Deep Learning (BEV033DLE)
Lecture 1.

Czech Technical University in Prague

- Biological neurons
- Visual cortex and pathway
- Artificial neuron models
- Neural network architectures
- Application examples
- Stochastic neurons

Organisational Matters

Teachers: Alexander Shekhovtsov, Jan Šochman and Boris Flach

Format: 1 lecture & 1 lab per week (6 credits), labs of two types (alternating)

- practical labs: implementation of selected methods (Python/PyTorch),
- theoretical labs: solving theoretical assignments
  assignments are published in advance, you are expected to present/discuss solutions

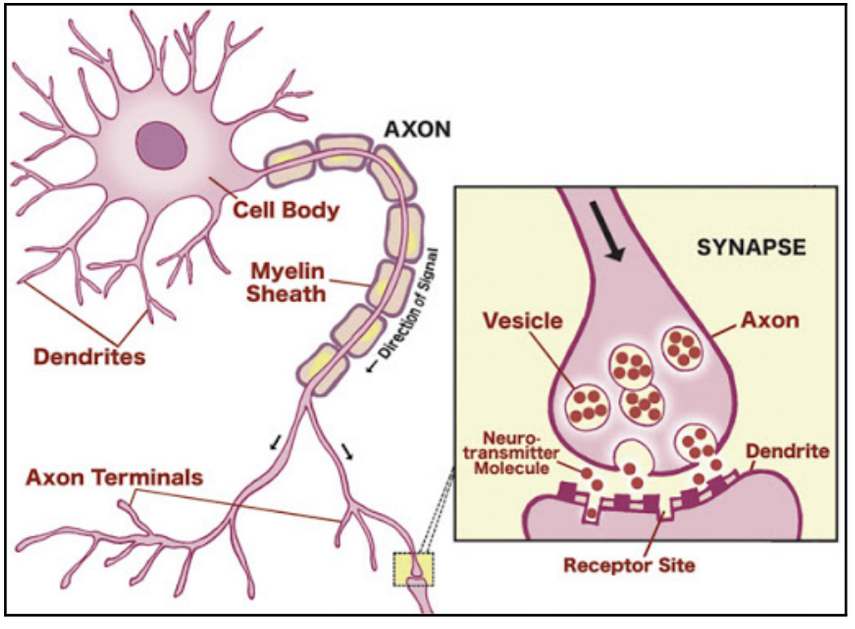
Grading: 50% practical labs + 50% written exam = 100% (+ bonus points)

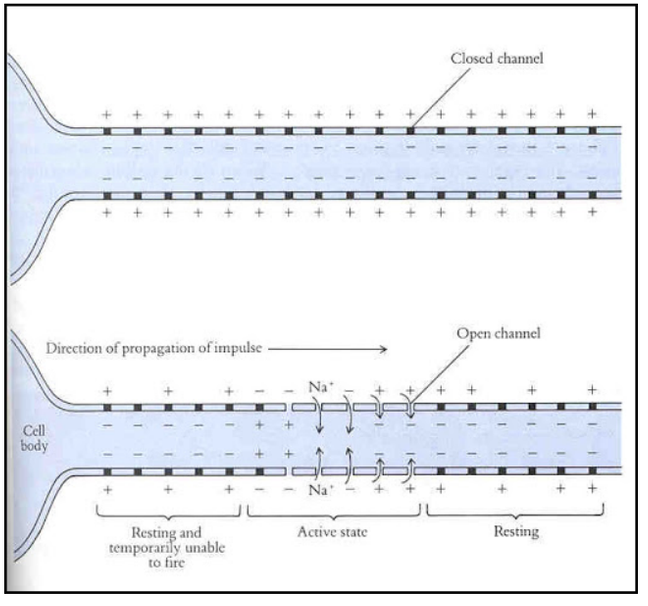
Prerequisites:

- calculus, linear algebra and optimisation
- basics of graph theory and related algorithms
- pattern recognition and machine learning (AE4B33RPZ)

More details: https://cw.fel.cvut.cz/wiki/courses/bev033dle/start

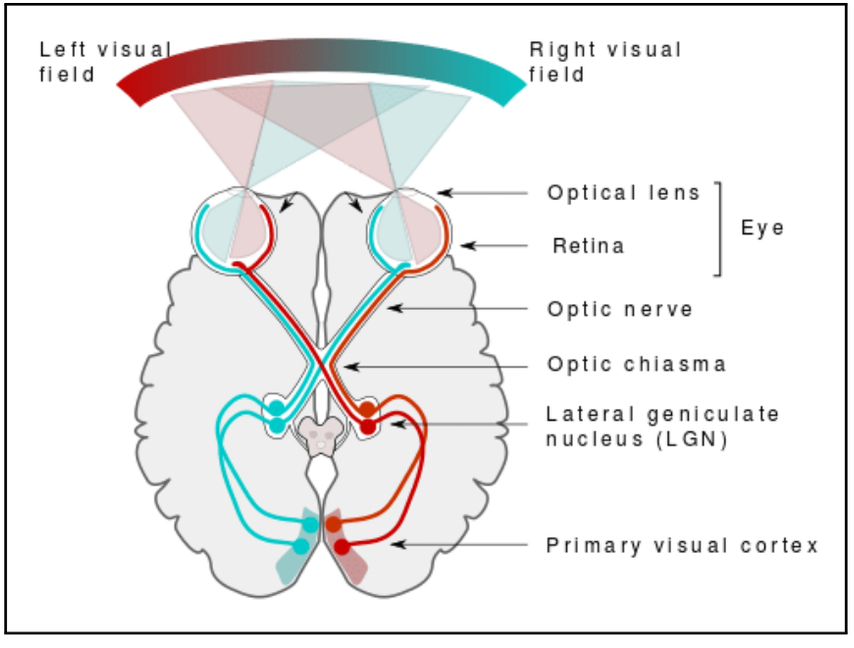
Biological neurons

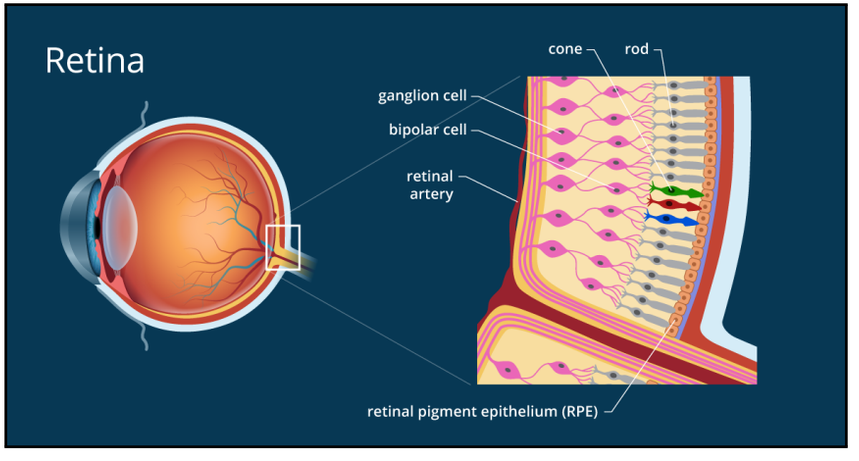




- physiology of neuron cells: dendrite, soma, axon, axon branches, synaptic connections
- resting state: interplay of selective ion channels and pumps leads to negative potential
- action potential: strong simultaneous excitation by incoming signals (dendrite) - triggers depolarisation at axon hillock - opens voltage gated ion channels - travelling electric signal (spike) along the axon
- spike arriving at synaptic endings - release of neuro-transmitters into the synaptic gap - excitation or inhibition of the postsynaptic neurons

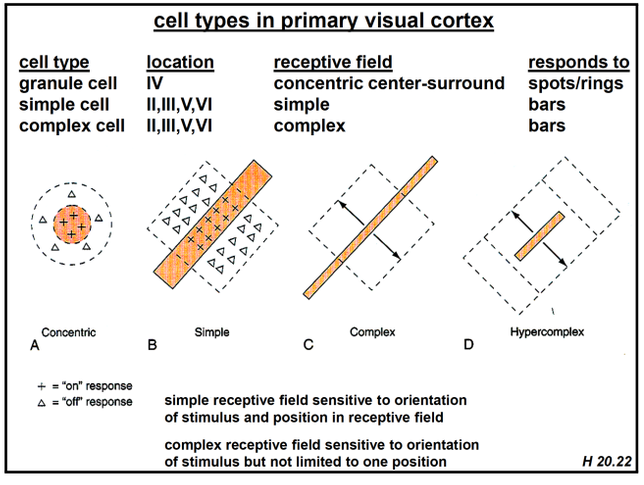
Biological neural networks: visual cortex





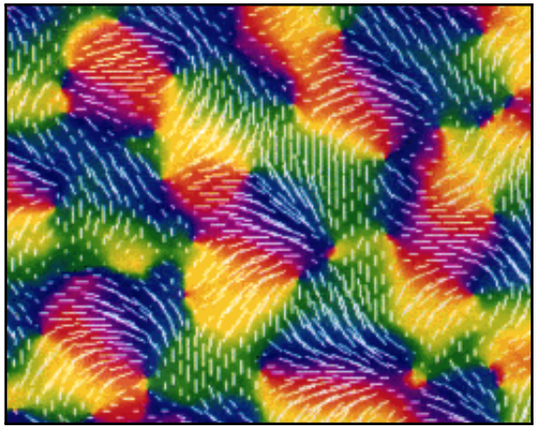
- Retina photoreceptors: $\sim 10^7$ cones and $\sim 10^8$ rods, several processing layers, last layer: $\sim 10^6$ ganglion cells ("center on" and "center off")
- Optical nerve - lateral geniculate nucleus - V1 first layer of visual cortex
- First layers in visual cortex resemble convolutional networks (with feedback)
- simple and complex cells in V1 have circular shaped receptive fields

Biological neural networks: visual cortex



Cell types in V1

David Hubel and Torsten Wiesel (Nobelprize, 1981)



Orientation columns in V1

- recording specific patterns that stimulate individual neurons, simple and complex cells in V1,
- retinotopy, ocular dominance columns, orientation columns,
- emergence of architecture/connectivity: blocking visual input from an eye input in critical development phases dramatically alters the connectivity pattern,
- absence of visual stimuli during critical development periods leads to deterioration of connectivity and suppresses emergency of cells responding to specific visual stimuli.

Artificial neurons

Artificial neuron as elementary computational unit (McCulloch-Pitts 1943)

$$y = f\left(\sum_i w_i x_i + b\right) = f(w^T x + b),$$

with $x = (x_1, \ldots, x_n)$ - inputs, $w = (w_1, \ldots, w_n)$ - weights, b - bias and $f : \mathbb{R} \to \mathbb{R}$ - a nonlinear activation function.

binary output: $y = 0, 1$, f: H - Heaviside function or $y = \pm 1$, f: sign - sign function

- represents a binary classifier
- Notice, that the mapping is invariant to parameter scaling $(w, b) \to (\lambda w, \lambda b)$
- networks with binary outputs & binary valued weights/biases are highly relevant for mobile applications
- the mapping is piece-wise constant, networks can not be learned by gradient descent

graded response: $y \in [0, 1]$, f: σ - sigmoid function or $y \in [-1, 1]$, f: $\tanh$

- squashed output (spike frequency)
- the mapping is differentiable, but can have vanishing gradients

Artificial neurons

Rectified Linear Units: $y = \mathbb{R}_+$, $f(x) = \max(0, x)$ - ReLU or $f(x) = \max(\alpha x, x)$ - Leaky ReLU

- the mapping is differentiable (a.e.),
- ReLU neurons can have vanishing gradients,
- notice, that the mapping scales linearly with $(w, b) \to (\lambda w, \lambda b)$
- LReLU neurons can be generalised to **maxout neurons** with tunable slopes $f(x) = \max(\alpha_1 x + b_1, \alpha_2 x + b_2)$

Softmax unit/layer: suppose the network outputs should represent class probabilities (conditioned on the input), i.e. they must be non-negative and sum up to one.

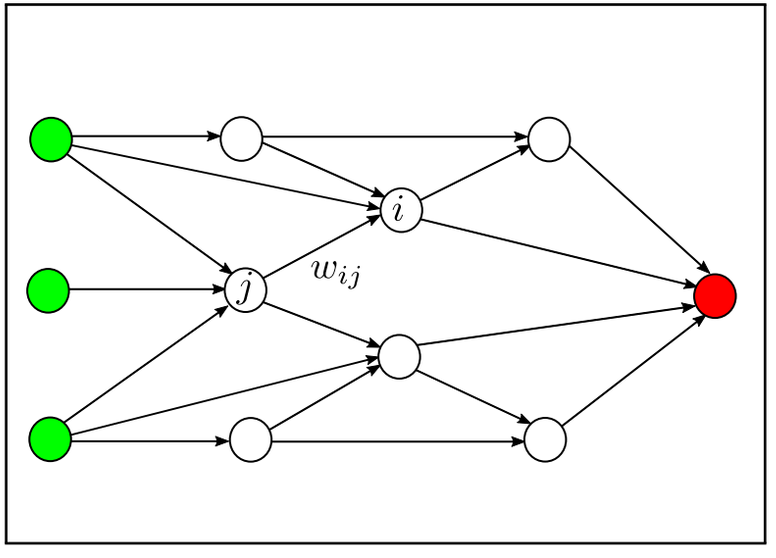
- $x = (x_1, \ldots, x_K)$ are **scores** for the K classes
- $y = (y_1, \ldots, y_K)$ are interpreted as class probabilities, where

$$y_k = \frac{e^{x_k}}{\sum_{m=1}^{K} e^{x_m}}$$

Network architectures

A general **feed forward network** is given by

- (V, E, w) a directed acyclic weighted graph
- $\{x_i \in \mathbb{R} \mid i \in V\}$ real valued variables associated with the nodes of the graph
- $\{f_i \mid i \in V\}$ activation functions associated with the nodes.



The network defines a mapping $x_I \mapsto x_T$, where $I \subset V$ are the input nodes and $T \subset V$ are the output nodes of the graph. It is defined recursively by

$$x_i = f_i(a_i) = f_i\Big(\sum_{j \in pa(i)} w_{ij} x_j\Big)$$

The value (output) of each node $i \in V$ becomes a function $x_i = F_i(x_I, w)$ of the inputs x_I and the network parameters w.

Network architectures

A **layered feed forward network** is organised in **layers** $V = \bigcup_{k=0}^{L} V^k$ with edges between nodes in consecutive layers only. We call V^0 the input layer, V^L the output layer and the remaining layers hidden.

The input-output mapping $F\colon x^0 \to x^L$ can be represented as composition of mappings

$$x^L = F(x^0) = f^L \circ A^L \circ \dots f^1 \circ A^1 x^0,$$

where

- x^k denotes the output vector of neurons in layer k,
- A^k denotes the linear (affine) mapping $A^k x^{k-1} = W^k x^{k-1} + b^k$,
- f^k denotes the elementwise application of the activation function of layer k.

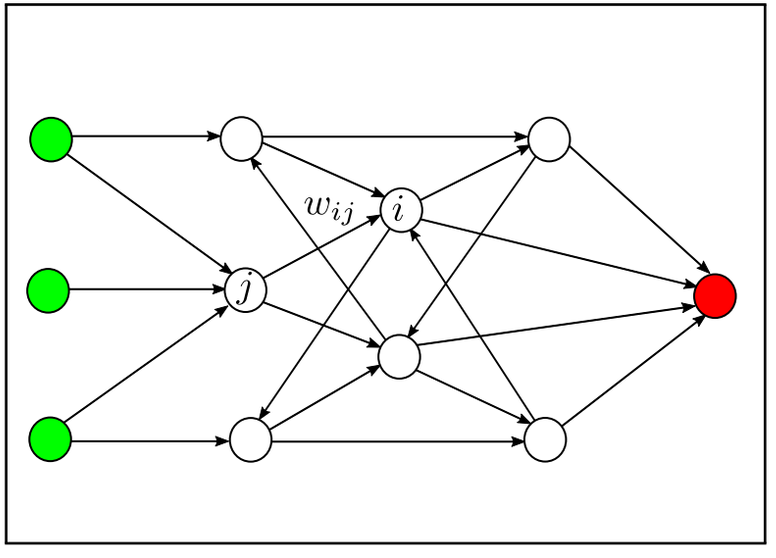
This representation allows to "modularise" the forward computation and the computation of the derivatives of the outputs w.r.t. to the network parameters.

An important special case of FFNs are **convolutional neural networks**, which will be discussed later in the course.

Network architectures

A general **recurrent** network is given by

- (V, E, w) a directed weighted graph (with cycles)
- $\{x_i \in \mathbb{R} \mid i \in V\}$ real valued variables associated with the nodes of the graph
- $\{f_i \mid i \in V\}$ activation functions associated with the nodes.



Notice, that this leads to a **dynamical system** even if we clamp the states of the input neurons.

$$x_i(t) = f_i(a_i) = f_i\Big(\sum_{j \in \mathcal{N}_i} w_{ij} x_j(t-1)\Big)$$

There are different possible **updating schedules**

- **parallel:** compute new outputs for all neurons and update them synchronously,
- **sequential:** fix an ordering of neurons and update them one at a time.

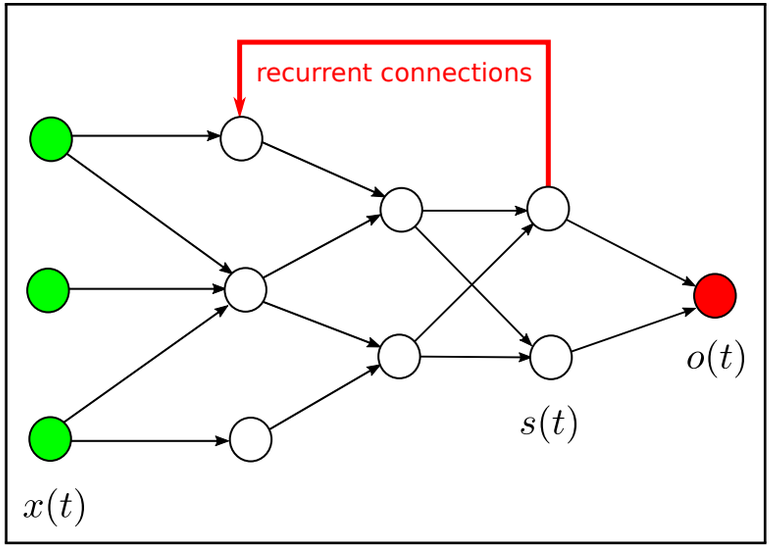
Network architectures

Recurrent networks for time sequence processing:

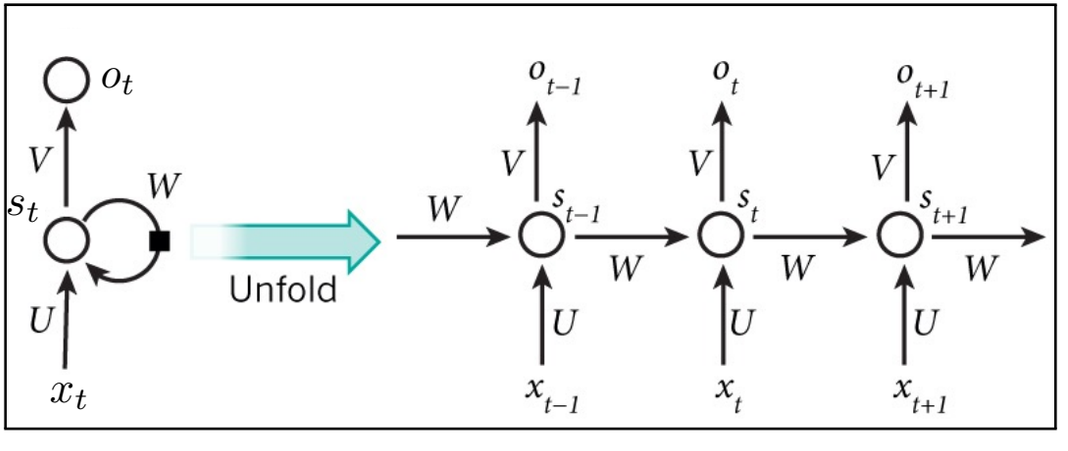
- consider layered networks with additional feedback connections from penultimate layer to first layer,
- this leads to the dynamical system

$$s(t) = F(x(t), s(t-1))$$
$$o(t) = G(s(t)),$$

where $x(t)$ denotes inputs, $s(t)$ denotes outputs of the penultimate layer and $o(t)$ denotes outputs

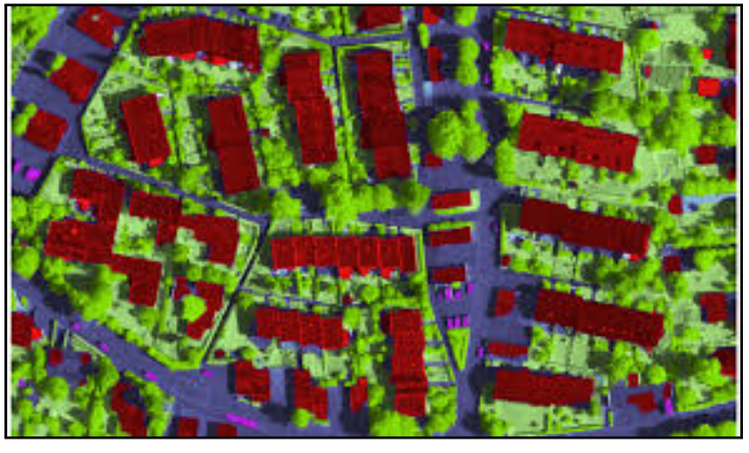


It can be "unfolded" in time

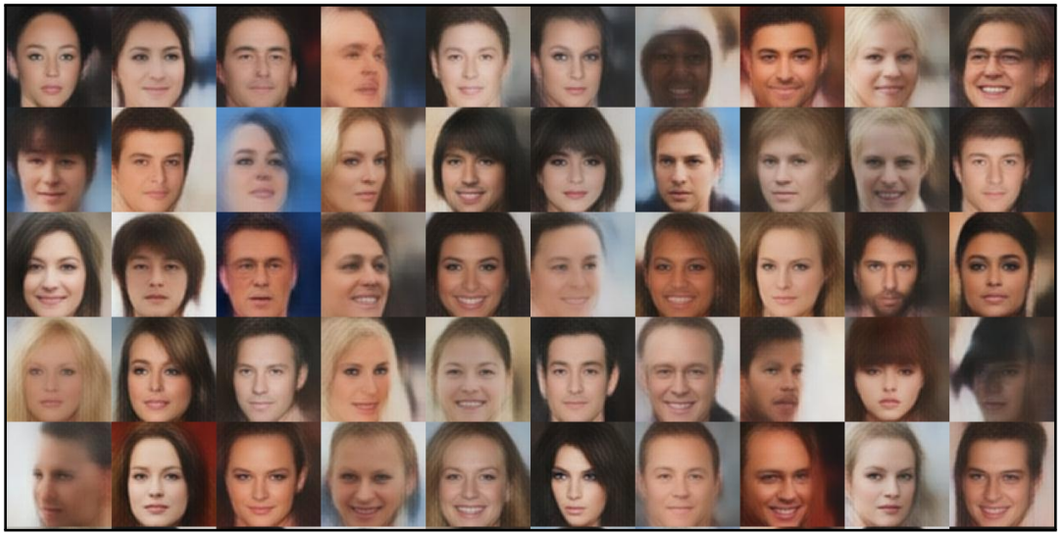


Application examples

Semantic segmentation: input image $\rightarrow$ CNN $\rightarrow$ output segment class probabilities for each pixel

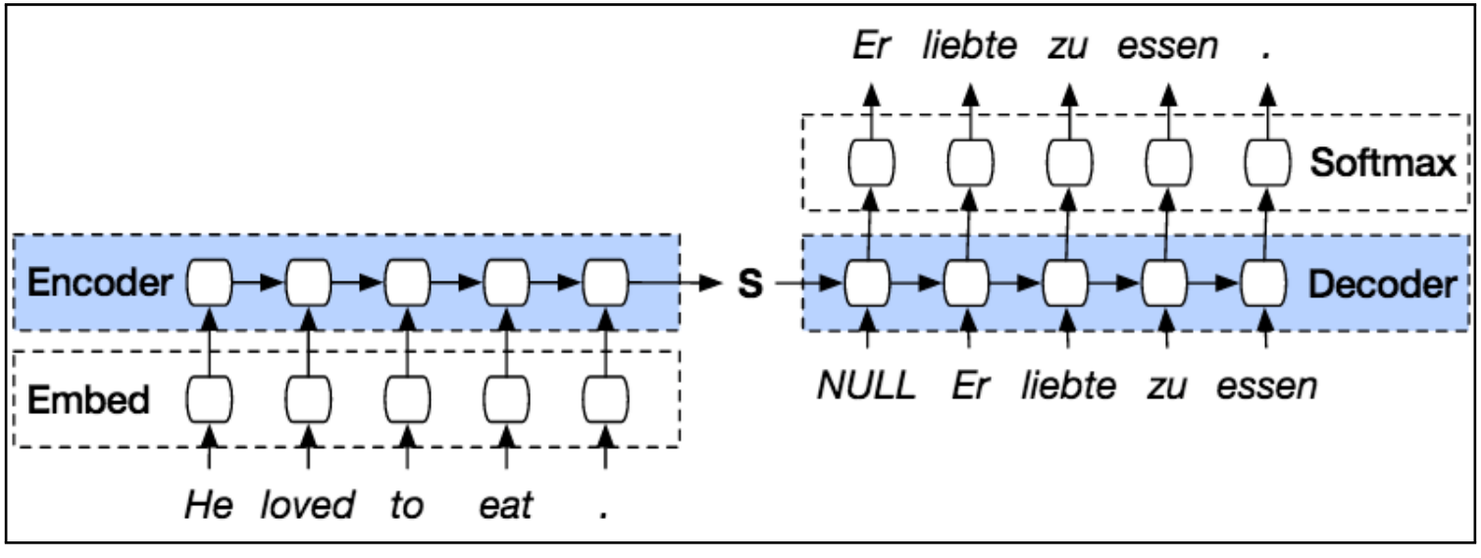


Generative networks: sample latent noise $Z \sim \mathcal{N}(0, C)$ $\rightarrow$ DNN $\rightarrow$ output image



Application examples

Natural Language Processing: machine translation by "sequence to sequence" RNNs



Sidestep: stochastic neurons

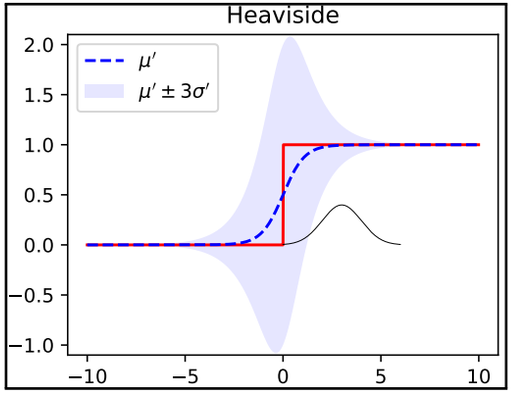
Why stochastic neurons?

- recognition: randomised predictors might be easier to learn (regularisation)
- networks with binary output neurons: make them accessible for learning

Stochastic neuron with additive injected noise

$$y = f\Big(\underbrace{\sum_i w_i x_i + b}_{a(x)} + z\Big)$$

z - noise



The **expected** output as a function of the input becomes smooth, even if the activation function is not. E.g. for $f = \mathrm{H}$ and $z \sim$ standard logistic distribution we obtain

$$\mathbb{E}_z[y] = \sigma\Big(\sum_i w_i x_i + b\Big)$$

because the sigmoid function σ is the c.d.f. F of the standard logistic distribution.

$$\mathbb{P}(y=1) = \mathbb{P}(a(x)+z \geqslant 0) = \mathbb{P}(z \geqslant -a(x)) = 1 - F(-a(x)) = F(a(x)) = \sigma(a(x)).$$