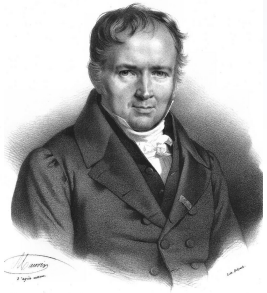
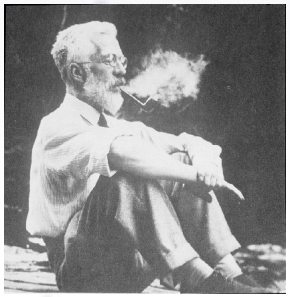
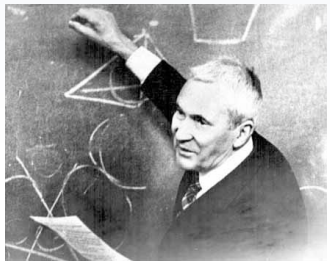
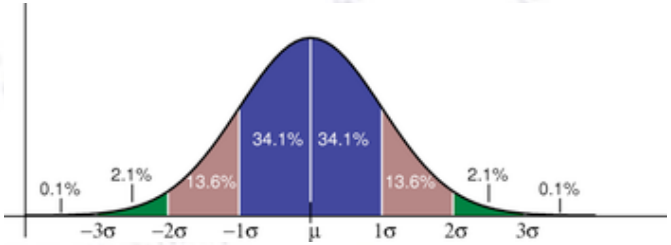
Neural Networks & Deep Learning



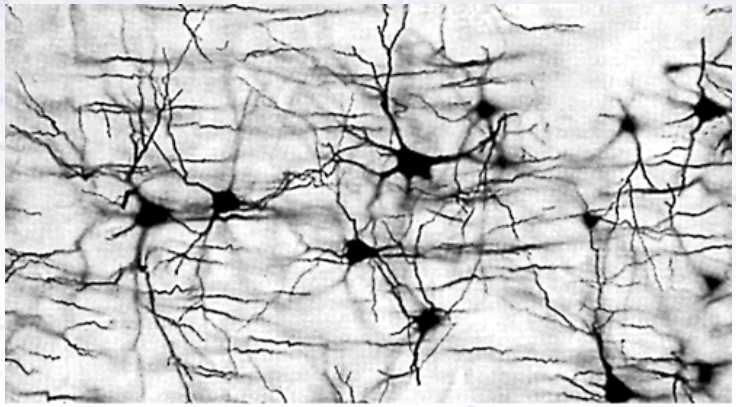





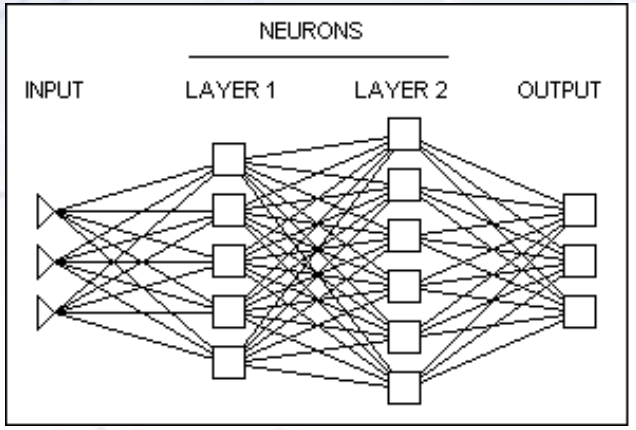
Troels C. Petersen (NBI)



"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

Neural Networks (NN)





In machine learning and related fields, artificial neural networks (ANNs) are computational models inspired by an animal's central nervous systems (in particular the brain) which is capable of ***machine learning*** *as well as* ***pattern recognition****.*

Neural networks *have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including* ***computer vision*** *and* ***speech recognition****.*

[Wikipedia, Introduction to Artificial Neural Network]

Neural Networks

Neural Networks combine the input variables using a "activation" function s(x) to assign, if the variable indicates signal or background.

The simplest is a single layer perceptron:

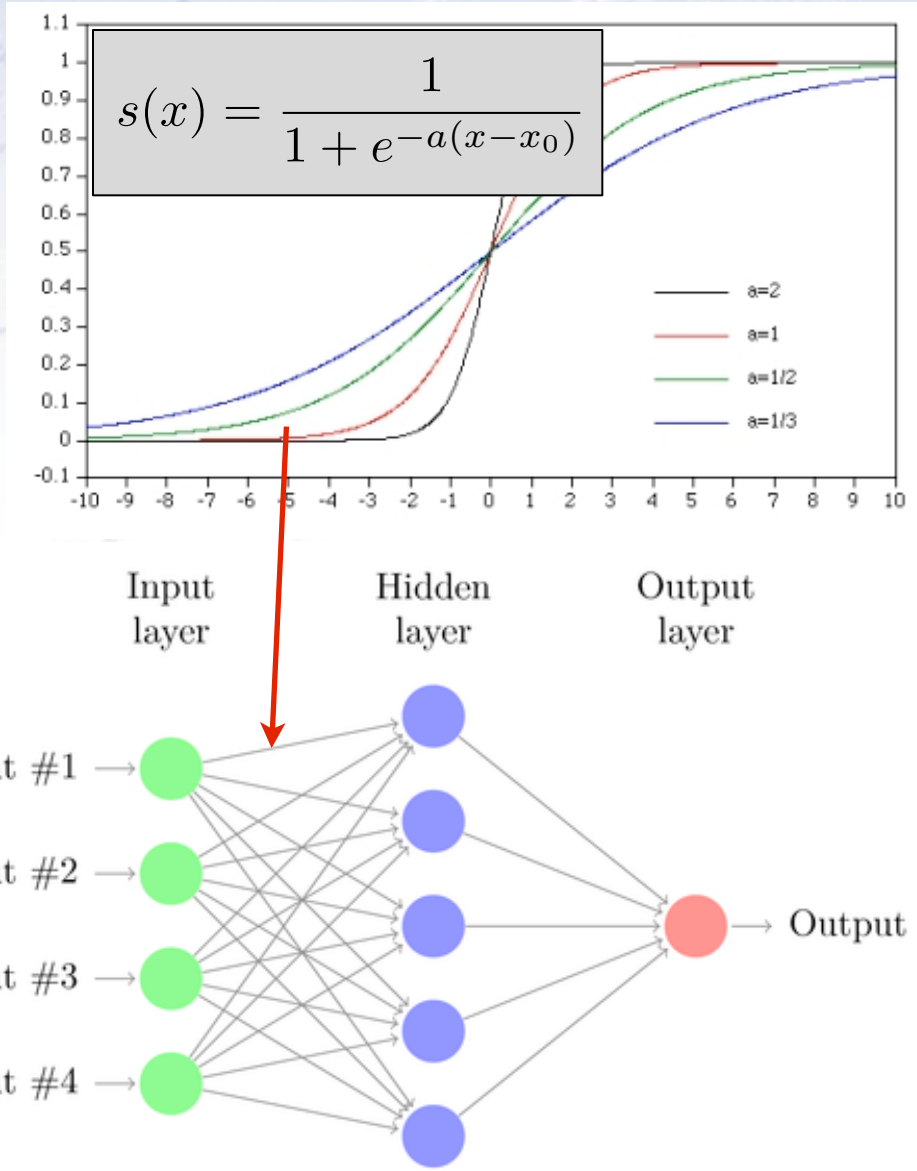
$$t(x) = s\left(a_0 + \sum a_i x_i\right)$$

This can be generalised to a multilayer perceptron:

$$t(x) = s\left(a_i + \sum a_i h_i(x)\right)$$

$$h_i(x) = s\left(w_{i0} + \sum w_{ij} x_j\right)$$

Activation function can be any "sigmoidal" function.



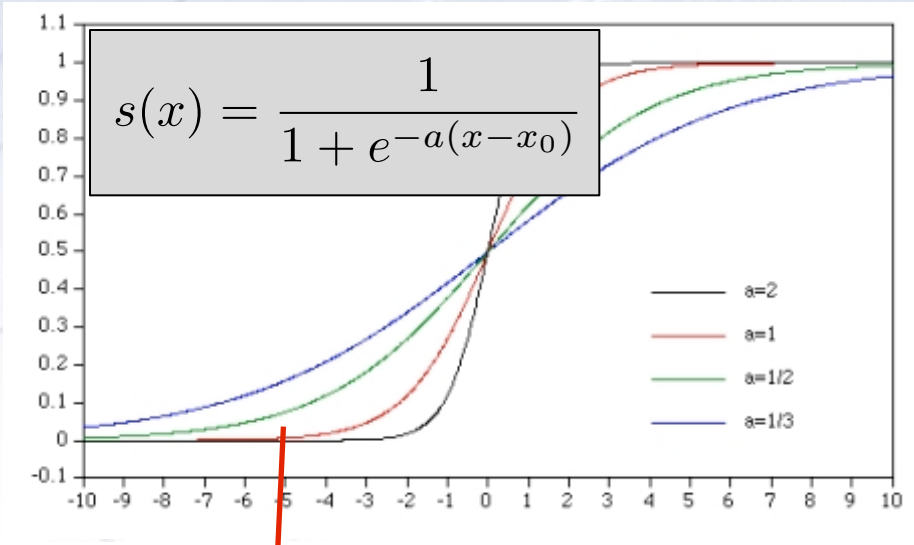
Neural Networks

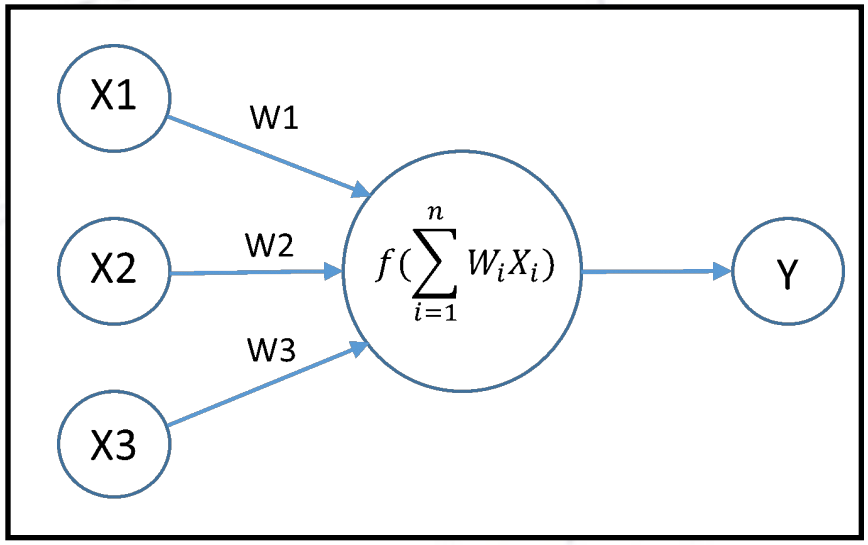
Neural Networks combine the input variables using a "activation" function s(x) to assign, if the variable indicates signal or background.

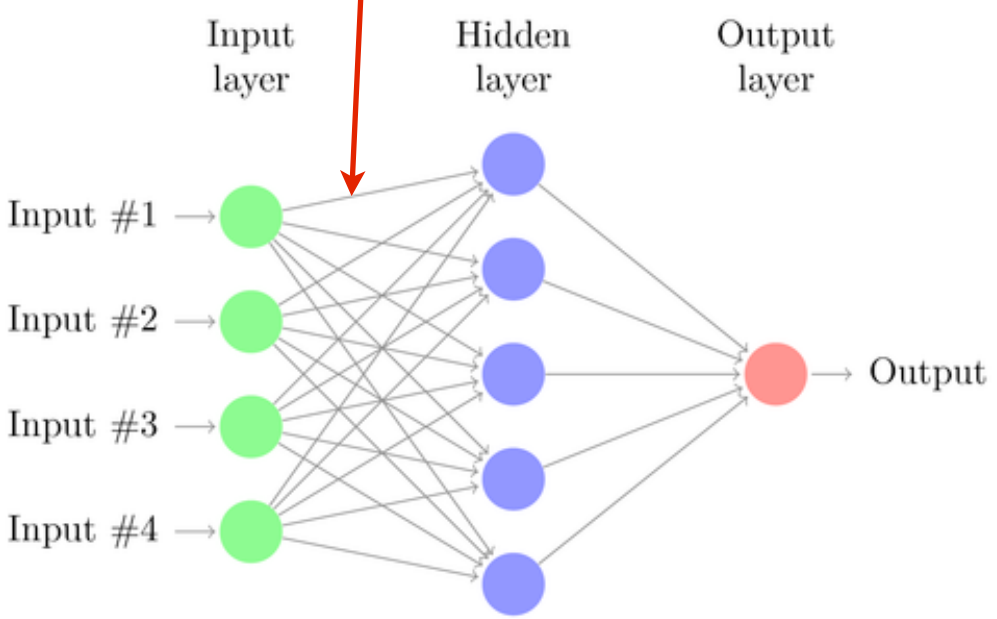
The simplest is a single layer perceptron:

$$t(x) = s\left(a_0 + \sum a_i x_i\right)$$

$$s(x) = \frac{1}{1 + e^{-a(x - x_0)}}$$







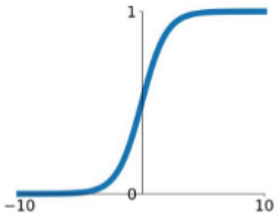
Activation Functions

There are many different activation functions, some of which are shown below. They have different properties, and can be considered a HyperParameter.

Activation Functions

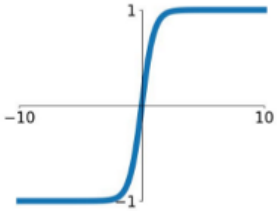
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



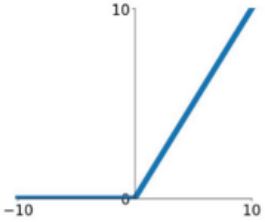
tanh

$$\tanh(x)$$



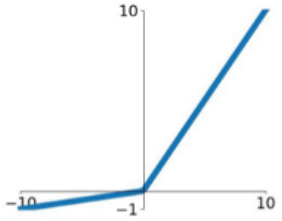
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

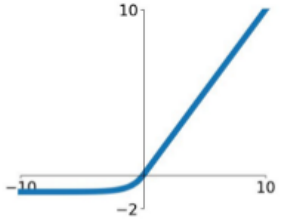


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

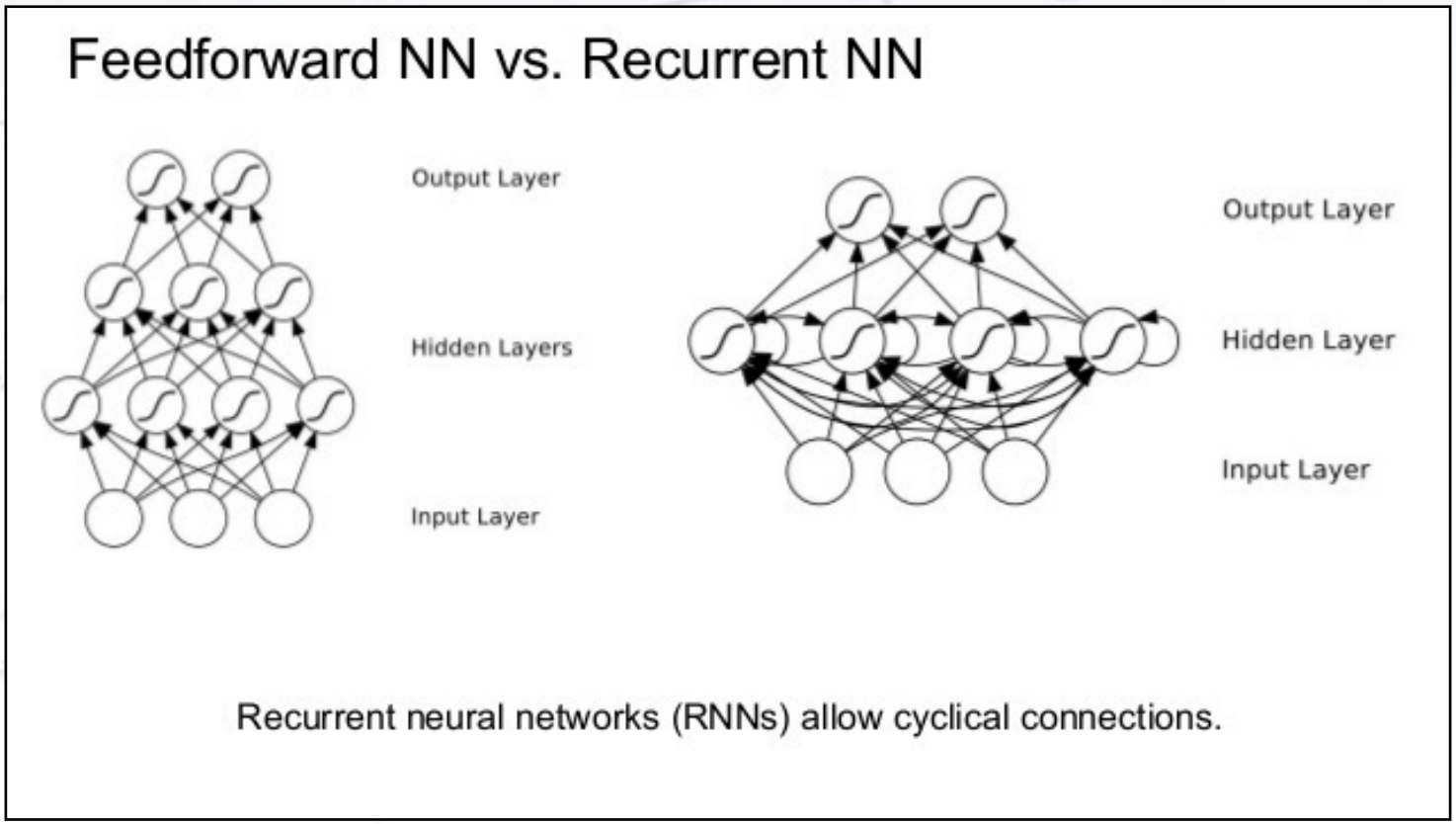


For a more complete list, check: https://en.wikipedia.org/wiki/Activation_function

Recurrent NN

Normally, the information from one layer is fed forward to the next layer in a feedforward Neural Network (NN).

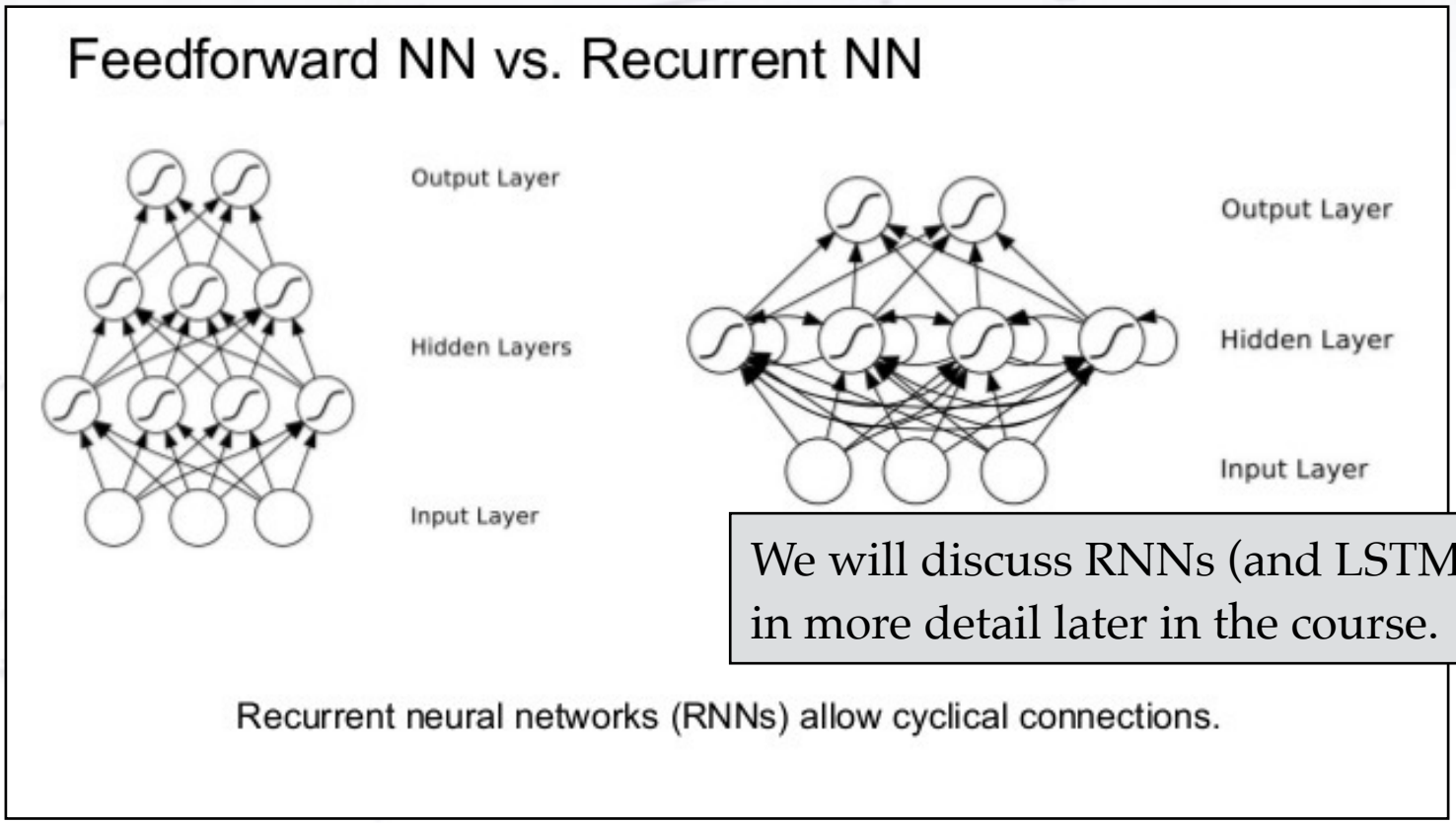
However, it may be of advantage to allow a network to give feedback, which is called a recurrent NN:



Recurrent NN

Normally, the information from one layer is fed forward to the next layer in a feedforward Neural Network (NN).

However, it may be of advantage to allow a network to give feedback, which is called a recurrent NN:

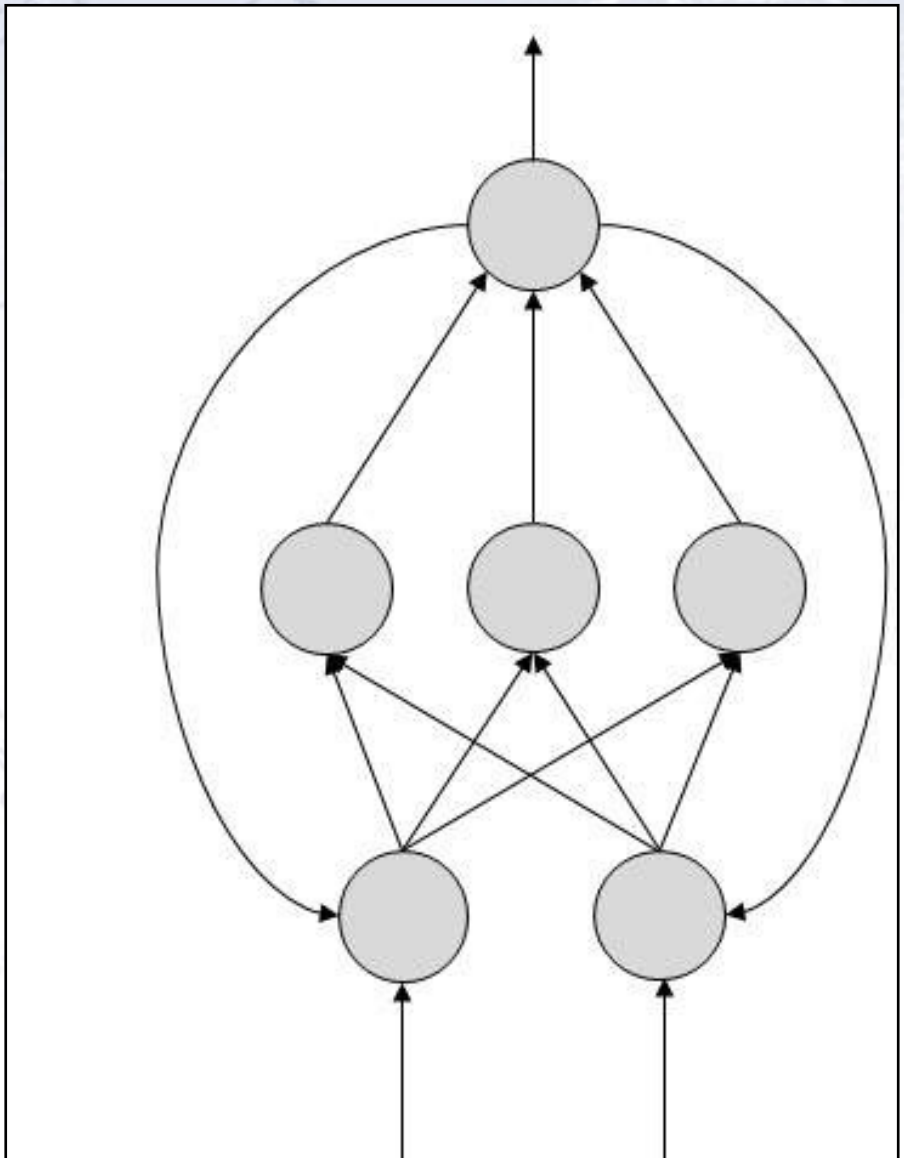


Feedback network

There is nothing that prohibits the use of feedback in the network.

In this way, one can pass information "back" in the network, allowing for input of "more advanced" neurons to earlier neurons.

Note, that it requires skill and knowledge (and time and hard work) to design the network that suits your problem!



Networks with "memory"

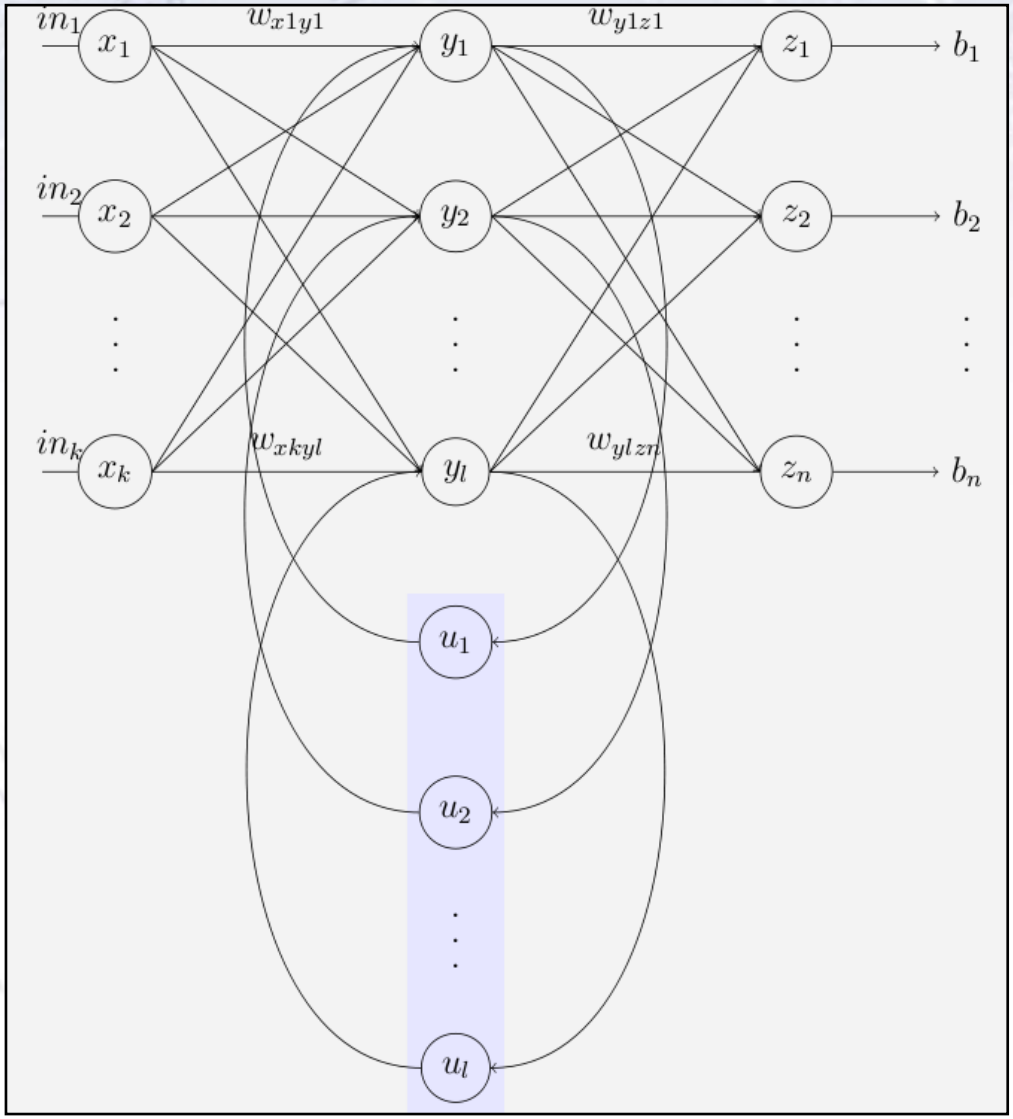
So-called Elman and Jordan networks

Allowing for feedback, one can also use this for providing "memory" of the last state(s) of the network.

This can be used for including "context" or "environment" in the network.

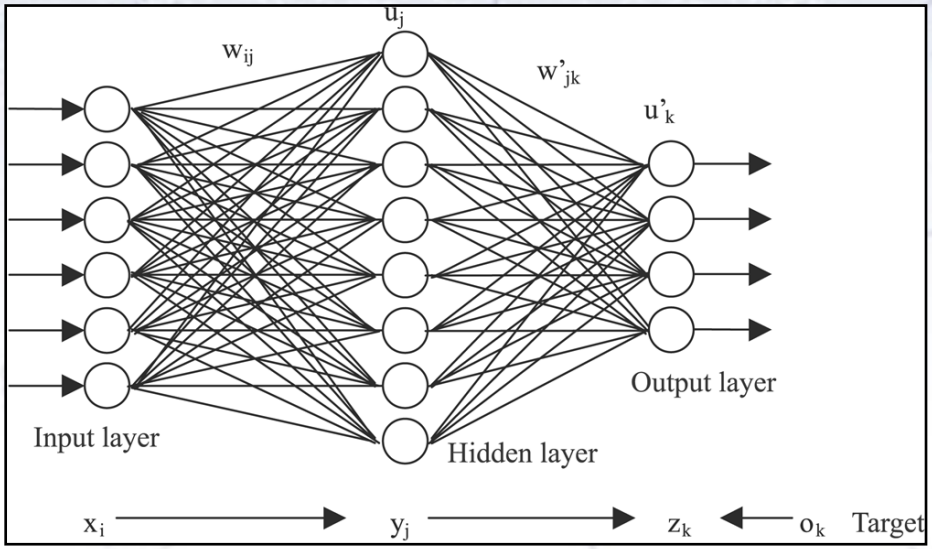
This can be used in case of e.g. a new user regarding adds, a new context regarding translation,

The keyword is Long Short-Term Memory (LSTM), if you want to look for more…



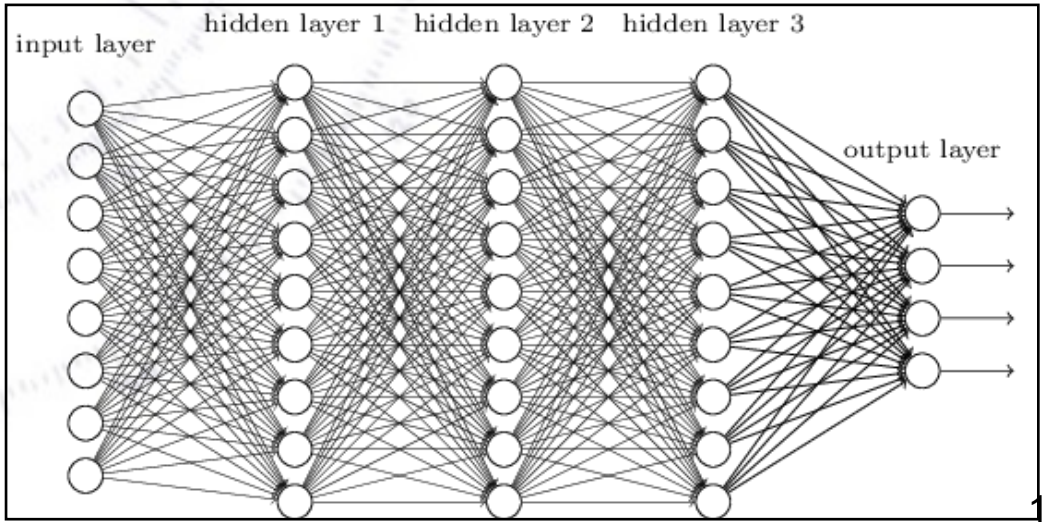
Deep Neural Networks

Deep Neural Networks (DNN) are simply (much) extended NNs in terms of layers!



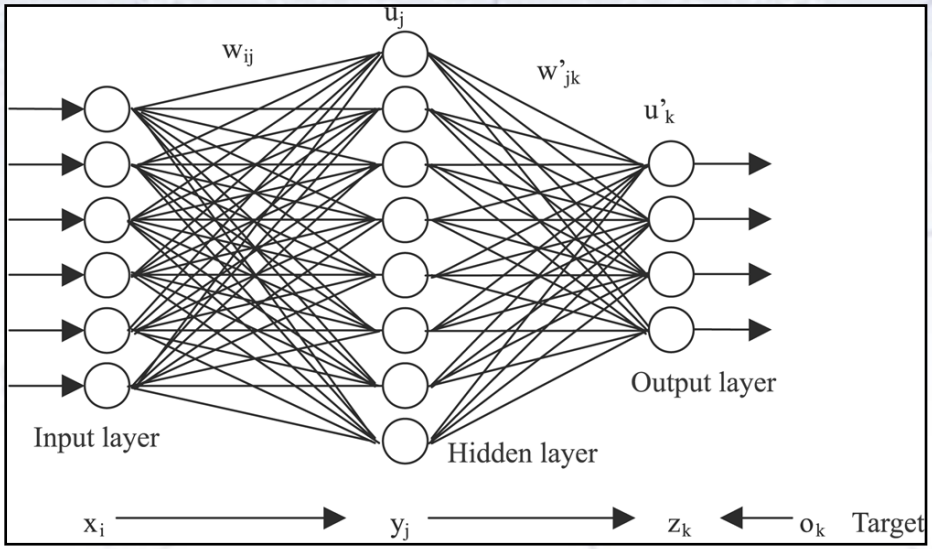
Instead of having just one (or few) hidden layers, many such layers are introduced.
This gives the network a chance to produce key features and use them for many different specialised tasks.

Currently, DNNs can have up to millions of neurons and connections, which compares to about the **brain of a worm**.



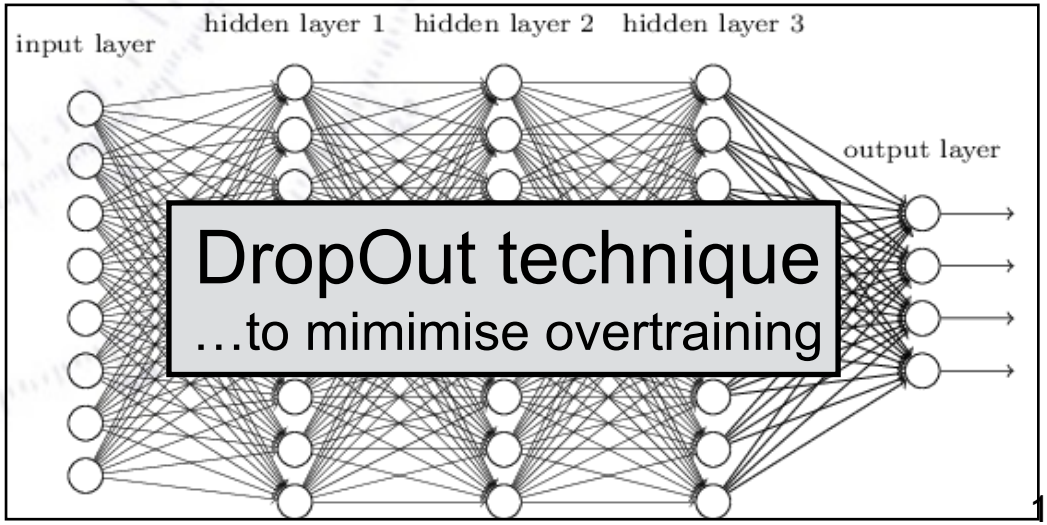
Deep Neural Networks

Deep Neural Networks (DNN) are simply (much) extended NNs in terms of layers!



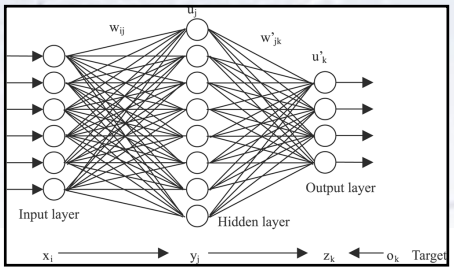
Instead of having just one (or few) hidden layers, many such layers are introduced.
This gives the network a chance to produce key features and use them for many different specialised tasks.

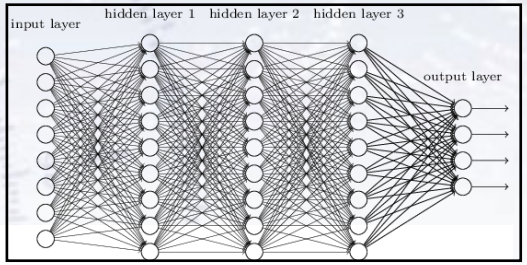
Currently, DNNs can have up to millions of neurons and connections, which compares to about the **brain of a worm**.

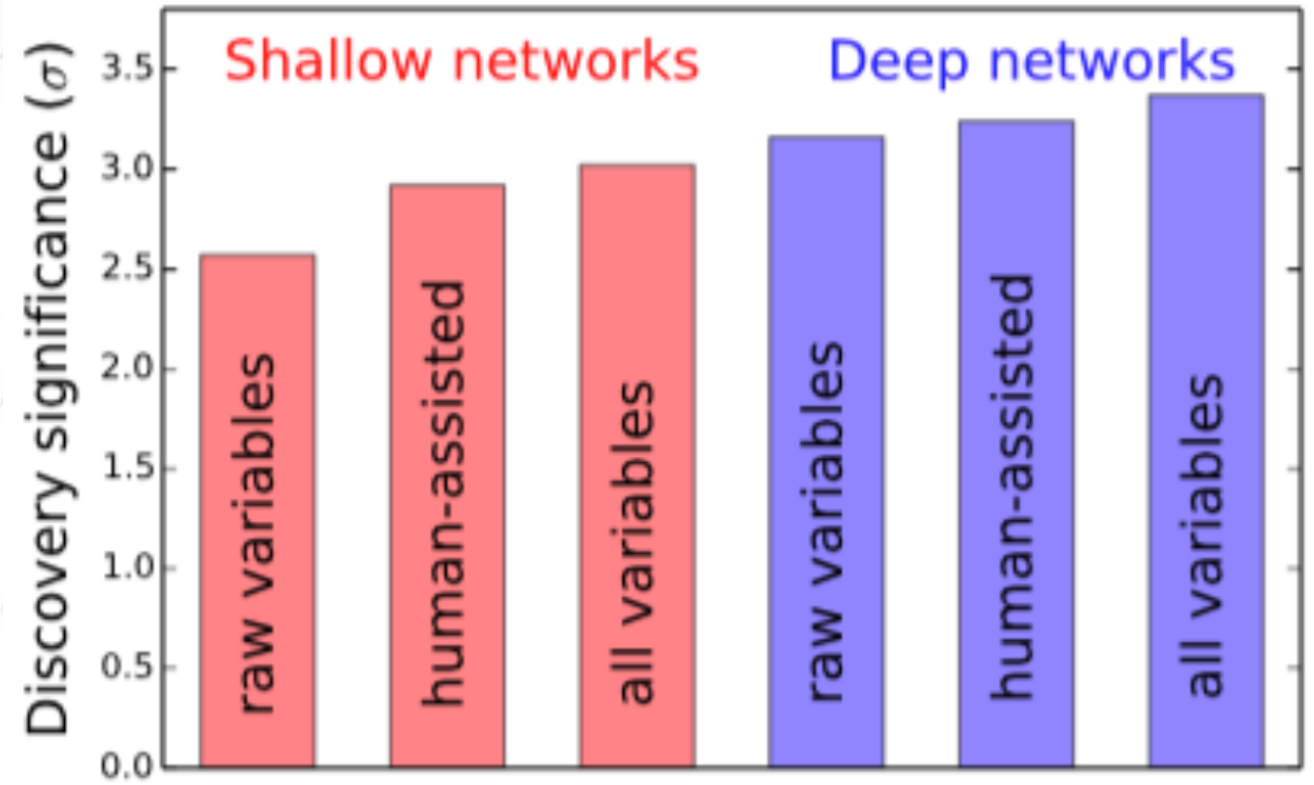


Deep Neural Networks

Deep Neural Networks likes to get both raw and "assisted" variables:

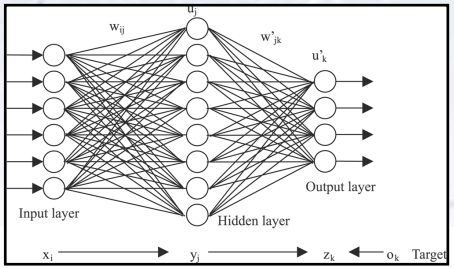


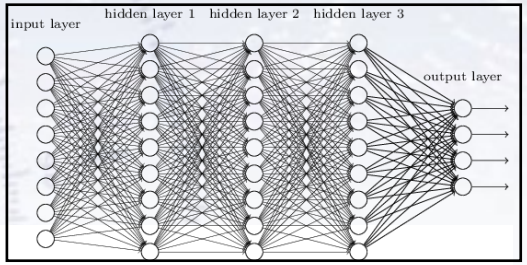


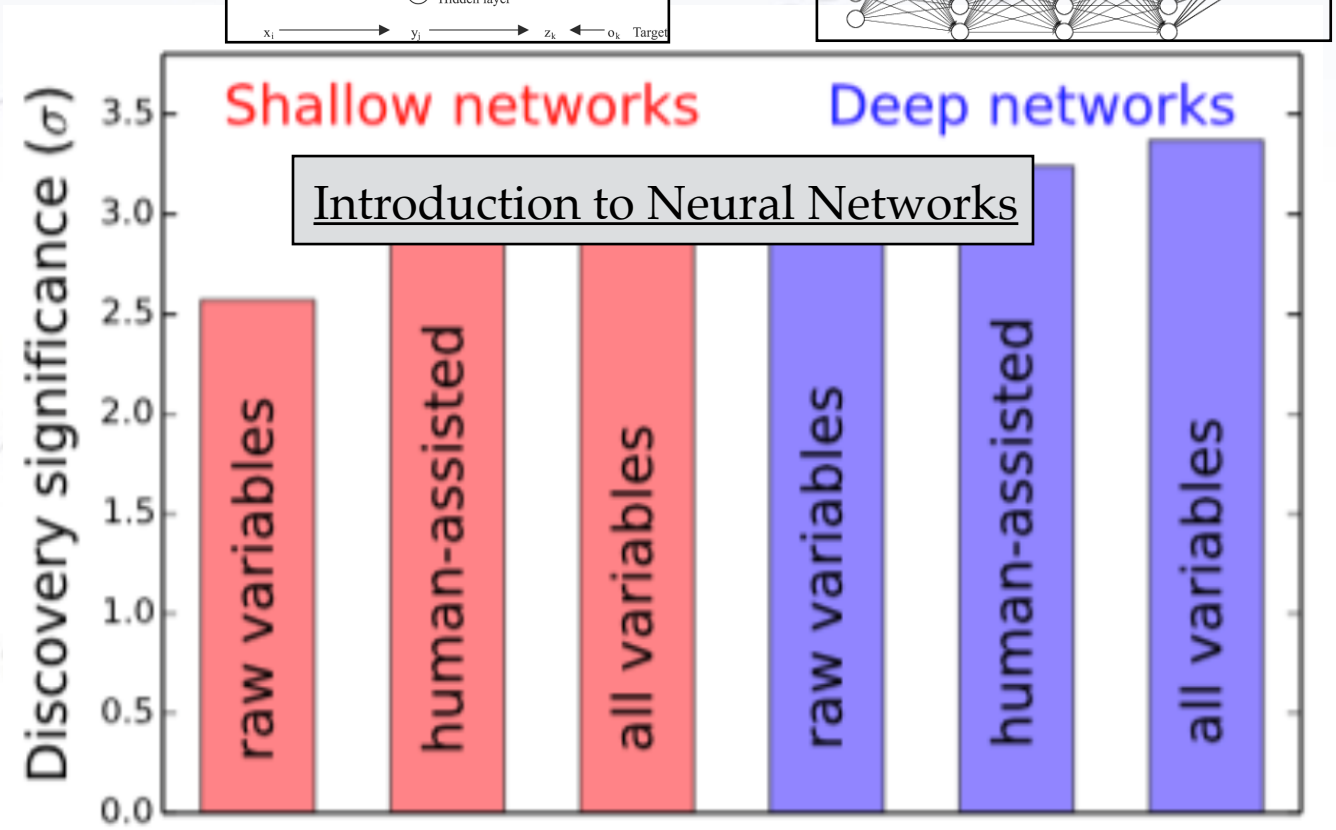


Deep Neural Networks

Deep Neural Networks likes to get both raw and "assisted" variables:



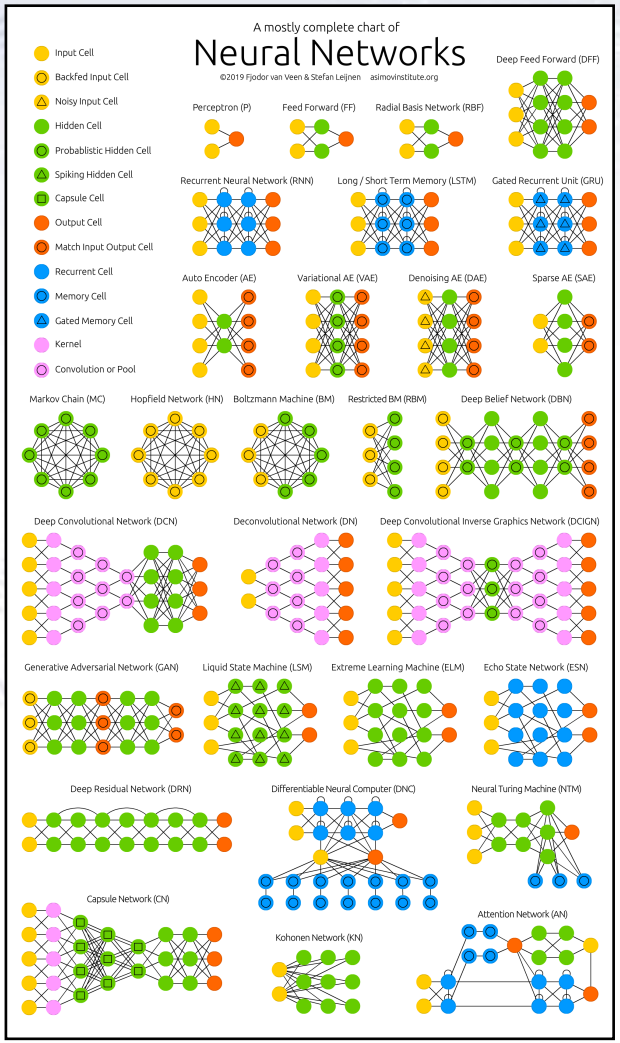




The role of NNs

The reason why NNs play such a central role is that they are versatile:

- Recurrent NNs (for time series)
- Convolutional NNs (for images)
- Adversarial NNs (for simulation)
- Graph NNs (for geometric data)
- etc.



Unlike trees, NNs typically make the "foundation" of all the more advanced ML paradigms. However, they are harder to optimise!
This is why trees a great for simpler tasks (i.e. data that typically fits into an excel sheet [2110.01889]), while NNs are typically used for the more advanced.

Have this in mind, when you attack problems with ML - and like any other project or analysis, it is typically good to get a "rough result" fast, and then to refine it from there.