Neuronale Netze Zusammenfassung

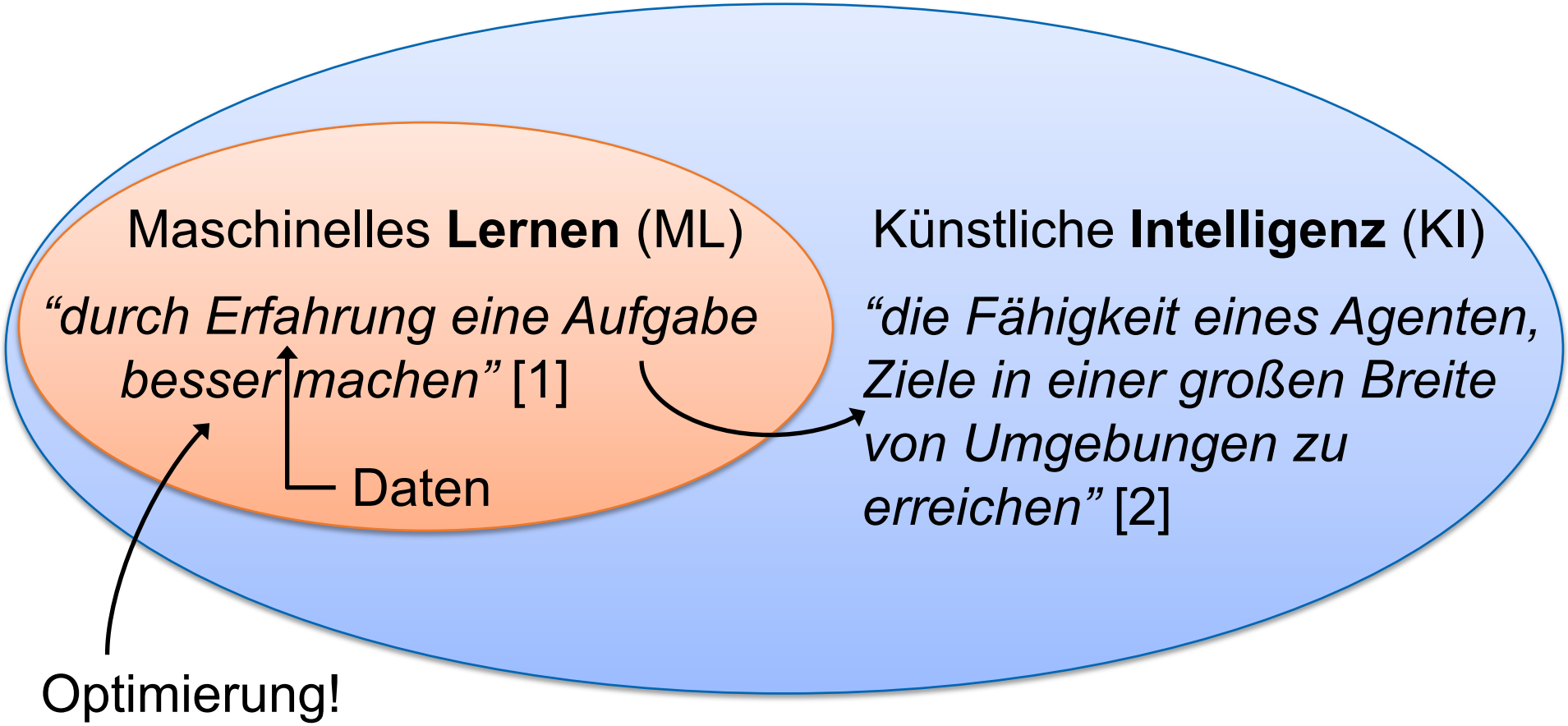
Prof. Dr.-Ing. Sebastian Stober

Artificial Intelligence Lab
Institut für Intelligente Kooperierende Systeme
Fakultät für Informatik
stober@ovgu.de

FACULTY OF
COMPUTER SCIENCE

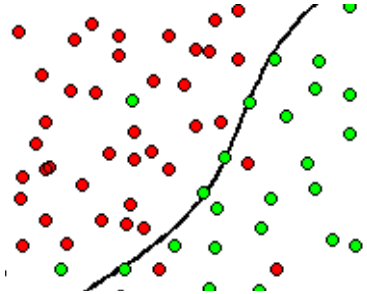
Künstliche Intelligenz & Maschinelles Lernen

Maschinelles Lernen



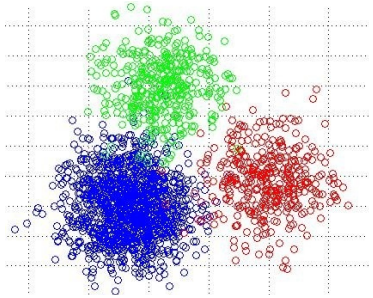
[1] T. Mitchell (1997). "Machine Learning", McGraw Hill.
[2] S. Legg; M. Hutter (2007). "Universal Intelligence: A Definition of Machine Intelligence". Minds & Machines. **17** (4): 391–444.

ML Problemklassen



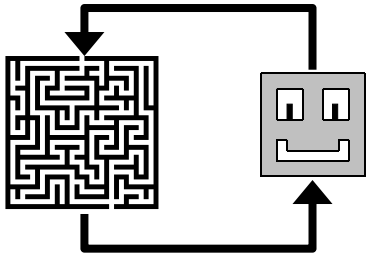
Überwachtes Lernen

(supervised learning)



Unüberwachtes Lernen

(unsupervised learning)



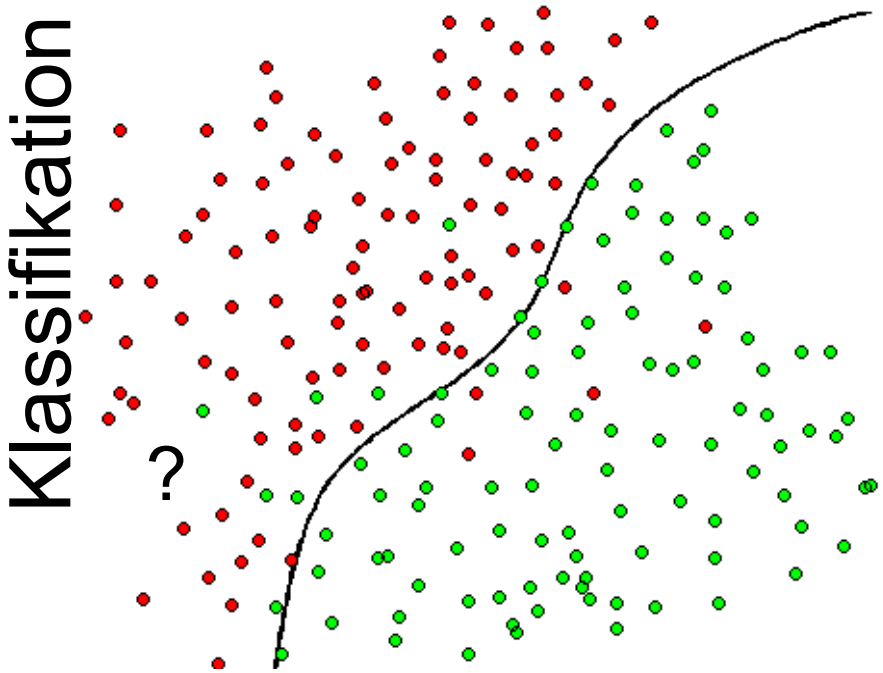
Bestärkendes Lernen

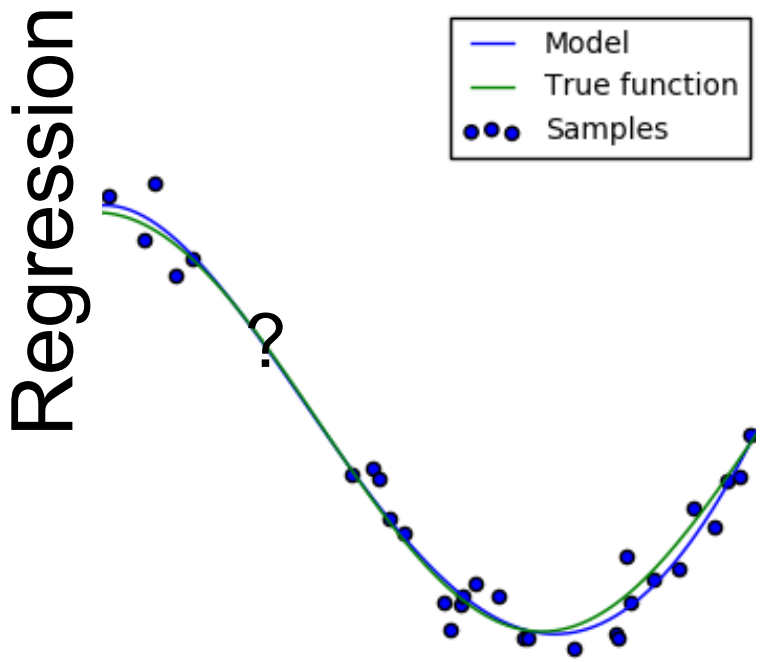
(reinforcement learning)

Überwachtes Lernen

(feste Lernaufgabe)

- annotierte Trainingsdaten (Beispielein- und Ausgaben)
- lerne Vorhersagemodell

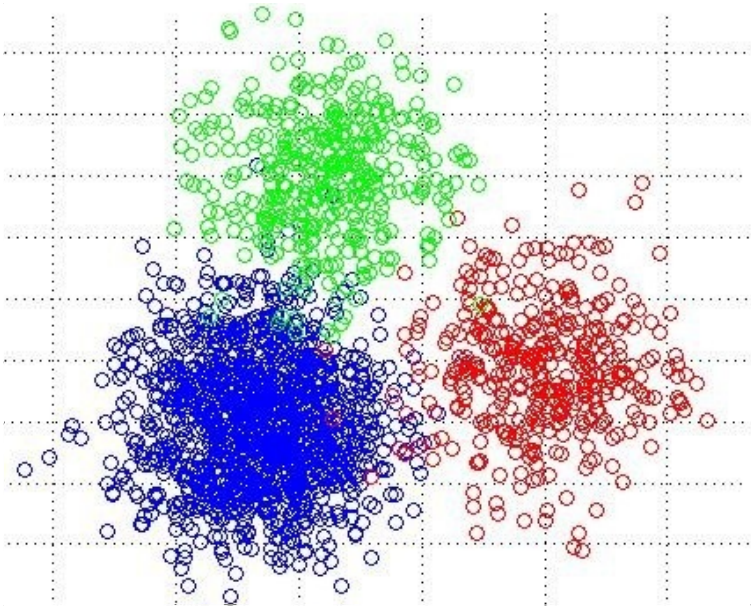




Unüberwachtes Lernen

(freie Lernaufgabe)

- Trainingdaten **ohne Annotationen**
- Lerne Struktur der Daten

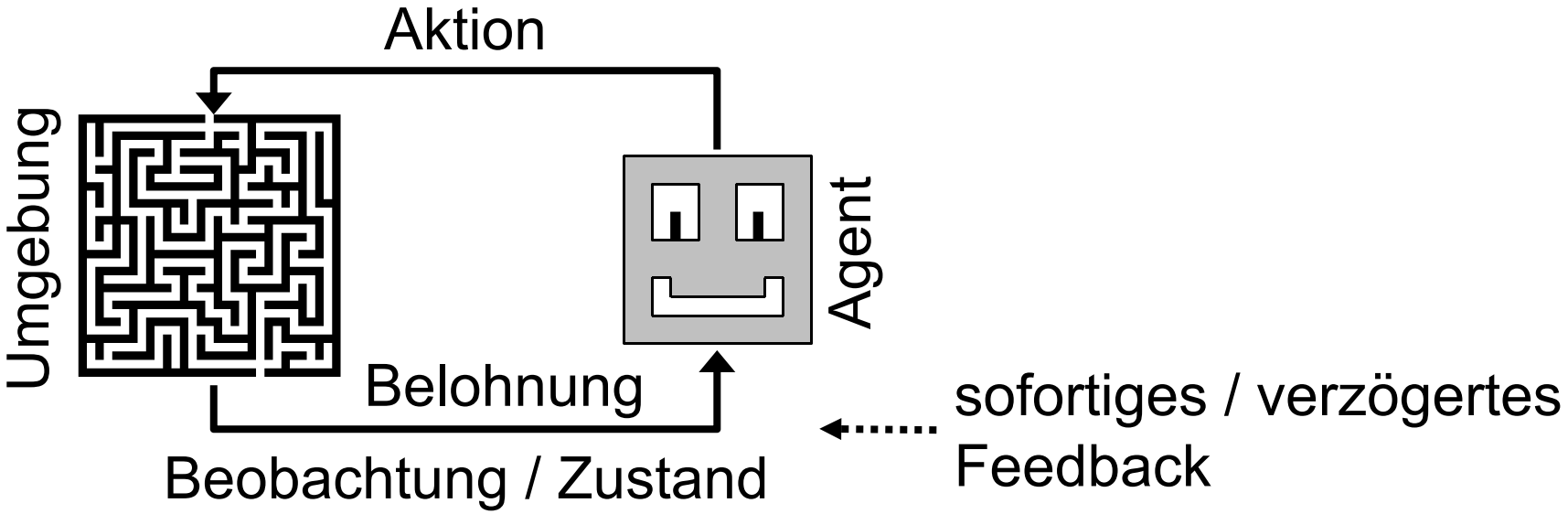


a) eigenständiges Ziel (Muster entdecken)

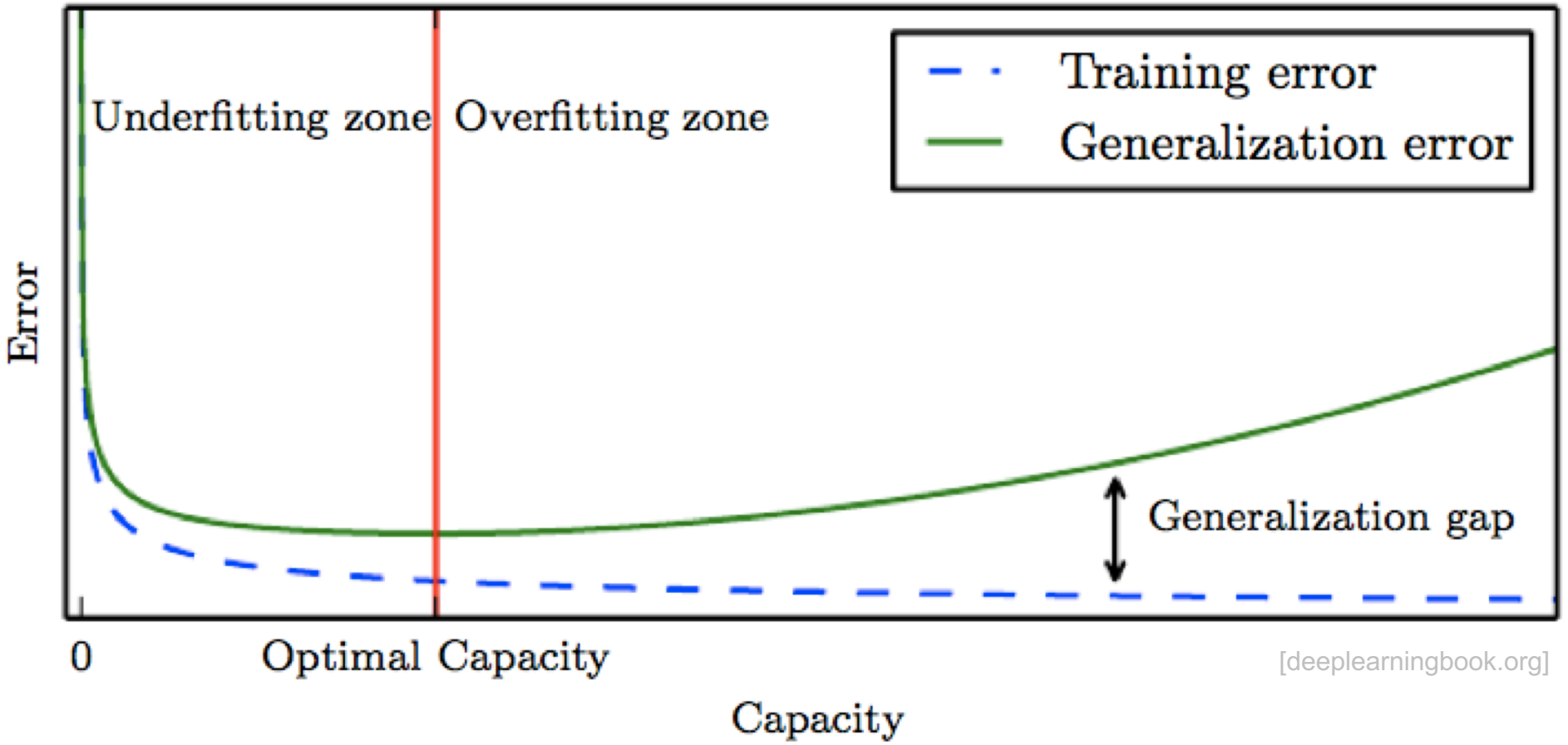
b) Zwischenschritt der Datenverarbeitung

Bestärkendes Lernen

- Trainingdaten durch Interaktion mit Umgebung
- Lerne Verhalten, welches die kumulative (verzögerte!) Belohnung über die Zeit maximiert



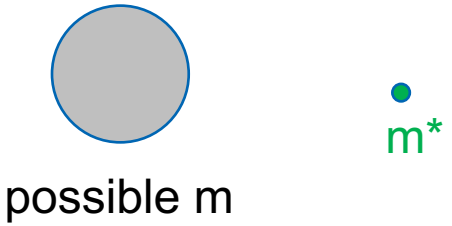
Modell-Kapazität



Bewertung & Selektierung von Modellen nur auf der Basis bisher ungesehener Daten!

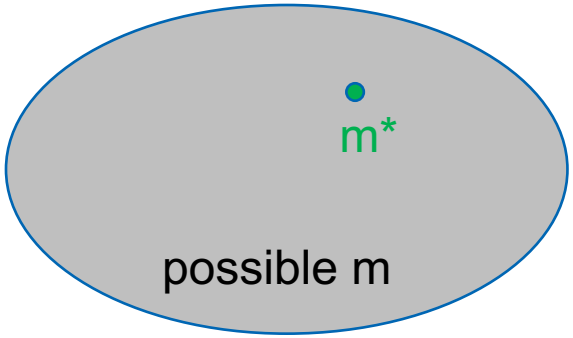
Bias-Variance Trade-Off

- high bias, low variance:

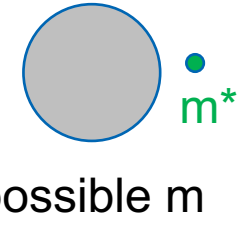


- low bias, high variance:

regularize!



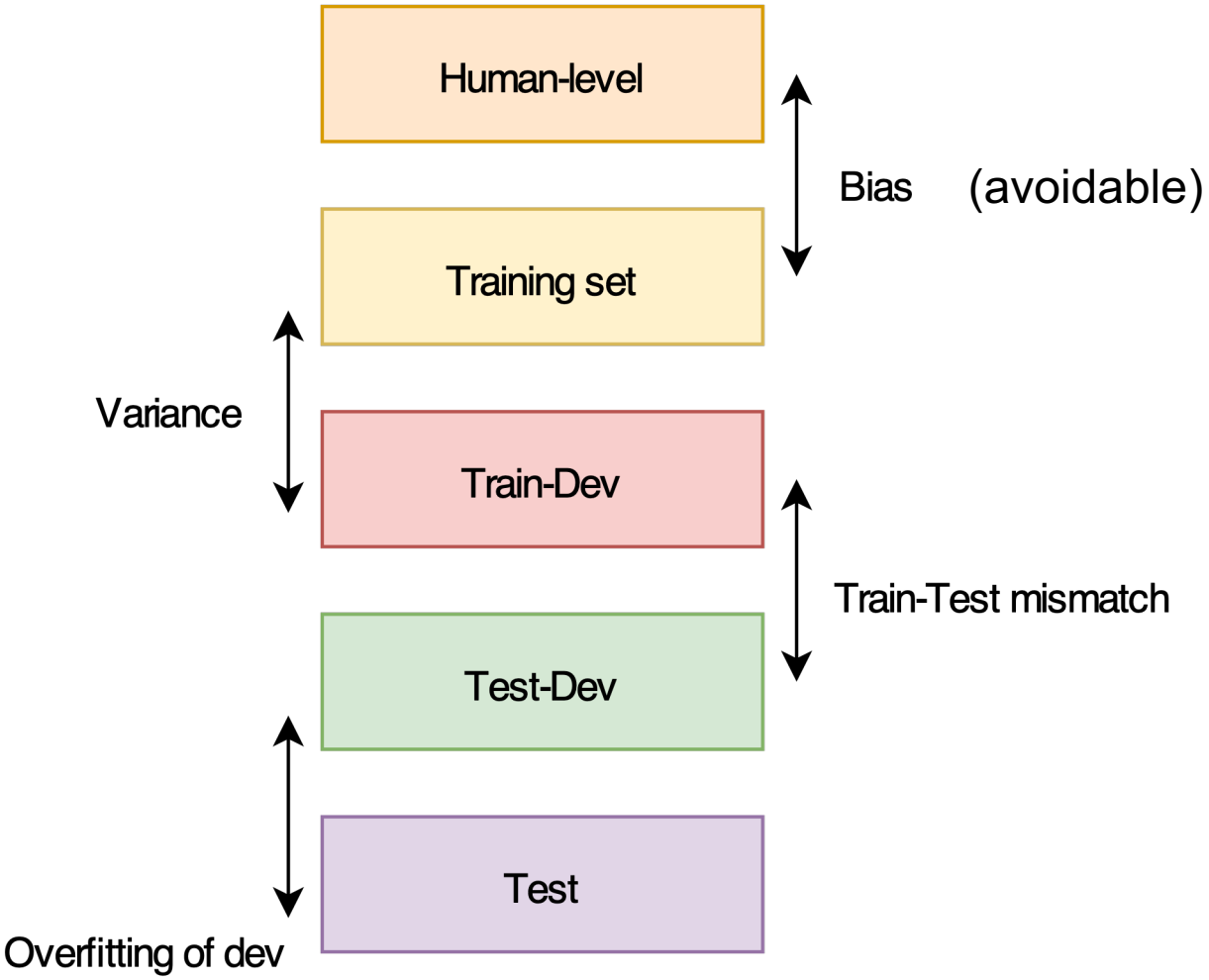
- good trade-off:



Regularization Techniques

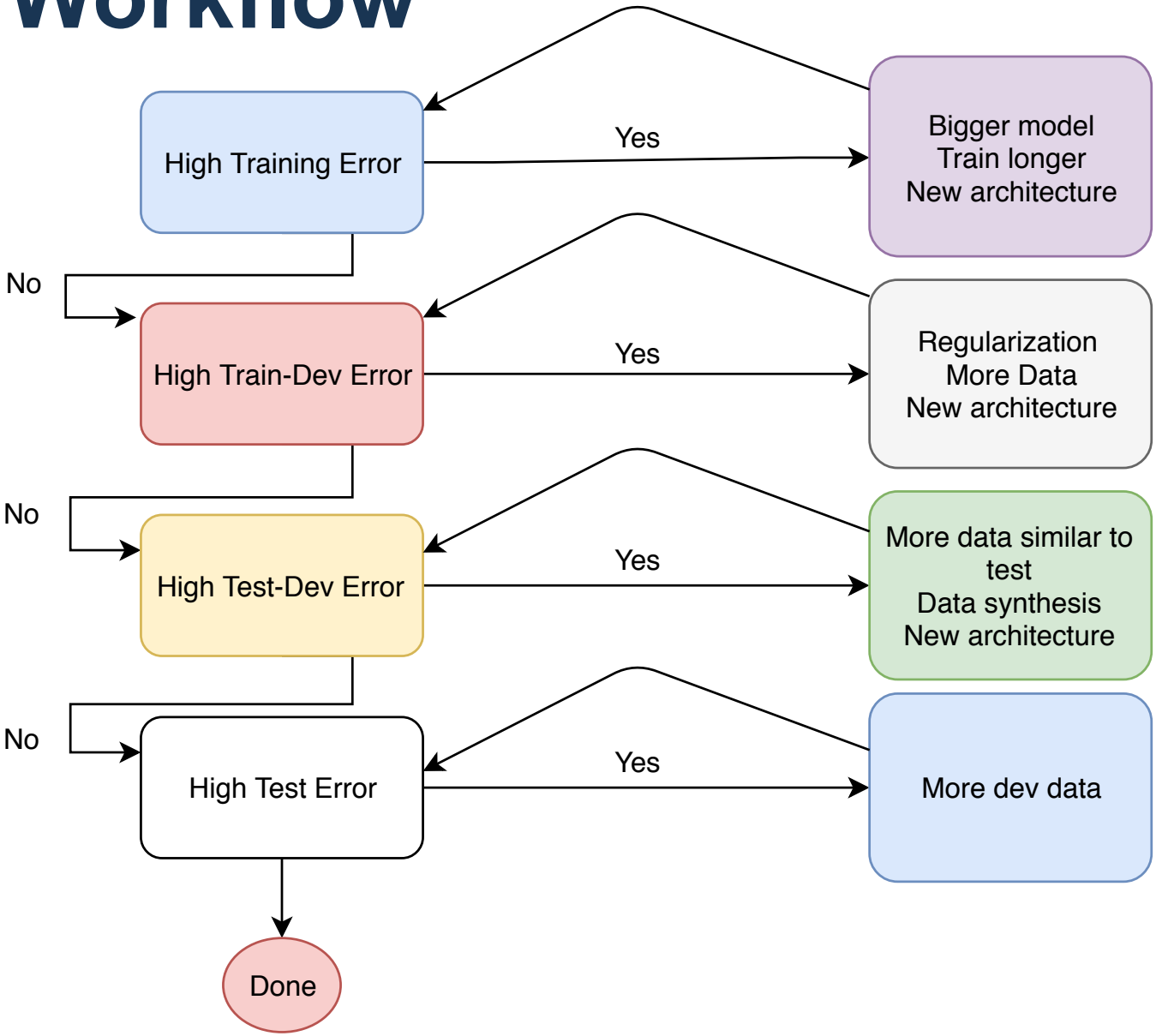
- parameter norm (L1/L2)
- early stopping
- dropout
- more data / data augmentation
- adding noise / denoising
- semi-supervised learning
- multi-task learning
- parameter tying & sharing
- sparse representations
- bagging / ensembles
- DropConnect = randomly set weights to zero
- (layer-wise) unsupervised pretraining
- adversarial training
- …

Error Factors



https://kevinzakka.github.io/2016/09/26/applying-deep-learning/

Workflow



(bias)

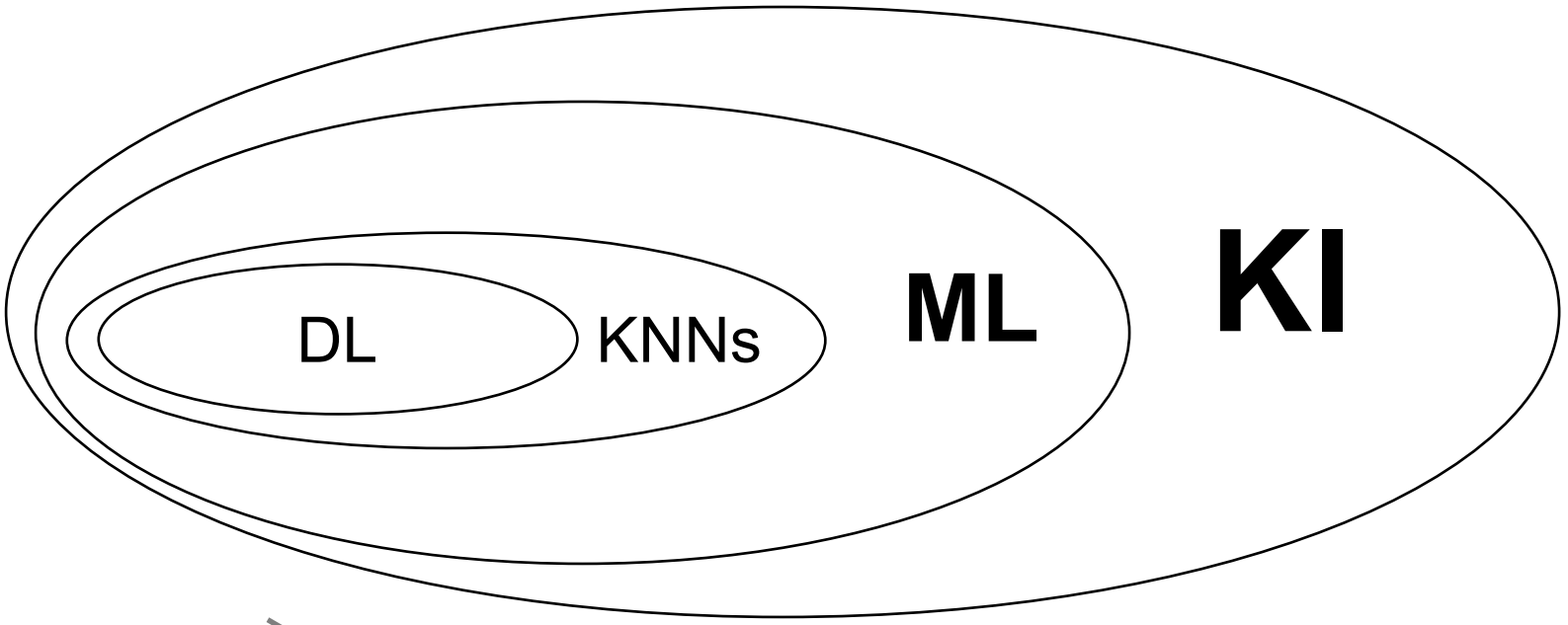
(variance)

(train-test data mismatch)

(overfit dev)

https://kevinzakka.github.io/2016/09/26/applying-deep-learning/

Thematische Einordnung



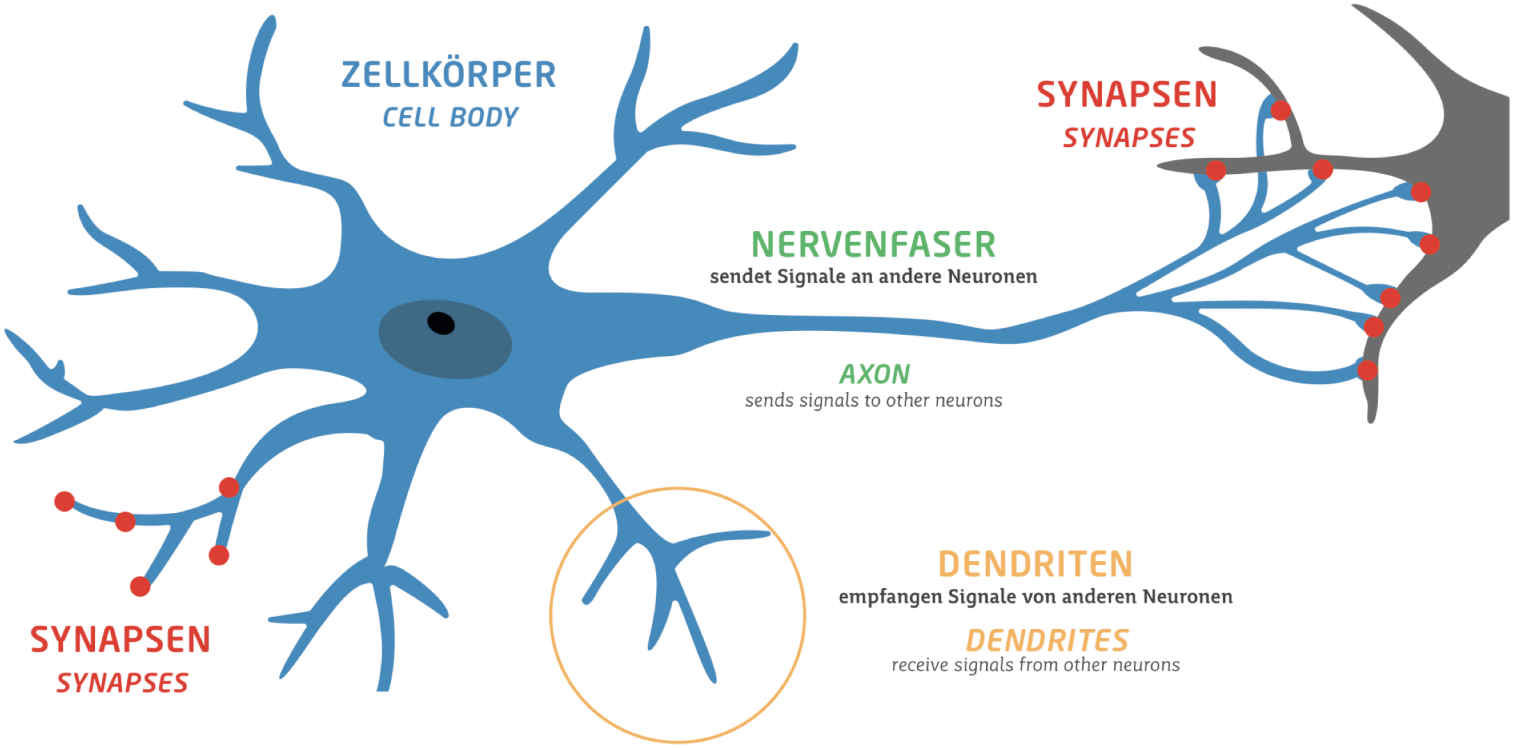
Tiefe Neuronale Netze „Deep Learning“ (DL)

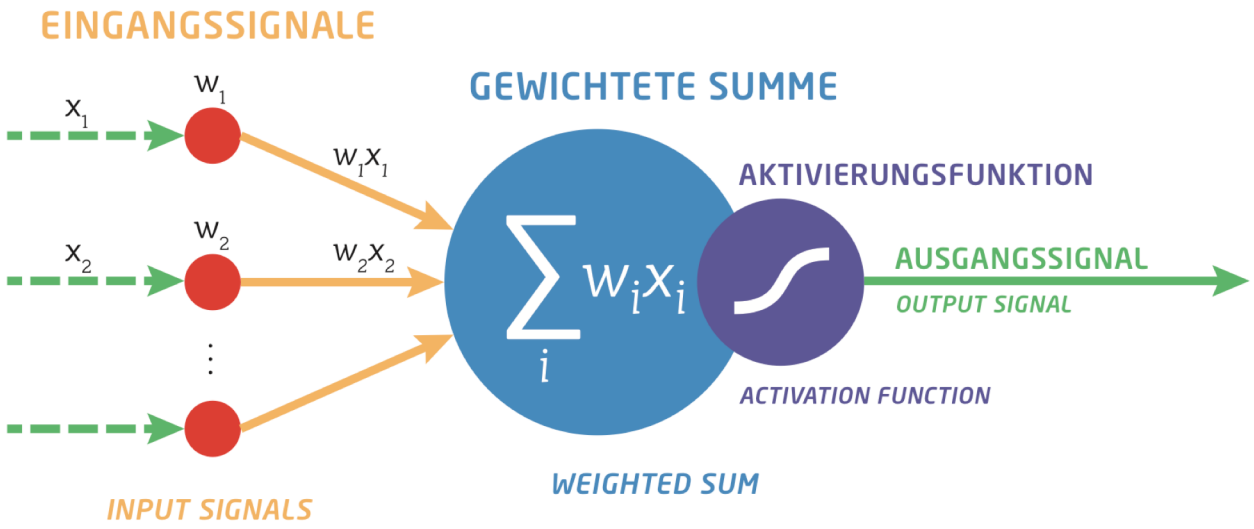
Künstliche Neuronale Netze

Maschinelles Lernen

Künstliche Intelligenz

Künstliche & Biologische Neuronen

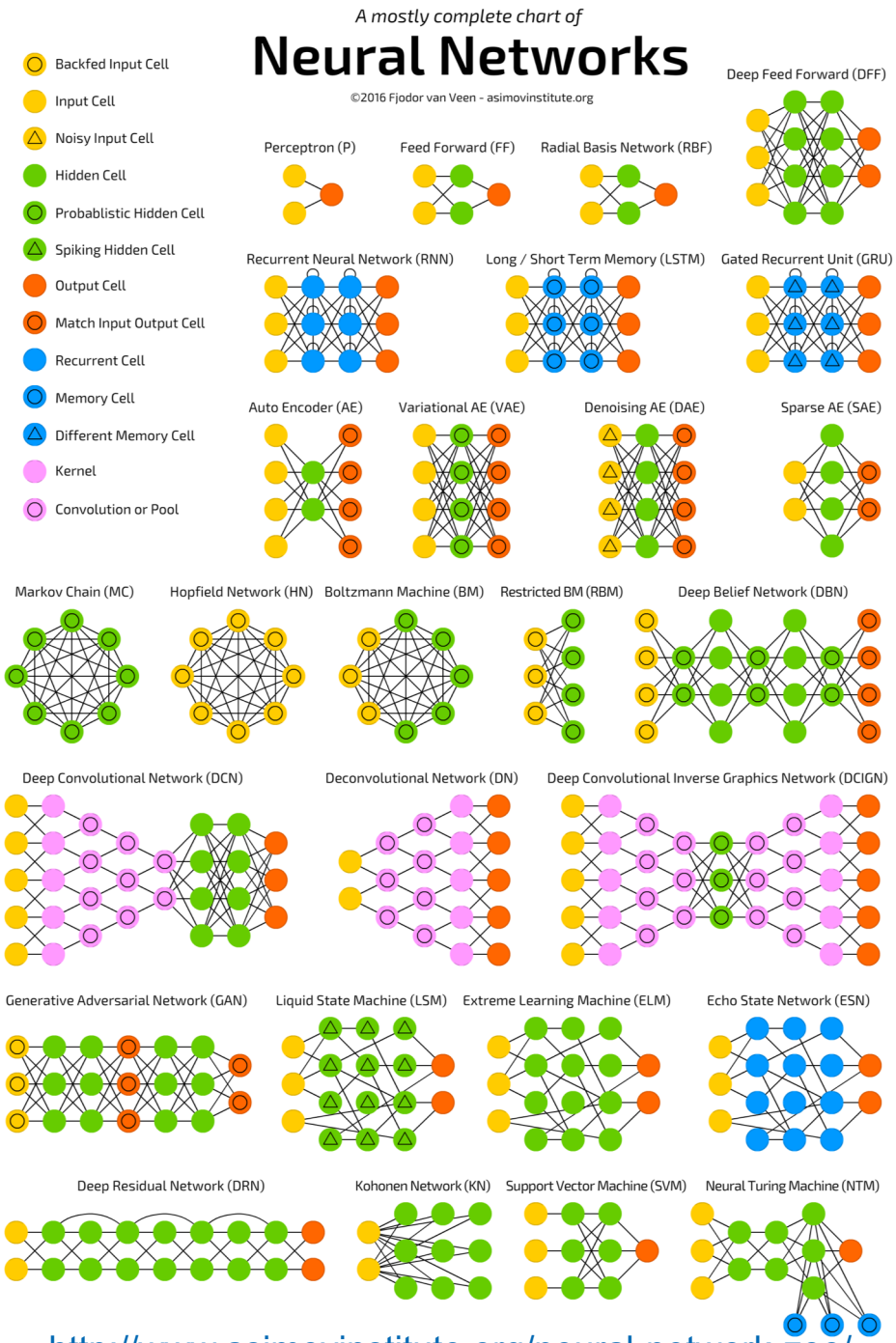




Der Neuronale Netze-Zoo

(nicht vollständig!)

- unterschiedliche Neuronentypen
- unterschiedliche Architekturen
- Gemeinsamkeiten?



http://www.asimovinstitute.org/neural-network-zoo/

Graphentheoretische Grundlagen

Graphentheoretische Grundlagen

Ein (gerichteter) **Graph** ist ein Tupel $G = (V, E)$, bestehend aus einer (endlichen) Menge V von **Knoten** oder **Ecken** und einer (endlichen) Menge $E \subseteq V \times V$ von **Kanten**.

Wir nennen eine Kante $e = (u, v) \in E$ **gerichtet** von Knoten u zu Knoten v.

Sei $G = (V, E)$ ein (gerichteter) Graph und $u \in V$ ein Knoten. Dann werden die Knoten der Menge

$$\text{pred}(u) = \{v \in V \mid (v, u) \in E\}$$

die **Vorgänger** des Knotens u
und die Knoten der Menge

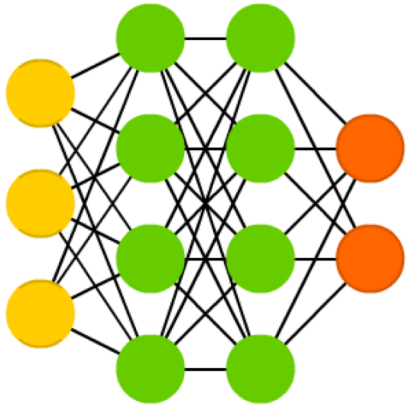
$$\text{succ}(u) = \{v \in V \mid (u, v) \in E\}$$

die **Nachfolger** des Knotens u genannt.

(englische Bezeichnungen: V=Vertices, E=Edges, pred=predecessors, succ=successors)

Allgemeine Netwerk-Definition

Ein (künstliches) **neuronales Netz** ist ein (gerichteter) Graph $G = (U, C)$, dessen Knoten $u \in U$ **Neuronen** oder **Einheiten** und dessen Kanten $c \in C$ **Verbindungen** genannt werden.



Die Menge U der Knoten wird partitioniert in

- die Menge U_{in} der **Eingabeneuronen**,
- die Menge U_{out} der **Ausgabeneuronen**, und
- die Menge U_{hidden} der **versteckten Neuronen**.

Es gilt

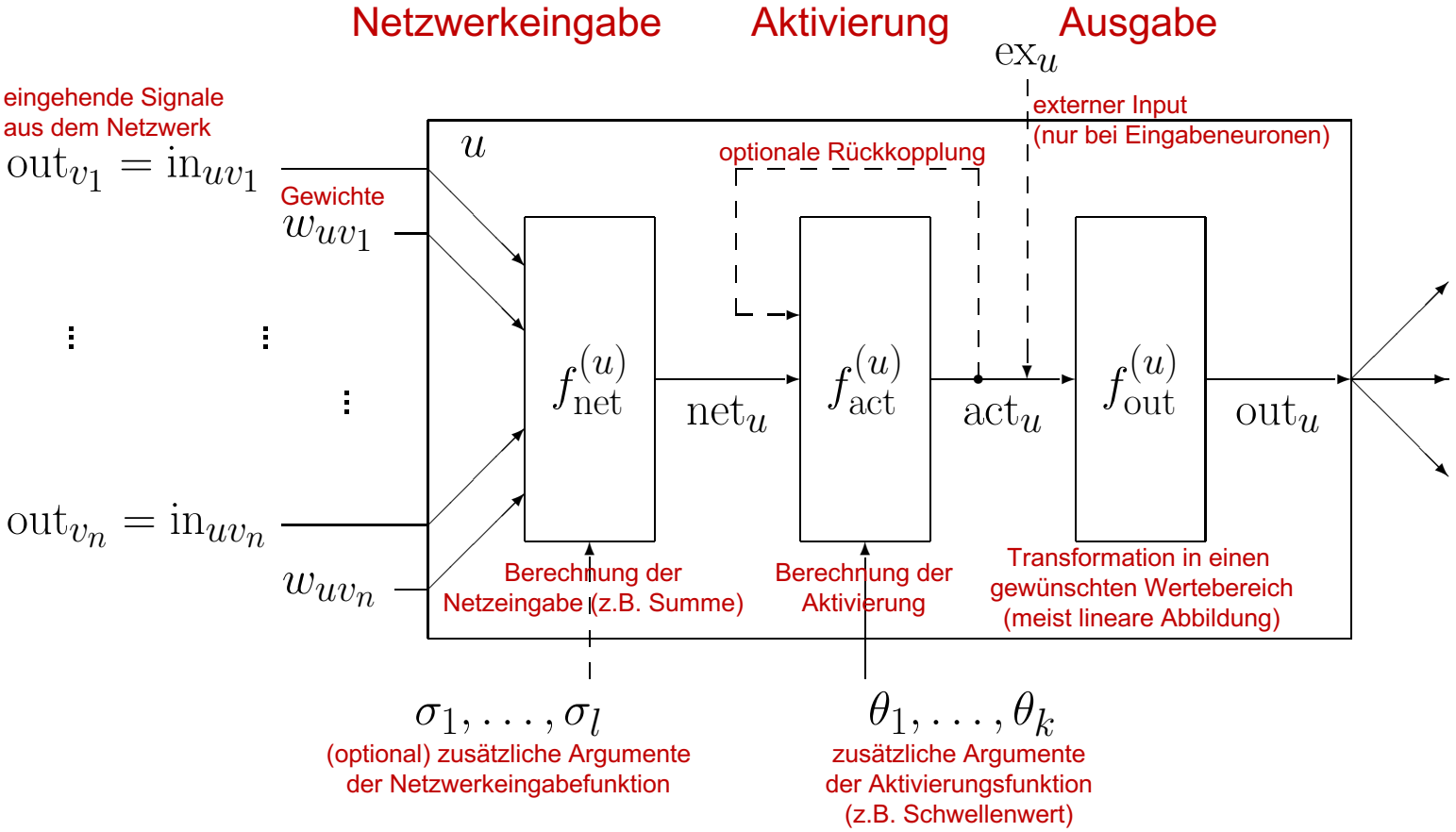
$$U = U_{\text{in}} \cup U_{\text{out}} \cup U_{\text{hidden}},$$

$$U_{\text{in}} \neq \emptyset, \qquad U_{\text{out}} \neq \emptyset, \qquad U_{\text{hidden}} \cap (U_{\text{in}} \cup U_{\text{out}}) = \emptyset.$$

(englische Bezeichnungen: U=Units, C=Connections)

Allgemeines Neuron

Ein verallgemeinertes Neuron verarbeitet numerische Werte



Gradientenabstieg

Allgemeinerer Ansatz:**Gradientenabstieg**.

Notwendige Bedingung: **differenzierbare Aktivierungs- und Ausgabefunktionen**.

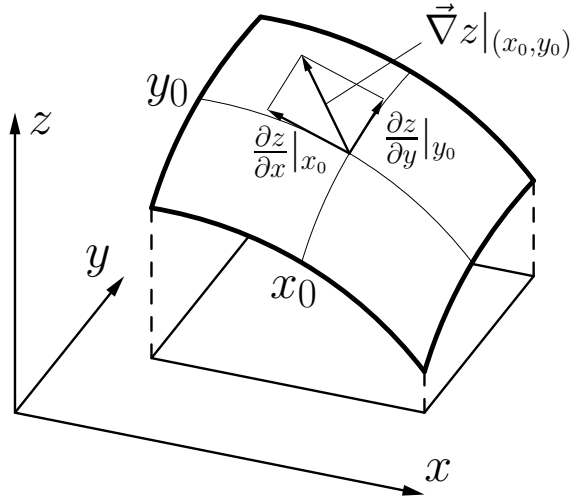


Illustration des Gradienten einer reellwertigen Funktion $z = f(x, y)$ am Punkt (x_0, y_0). Dabei ist $\vec{\nabla} z|_{(x_0,y_0)} = \left(\frac{\partial z}{\partial x}|_{x_0}, \frac{\partial z}{\partial y}|_{y_0}\right)$.

Error Backpropagation

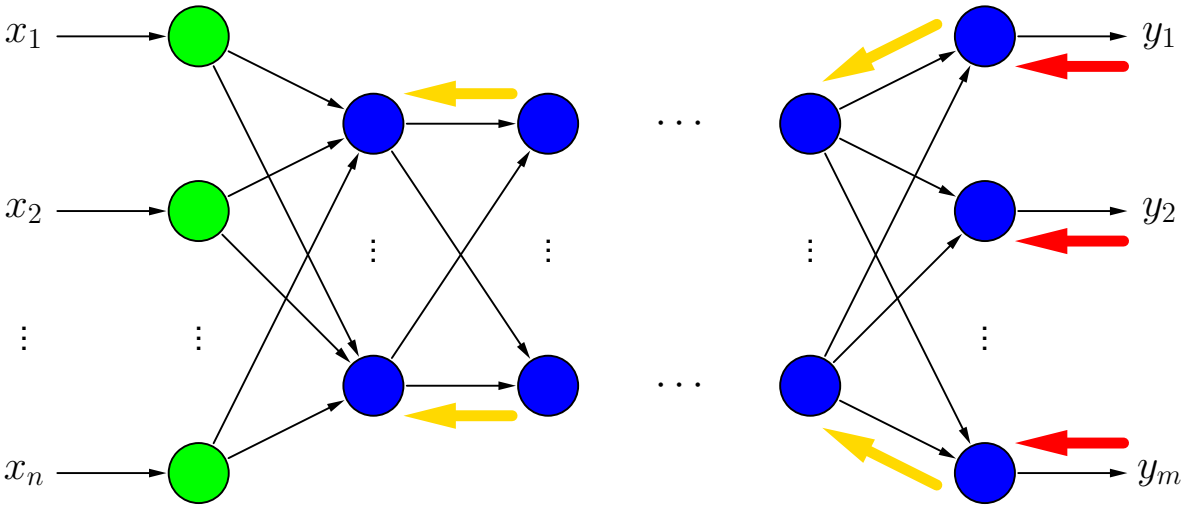
Aktivierungsfunktion: logistisch
Ausgabefunktion: Identität
impliziter Biaswert

$$\forall u \in U_{\text{in}}: \quad \text{out}_u^{(l)} = i_u^{(l)}$$

Setzen der Eingabe

$$\forall u \in U_{\text{hidden}} \cup U_{\text{out}}: \quad \text{out}_u^{(l)} = \left(1 + \exp\left(-\sum_{p \in \text{pred}(u)} w_{up}\, \text{out}_p^{(l)}\right)\right)^{-1}$$

Vorwärtsweitergabe der Eingabe



Fehlerrückübertragung

Fehlerbestimmung

Rückwärts-propagation:

$$\forall u \in U_{\text{hidden}}: \quad \delta_u^{(l)} = \left(\sum_{s \in \text{succ}(u)} \delta_s^{(l)} w_{su}\right) \lambda_u^{(l)}$$

$$\forall u \in U_{\text{out}}: \quad \delta_u^{(l)} = \left(o_u^{(l)} - \text{out}_u^{(l)}\right) \lambda_u^{(l)}$$

Aktivierungs-ableitung:

$$\lambda_u^{(l)} = \text{out}_u^{(l)} \left(1 - \text{out}_u^{(l)}\right)$$

Gewichts-änderung:

$$\Delta w_{up}^{(l)} = \eta\, \delta_u^{(l)}\, \text{out}_p^{(l)}$$

Algorithmus-Skizze (online)

gegeben: MLP mit $G = (U, C)$, Lernrate η, Trainingsbeispiele L_{fixed}

Initialisierung aller Gewichte (Zufallswerte)

wiederhole:

für jedes Trainingsbeispiel $l = (\vec{\imath}^{(l)}, \vec{o}^{(l)}) \in L_{\text{fixed}}$

Eingabe, Vorwärtsberechnung der Aktivierungen und Ausgabe:

$$\forall u \in U_{\text{in}}: \quad \text{out}_u^{(l)} = i_u^{(l)}$$

$$\forall u \in U_{\text{hidden}} \cup U_{\text{out}}: \quad \text{out}_u^{(l)} = \left(1 + \exp\left(-\sum_{p \in \text{pred}(u)} w_{up}\, \text{out}_p^{(l)}\right)\right)^{-1}$$

Fehlerberechnung und Rückübertragung (Backpropagation):

$$\forall u \in U_{\text{out}}: \quad \delta_u^{(l)} = \left(o_u^{(l)} - \text{out}_u^{(l)}\right) \lambda_u^{(l)}$$

$$\forall u \in U_{\text{hidden}}: \quad \delta_u^{(l)} = \left(\sum_{s \in \text{succ}(u)} \delta_s^{(l)} w_{su}\right) \lambda_u^{(l)}$$

mit Ableitung der Aktivierungsfunktion $\lambda_u^{(l)} = \text{out}_u^{(l)} \left(1 - \text{out}_u^{(l)}\right)$

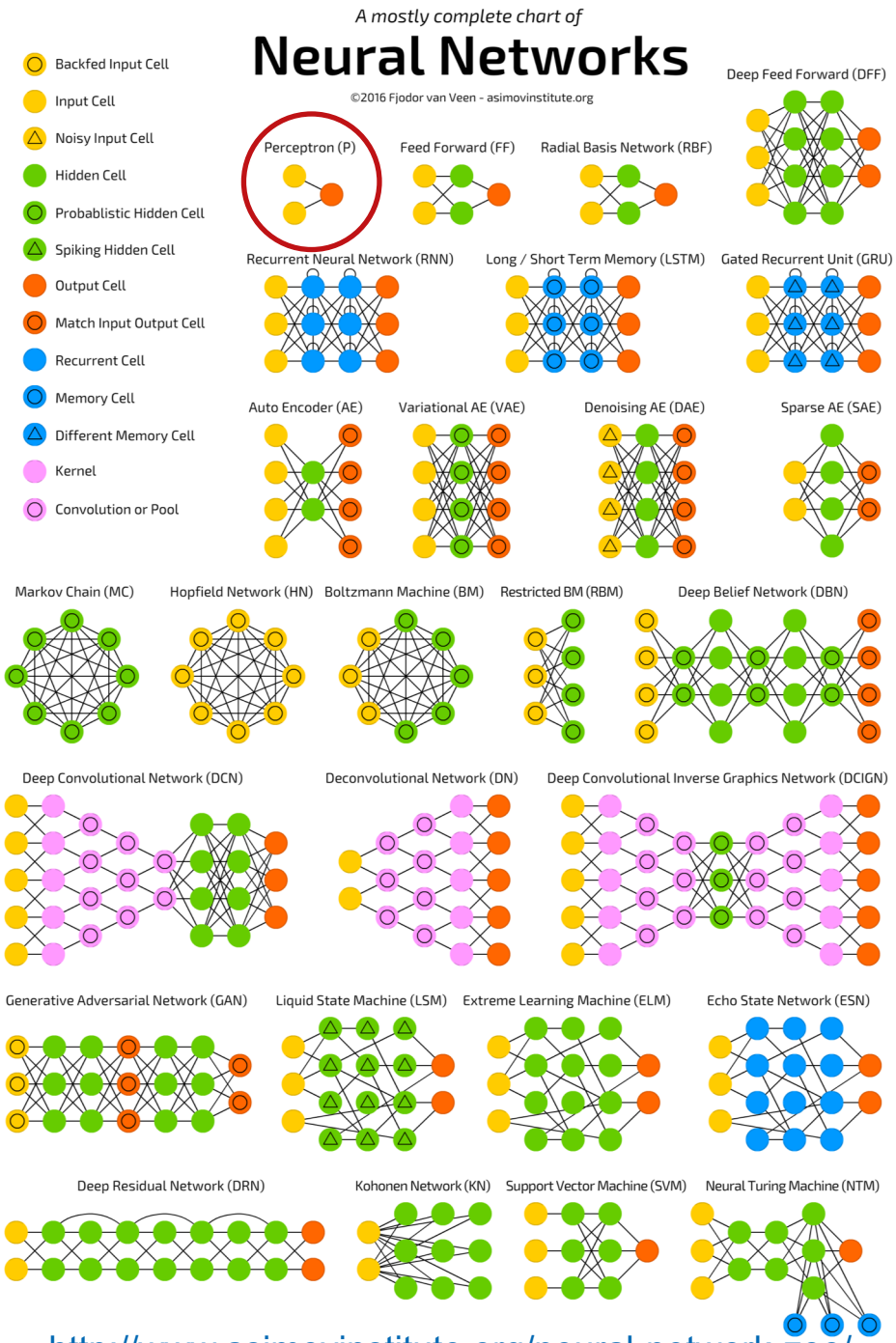
Berechnung der Gewichtsänderung $\Delta w_{up}^{(l)} = \eta\, \delta_u^{(l)}\, \text{out}_p^{(l)}$ und Update

bis Stopkriterium erreicht

Der Neuronale Netze-Zoo

(nicht vollständig!)

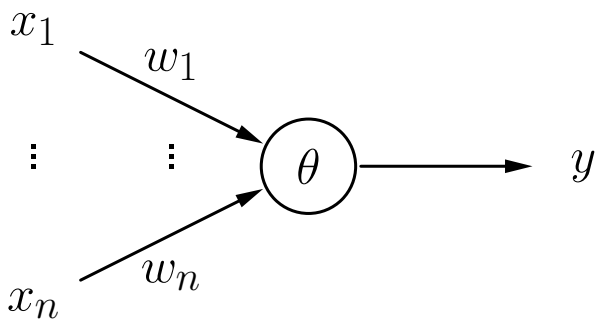
- Schwellenwertelem. (Perceptron)



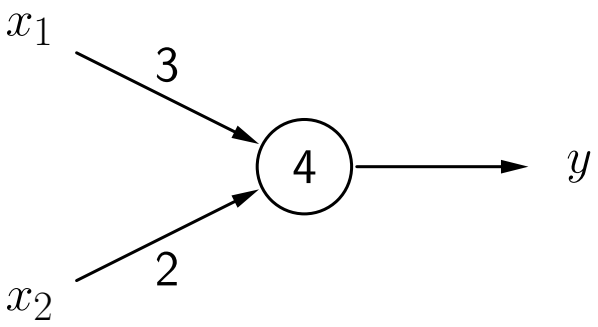
http://www.asimovinstitute.org/neural-network-zoo/

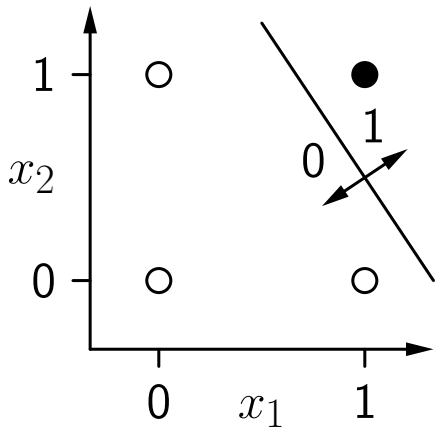
Schwellenwertelement

$$y = \begin{cases} 1, & \text{falls} \quad \vec{x}\vec{w} = \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0, & \text{sonst.} \end{cases}$$



Schwellenwertelement für $x_1 \wedge x_2$.





Delta-Regel

Formale Trainingsregel: Sei $\vec{x} = (x_1, \ldots, x_n)$ ein Eingabevektor eines Schwellenwertelements, o die gewünschte Ausgabe für diesen Eingabevektor, und y die momentane Ausgabe des Schwellenwertelements. Wenn $y \neq o$, dann werden Schwellenwert θ und Gewichtsvektor $\vec{w} = (w_1, \ldots, w_n)$ wie folgt angepasst, um den Fehler zu reduzieren:

$$\begin{aligned} \theta^{(\text{neu})} &= \theta^{(\text{alt})} + \Delta\theta \quad \text{wobei} \quad \Delta\theta &&= -\eta(o - y), \\ \forall i \in \{1, \ldots, n\}: \quad w_i^{(\text{neu})} &= w_i^{(\text{alt})} + \Delta w_i \quad \text{wobei} \quad \Delta w_i &&= \eta(o - y)x_i, \end{aligned}$$

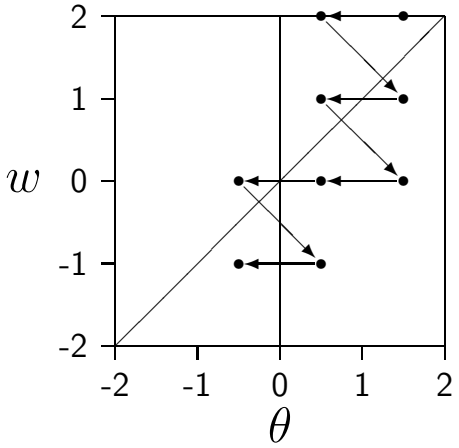
wobei η ein Parameter ist, der **Lernrate** genannt wird. Er bestimmt die Größenordnung der Gewichtsänderungen. Diese Vorgehensweise nennt sich **Delta-Regel** oder **Widrow–Hoff–Procedure** [Widrow and Hoff 1960].

Online-Training: Passe Parameter nach jedem Trainingsmuster an.

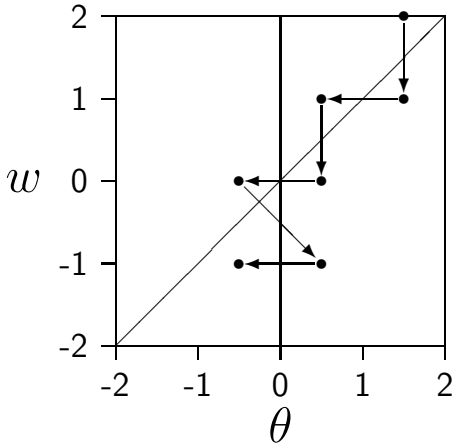
Batch-Training: Passe Parameter am Ende jeder **Epoche** an,
d.h. nach dem Durchlaufen aller Trainingsbeispiele.

Trainieren von Schwellenwertelementen

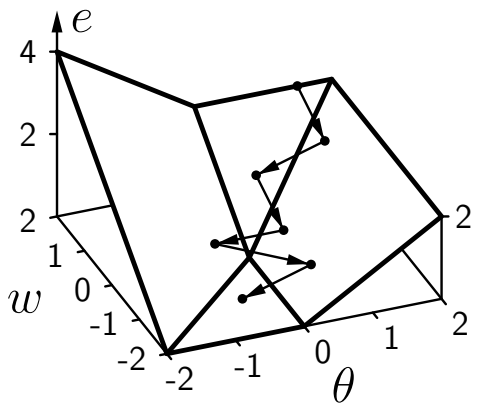
Beispieltrainingsprozedur: Online- und Batch-Training.



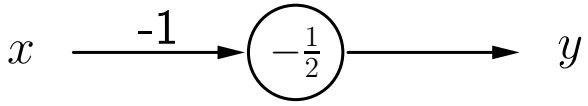
Online-Lernen

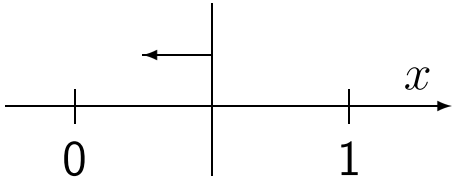


Batch-Lernen



Batch-Lernen

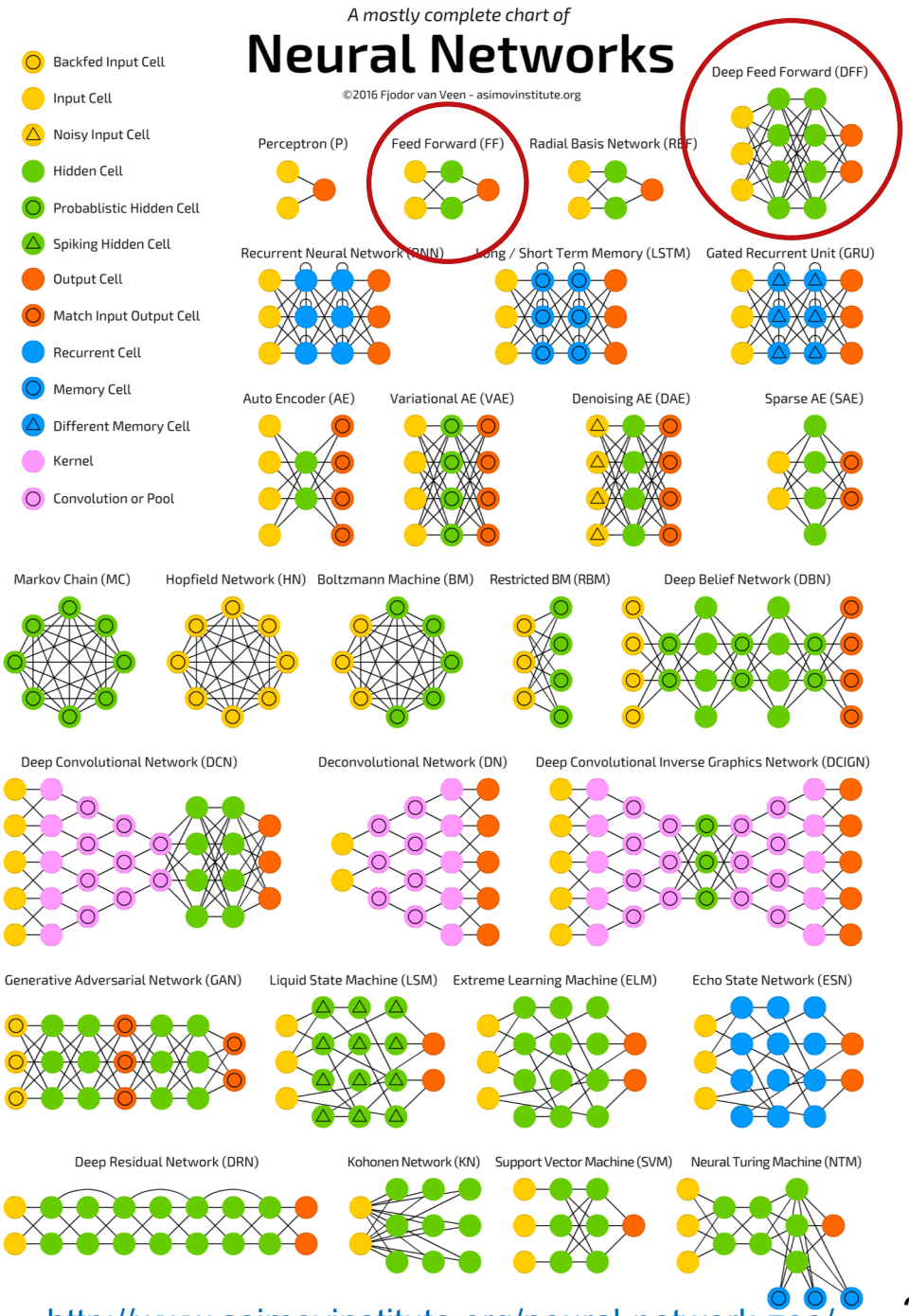




Der Neuronale Netze-Zoo

(nicht vollständig!)

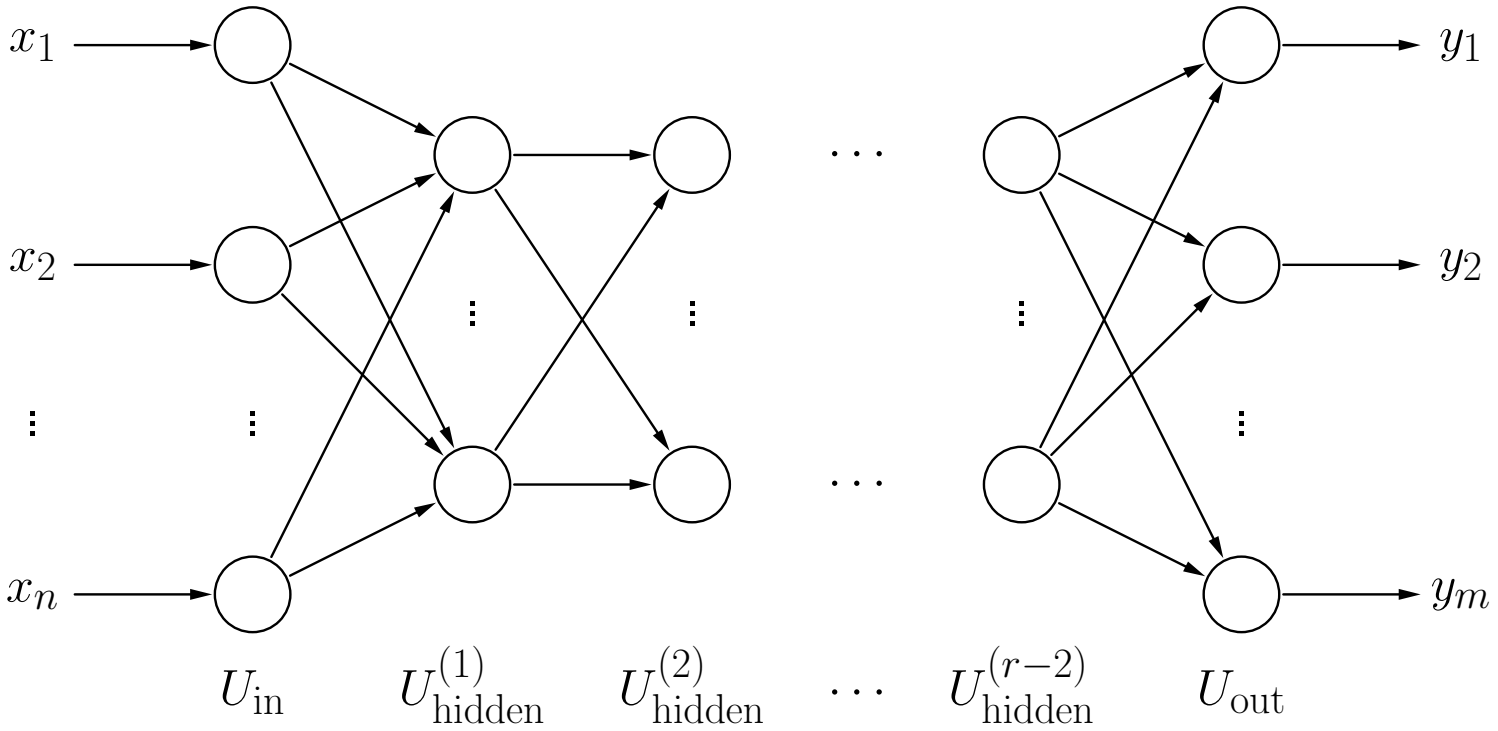
- Schwellenwertelem. (Perceptron)
- MLPs



http://www.asimovinstitute.org/neural-network-zoo/

Mehrschichtige Perzeptren (MLPs)

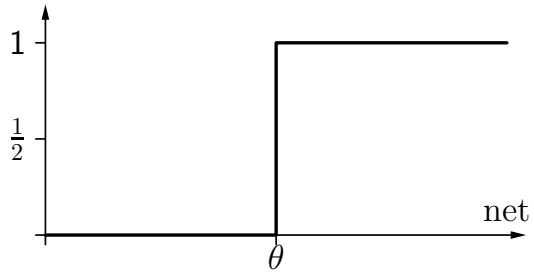
Allgemeine Struktur eines mehrschichtigen Perzeptrons



Sigmoide Aktivierungsfunktionen

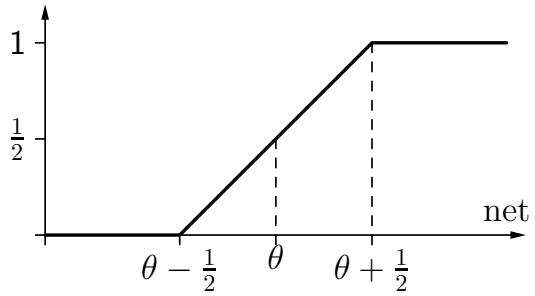
Stufenfunktion:

$$f_{\text{act}}(\text{net}, \theta) = \begin{cases} 1, & \text{falls net} \geq \theta, \\ 0, & \text{sonst.} \end{cases}$$



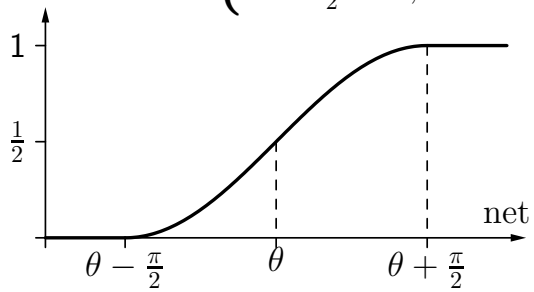
Semilineare Funktion:

$$f_{\text{act}}(\text{net}, \theta) = \begin{cases} 1, & \text{falls net} > \theta + \frac{1}{2}, \\ 0, & \text{falls net} < \theta - \frac{1}{2}, \\ (\text{net} - \theta) + \frac{1}{2}, & \text{sonst.} \end{cases}$$



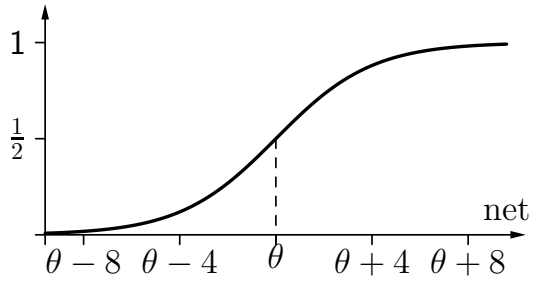
Sinus bis Sättigung:

$$f_{\text{act}}(\text{net}, \theta) = \begin{cases} 1, & \text{falls net} > \theta + \frac{\pi}{2}, \\ 0, & \text{falls net} < \theta - \frac{\pi}{2}, \\ \frac{\sin(\text{net} - \theta) + 1}{2}, & \text{sonst.} \end{cases}$$



Logistische Funktion:

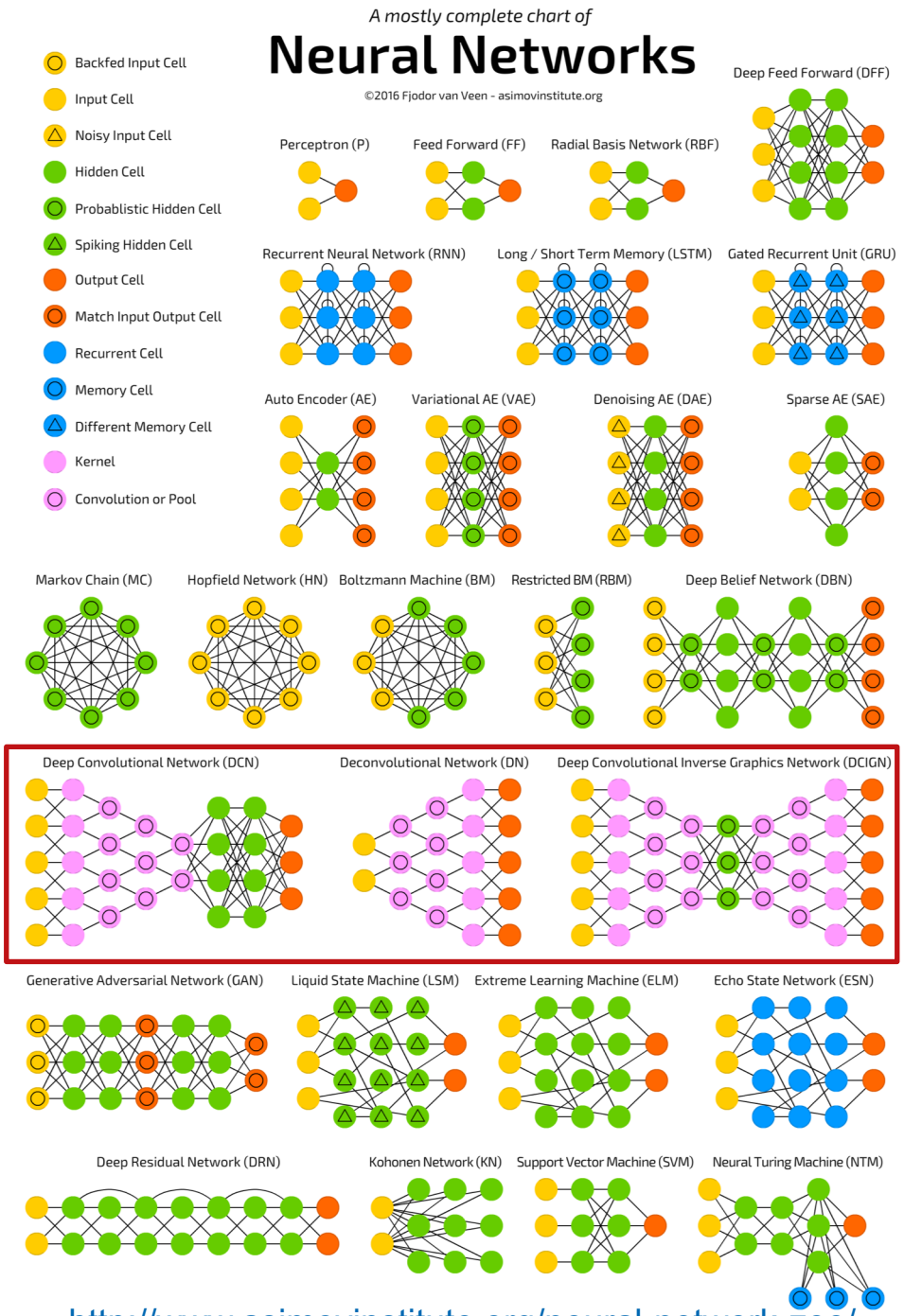
$$f_{\text{act}}(\text{net}, \theta) = \frac{1}{1 + e^{-(\text{net} - \theta)}}$$



Der Neuronale Netze-Zoo

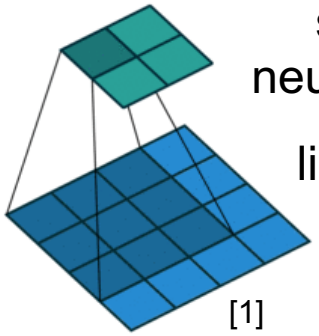
(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs



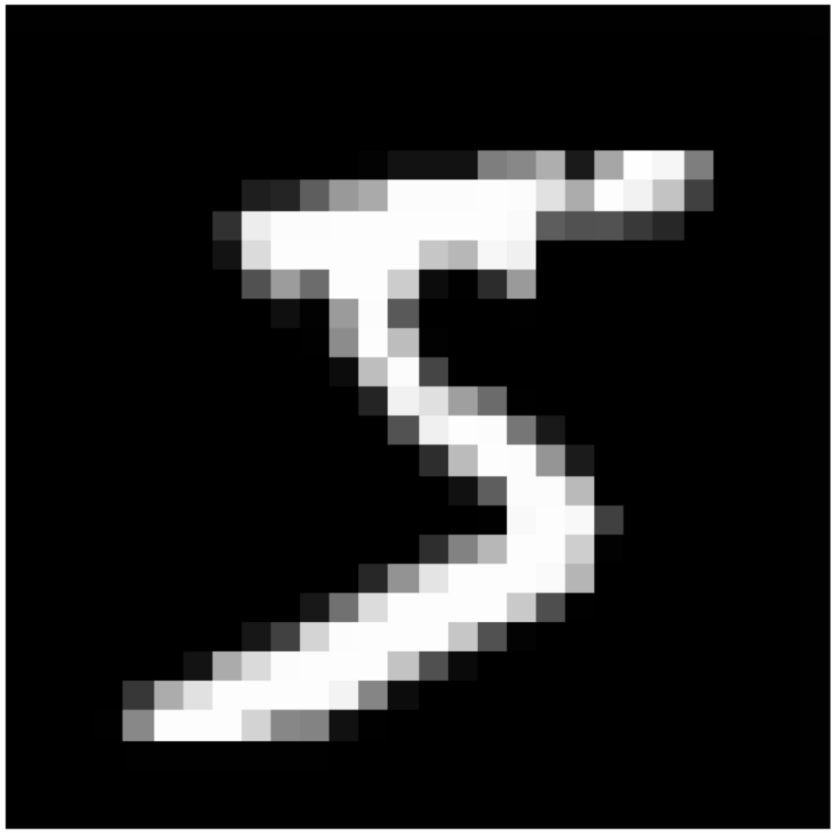
http://www.asimovinstitute.org/neural-network-zoo/

Convolutional Neural Nets (CNNs)



specialized group of neurons in visual cortex

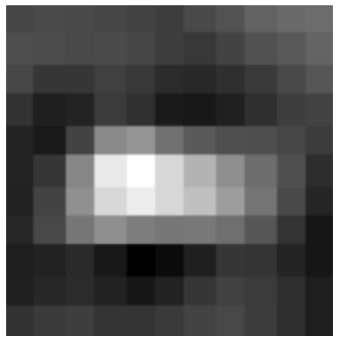
limited & overlapping receptive fields with same filter



example from MNIST dataset
http://yann.lecun.com/exdb/mnist/

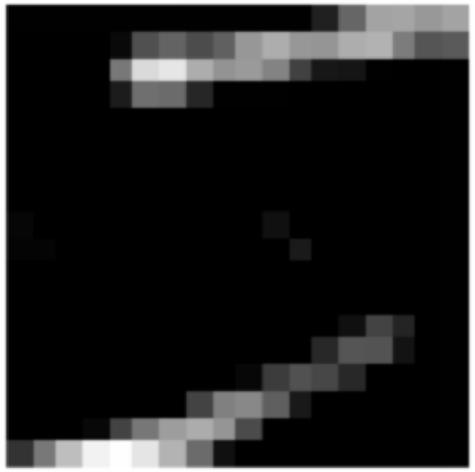
2D input

×



learnable filter (feature)

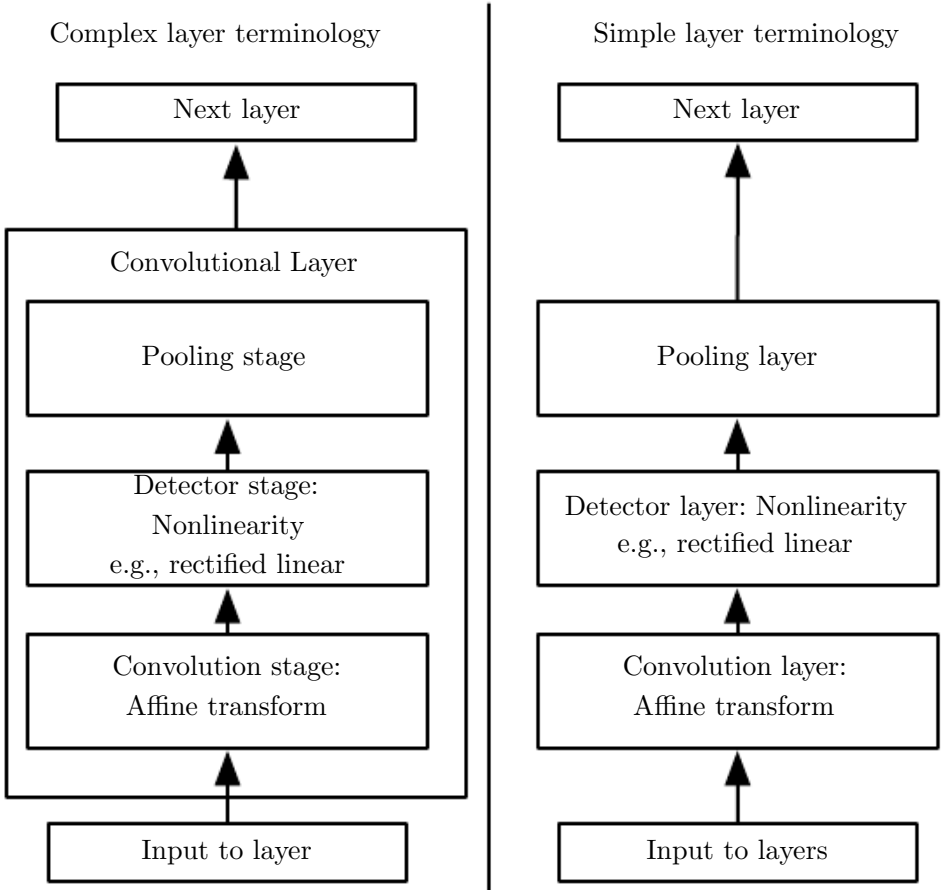
=

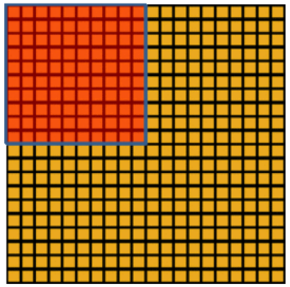


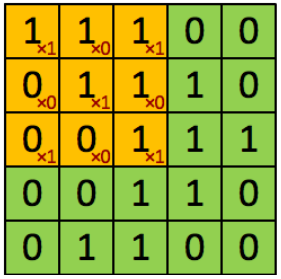
2D output (feature map)

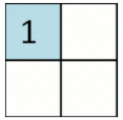
[1] http://deeplearning.net/software/theano/tutorial/

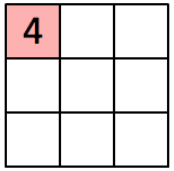
Convolutional Layers







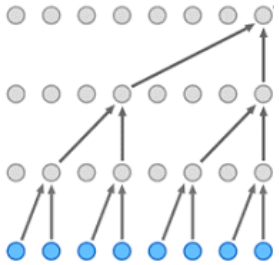




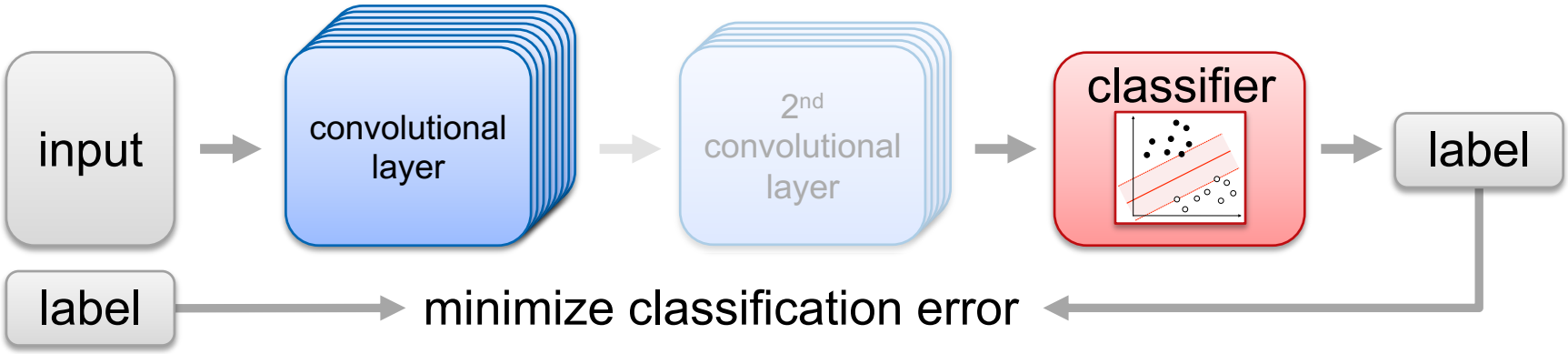
design choices
(hyper params)

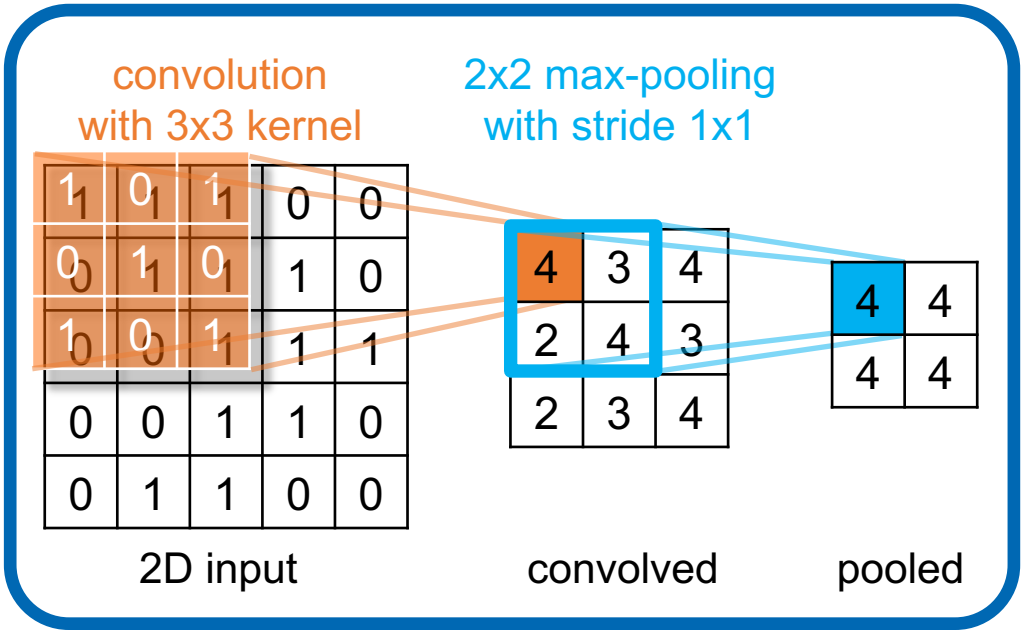
- pool shape
- pool op
- pool stride
- non-linearity
- #kernels
- kernel shape
- kernel stride
- biases?
- padding mode

variant:
dilated
convolution



Convolutional Neural Nets (CNNs)

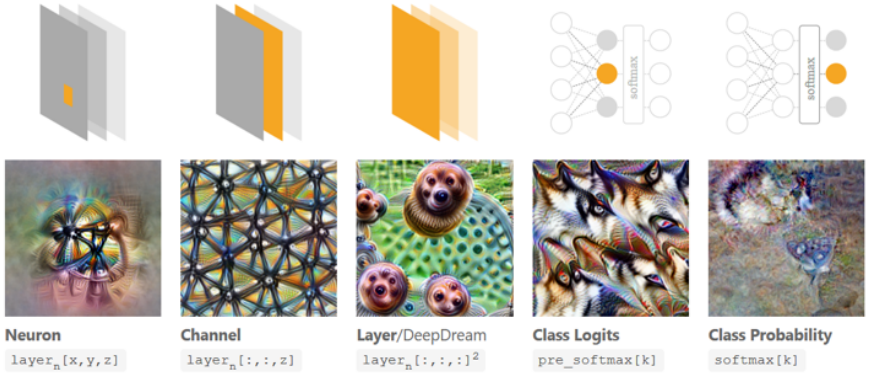




- **local connectivity**
- **parameter sharing**
- translation **equivariance**
- involves non-linear transform (activation function) after conv.
- pool size controls amount of invariance to input translations
- stride (step size) controls non-linear sub-sampling

Introspection

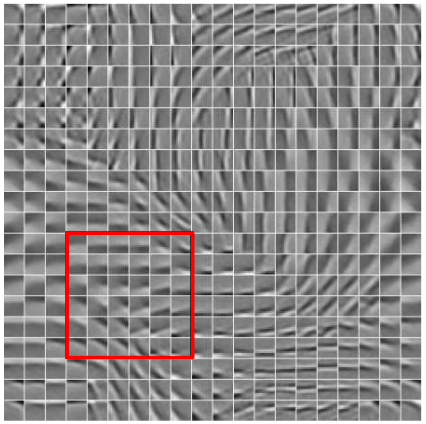
feature visualization (optimize input)



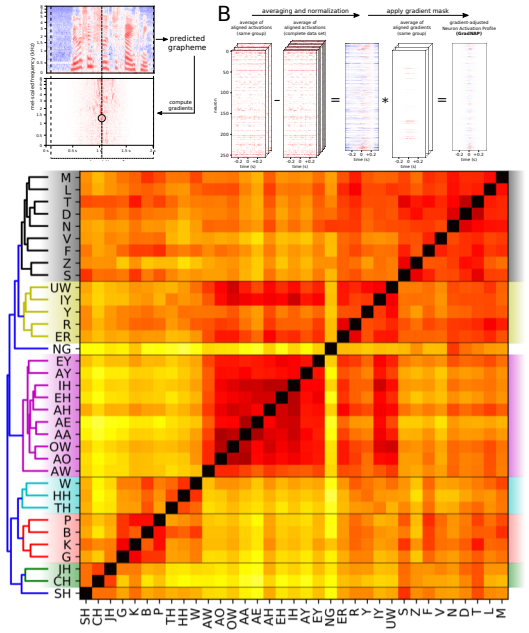
relevance / saliency analysis (for given input)

(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat'

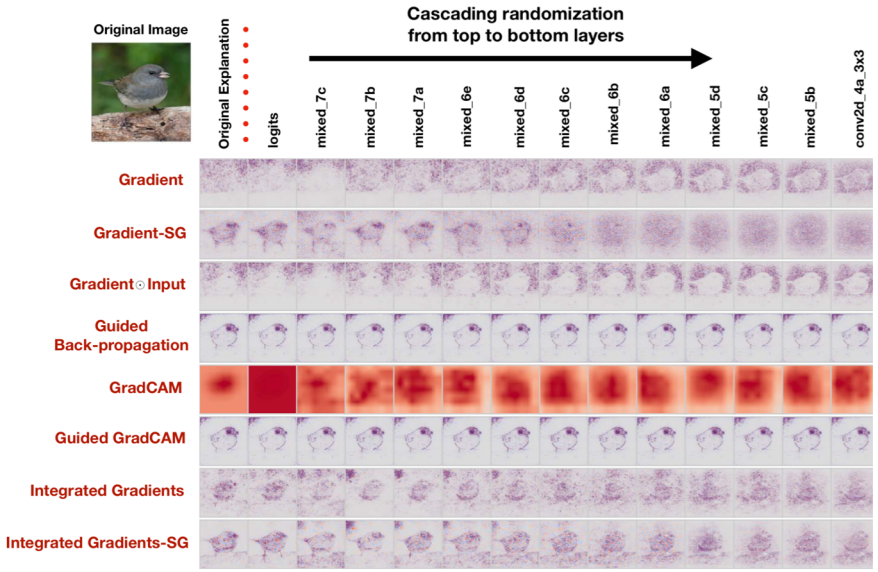
feature topography
(improve interpretability)



neuron activation profiles



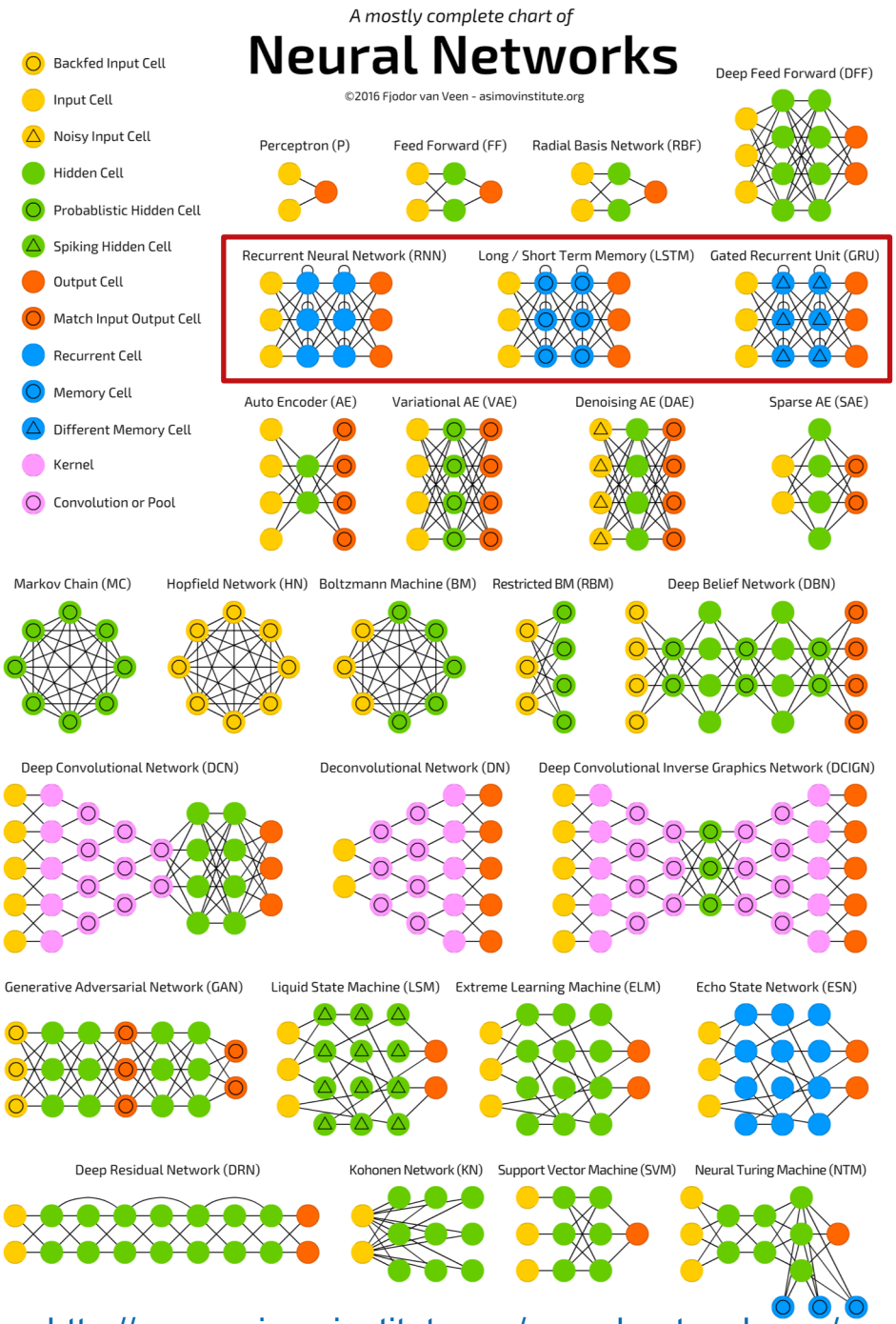
=> sanity checks!



Der Neuronale Netze-Zoo

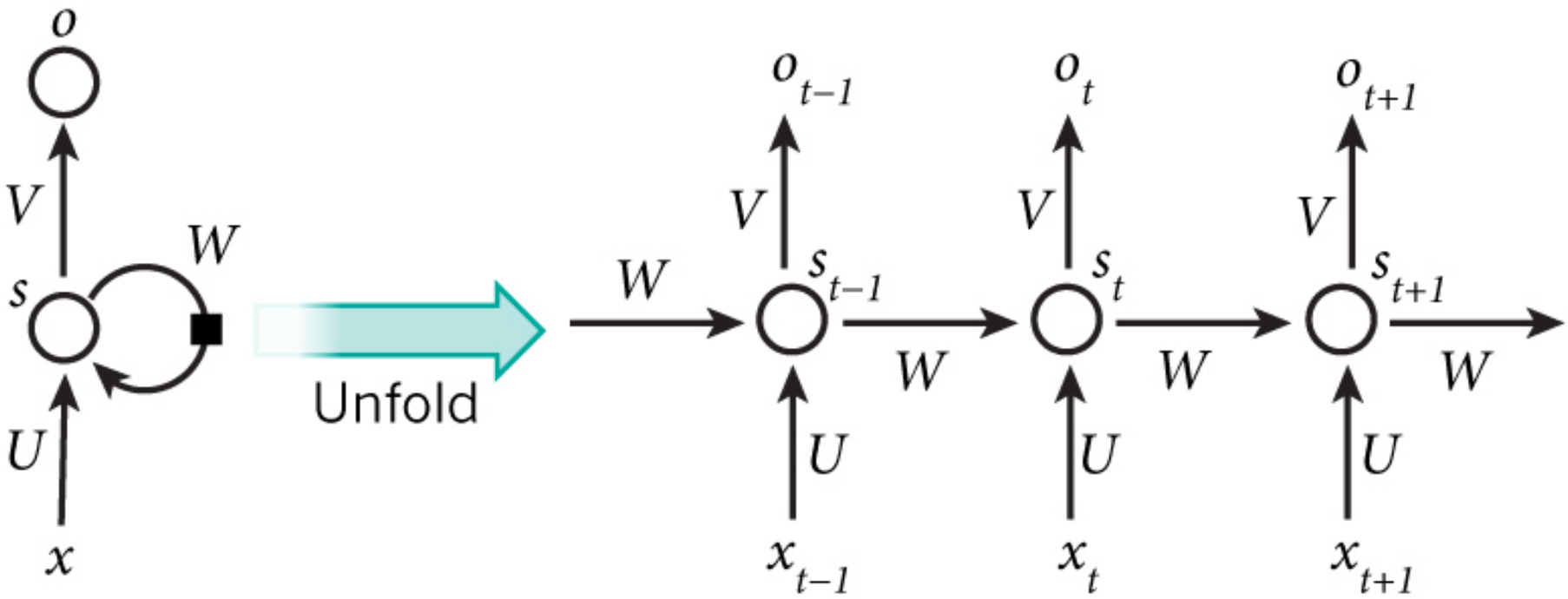
(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs



http://www.asimovinstitute.org/neural-network-zoo/

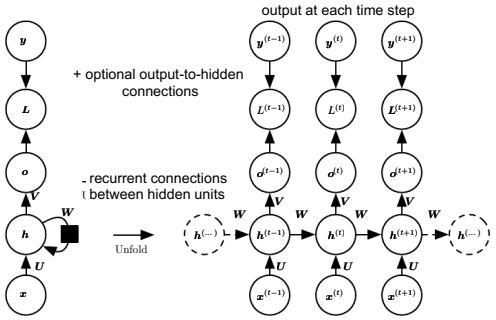
Recurrent Neural Nets



LeCun, Bengio & Hinton. ”Deep Learning.” *nature* 521.7553 (2015)

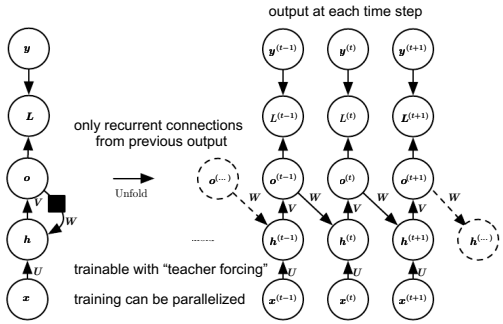
recursive networks

sequence to sequence (same length)



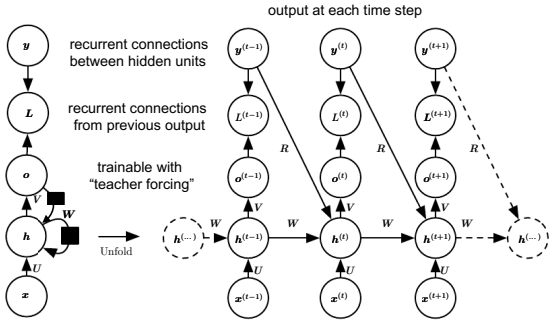
can compute any function computable by a Turing machine (universal function approximator)

sequence to sequence (same length)



lacks important information from past unless o is very high-dimensional & rich

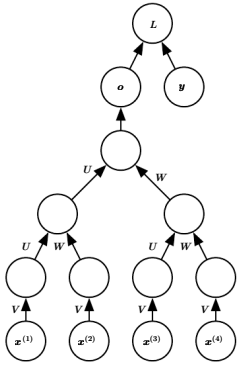
sequence to sequence (same length)



can model arbitrary distribution over sequences of y given sequences of x

sequence-to-sequence

complex structure to fixed-size vector



recursive neural network

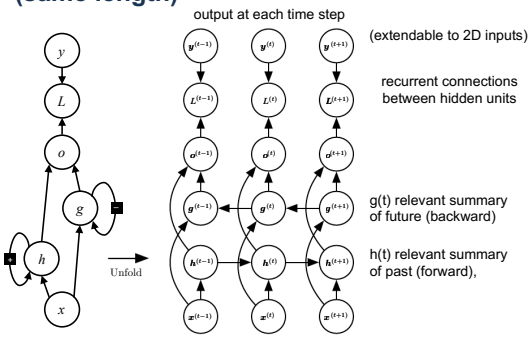
generalization of RNNs

computational graph (given from external tool such as parser) structured as deep tree

generalization

RNNs

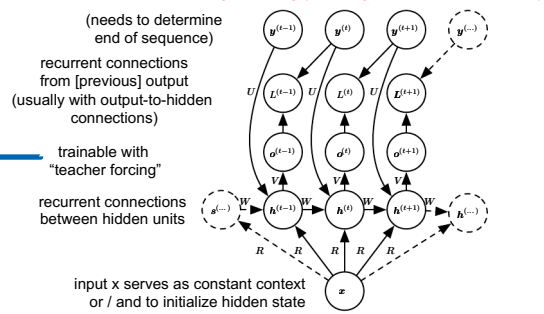
bi-directional sequence to sequence (same length)



can model dependencies on both the past and the future

bi-directional

fixed-size ("context") vector to sequence

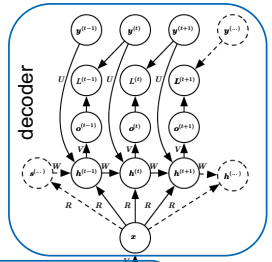


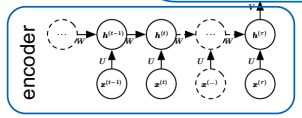
decoder (writer): generate output sequence from hidden state (= decoder part of encoder-decoder architecture)

decoder

sequence to sequence (variable length)

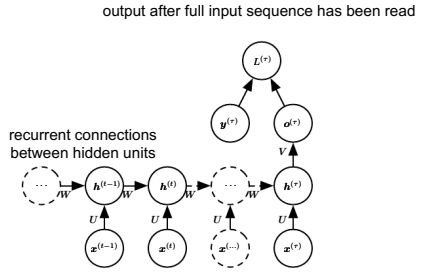
encoder-decoder architecture





encoder-decoder

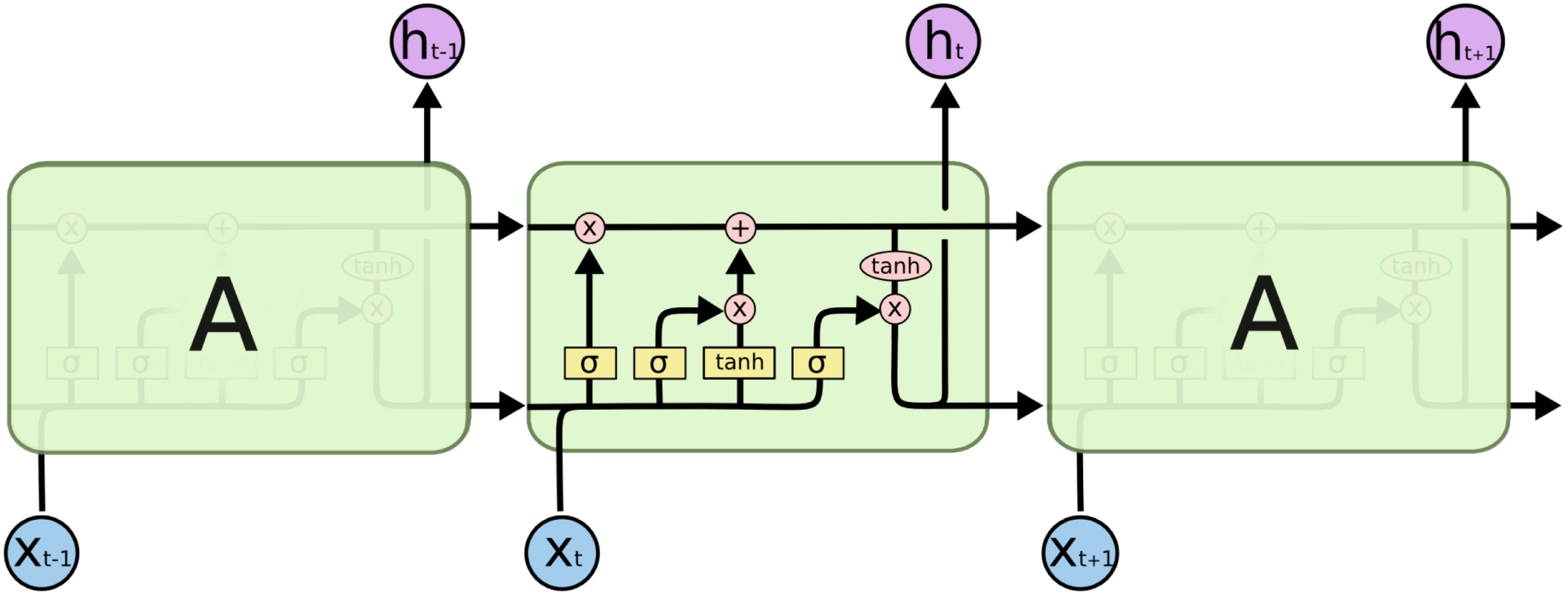
sequence to fixed-size vector



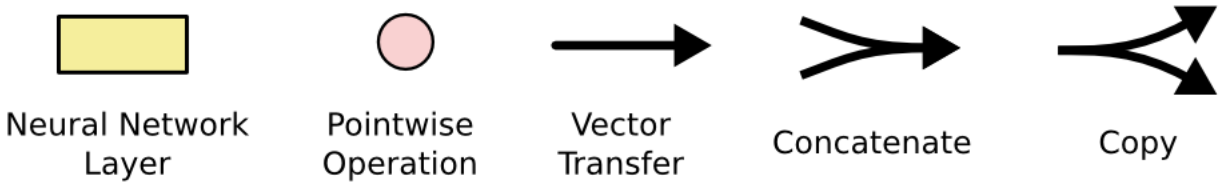
encoder (reader): read input sequence, generate hidden state (= encoder part of encoder-decoder architecture)

encoder

LSTM



The repeating module in an LSTM contains four interacting layers.



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNNs

- work well for sequential data
  - time series (with low sampling rate)
  - texts (translation, discourse, sentiment, ...)
- support variable-length input
  - including long-term dependencies
- are hard to parallelize

Beispiel: CNN+RNN

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
 A person riding a motorcycle on a dirt road.	 Two dogs play in the grass.	 A skateboarder does a trick on a ramp.	 A dog is jumping to catch a frisbee.
 A group of young people playing a game of frisbee.	 Two hockey players are fighting over the puck.	 A little girl in a pink hat is blowing bubbles.	 A refrigerator filled with lots of food and drinks.
 A herd of elephants walking across a dry grass field.	 A close up of a cat laying on a couch.	 A red motorcorcycle parked on the side of the road.	 A yellow school bus parked in a parking lot.

https://arxiv.org/abs/1411.4555

Beispiel: CNN+RNN

What vegetable is on the plate?
Neural Net: broccoli
Ground Truth: broccoli

What color are the shoes on the person's feet ?
Neural Net: brown
Ground Truth: brown

How many school busses are there?
Neural Net: 2
Ground Truth: 2

What sport is this?
Neural Net: baseball
Ground Truth: baseball

What is on top of the refrigerator?
Neural Net: magnets
Ground Truth: cereal

What uniform is she wearing?
Neural Net: shorts
Ground Truth: girl scout

What is the table number?
Neural Net: 4
Ground Truth:40

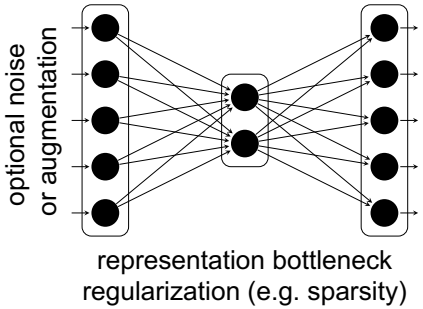
What are people sitting under in the back?
Neural Net: bench
Ground Truth: tent

https://avisingh599.github.io/deeplearning/visual-qa/

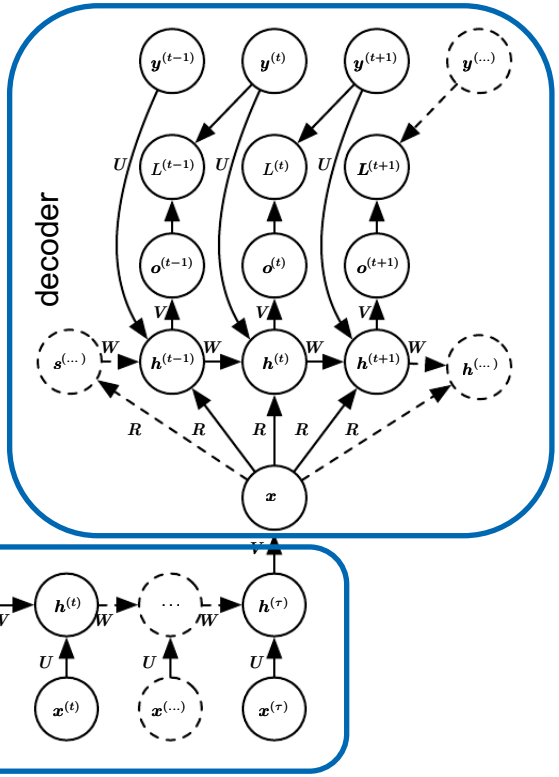
Autoencoders

structure: encoder + decoder
- arbitrarily complex
- often symmetrical

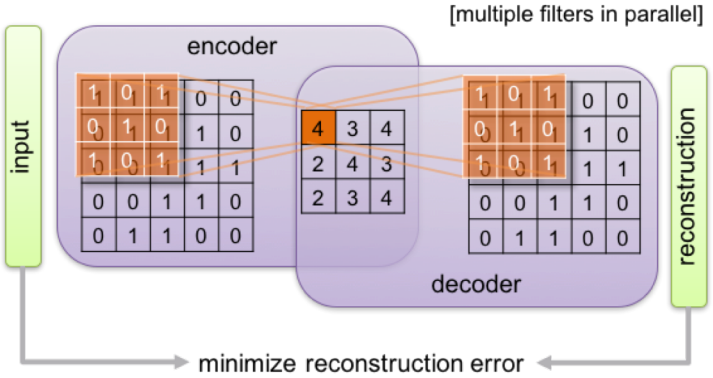
objective: reconstruct inputs
- often under constraints (capacity, sparsity etc.)



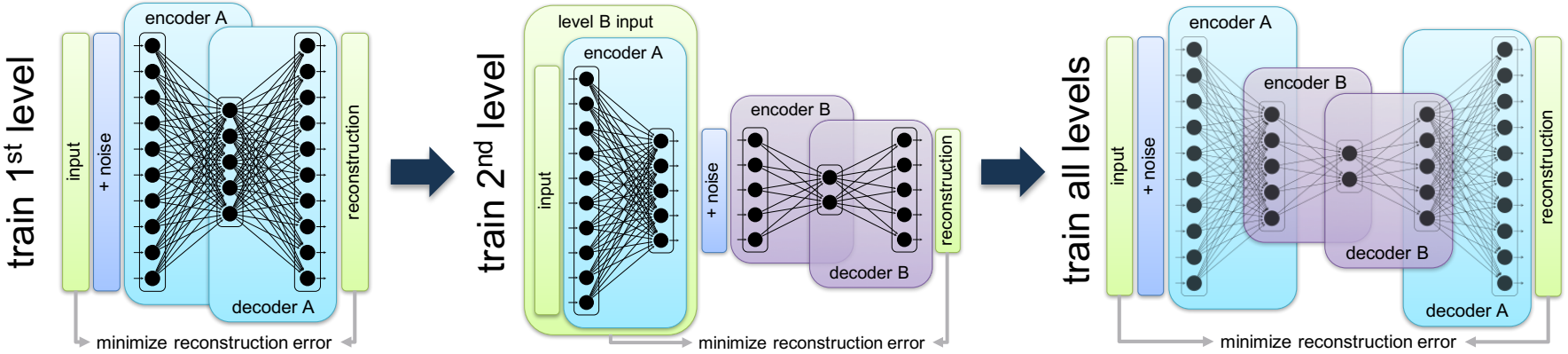
Recurrent Autoencoder



Convolutional Autoencoder



Stacked (Denoising) Autoencoder

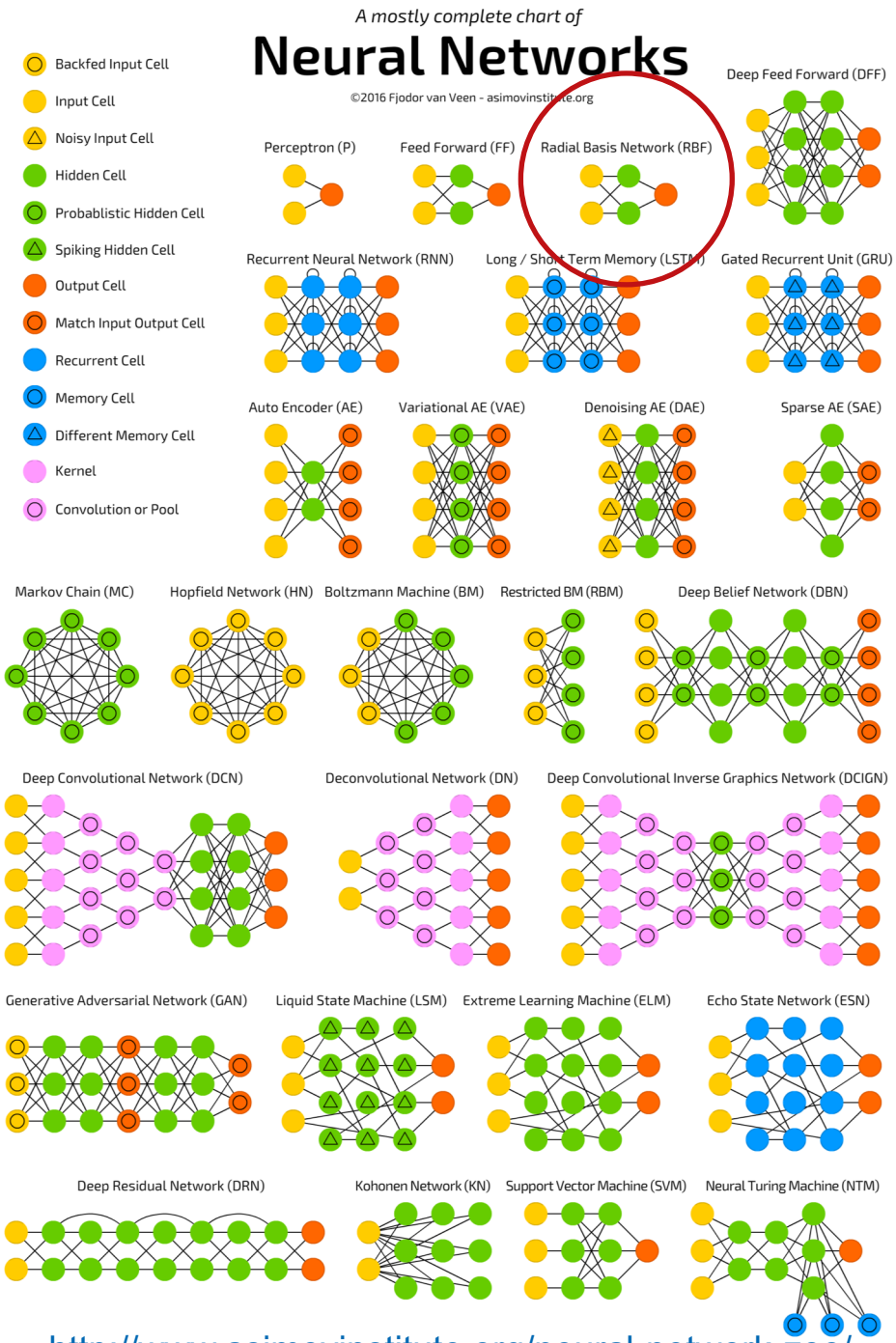


Abstandsbasierte Netze

Der Neuronale Netze-Zoo

(nicht vollständig!)

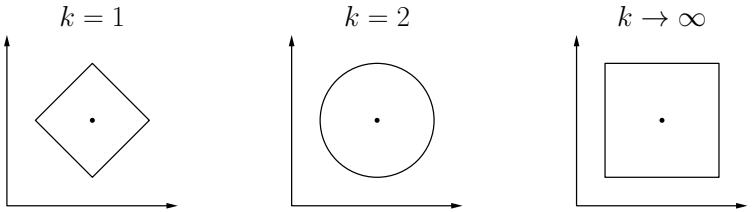
- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze



http://www.asimovinstitute.org/neural-network-zoo/

RBF-Netze

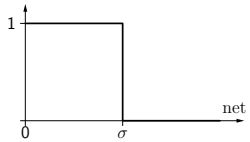
f_{net}: Abstandsfunktion



f_{act}: radiale Aktivierungsfunktion

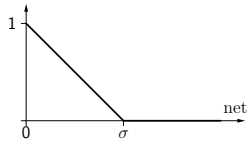
Rechteckfunktion:

$$f_{act}(net, \sigma) = \begin{cases} 0, & \text{falls } net > \sigma, \\ 1, & \text{sonst.} \end{cases}$$



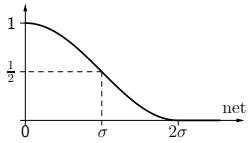
Dreiecksfunktion:

$$f_{act}(net, \sigma) = \begin{cases} 0, & \text{falls } net > \sigma, \\ 1 - \frac{net}{\sigma}, & \text{sonst.} \end{cases}$$



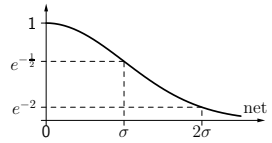
Kosinus bis Null:

$$f_{act}(net, \sigma) = \begin{cases} 0, & \text{falls } net > 2\sigma, \\ \frac{\cos(\frac{\pi}{2\sigma} net)+1}{2}, & \text{sonst.} \end{cases}$$

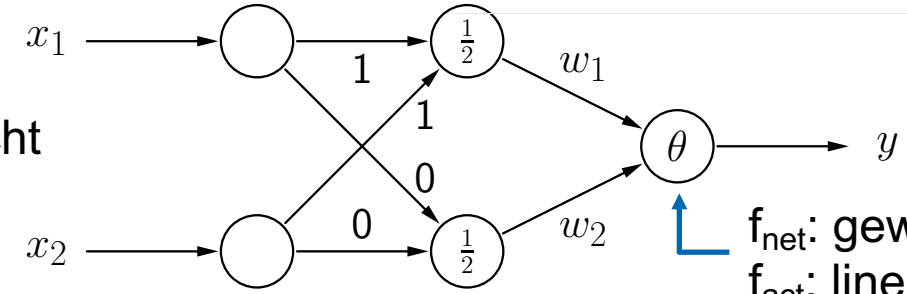


Gaußsche Funktion:

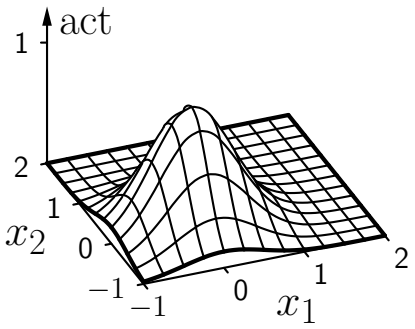
$$f_{act}(net, \sigma) = e^{-\frac{net^2}{2\sigma^2}}$$



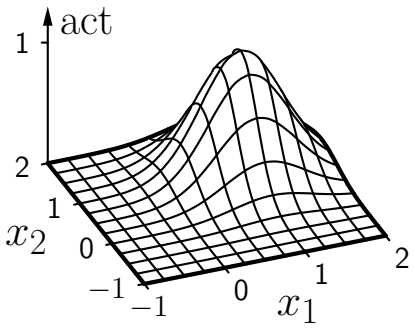
nur 1 versteckte Schicht



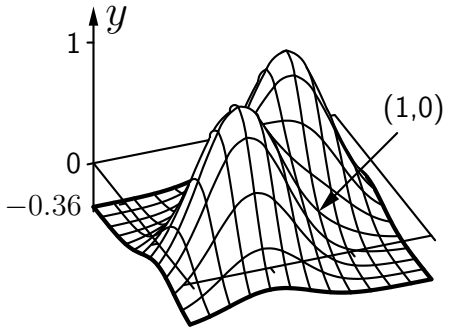
f_{net}: gewichtete Summe
f_{act}: lineare Funktion



basis function (0,0)



basis function (1,1)

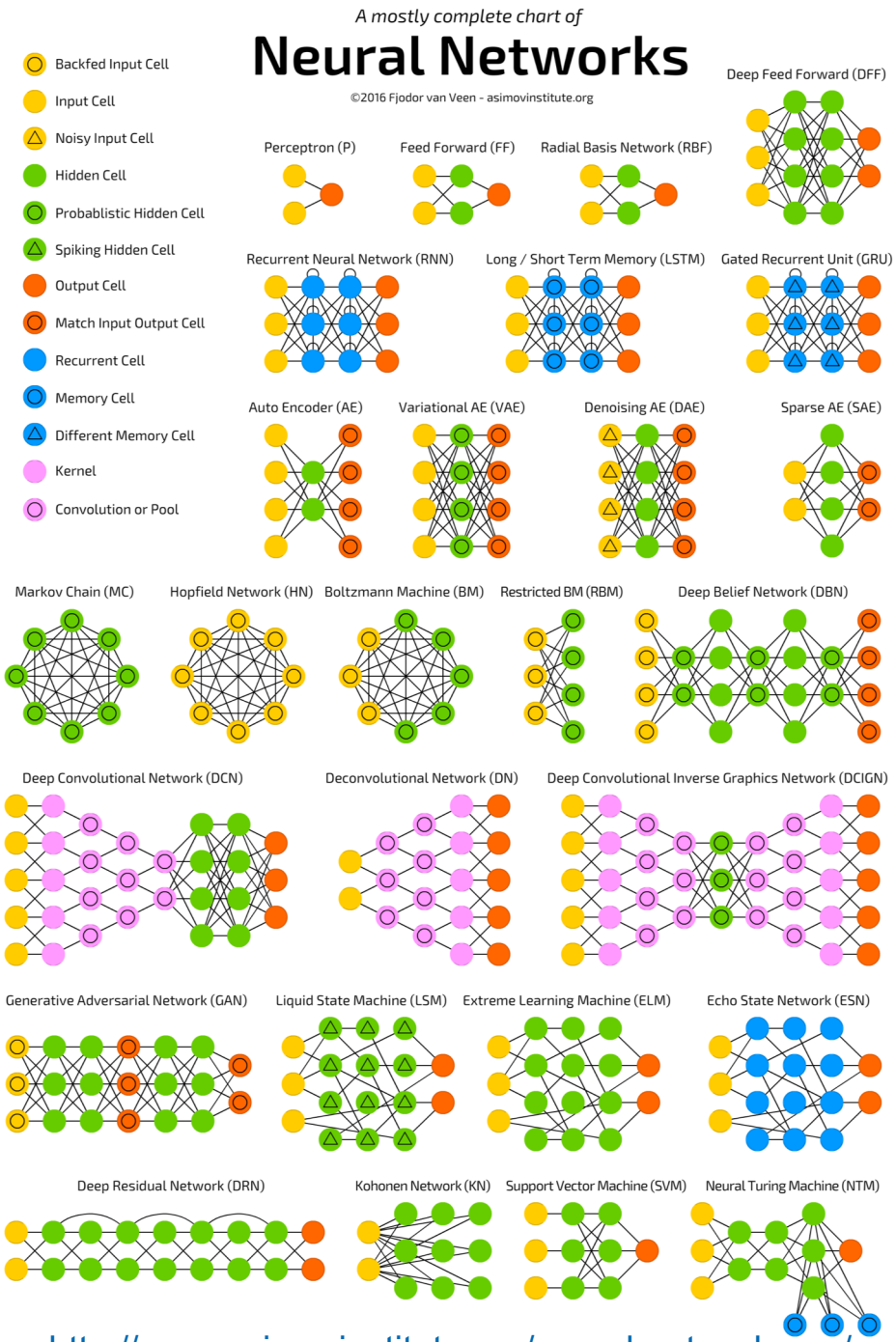


output

Der Neuronale Netze-Zoo

(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze
- LVQ

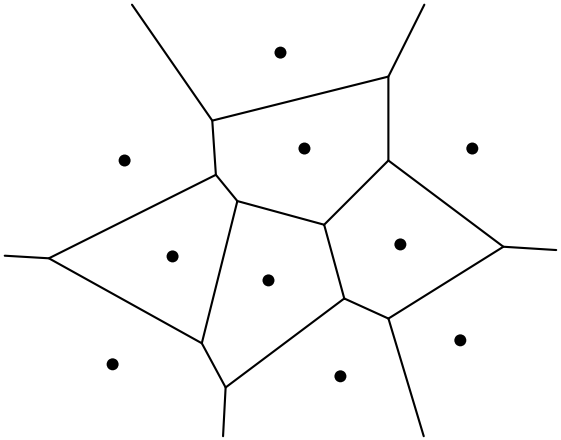


http://www.asimovinstitute.org/neural-network-zoo/

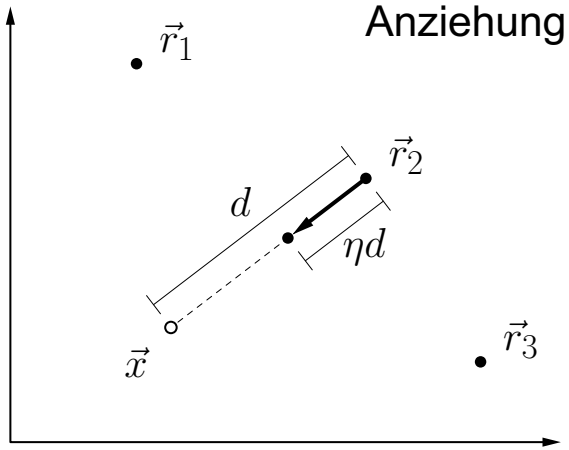
LVQ

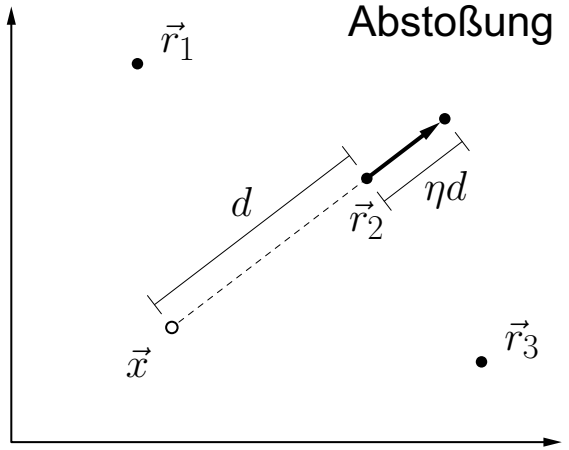
- wie RBF-Netz
  ohne Ausgabeschicht
- Winner-Takes-All
  Ausgabefunktion

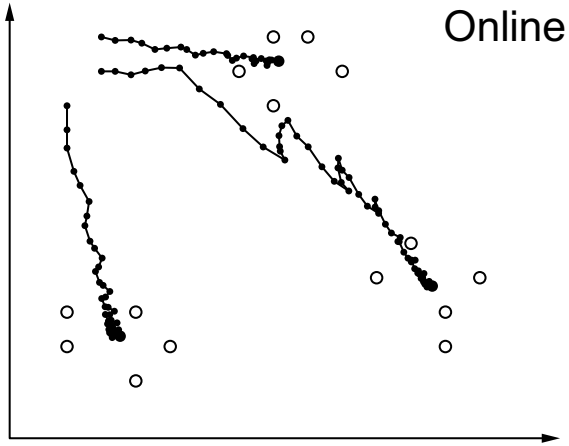
Voronoi-Diagramm
=> Quantisierung

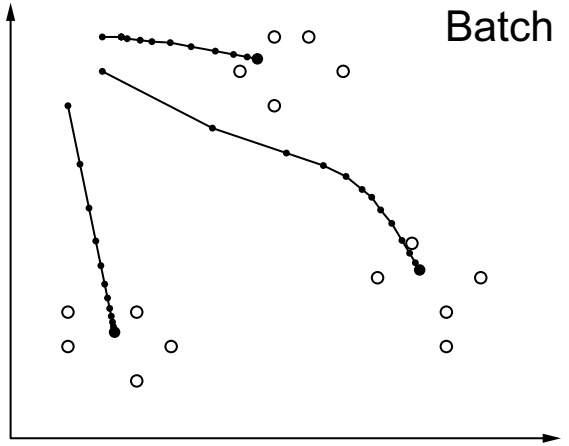


Anpassung der Refenrenzvektoren





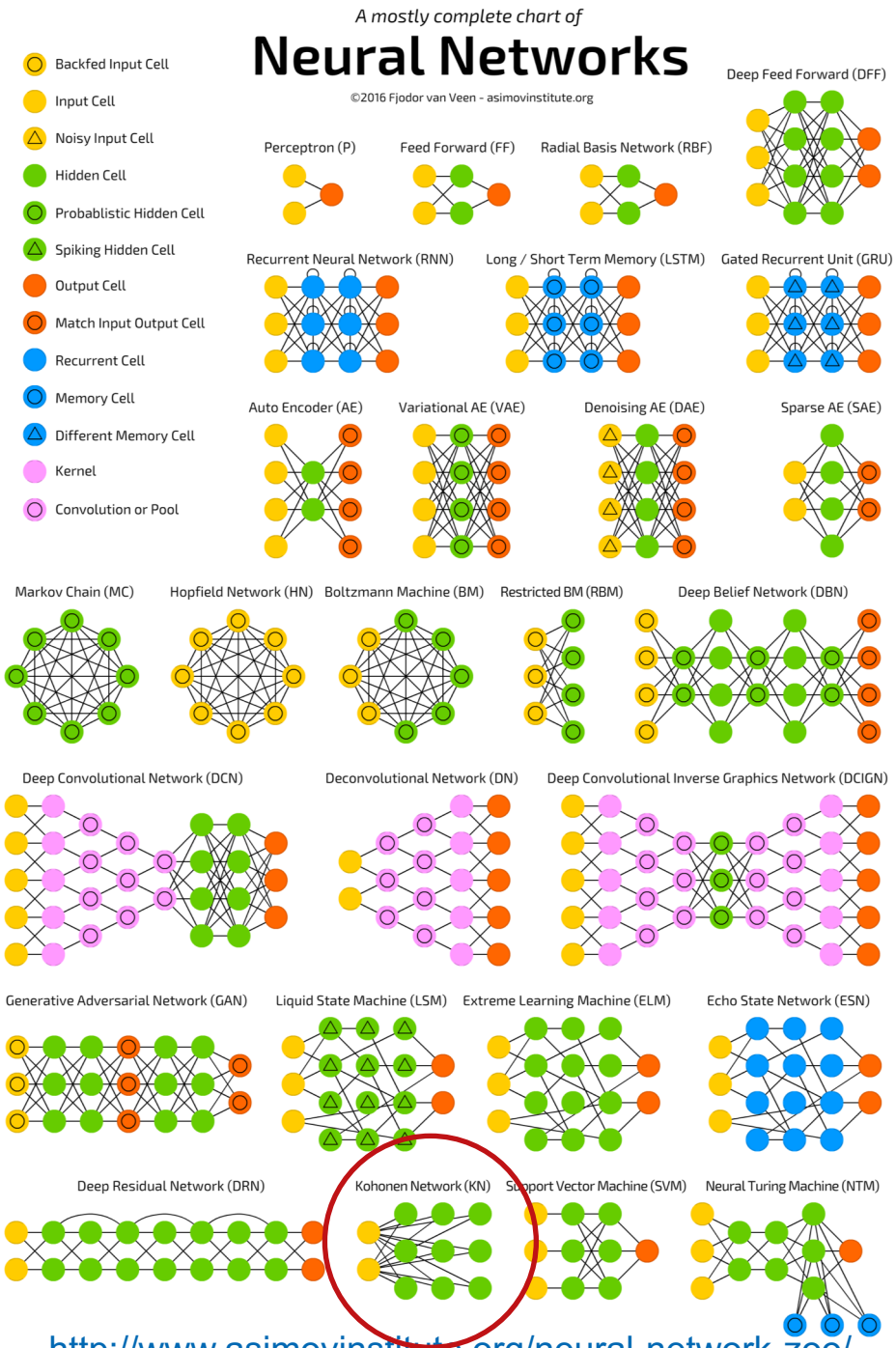




Der Neuronale Netze-Zoo

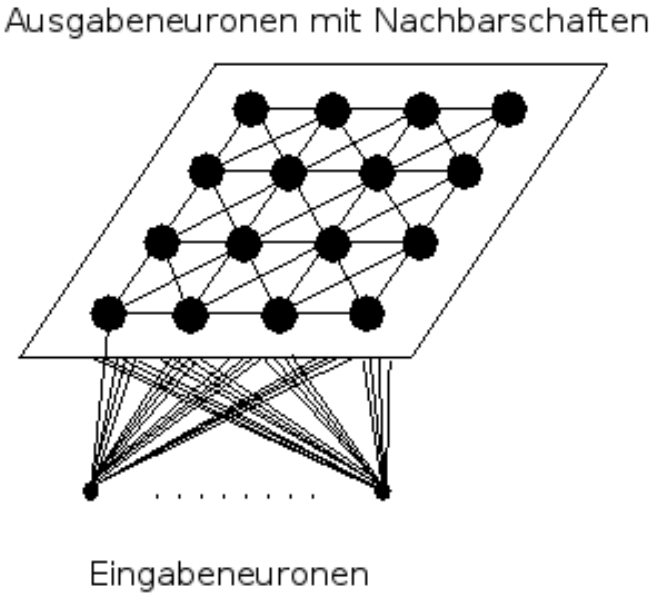
(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze
- LVQ & SOMs



http://www.asimovinstitute.org/neural-network-zoo/

SOMs



Finde topologieerhaltende Abbildung durch Beachtung der Nachbarschaft

Anpassungsregel für Referenzvektor:

$$\vec{r}_u^{(\text{new})} = \vec{r}_u^{(\text{old})} + \eta(t) \cdot f_{\text{nb}}(d_{\text{neurons}}(u, u_*), \varrho(t)) \cdot (\vec{x} - \vec{r}_u^{(\text{old})}),$$

u_* ist das Gewinnerneuron (Referenzvektor am nächsten zum Datenpunkt).

Die Funktion f_{nb} ist eine radiale Funktion.

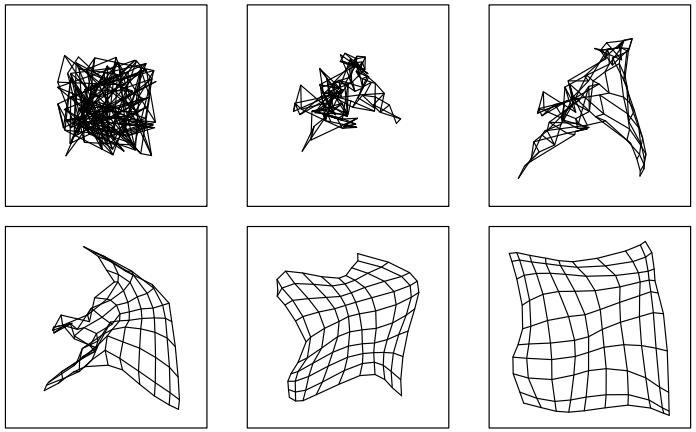
Zeitabhängige Lernrate

$$\eta(t) = \eta_0 \alpha_\eta^t, \quad 0 < \alpha_\eta < 1, \qquad \text{oder} \qquad \eta(t) = \eta_0 t^{\kappa_\eta}, \quad \kappa_\eta > 0.$$

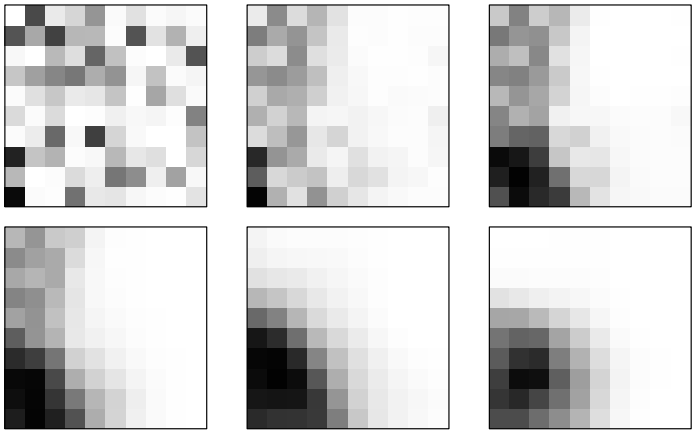
Zeitabhängiger Nachbarschaftsradius

$$\varrho(t) = \varrho_0 \alpha_\varrho^t, \quad 0 < \alpha_\varrho < 1, \qquad \text{oder} \qquad \varrho(t) = \varrho_0 t^{\kappa_\varrho}, \quad \kappa_\varrho > 0.$$

Visualisierung (2D-Eingaberaum)
Referenzvektoren => “Entfaltung”



Visualisierung (2D-Ausgaberaum)
Aktivierung für eine Beispieleingabe

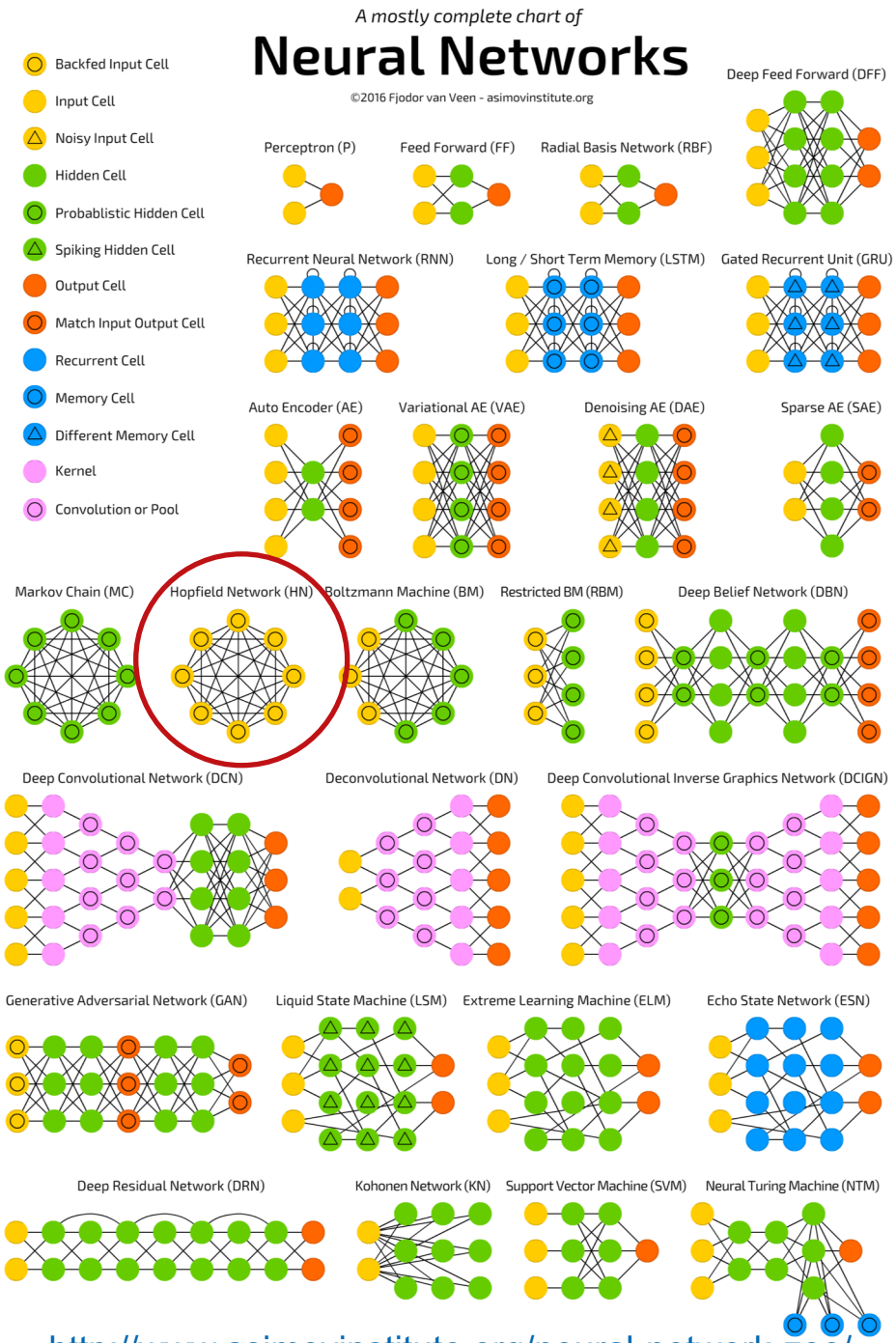


Energiebasierte Netze

Der Neuronale Netze-Zoo

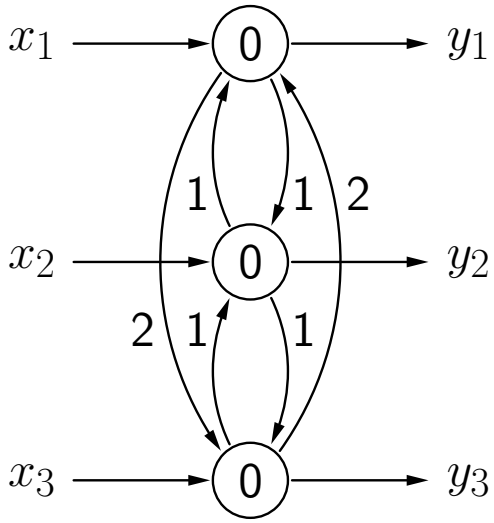
(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze
- LVQ & SOMs
- Hopfield-Netze

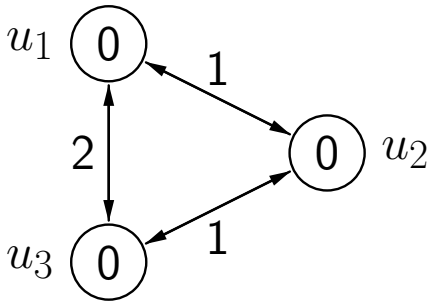


http://www.asimovinstitute.org/neural-network-zoo/

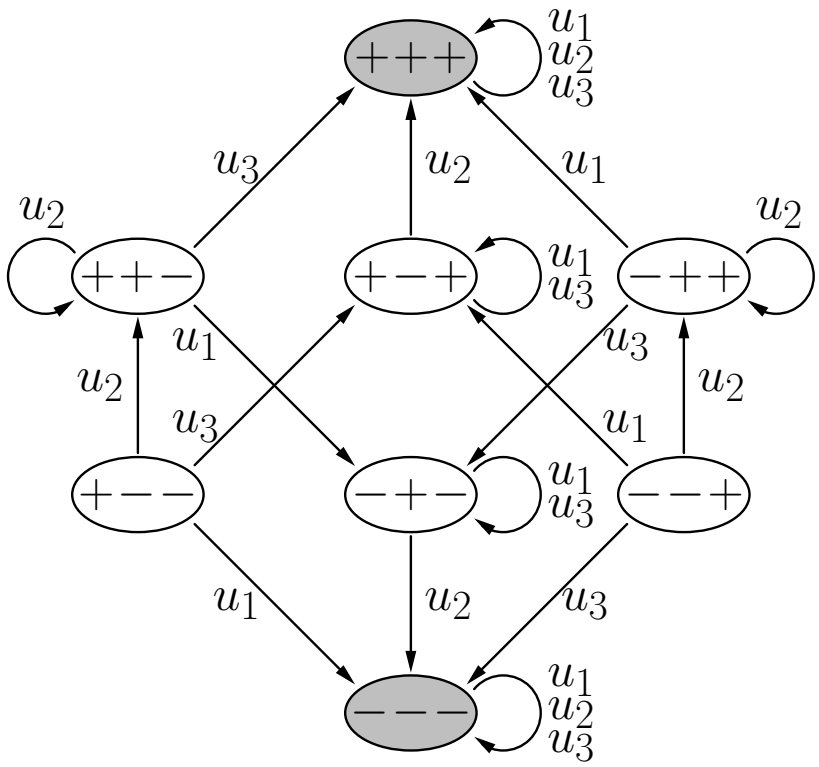
Hopfield-Netze



vereinfachte Darstellung

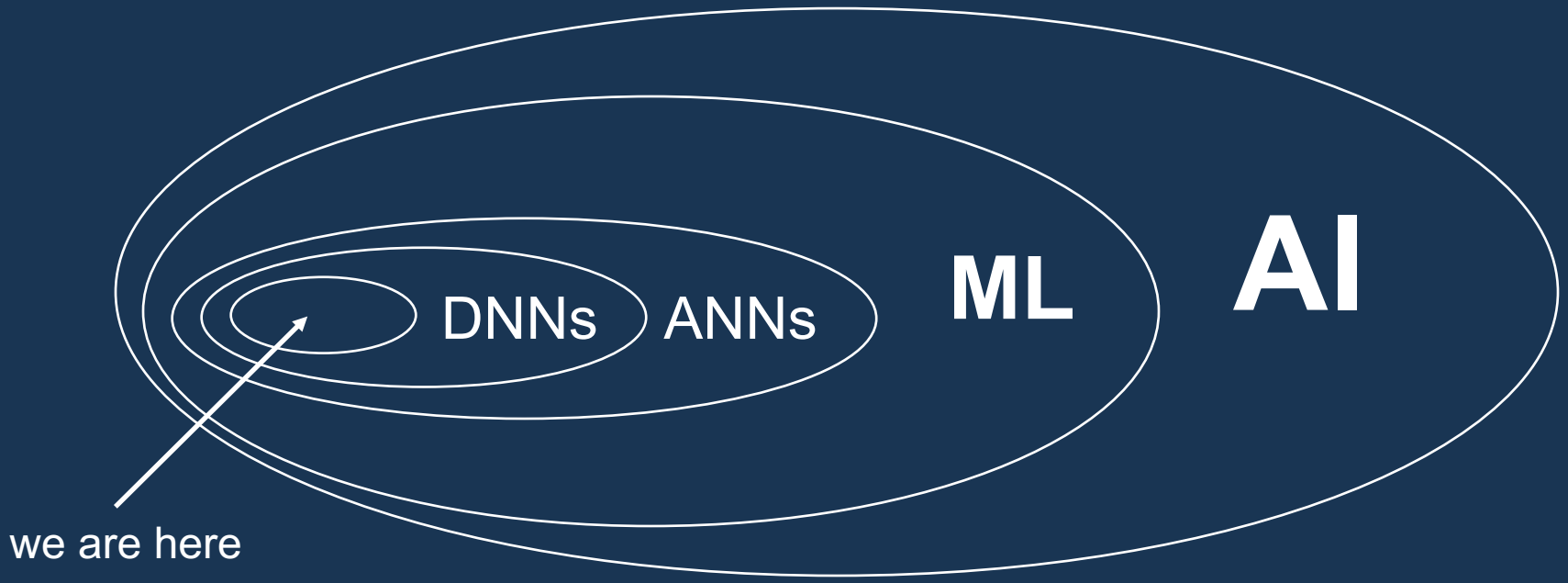


Zustandsgraph



- Konvergenz bei asynchroner Verarbeitung (in fester Reihenfolge)
- Endzustand (lokales) Energieminimum
- Verwendung als assoziativer Speicher oder zur Optimierung

Generative Models



Generative Training

- Given training data, generate new samples from same distribution!

training data ~ $p_{data}(x)$

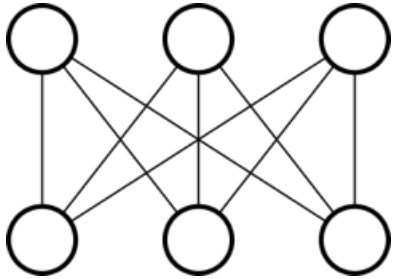
generated samples ~ $p_{model}(x)$

=> train $p_{model}(x)$ to approximate $p_{data}(x)$

- here: capture dependencies between pixels

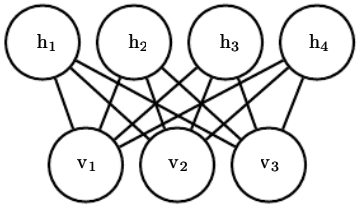
images from Denton et al. 2015

undirected model



(energy-based)

Restricted Boltzmann Machine (RBM)



hidden variables
(conditionally independent given visible variables)

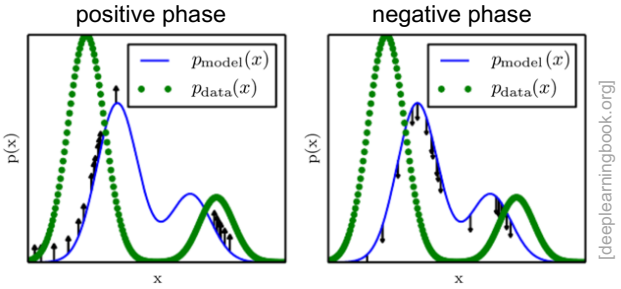
visible variables
(conditionally independent given hidden variables)

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}$$

$$P(\mathbf{v} = \boldsymbol{v}, \mathbf{h} = \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right)$$

"partition function" – for probability normalization
not tractable (sum over **many** values)

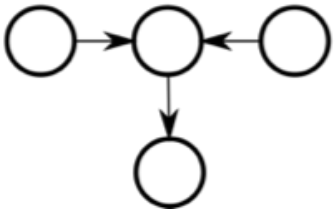
RBM Training



make samples from training data more likely

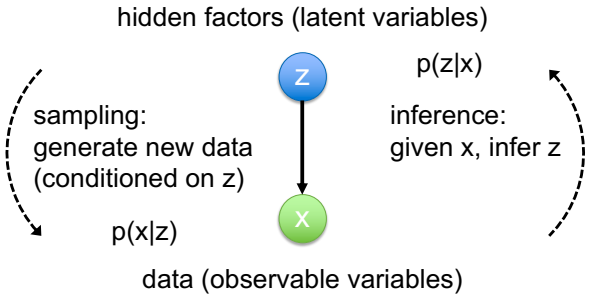
make samples from model less likely

directed model



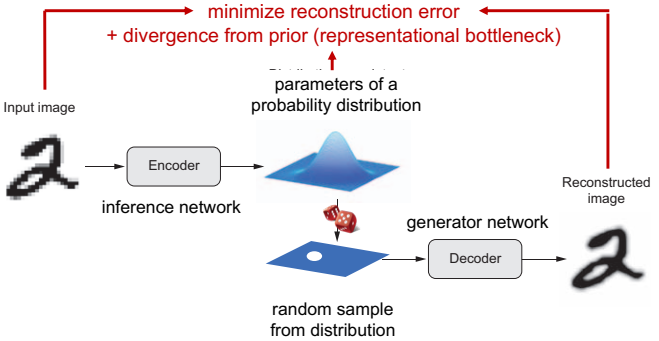
(causal structure)

Directed Graphical Model



Use DNNs to parameterize and represent conditional distributions!

Variational Autoencoder (VAE)

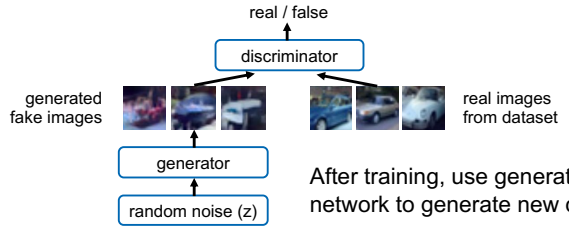


adapted from F. Chollet (2017) "Deep Learning with Python", Manning

Generative Adversarial Net (GAN)

generator net (like VAE decoder):
try to fool the discriminator by generating real-looking data

discriminator net:
try to distinguish between real and fake data



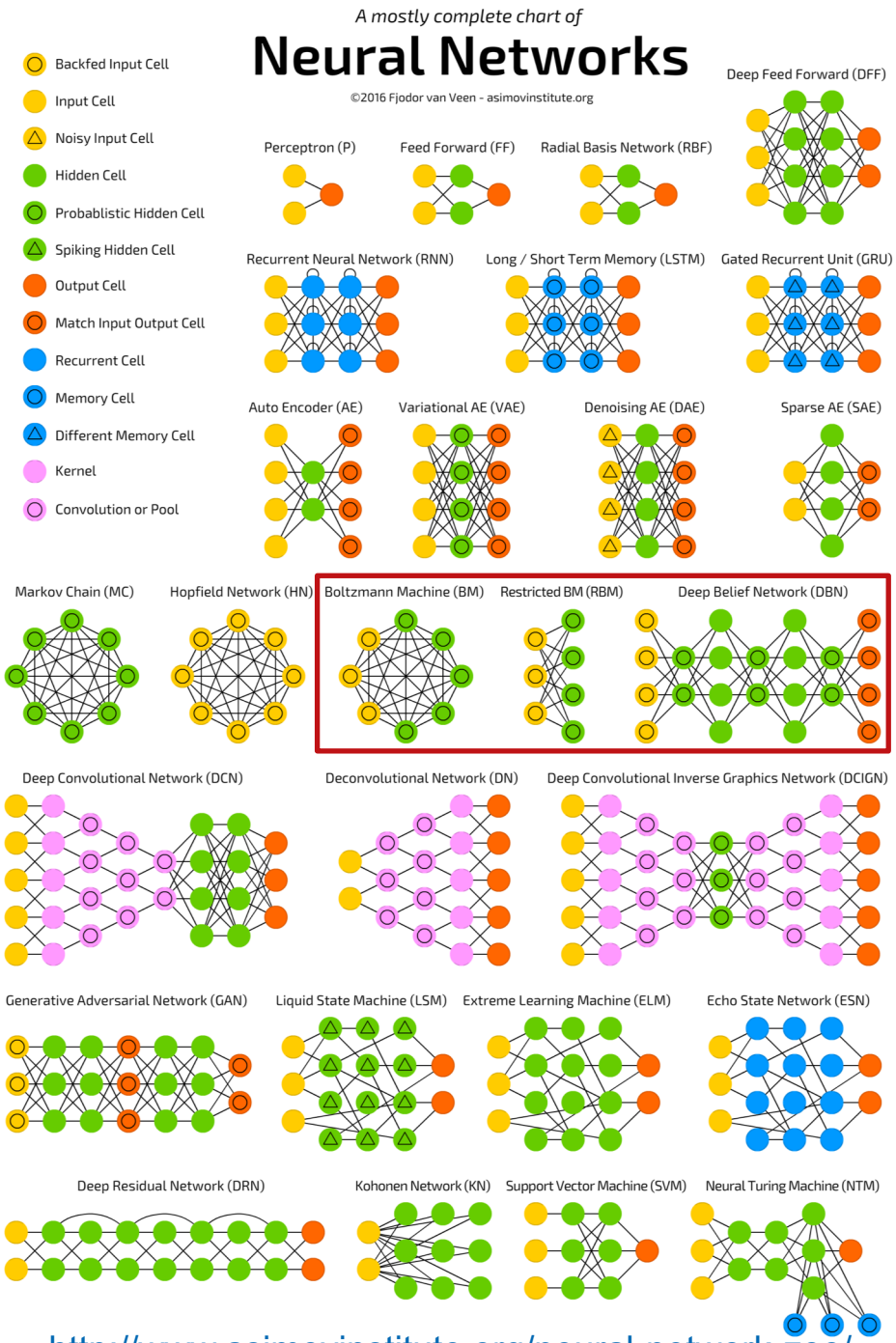
After training, use generator network to generate new data.

images from Denton et al. 2015

Der Neuronale Netze-Zoo

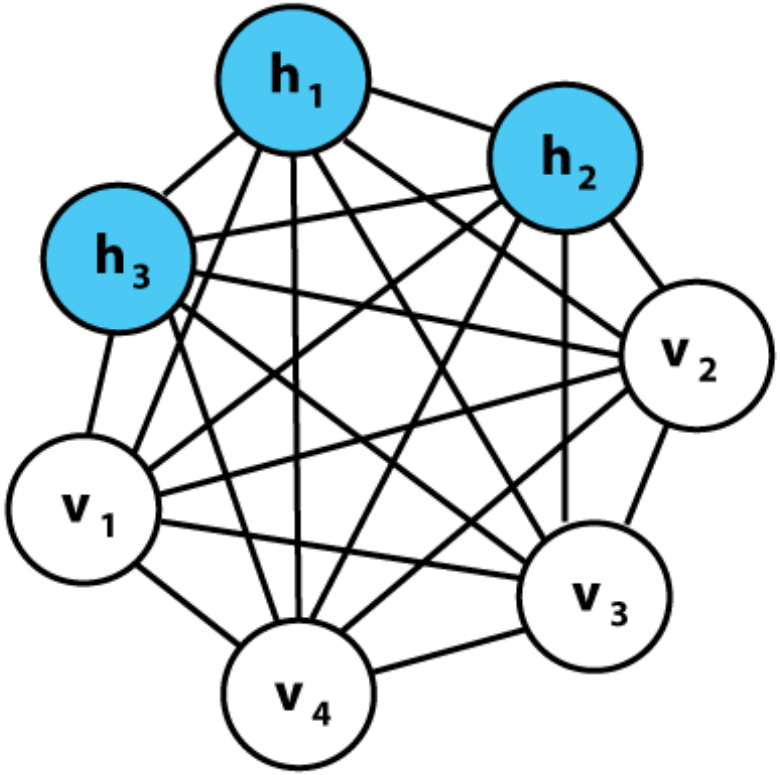
(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze
- LVQ & SOMs
- Hopfield-Netze
- BMs, RBMs & DBNs



http://www.asimovinstitute.org/neural-network-zoo/

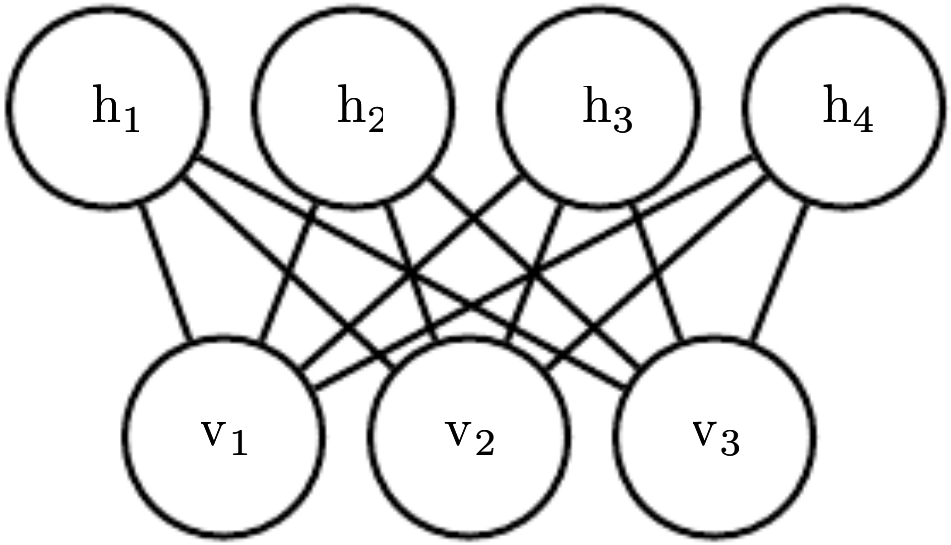
Boltzmann-Maschinen (1985)



Keine Neuronen sondern Zufallsvariablen,
die sich gegenseitig beeinflussen!

Geoffrey Hinton
(Univ. of Toronto / Google)

Restricted Boltzmann Machine (RBM)



hidden variables
(conditionally independent given visible variables)

visible variables
(conditionally independent given hidden variables)

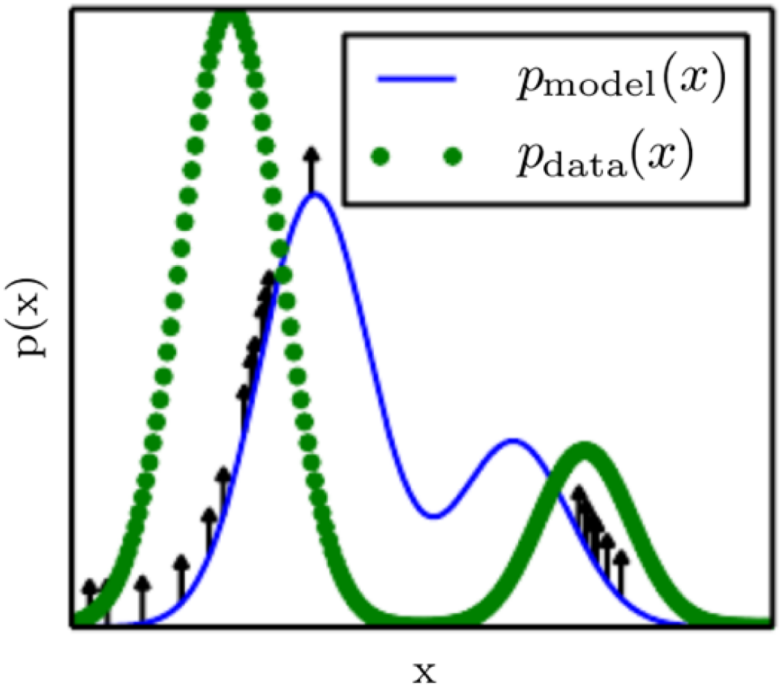
$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}$$

$$P(\mathbf{v} = \boldsymbol{v}, \mathbf{h} = \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right)$$

"partition function" – for probability normalization not tractable (sum over **many** values)

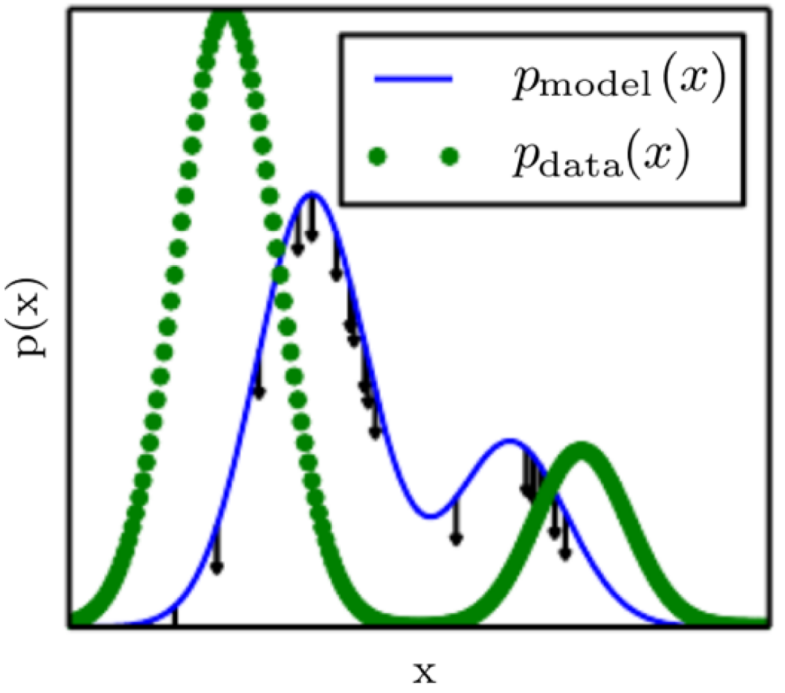
RBM Training

positive phase



make samples from training data more likely

negative phase



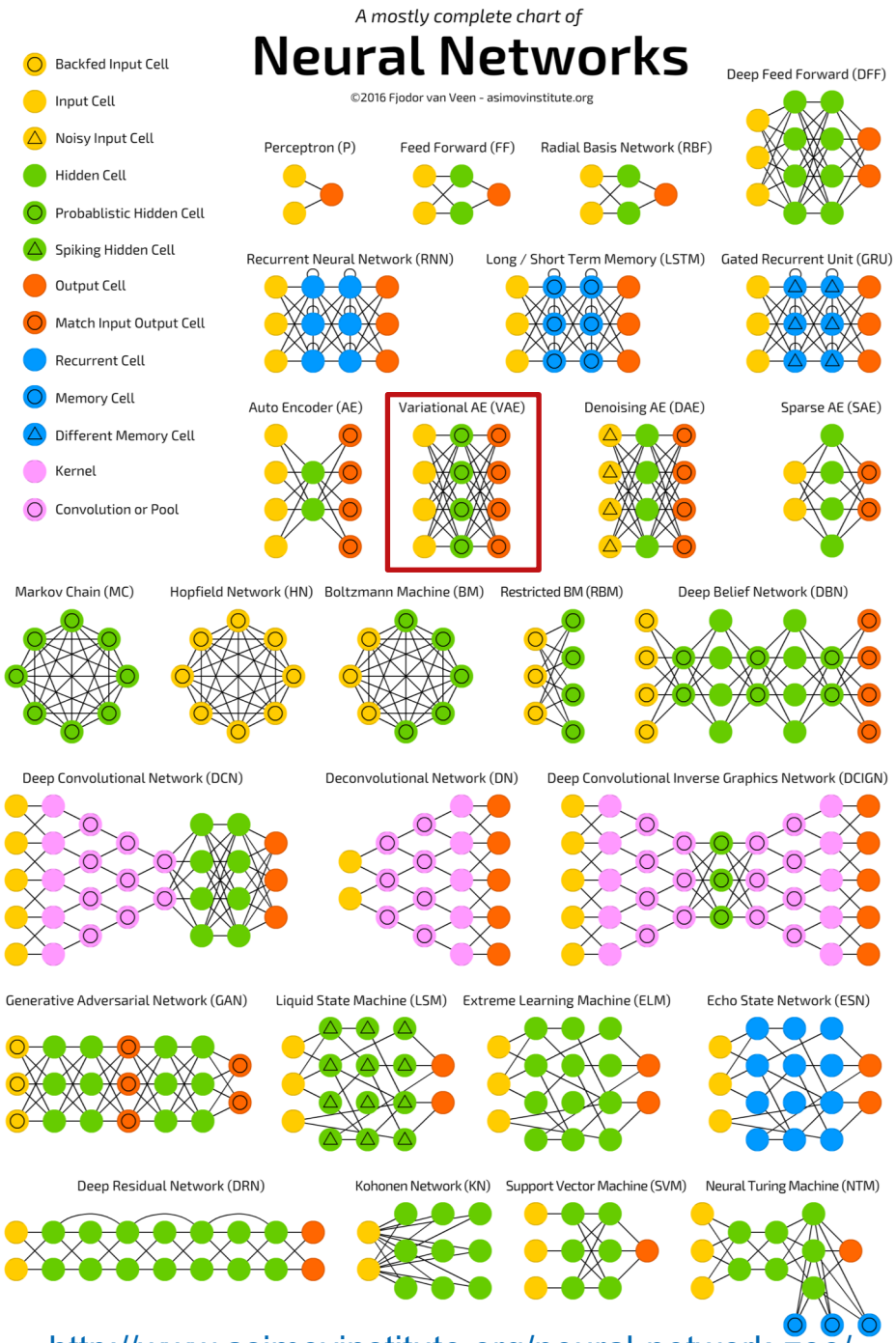
make samples from model less likely

[deeplearningbook.org]

Der Neuronale Netze-Zoo

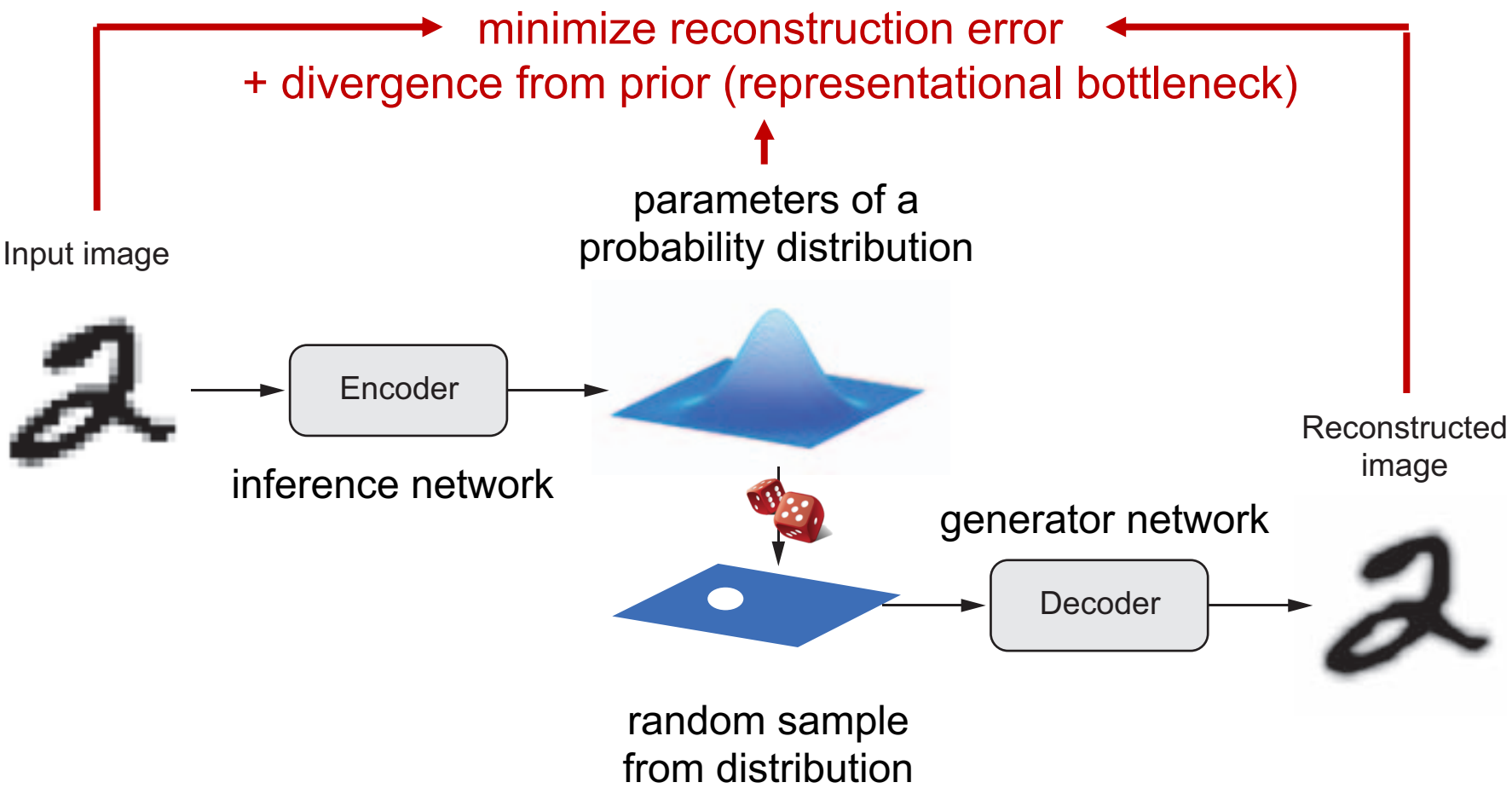
(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze
- LVQ & SOMs
- Hopfield-Netze
- BMs, RBMs & DBNs
- VAEs



http://www.asimovinstitute.org/neural-network-zoo/

Variational Autoencoder (VAE)



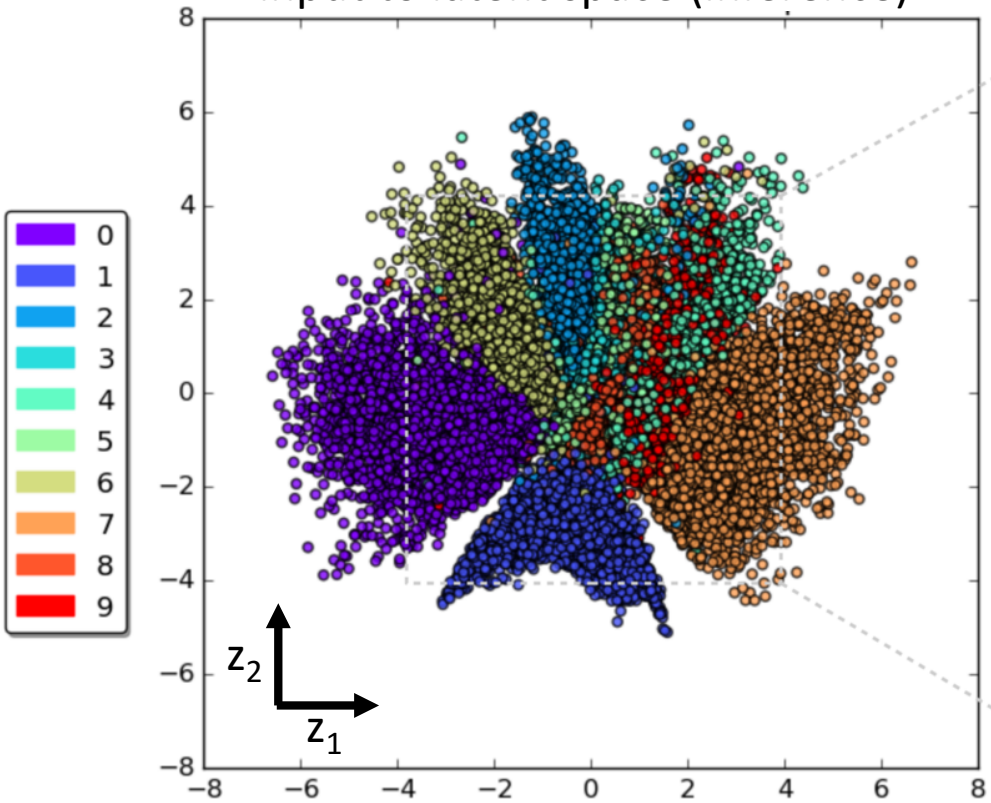
adapted from F. Chollet (2017) “Deep Learning with Python”, Manning

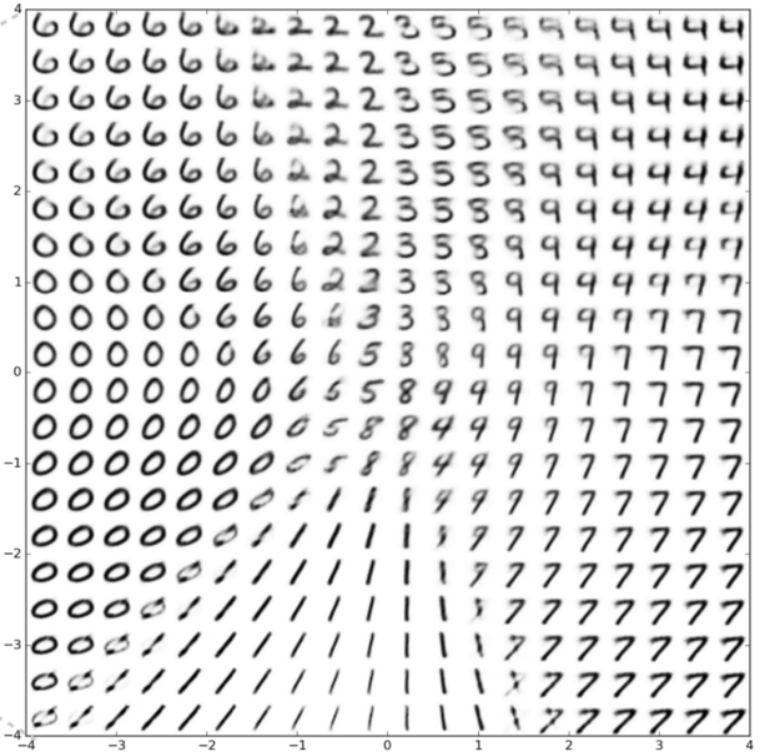
VAE Introspection

Latent Space Visualization (for MNIST dataset)

input to latent space (inference)

latent space to output (generation)

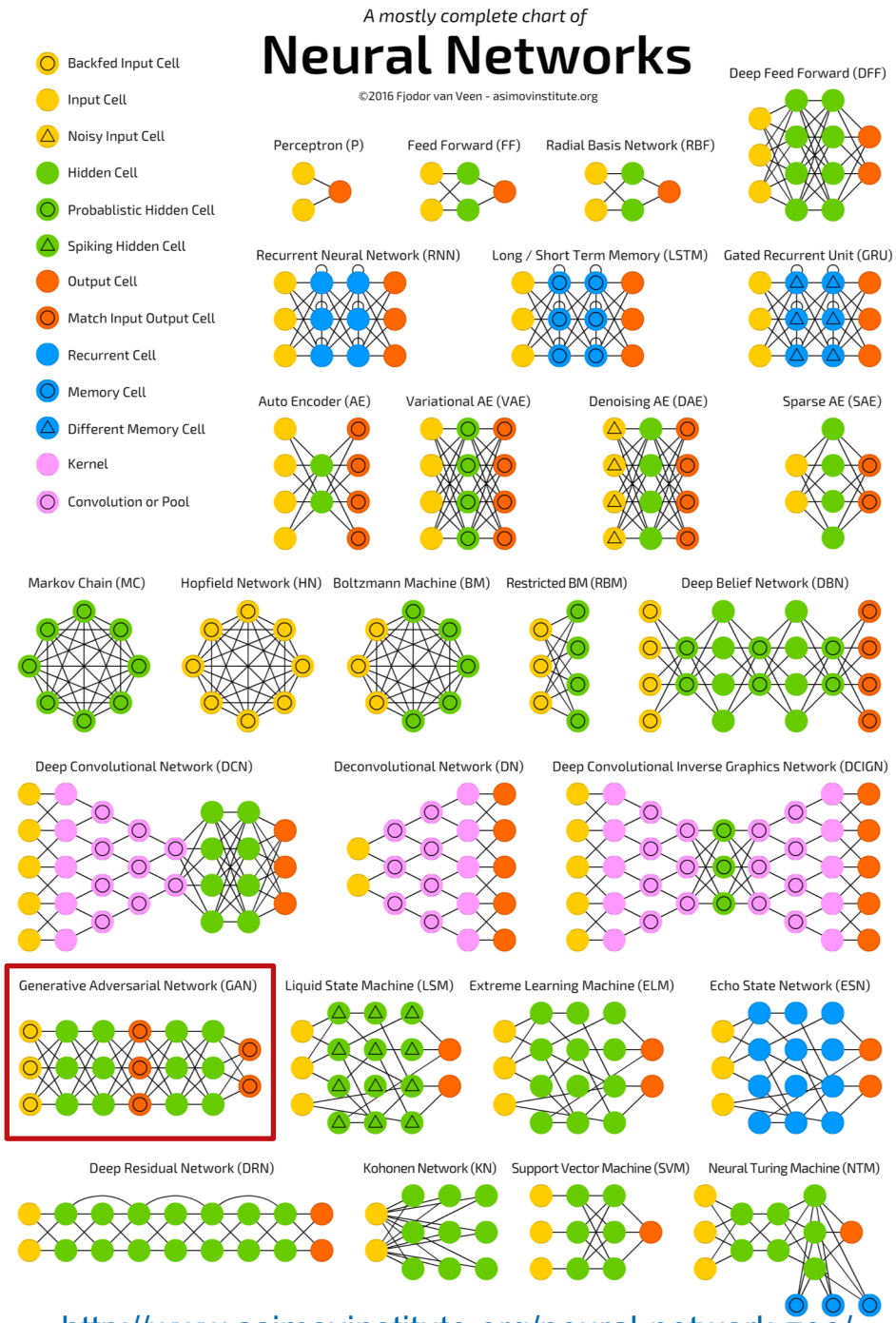




Der Neuronale Netze-Zoo

(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze
- LVQ & SOMs
- Hopfield-Netze
- BMs, RBMs & DBNs
- VAEs
- GANs

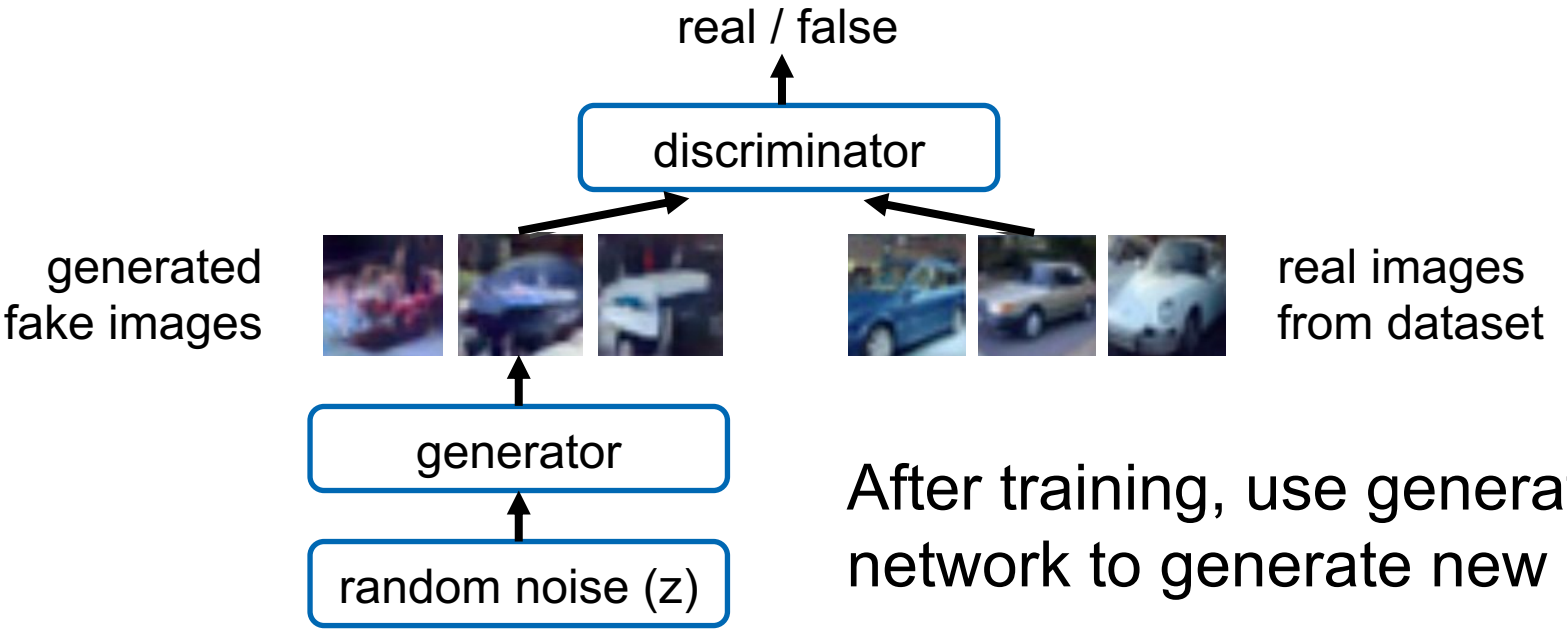


http://www.asimovinstitute.org/neural-network-zoo/

Generative Adversarial Net (GAN)

generator net (like VAE decoder):
try to fool the discriminator by generating real-looking data

discriminator net:
try to distinguish between real and fake data



After training, use generator network to generate new data.

images from Denton et al. 2015

Der Neuronale Netze-Zoo

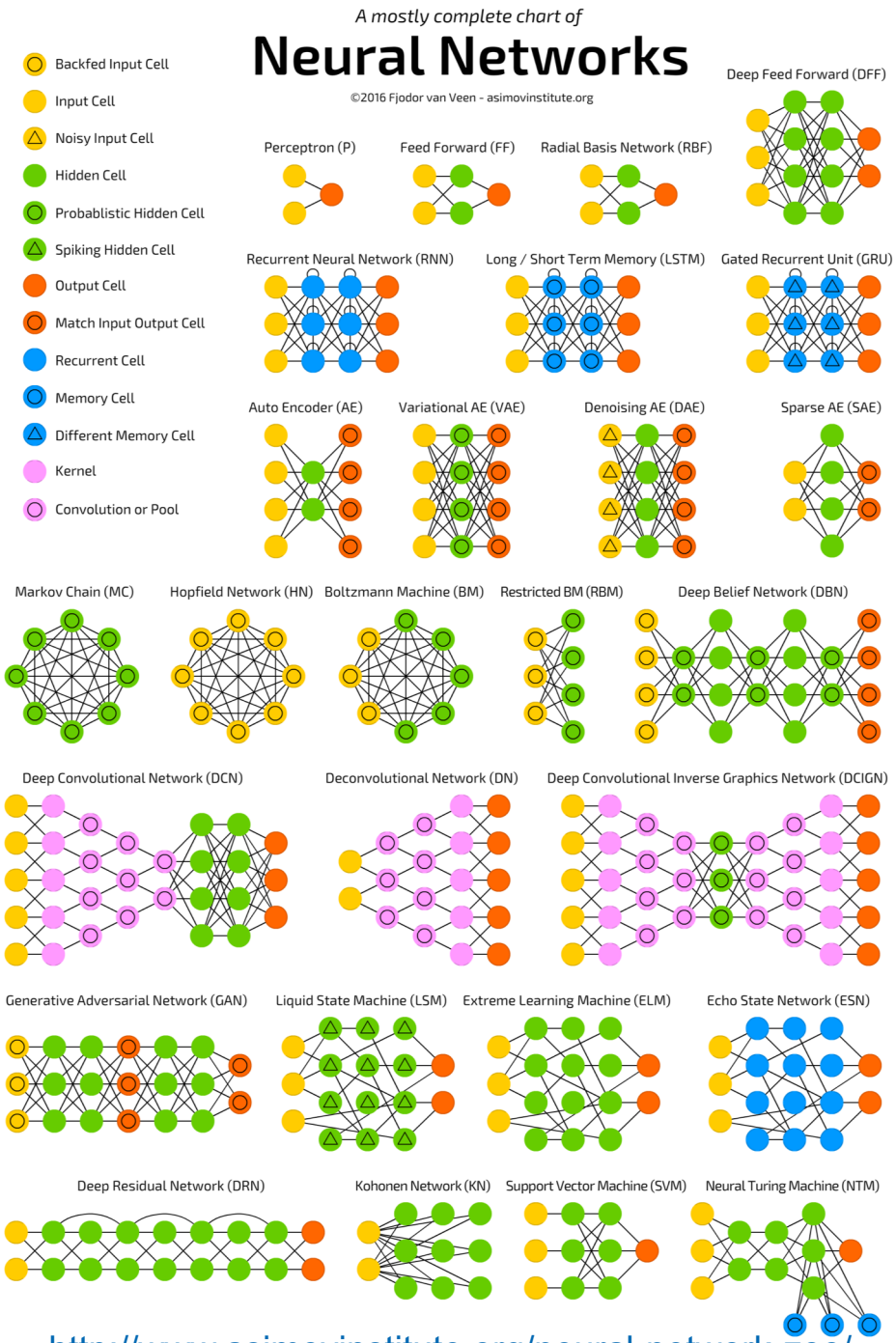
(nicht vollständig!)

- Schwellenwertelem. (Perceptron)
- MLPs
- CNNs
- RNNs
- RBF Netze
- LVQ & SOMs
- Hopfield-Netze
- BMs, RBMs & DBNs
- VAEs
- GANs

Abstand

Energie

generativ



http://www.asimovinstitute.org/neural-network-zoo/

Ausblick

- Responsible Data Science (Projekt)
  - 16. Juli (Workshop)
  - 22. November (öffentliche Präsentation)
- Introduction to Deep Learning (jedes WiSe)
  - Vorlesung (flipped) + Übung + Tutorium
- Learning Generative Models (jedes SoSe)
  - Vorlesung (flipped) + Übung
- Music Information Retrieval (jedes WiSe)
  - Vorlesung + Übung