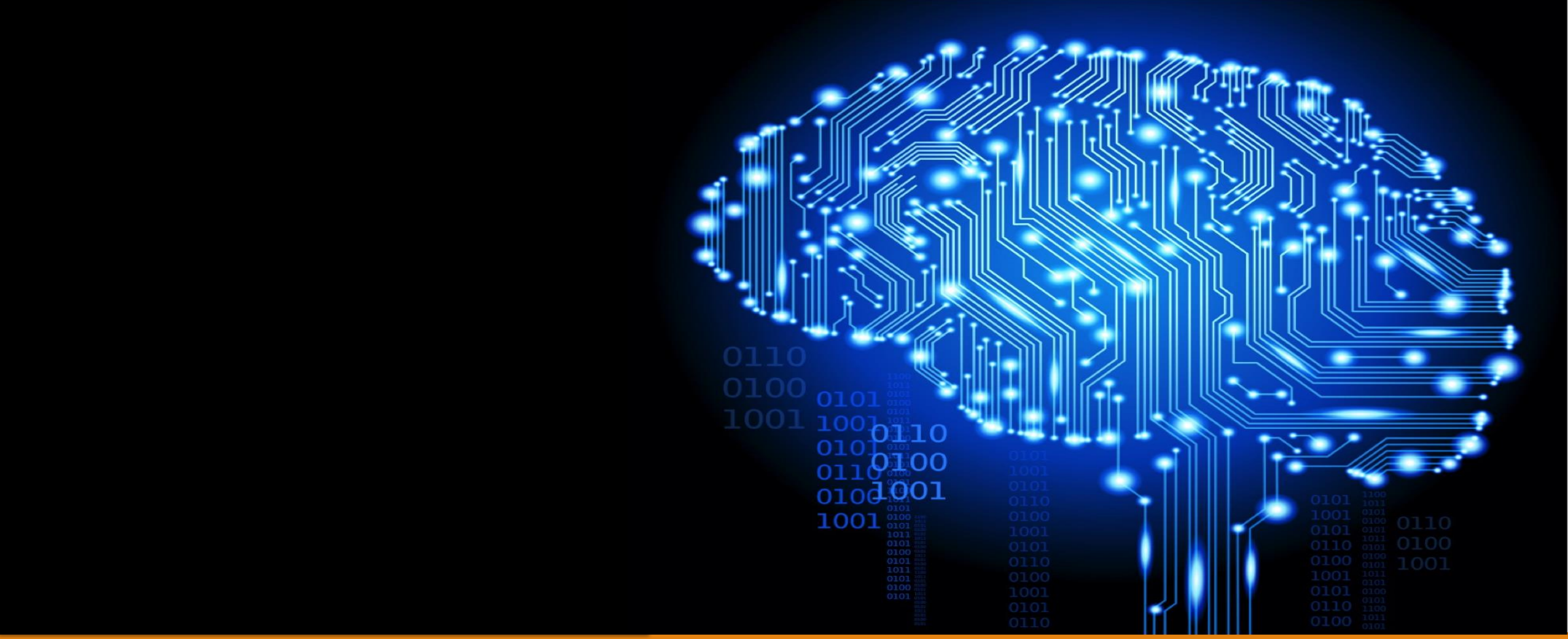


Lecture 2: Modular Learning

Deep Learning @ UvA

Lecture Overview

- Modularity in deep learning
- Deep learning nonlinearities
- Gradient-based learning
- Chain rule
- Backpropagation

Neural networks in math

A family of parametric non-linear and hierarchical representation learning functions, which are massively optimized with stochastic gradient descent to encode domain knowledge, i.e. domain invariances, stationarity.

$$a_L(x; \theta_{1,\dots,\mathrm{L}}) = h_L\left(h_{L-1}(\dots(h_1(x, \theta_1), \dots), \theta_{L-1}), \theta_L\right)$$

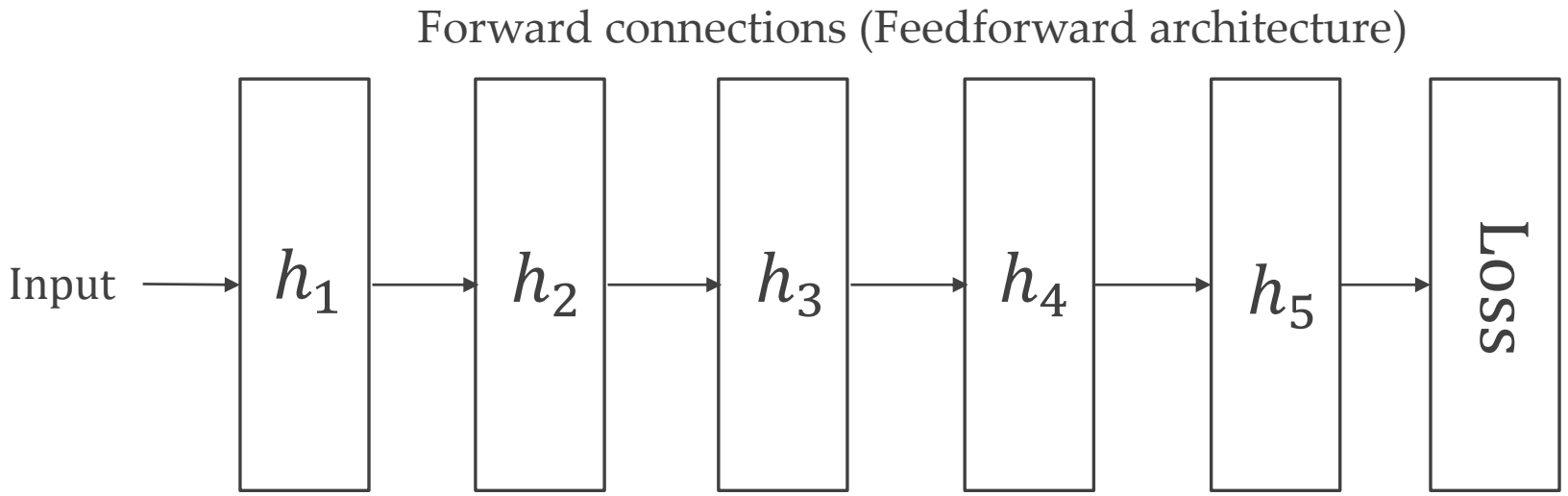
- We can simplify the notation by

$$a_L = h_L \circ h_{L-1} \circ \cdots \circ h_1 \circ x$$

where all functions h_l have a parameter θ_l

Neural networks in blocks

- We can visualize $a_L = h_L \circ h_{L-1} \circ \cdots \circ h_1 \circ x$ as a cascade of blocks



What is a module?

- Module ⇔ Building block ⇔ Transformation ⇔ Function
- A module receives as input either data x or another module's output
- A module returns an output a based on its activation function $h(...)$
- A module may or may not have trainable parameters w

Requirements

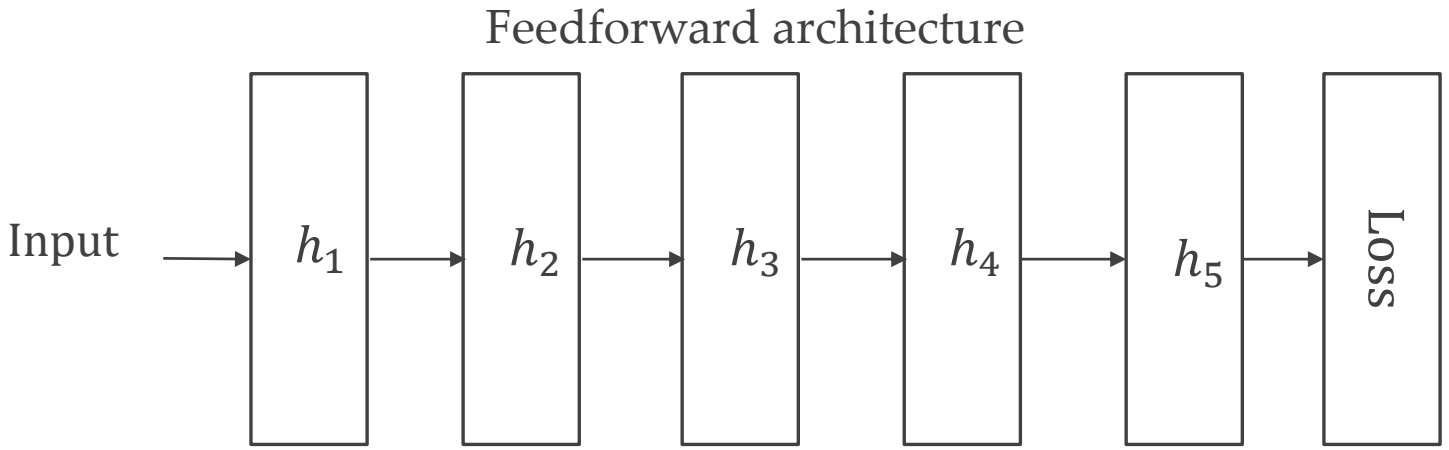
(1) The activation functions must be **1st-order differentiable (almost) everywhere**

(2) Take special care when there are cycles in the architecture of blocks

- No other requirements
- We can build as complex hierarchies as we want

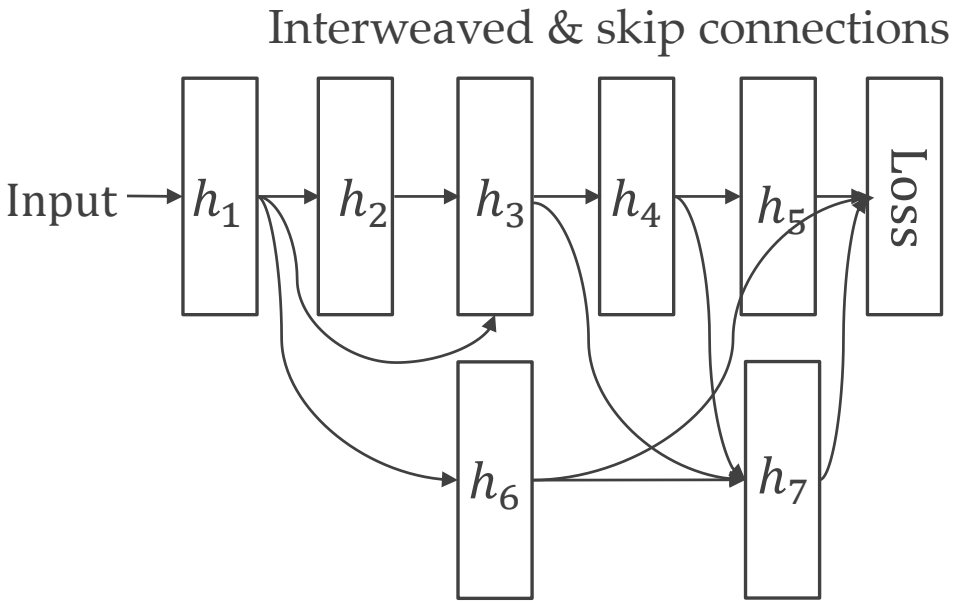
Feedforward model

- The vast majority of models
- Almost all CNNs out there
- As simple as it gets



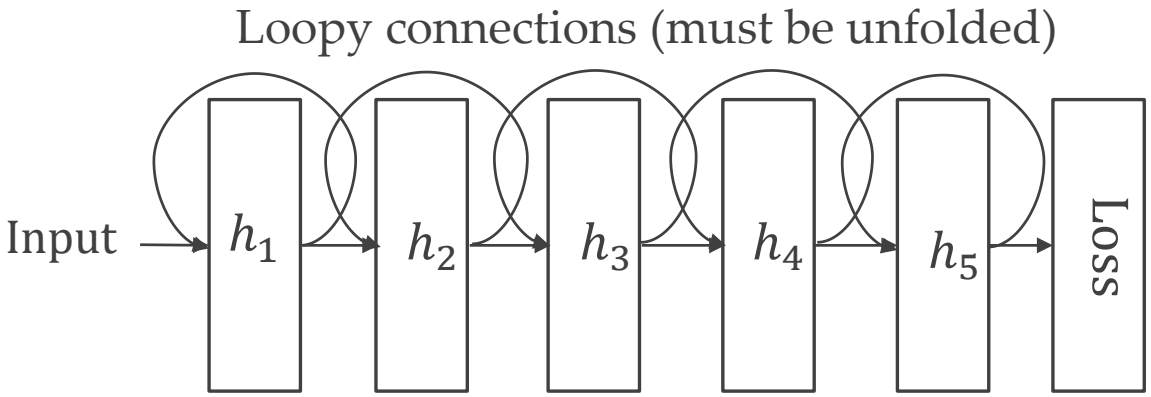
Directed acyclic graph models

- We can mix up our hierarchies
- Makes sense when good knowledge of problem domain
- Makes sense when combining multiple inputs & modalities
  - *E.g.*, RGB & LIDAR



Loopy connections

- Module's past output is module's future input
- We must take care of cycles, *i.e.*, unfold the graph (later lecture)
- Often used with memory models, recurrent models



Hierarchies of modules

- Data efficient modules and hierarchies
  - Trade-off between model complexity and efficiency
  - Often, more training iterations with a "weaker" model better than fewer with a "stronger" one
  - ReLUs are basically half-linear functions, but give SoTA (also) because they train faster
- Not too complex modules, better complex hierarchies
  - Again, ReLUs are basically half-linear functions, but give SoTA
- Use parameters smartly
  - Often, the real constraint is GPU memory. What is the best way to distribute our parameters?
- Compute modules in the right order to feed next modules