
Scene Labeling

- segment the image
  - classify every pixel
  - autoencoder/decoder structure



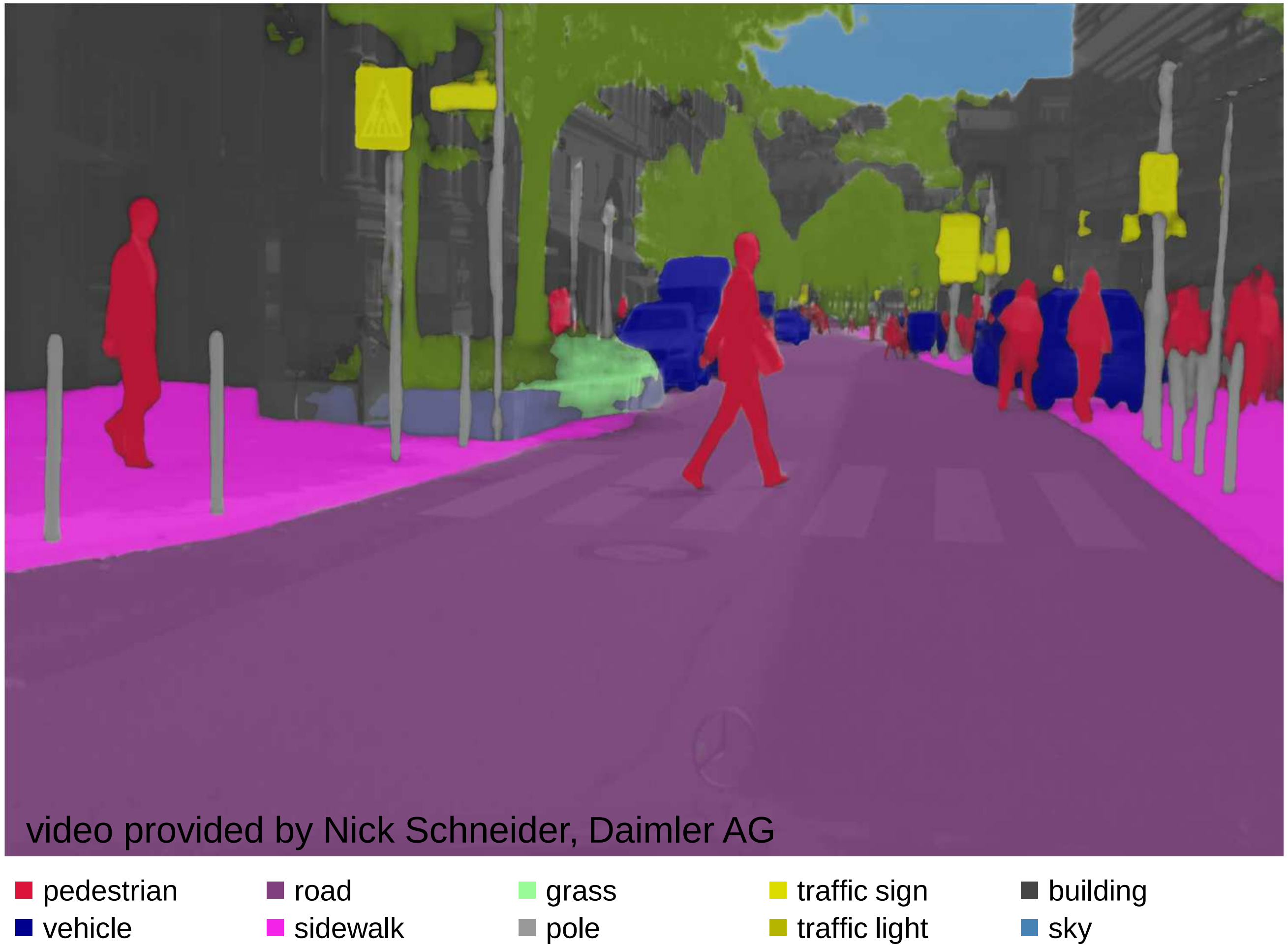
taken from: J. Long, E. Shelhamer, T. Darrell, „Fully Convolutional Networks for Semantic Segmentation“, CVPR, 2015

Scene Labeling



taken from: J. Long, E. Shelhamer, T. Darrell, „Fully Convolutional Networks for Semantic Segmentation“, CVPR, 2015

Scene Labeling



best performance on *Cityscapes* dataset > 80% accuracy

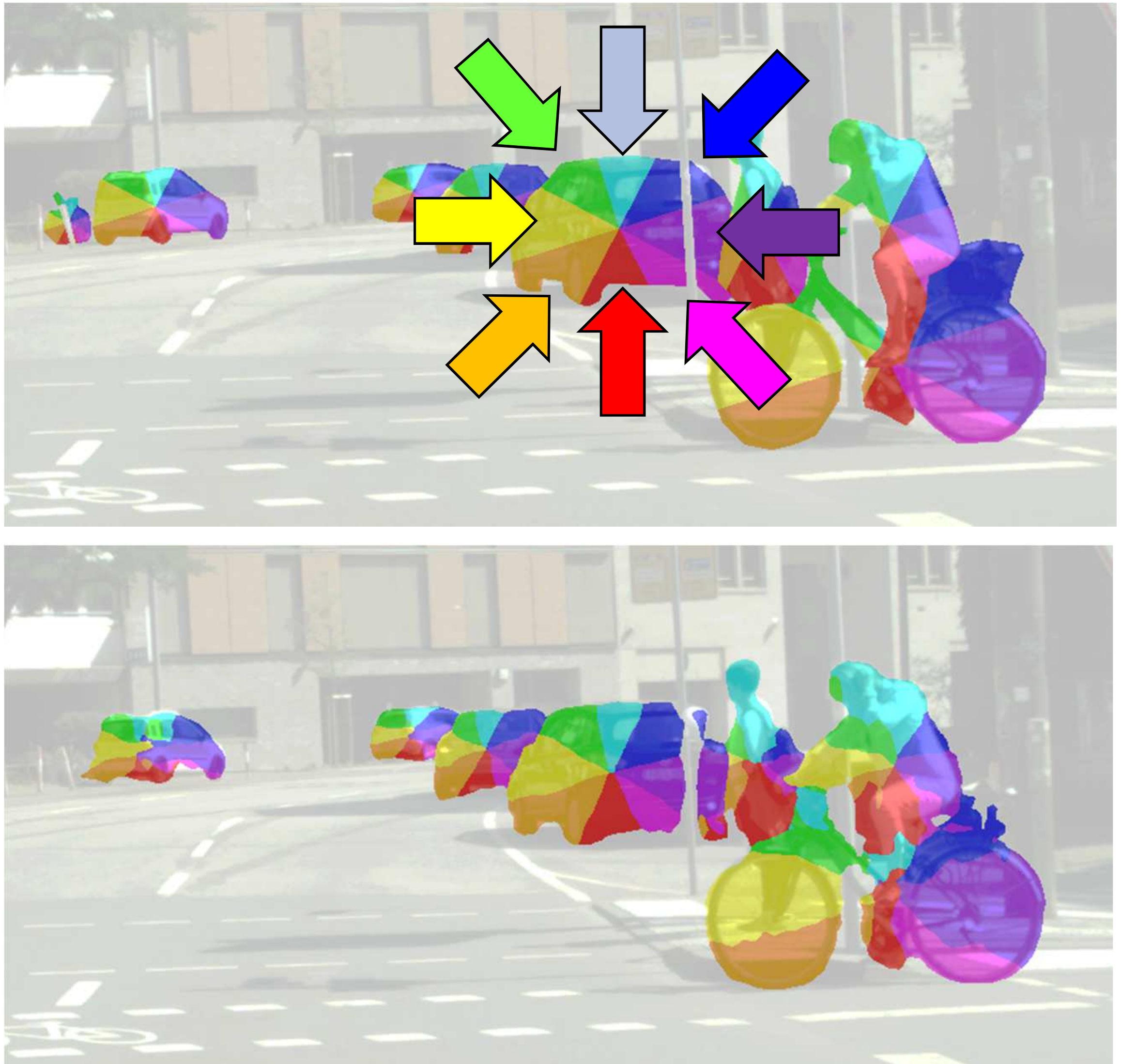
Instance Labeling

semantic labels do not provide object boundaries!

taken from: J. Uhrig, M. Cordts, U. Franke, T. Brox, Pixel-level encoding and depth layering for instance-level semantic segmentation, Germ. Conf. on Pattern Recognition, 2016/ provided by Nick Schneider, Daimler AG

Instance Labeling

idea:
label direction
to center of object



taken from: J. Uhrig, M. Cordts, U. Franke, T. Brox, Pixel-level encoding and depth layering for instance-level semantic segmentation, Germ. Conf. on Pattern Recognition, 2016/ provided by Nick Schneider, Daimler AG

Instance Segmentation

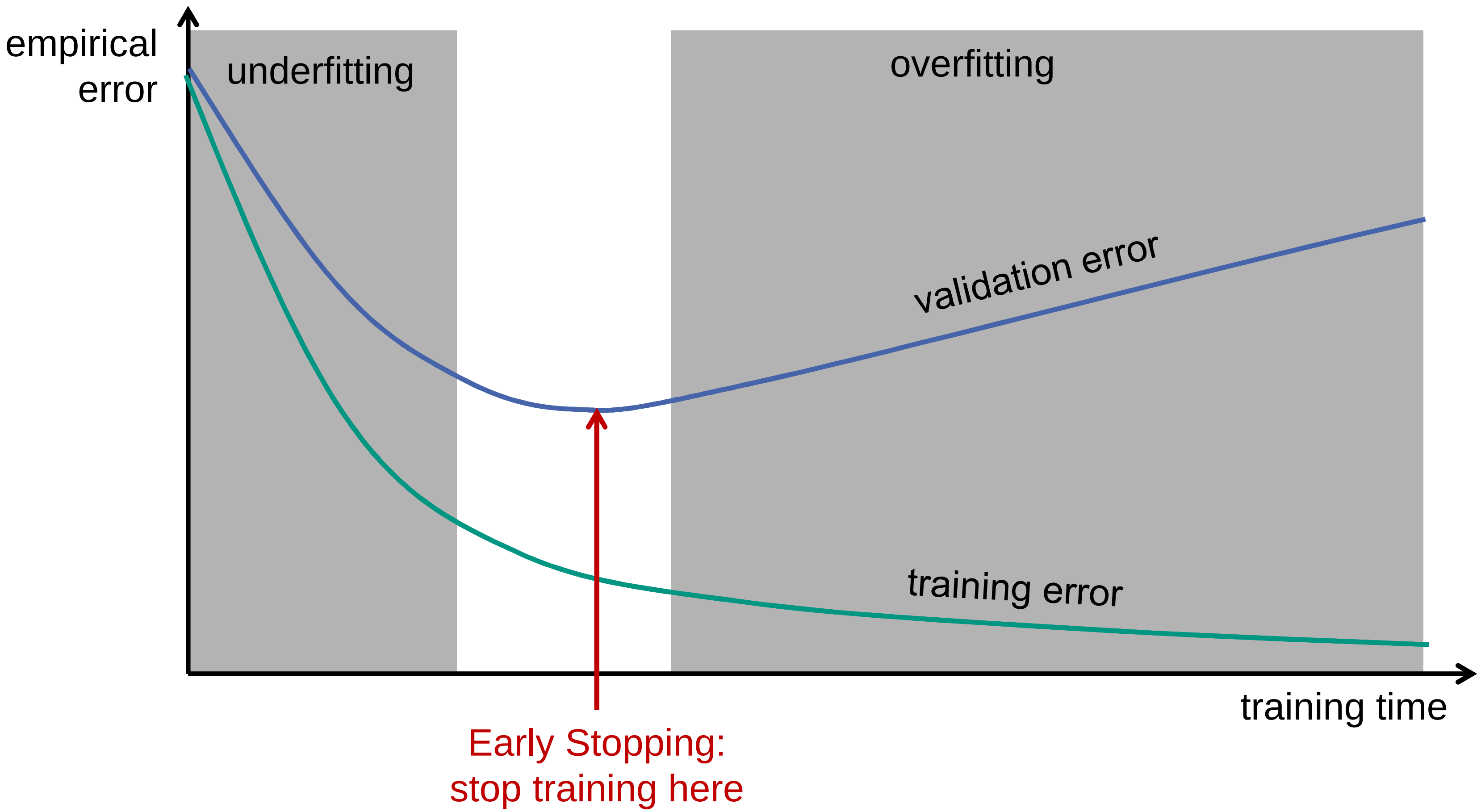
video provided by Nick Schneider, Daimler AG

TECHNIQUES FOR DEEP LEARNING

Principles for Training MLPs

- There‘s no data like more data!
  - remind slides 10/60-62 on 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



Why does early stopping work as regularisation technique?

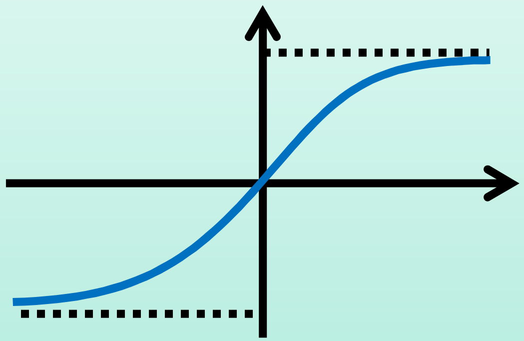
- early stopping prefers small weights
- small weights means little non-linearity (see next slide)

Regularisation by Small Weights

- assume perceptron with small absolute weights

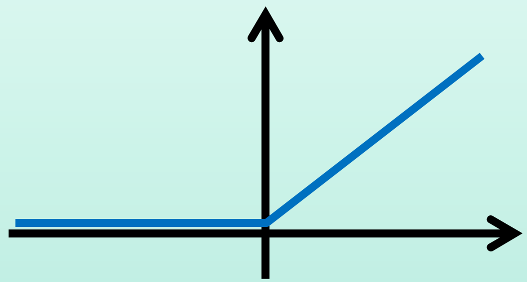
$$y = f_{act}\Big(w_0 + \sum_{i=1}^{m} w_i x_i\Big)$$

sigmoidal/hyperbolic tangent activation



$$y \approx w_0 + \sum_{i=1}^{m} w_i x_i$$

ReLu activation

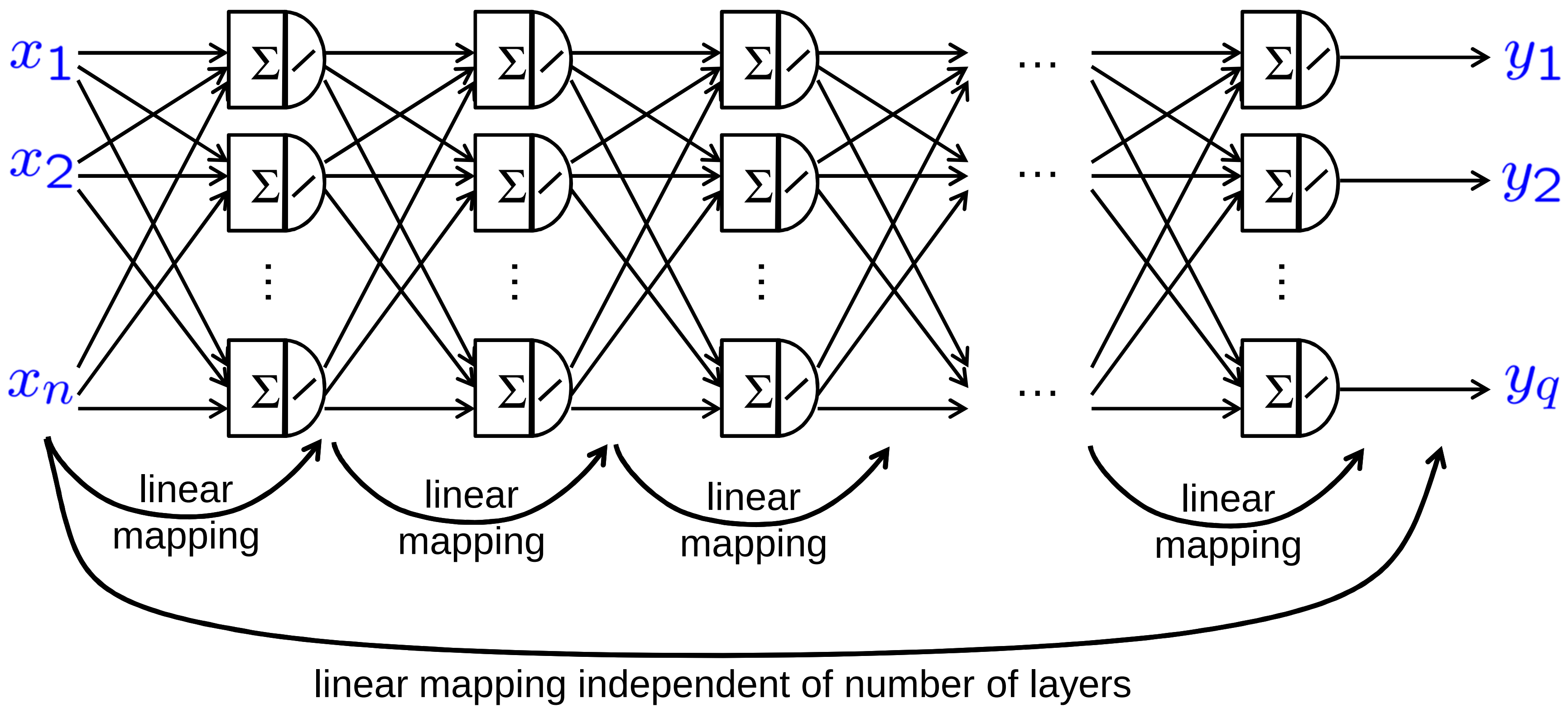


$$y \approx \begin{cases} 0 & \text{if } w_0 < 0 \\ w_0 + \sum_{i=1}^{m} w_i x_i & \text{if } w_0 > 0 \end{cases}$$

$\rightarrow$ small weights foster linear behavior of perceptrons

Regularisation by Small Weights

- assume fully connected network with linear activation



$\rightarrow$ linear behavior of perceptrons reduces non-linear expressiveness

$\rightarrow$ regularisation

Weight Decay / L2-Regularisation

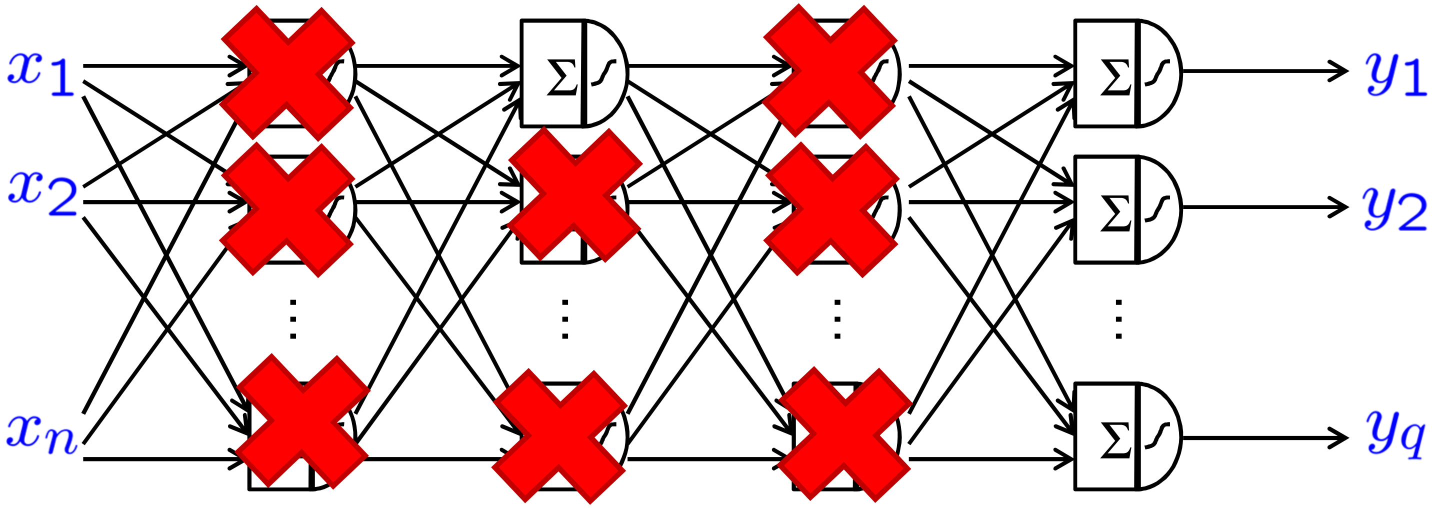
- extend goal of training by regulatisation term

$$\min_{\vec{w}} \underbrace{\sum_{j=1}^{p} err\left(f_{MLP}^{\vec{w}}(\vec{x}^{(j)}), \vec{t}^{(j)}\right)}_{\text{1st goal: minimize error on training set}} + \lambda \cdot \underbrace{\sum_{\text{weights } i} w_i^2}_{\text{2nd goal: prefer small weights (regularisation)}}$$

$\lambda \geq 0$ is a parameter of the training algorithm that has to be chosen by trial and error

Dropout

- regularization by randomly switching off perceptrons during training



- dropout forces the neural network to store relevant information in a distributed way
- dropout reduces overfitting

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 $S \subseteq \{1, \dots, p\}$

calculate gradient from subset of all training examples. Subsets typically cacle through all examples

advantages:

- speed up
- a little bit less overfitting

Modifications of Gradient Descent

- gradient descent with momentum

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

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

with $\alpha \geq 0$ a parameter that controls consistency of subsequent steps

advantages:

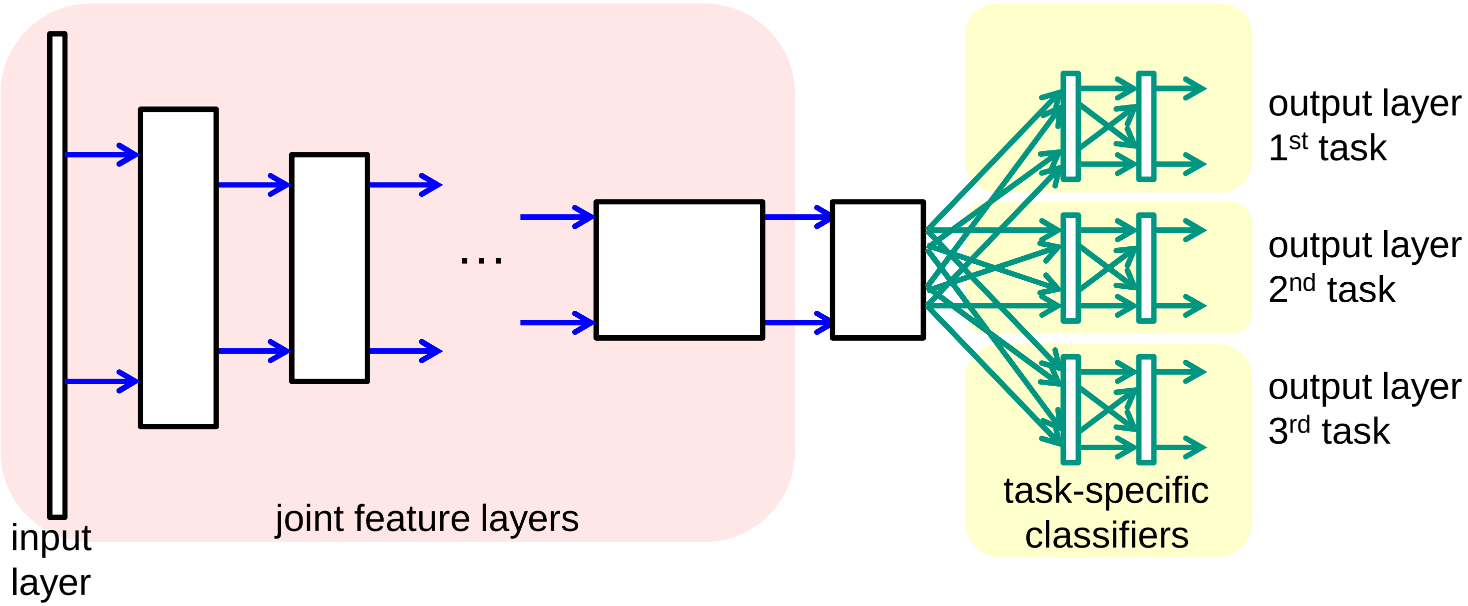
– speed up in flat areas

– less zig zagging

Multi Task Learning

- idea: learn several related tasks in a single network

  example: scene labeling + instance labeling + depth estimation



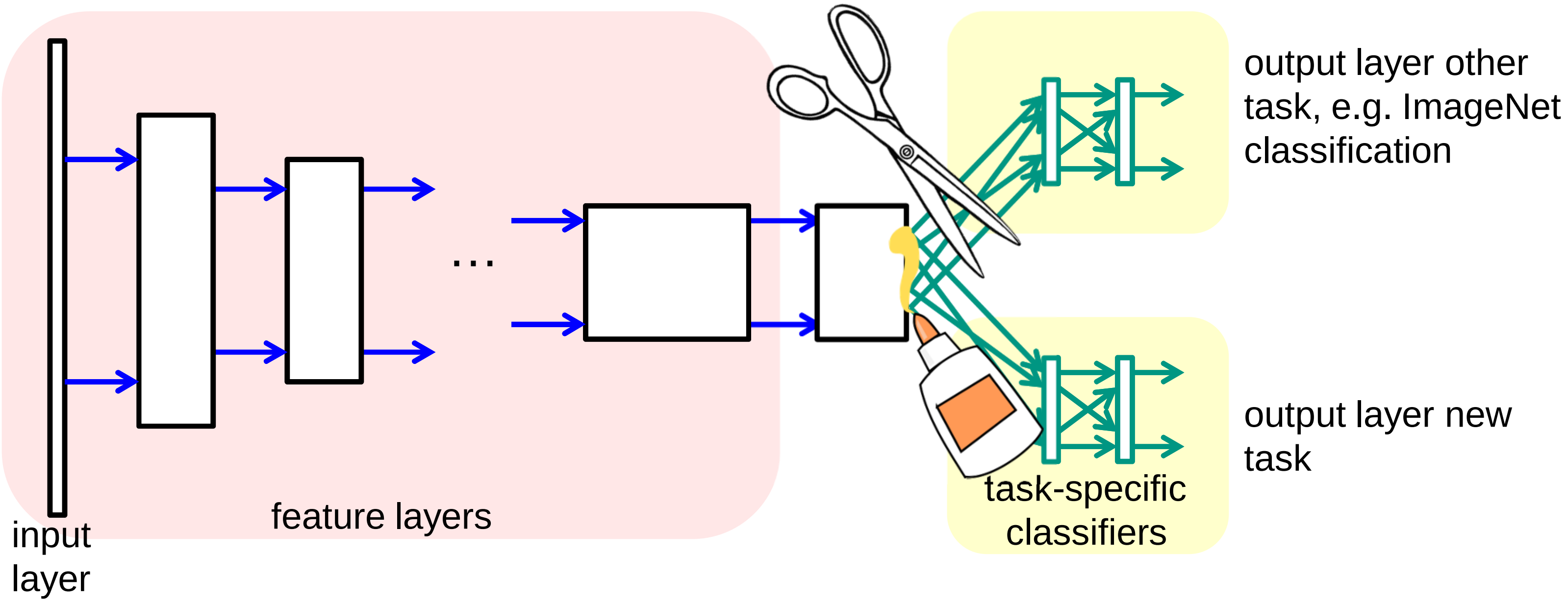
advantages:

– force network to develop common features in hidden layer

– reduce overfitting to a single task

Usage of Pre-Trained Feature Networks

- 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

Popular Architectures

net	layers	kernel sizes	reference
LeNet5 (1998) (historical)	2 conv. layers 2 max pooling layers 2 fully conn. layers	5x5	Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. *Gradient-based learning applied to document recognition*. Proc. of the IEEE, 1998
AlexNet (2012)	5 conv. layers 3 max pooling layers 3 fully conn. layers	3x3 – 11x11	A. Krizhevsky, I. Sutskever, G. E. Hinton. *Imagenet classification with deep convolutional neural networks*. NIPS 2012
VGG (2014)	13 conv. layers 3 max pooling layers 3 fully conn. layers	3x3 (1x1)	K. Simonyan, A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. arXiv 2014
ResNet (2015)	152 conv. layers (residual blocks) 2 pooling layers	3x3 (7x7)	K. He, X. Zhang, S. Ren, J. Sun. *Deep residual learning for image recognition*. CVPR 2016
GoogLeNet (2015)	9 inception blocks	1x1, 3x3, 5x5	C. Szegedy, et al. *Going deeper with convolutions*. CVPR 2015

SUMMARY: DEEP LEARNING

Pattern Recognition: the Complete Picture

feature extraction → classifier / ensemble → prediction c

training & generalization

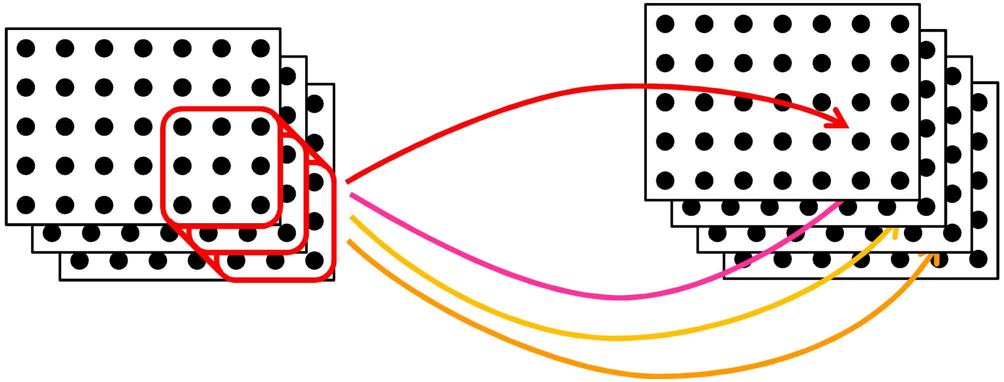
features	classifiers	ensembles	generalization
smart (specific)	SVM	free ensemble	validation
HOG	threshold	bagging	cross-validation
Haar	decision tree	boosting	data tuning
LBP	cascade	decision forest	early stopping
neural features	neural network	multi-class	randomization
…	…	…	…

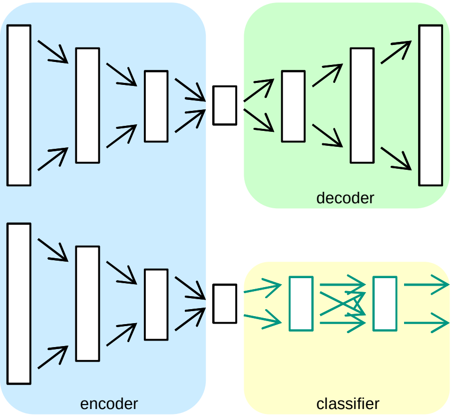
Pattern Recognition: the Complete Picture cont.

features

- smart (specific)
- HOG
- Haar
- LBP
- neural features
- …

- neural features
  – generated by autoencoder networks
  – generated by convolutional networks
  – neural features often perform better than "traditional" features (e.g. HOG) even if trained on different images



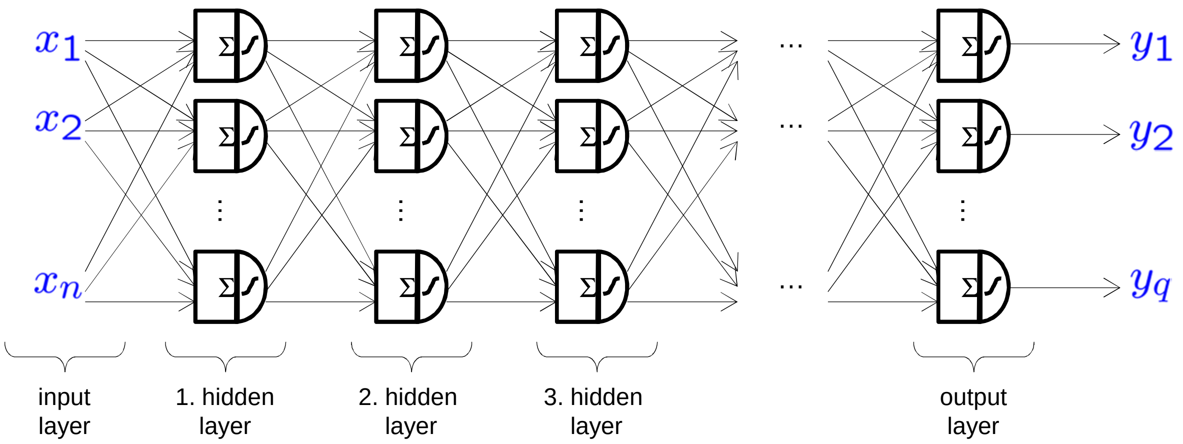


Pattern Recognition: the Complete Picture cont.

classifiers

- SVM
- threshold
- decision tree
- cascade
- neural network
- …

- artificial neural network
  - highly parameterized
  - nonlinear functions
  - build out of simple blocks (perceptrons)
  - layered layout



Pattern Recognition: the Complete Picture cont.

generalization

- validation
- cross-validation
- data tuning
- early stopping
- randomization
- …

- early stopping
  - monitor validation error
  - stop training at minimum
- weight decay/L2-regularisation
  - preference for small weights
  - punish nonlinearity
- dropout
  - randomly switch off percpetrons
  - foster distributed representation
- multi task learning
  - train related tasks with same network
  - share feature layers
- reuse of layers and networks
  - reuse trained feature layers
  - replace and retrain classification layers
  - little retraining of all weights

References

- books and articles

  - I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press, 2016. online: http://www.deeplearningbook.org
    extensive description of deep learning techniques
  - Y. LeCun, Y. Bengio, G. E. Hinton. *Deep Learning*. nature 521, pp. 436-444, 2015
    brief overview paper, not very deep but good to get a rough idea
  - L. Deng, D. Yu. *Deep learning: methods and applications*. Foundations and Trends in Signal Processing 7:3–4, pp. 197-387, 2014. online: http://ftp.nowpublishers.com/article/Details/SIG-039
    extensive overview and introduction, focus on natural language processing
  - Y. Bengio. Learning Deep Architectures for AI. Foundations and Trends in Machine Learning 2:1, pp. 1-127, 2009. online: http://www.nowpublishers.com/article/Details/MAL-006
    extensive introduction and overview, does not contain newer developments

References

- toolboxes
  - Caffe (UC Berkeley), http://caffe.berkeleyvision.org
    simple to use deep learning engine, provides pretrained networks, good for beginners
  - Tensorflow (Google), https://www.tensorflow.org
    deep learning engine that allows more extensions of existing approaches, good for experienced persons