Topic 6
Introduction to Neural Networks (NN) & Deep Learning (DL)

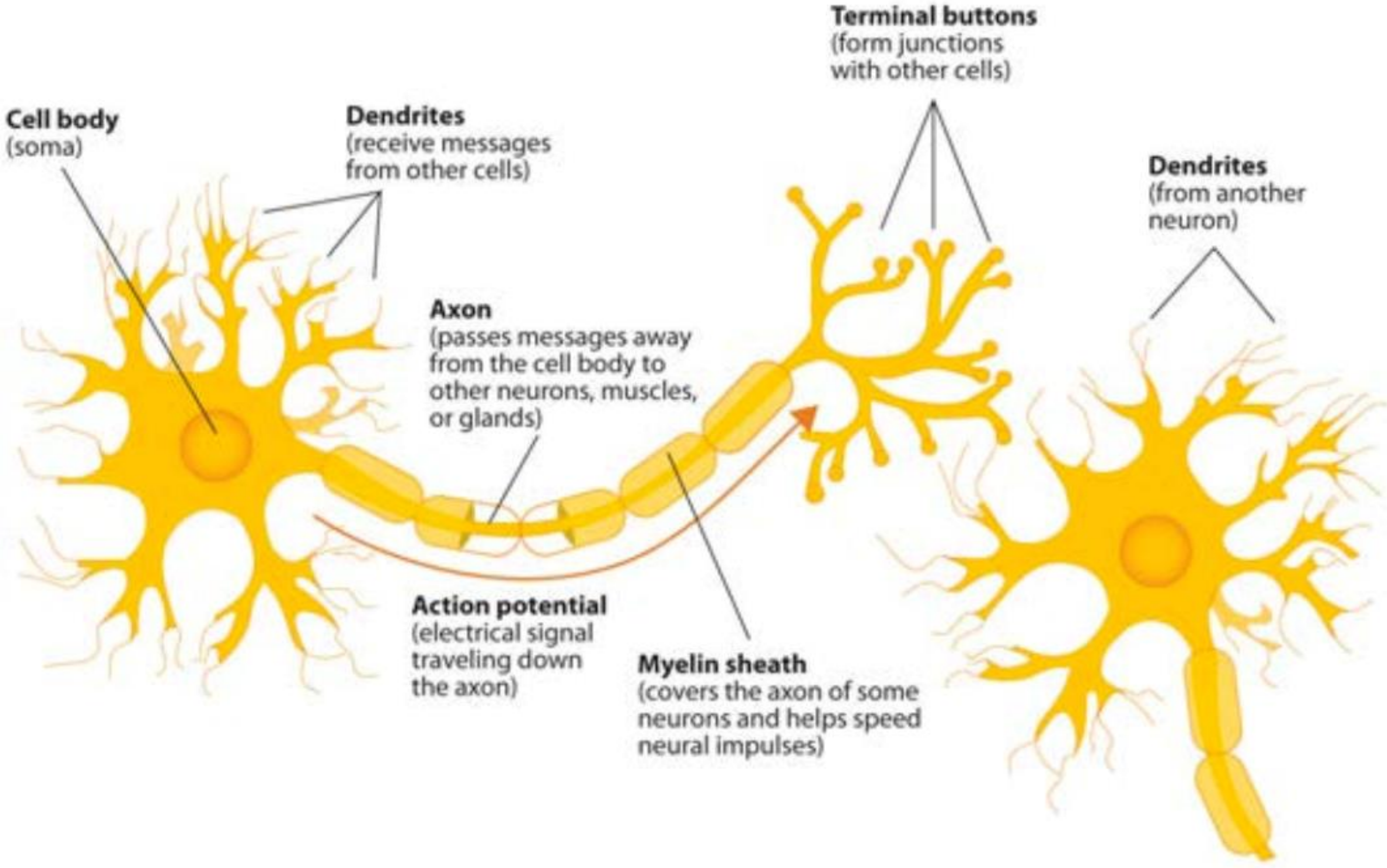
ST0249 (AIML) AI & MACHINE LEARNING

Learning Outcomes

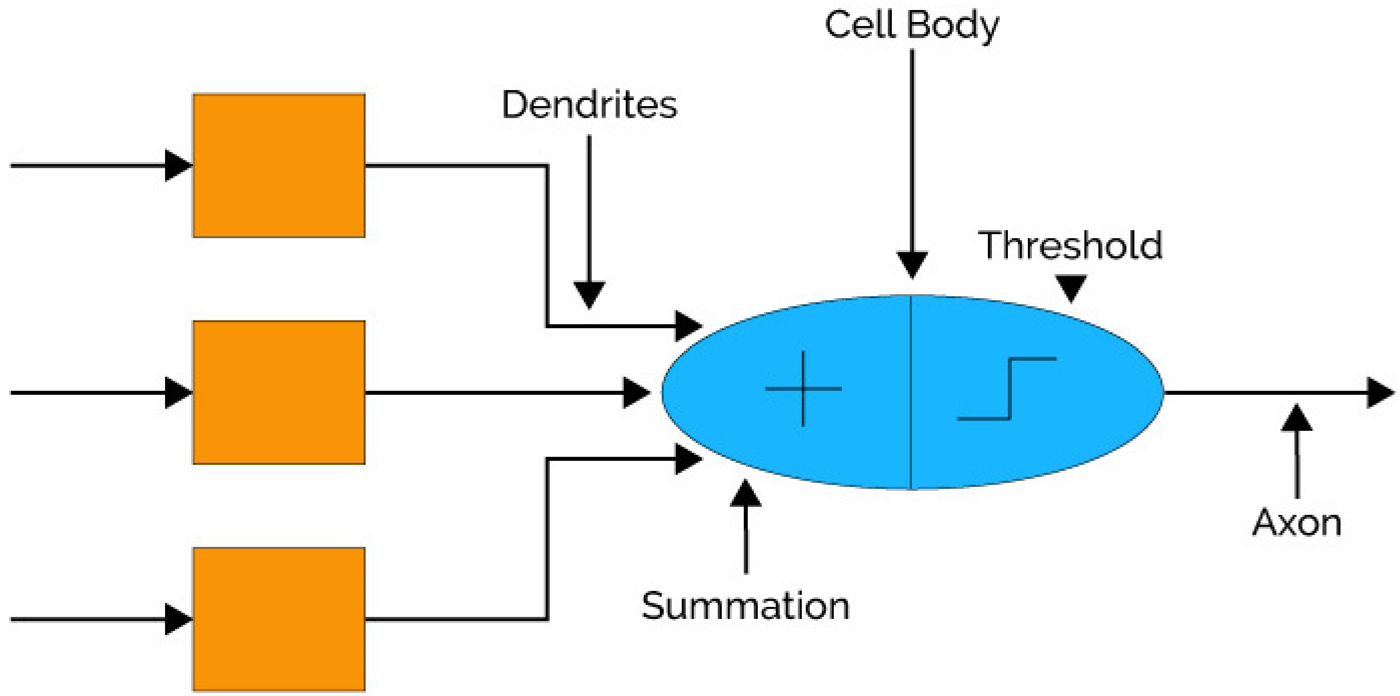
- Understand Neural Networks and Deep Learning
  - Understand Neural Networks
  - Explain machine learning based on learning data representations
  - Understand what is Deep Learning
  - Applications of Deep Learning

Understanding Neural Networks

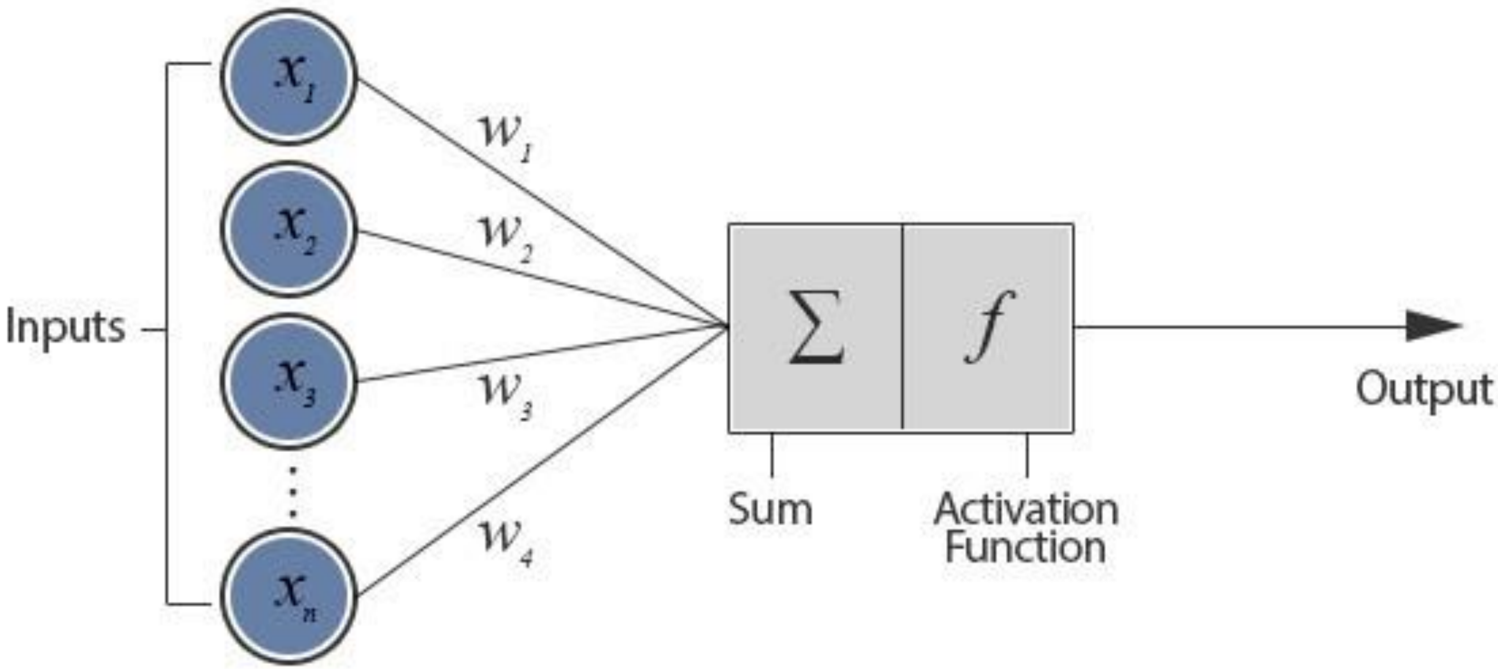
Biological Neuron



Biological Neuron (Simplification)

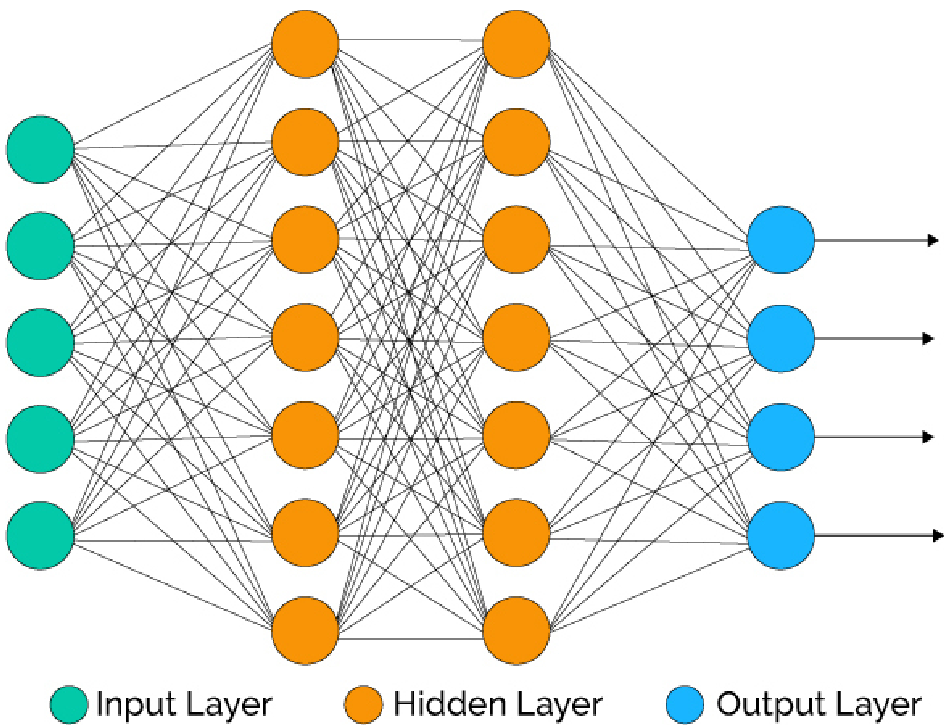


Artificial Neuron



Neural Network

A typical **Neural Network** contains a large number of artificial neurons called units arranged in a series of layers.



Activation Function

The **activation function** of a node defines the output of that node given an input or set of inputs. In artificial neural networks this function is also called the **transfer function**.

Common activation functions (for hidden layers)

- Sigmoid/Logistic
- Tanh (Hyperbolic Tangent)
- ReLU (Rectified Linear Unit) [Leaky ReLU and Maxout]

Common activation functions (for output layers)

- Linear (for regression)
- Softmax (for classification)

source https://en.wikipedia.org/wiki/Activation_function

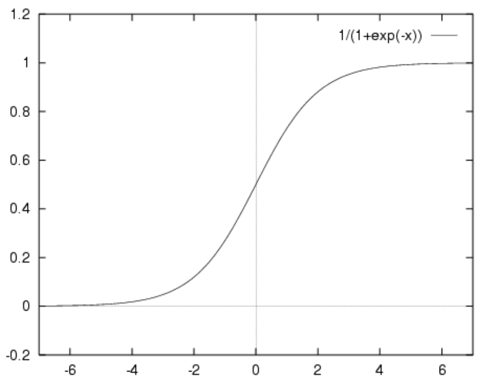
Activation Function

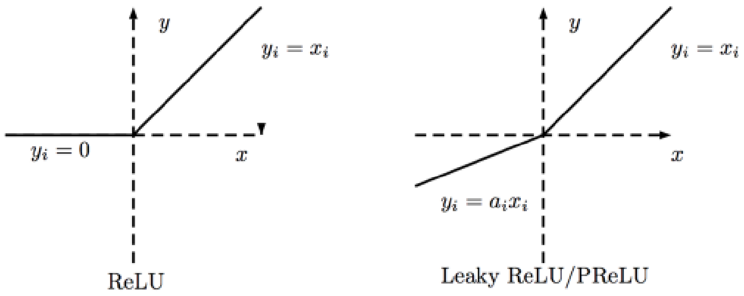
- Why is an activation function needed?
  - If we do not apply a Activation function then the output signal would simply be a simple linear function. Now, a linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data.
  - A Neural Network without Activation function would simply be a **Linear regression Model**, which has limited power and does not performs good most of the times.
- An Artificial Neural Network is a **Universal Function Approximator**
  - It means that they can compute and learn any function at all. Almost any process we can think of can be represented as a functional computation in Neural Networks.
  - Using a non linear Activation we are able to generate non-linear mappings from inputs to outputs.

source https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Problems with Activation Functions

- Vanishing gradient problem for Sigmoids
  - Updating NN weights is affected by gradient of error function which may become smaller and smaller
  - Sigmoid gradient is limited to 0..1
- Dead neurons problem for ReLU
  - Problem is that some neurons may have their weights updated into the 0 (dead) region and stop having any output.
  - Leaky ReLU or Maxout solves this



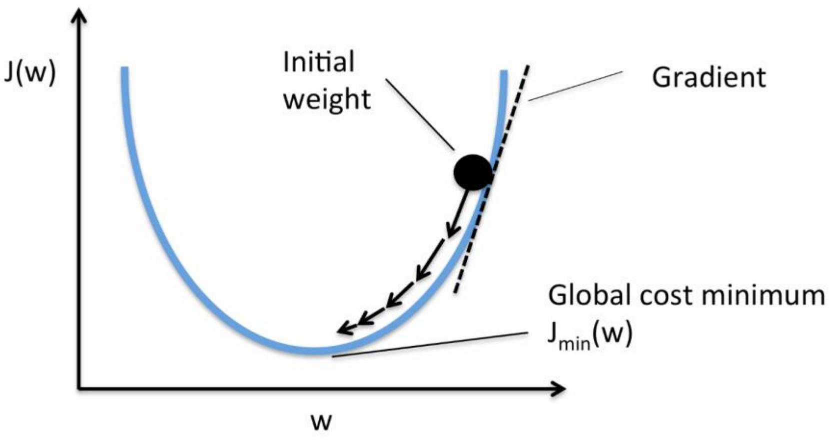


No vanishing gradient problem.
Almost all deep learning Models use ReLU nowadays.

Training the Neural Network

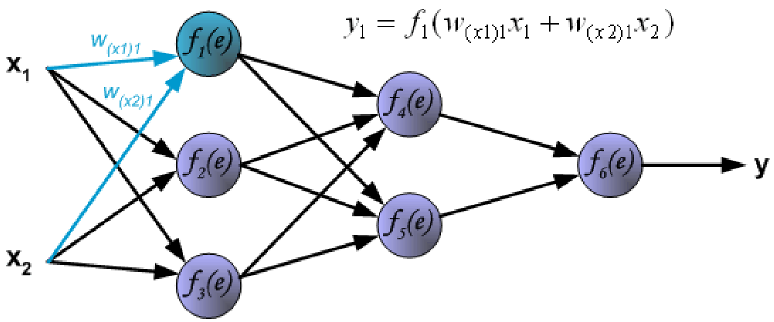
- A key feature of neural networks is an iterative learning process in which data cases (rows) are presented to the network one at a time, and the weights associated with the input values are adjusted each time.
- After all cases are presented, the process often starts over again.
- During this learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of input samples.

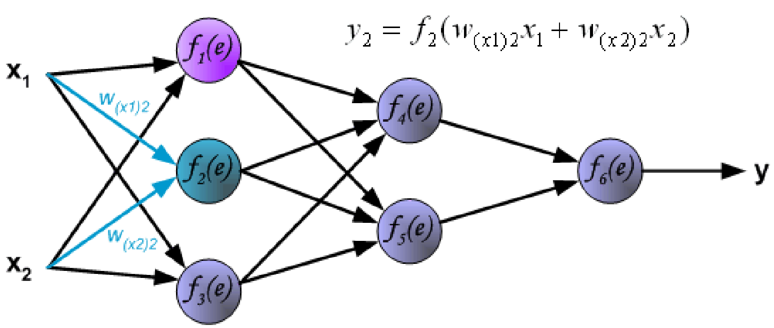
- The idea is to adjust the weights so that the total error moves in the direction of the minimum. The "gradient" determines the "direction" for weight adjustment.



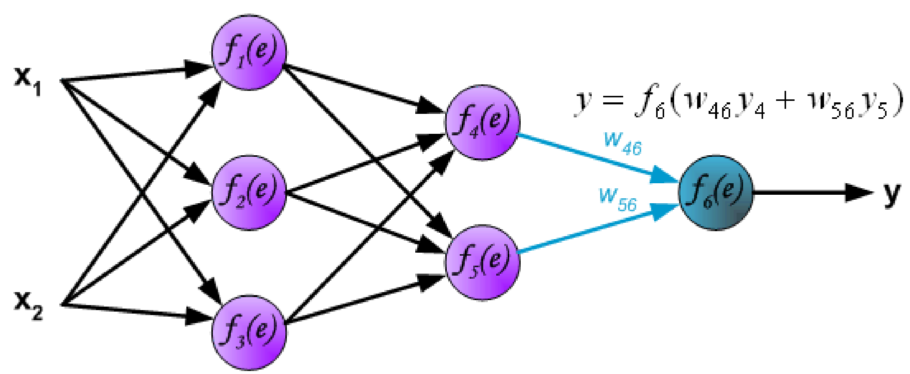
source https://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/

Understand Backpropagation Visually (Forward Pass)



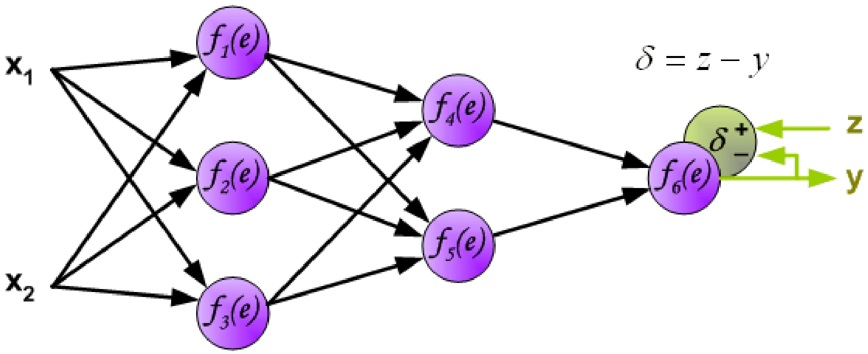


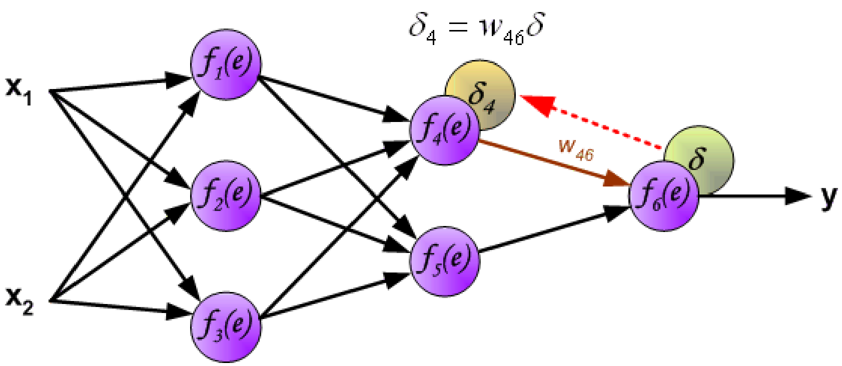
...

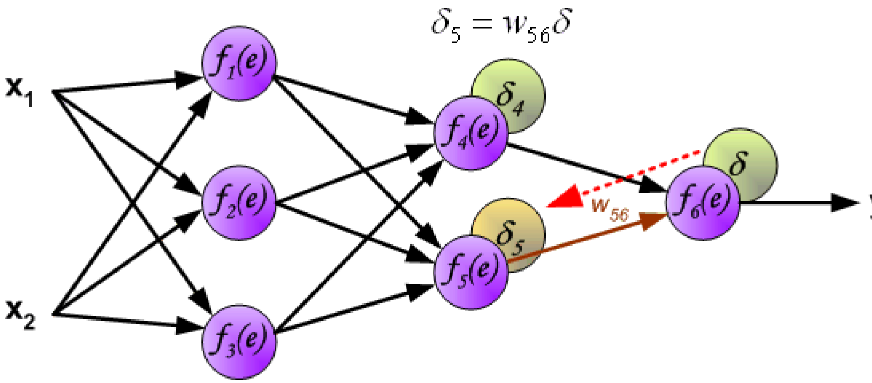


source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Backward Pass)



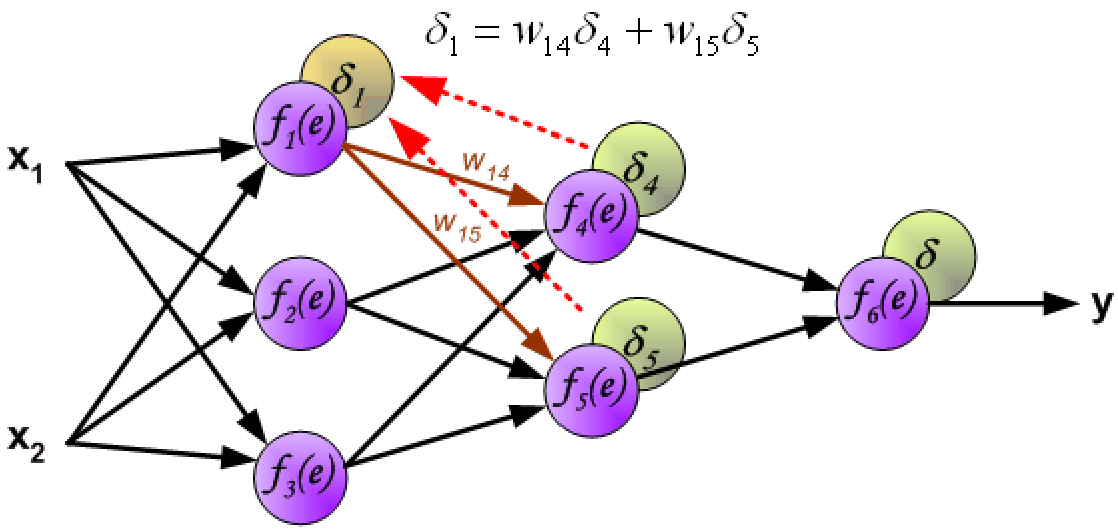




For many years the effective method for training multilayer networks has been unknown. Only in the middle 1980s the backpropagation algorithm has been worked out. The idea is to propagate error signal δ back to all neurons, which output signals were input for discussed neuron.

source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

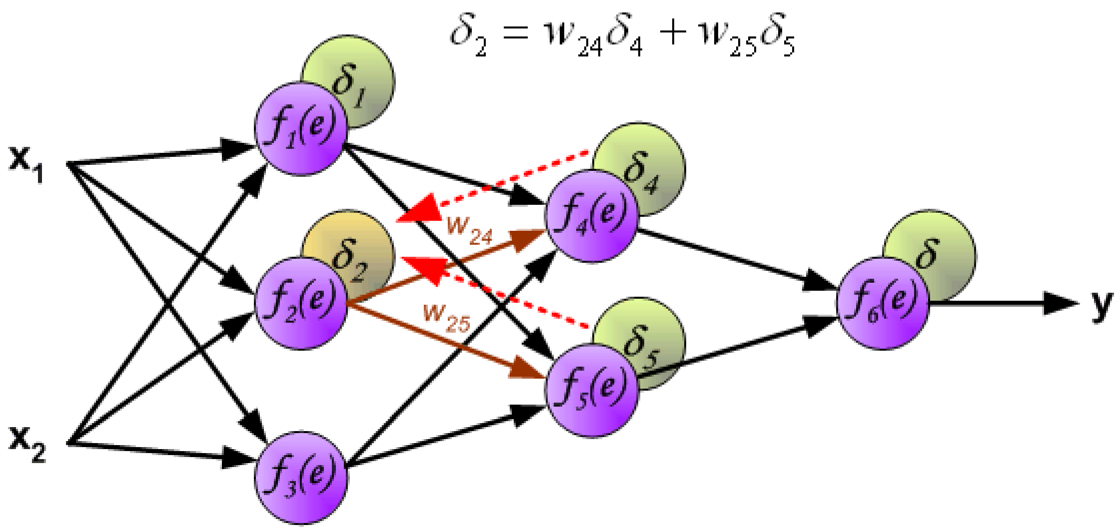
Understand Backpropagation Visually (Backward Pass)



The weights' coefficients w_{mn} used to propagate errors back are equal to this used during computing output value. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). This technique is used for all network layers.

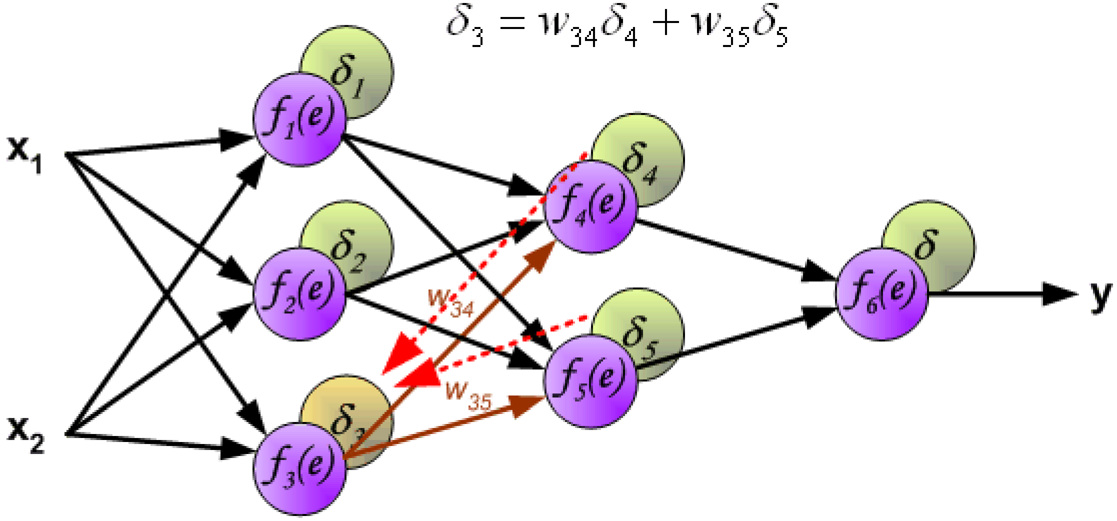
source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Backward Pass)



source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Backward Pass)



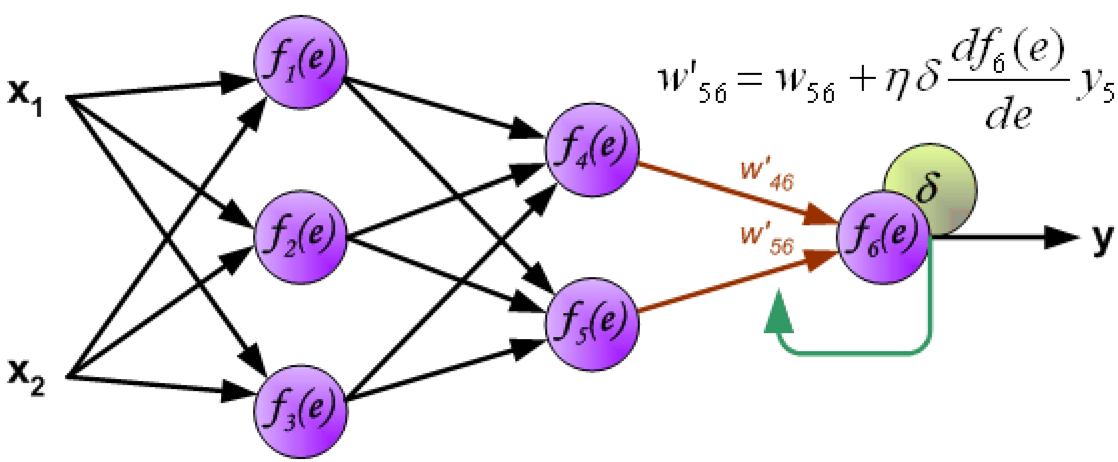
source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Update Weights)

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below *df(e)/de* represents derivative of neuron activation function (which weights are modified).

$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$



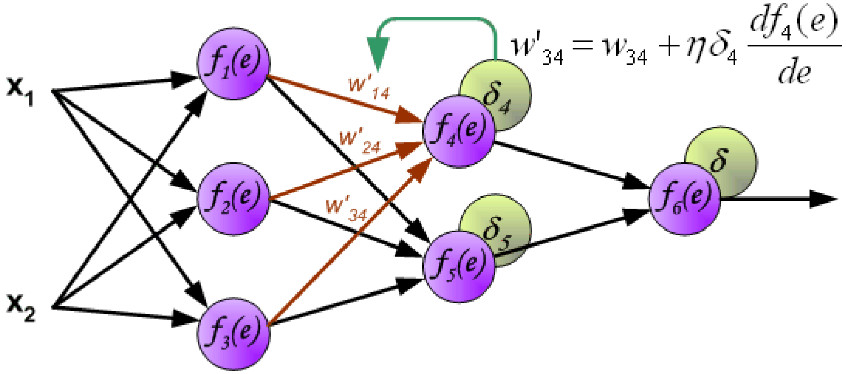
source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Update Weights)

$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$

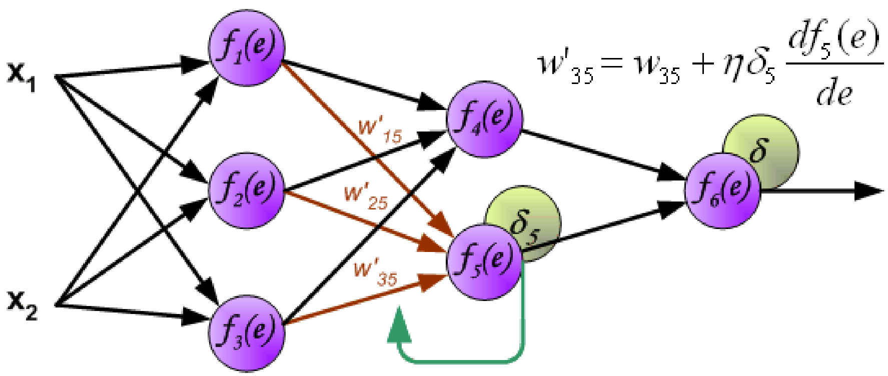
$$w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3$$



$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

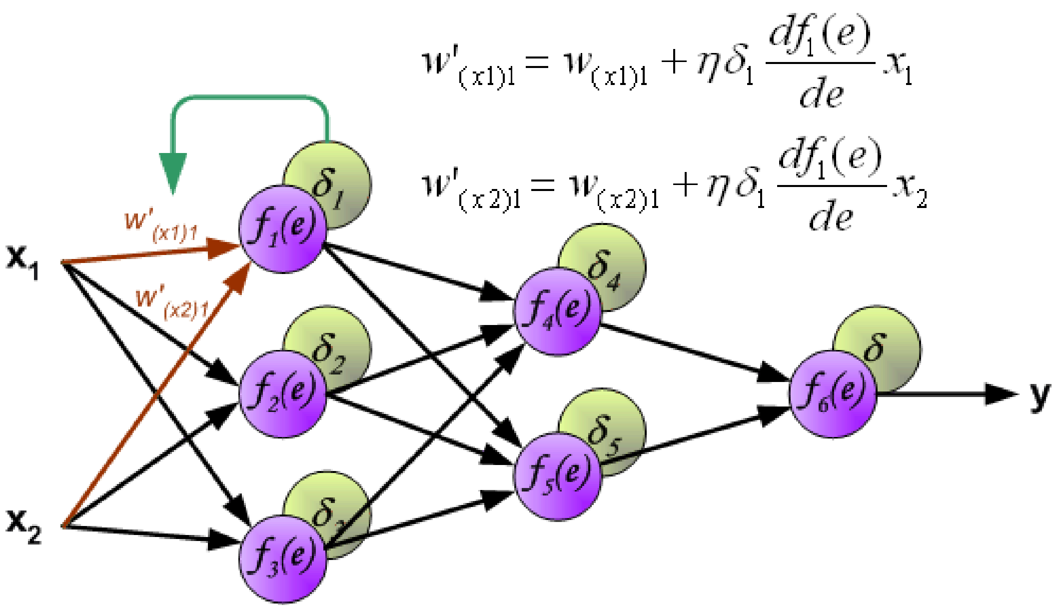
$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$



source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Update Weights)

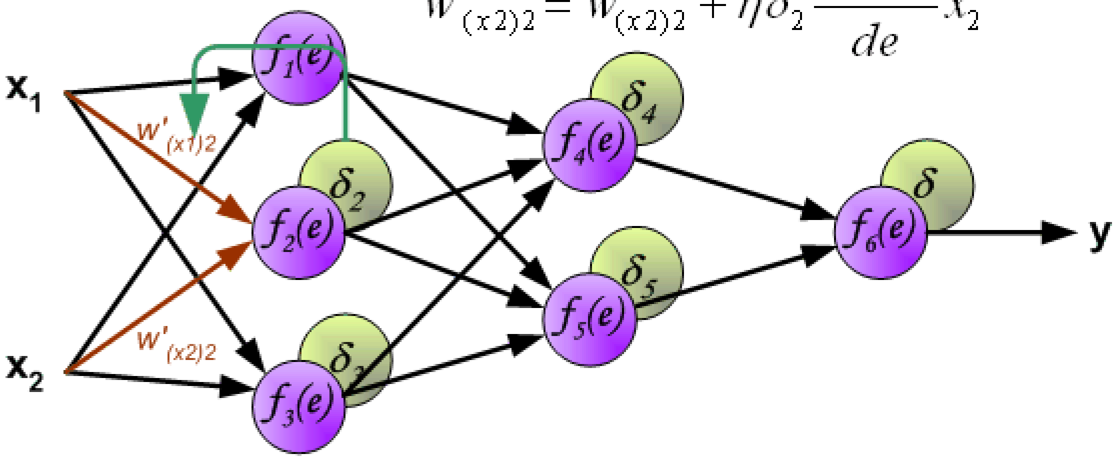


source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Update Weights)

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$

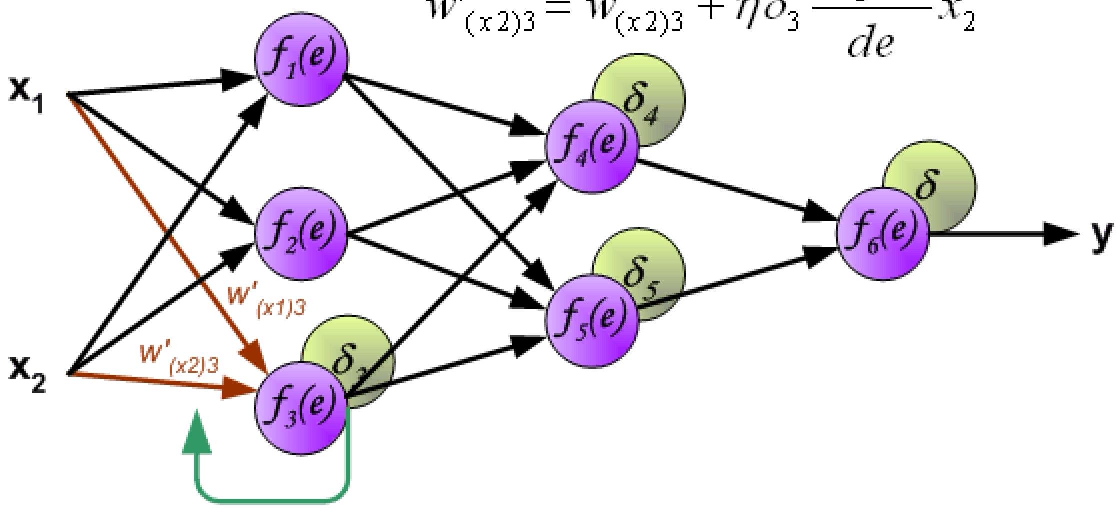


source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Understand Backpropagation Visually (Update Weights)

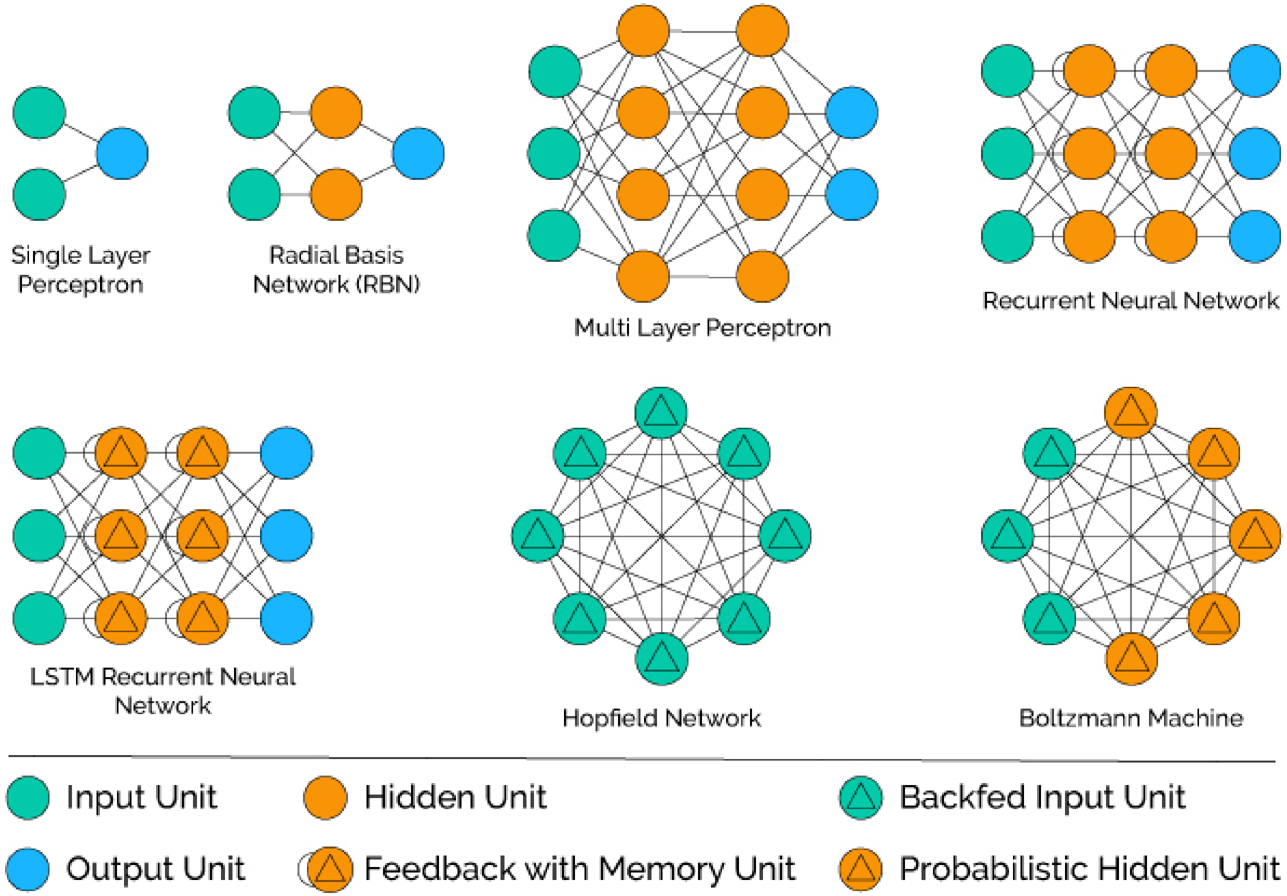
$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2$$



source http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

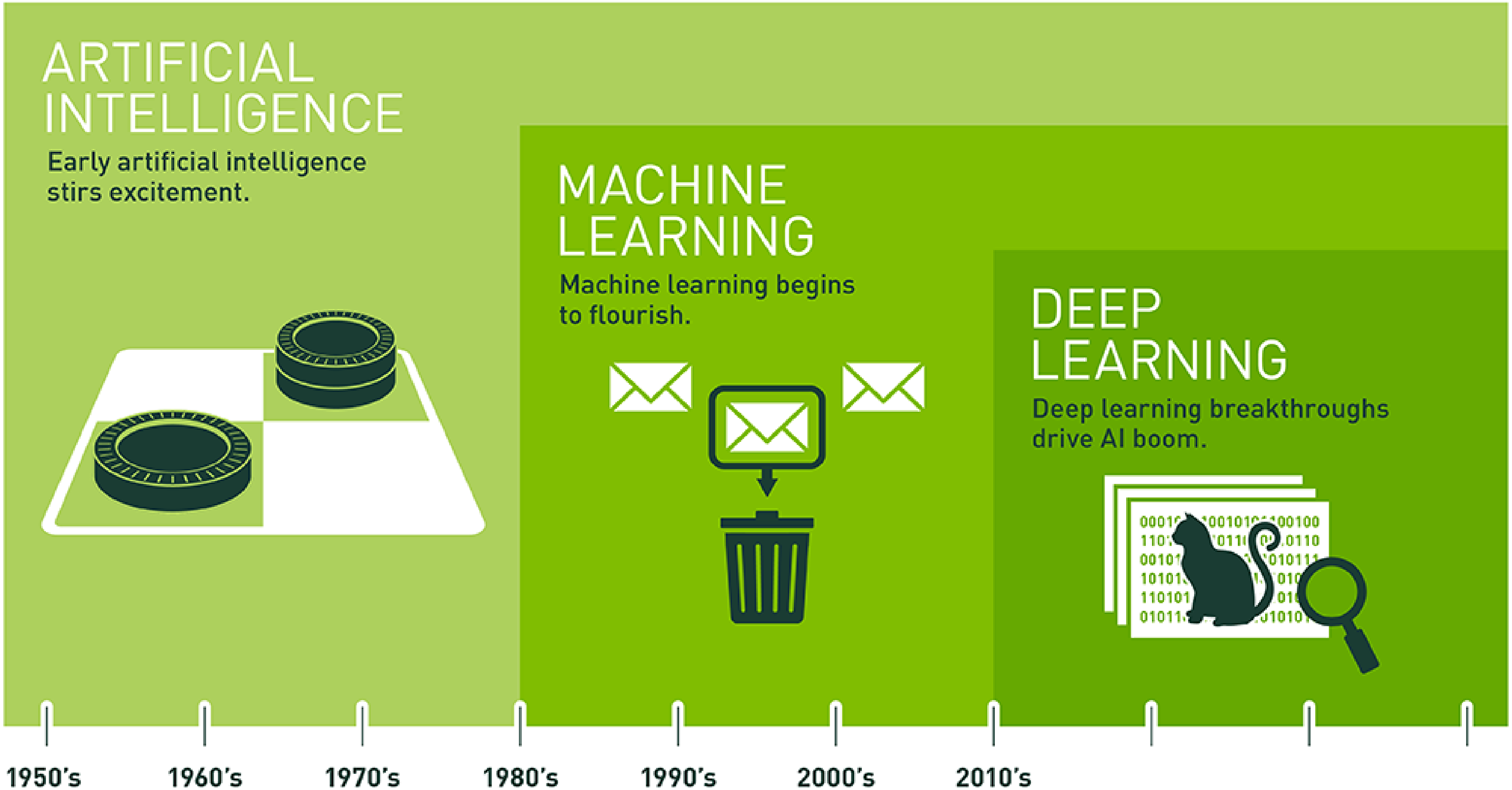
Types of Neural Networks



source https://www.xenonstack.com/blog/data-science/overview-of-artificial-neural-networks-and-its-applications

Parameter	Types	Description
Based on connection pattern	FeedForward, Recurrent	• Feedforward - In which graphs have no loops. • Recurrent - Loops occur because of feedback.
Based on the number of hidden layer	Single layer, Multi-Layer	• Single Layer - Having one hidden layer. E.g. , Single Perceptron • Multilayer - Having multiple hidden layers. Multilayer Perceptron
Based on nature of weights	Fixed, Adaptive	• Fixed - Weights are fixed a priori and not changed at all. • Adaptive - Weights are updated and changed during training.
Based on Memory unit	Static, Dynamic	• Static - Memoryless unit. The current output depends on the current input. E.g. , Feedforward network • Dynamic - Memory unit - The output depends upon the current input as well as the current output. E.g. , Recurrent Neural Network

What is Deep Learning?

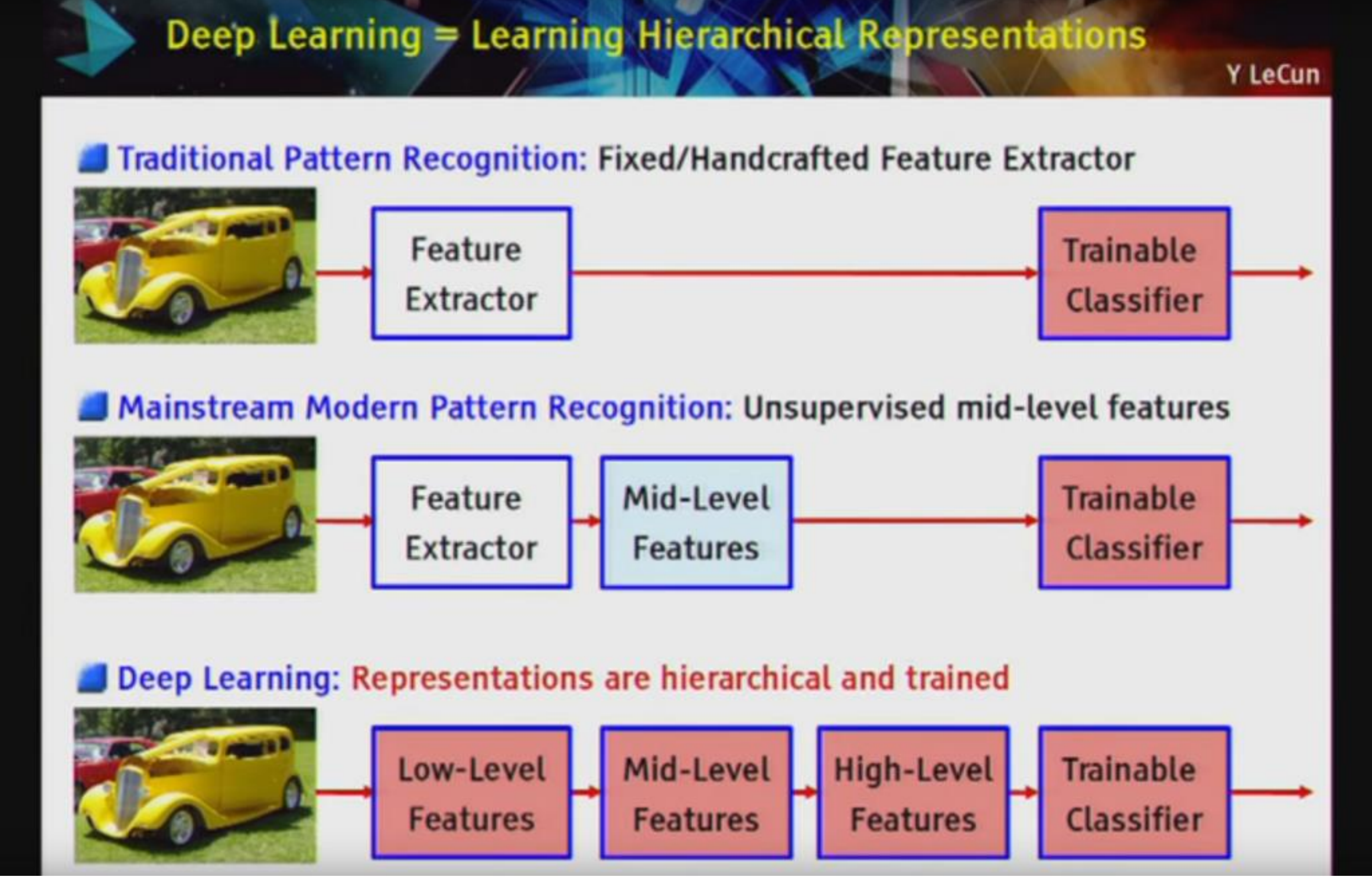


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Deep Learning

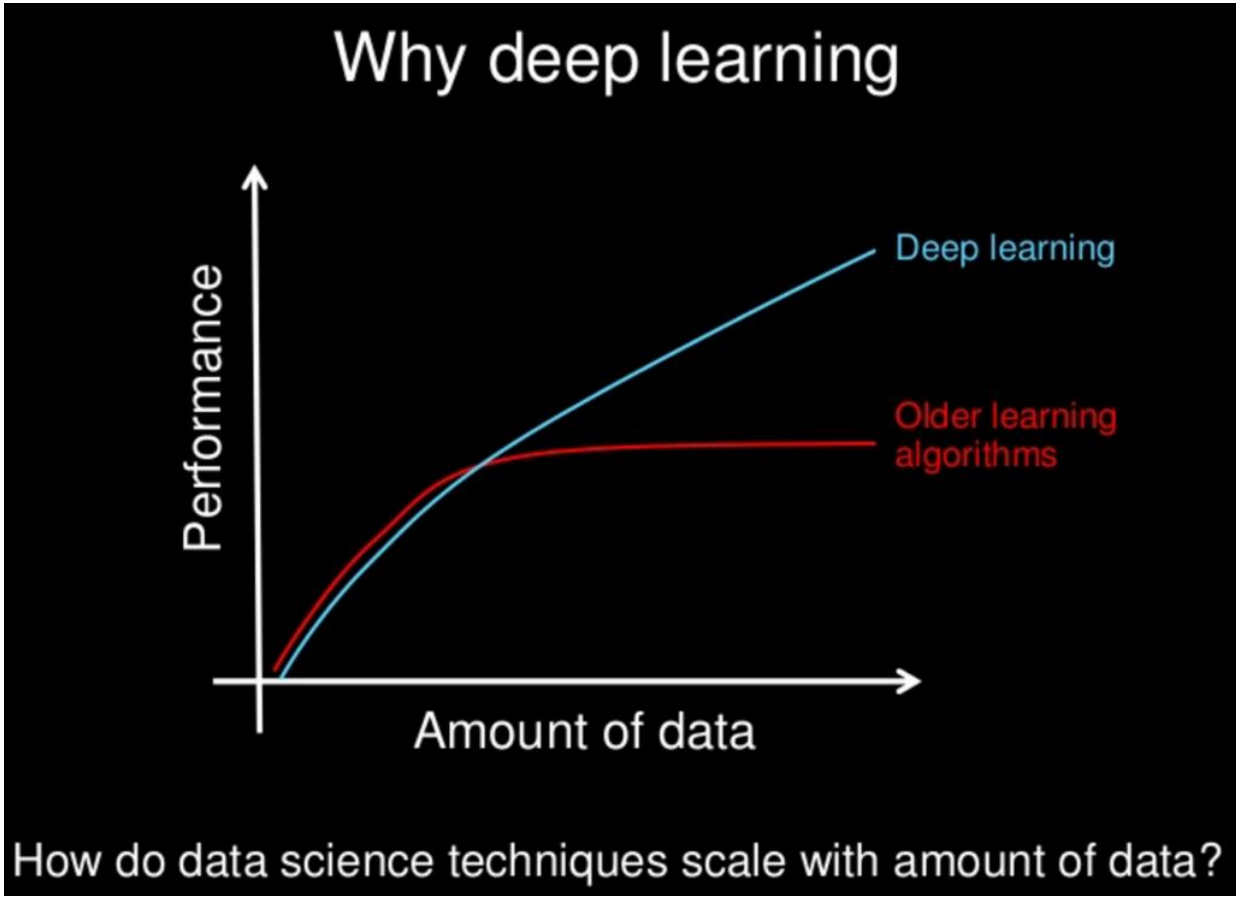
Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.



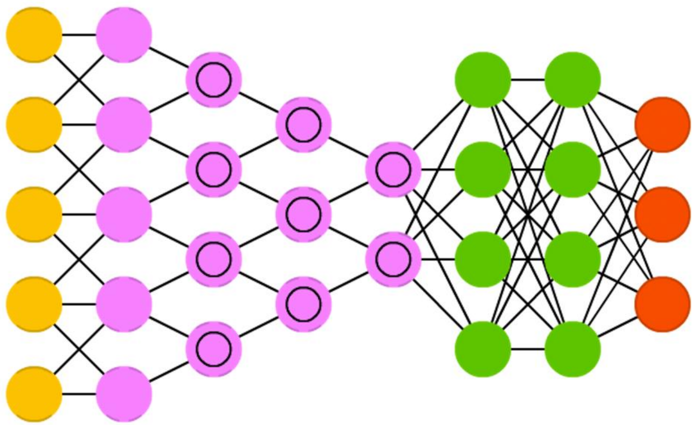
source https://machinelearningmastery.com/what-is-deep-learning/

Deep Learning — Scalable



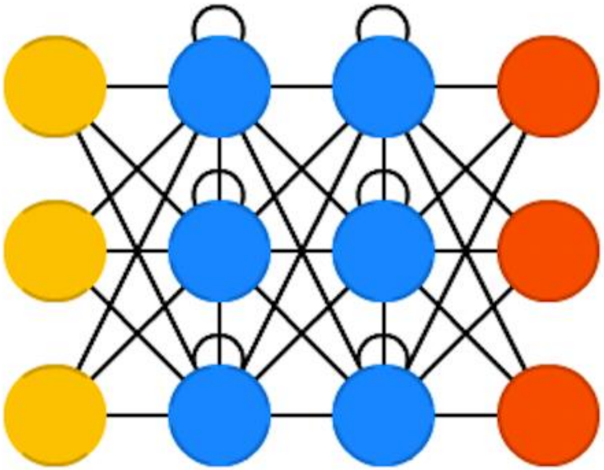
Types of Deep Learning Neural Networks: CNN, RNN, RBM, DBN

Convolutional Neural Network (CNN)



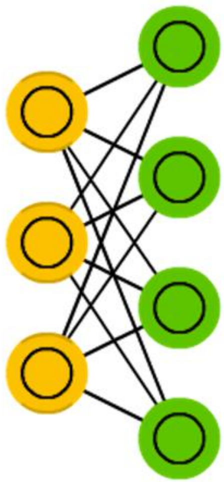
Convolutional neural networks (CNN) are quite different from most other networks. They are primarily used for image processing but can also be used for other types of input such as audio. A typical use case for CNNs is where you feed the network images and the network classifies the data, e.g. it outputs “cat” if you give it a cat picture and “dog” when you give it a dog picture.

Recurrent Neural Network (RNN)



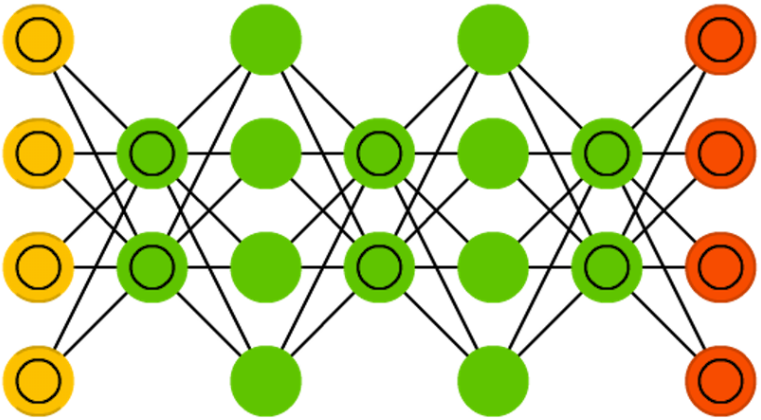
Recurrent neural networks (RNN) are Feed Forward NNs with a time twist: they are not stateless; they have connections between passes, connections through time. Neurons are fed information not just from the previous layer but also from themselves from the previous pass. This means that the order in which you feed the input and train the network matters: feeding it “milk” and then “cookies” may yield different results compared to feeding it “cookies” and then “milk”.

Restricted Boltzmann Machine (RBM)



RBMs can be trained like Feed Forward NNs with a twist: instead of passing data forward and then back-propagating, you forward pass the data and then backward pass the data (back to the first layer). Like the Hopfield Network, the RBM behaves like associative memory which remembers the patterns it has been trained on.

Deep Belief Network (DBN)



Deep belief networks (DBN) is the name given to stacked architectures of mostly RBMs. These networks have been shown to be effectively trainable stack by stack, where each RBM only has to learn to encode the previous network.

- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Deep Belief Network (DBN)
- Recursive Neural Tensor Network (RNTN)
- Restricted Boltzmann Machine (RBM)

Applications	
Text Processing	RNTN, RNN
Image Recognition	CNN, DBM
Object Recognition	CNN, RNTN
Speech Recognition	RNN
Time series Analysis	RNN
Unlabeled data – pattern recognition	RBM

Applications of Deep Learning

SUPERVISED AND UNSUPERVISED

Deep learning

Deep learning (DL) has the ability to work with unstructured data or very high dimensional data such as speech, sequences, images, videos with better results that other machine learning algorithms.

Going back to our MCQ test example, suppose that the questions and answers are only available verbally. We need the DL algorithm to be correctly identify the different parts of speed and apply natural language processing (NLP)

Deep Learning — both supervised and unsupervised applications

SUPERVISED APPLICATION	UNSUPERVISED APPLICATION
Image Recognition	Feature Learning
Speech Recognition	Autoencoders

Use cases for DL

Image classification/object recognition (convolutional networks)

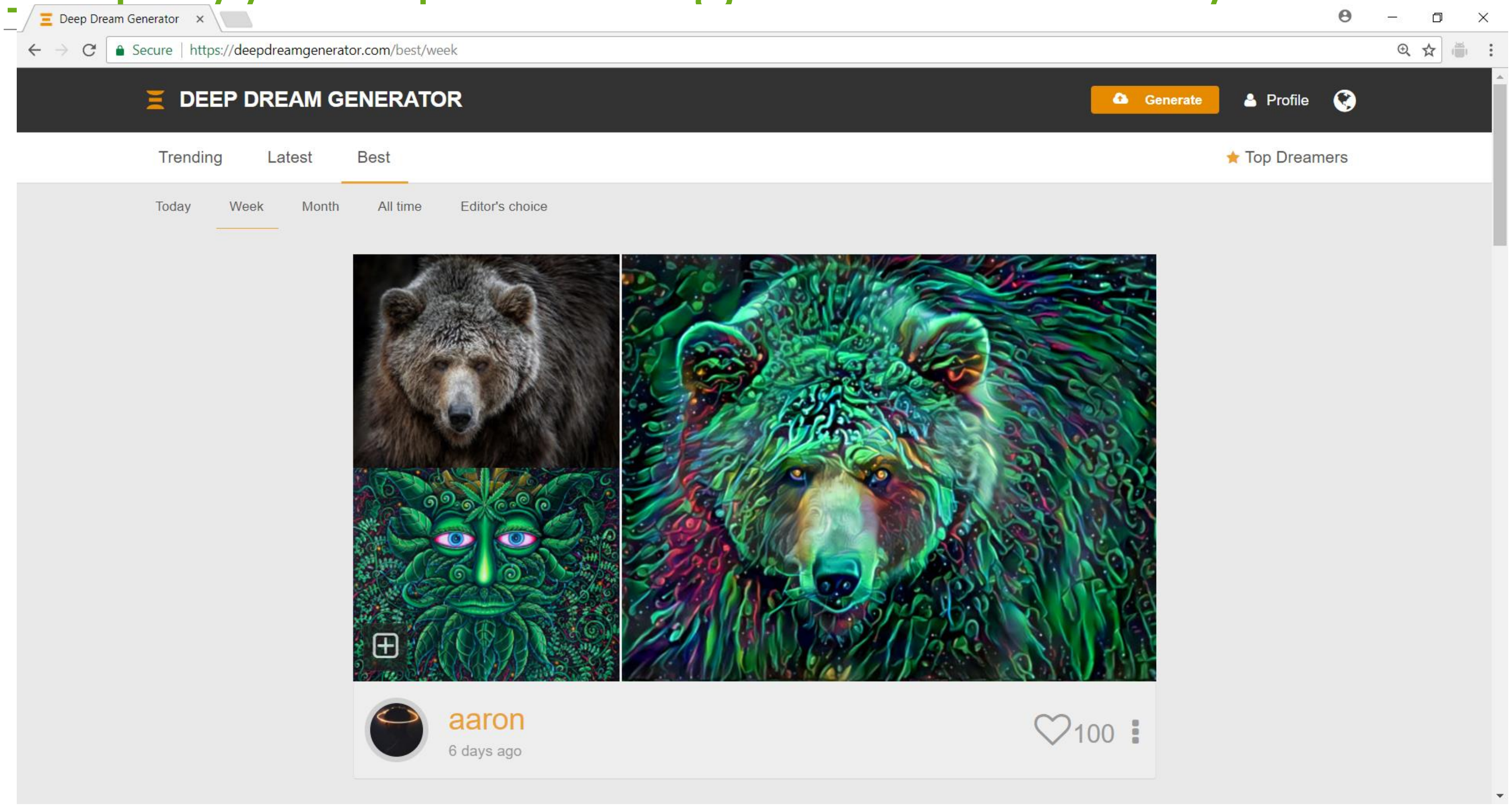
Processing sequential data (RNN/LSTM/GRU) such as language processing or forecasting

Complex decision making (deep reinforcement learning)

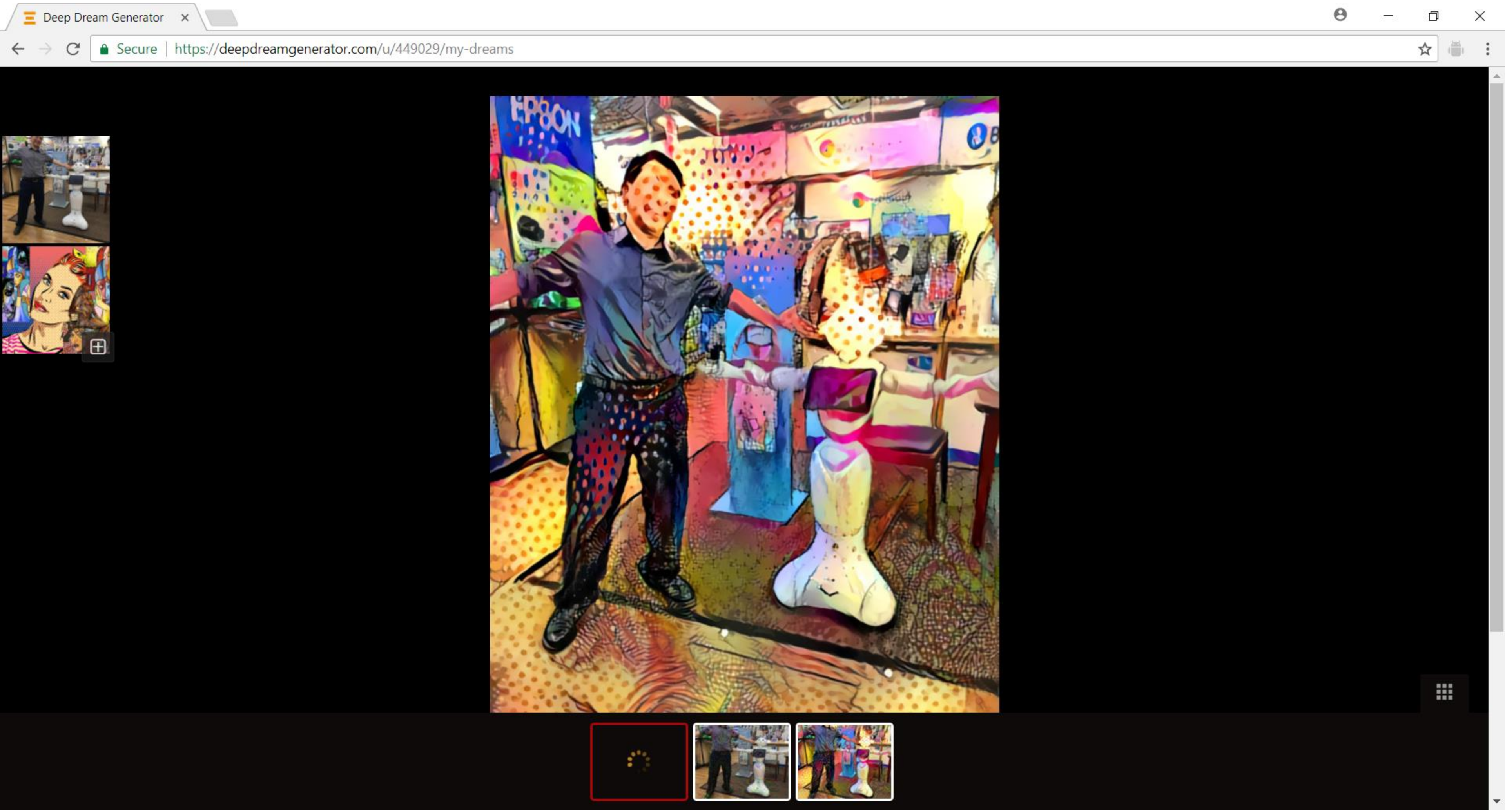
There are also a number of other applications that are common, such as image segmentation and super-resolution (fully convolutional networks) and similarity matching (Siamese networks).

Activity

https://deepdreamgenerator.com/

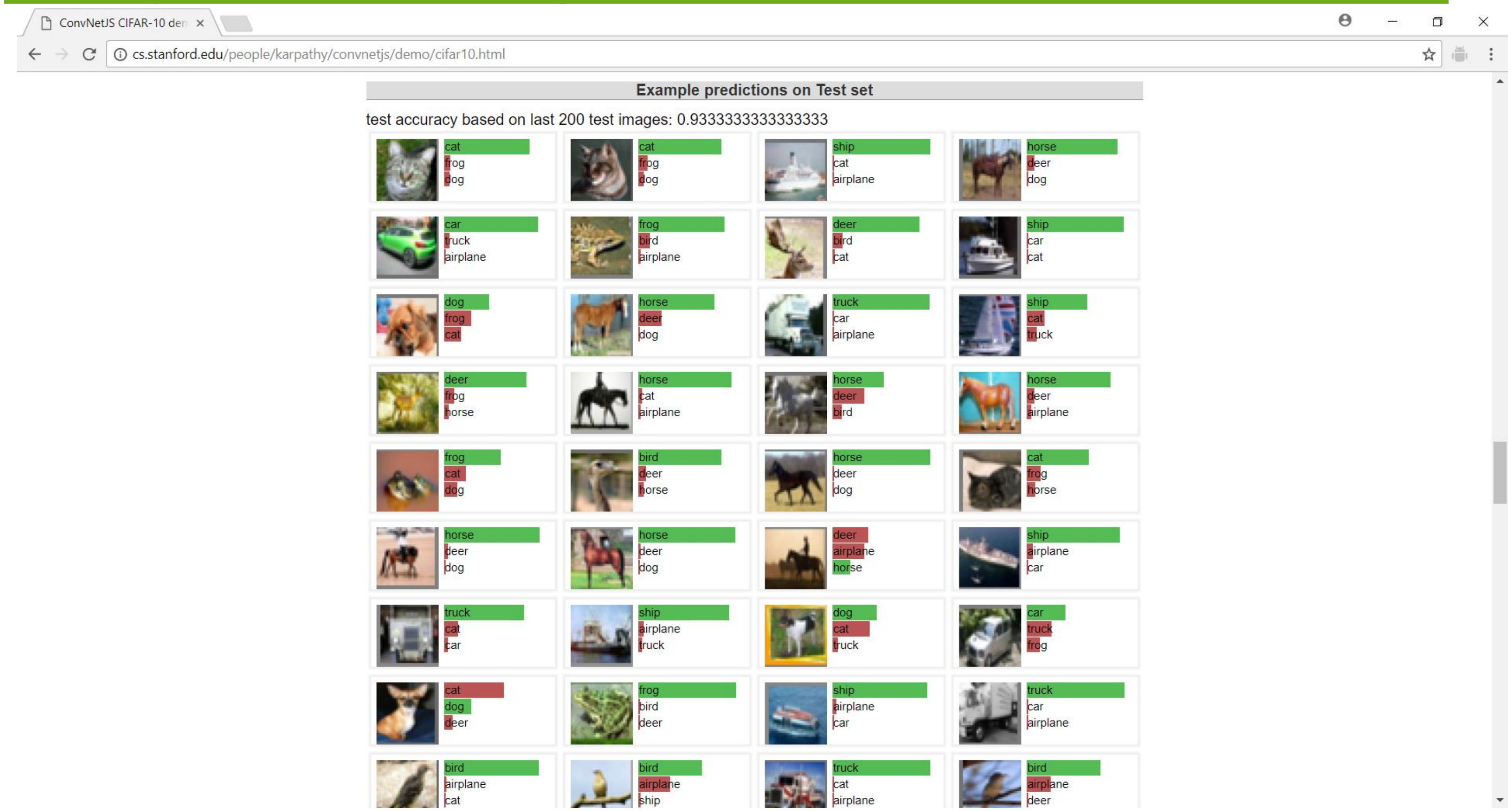


source: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
source: https://deepdreamgenerator.com/


Deep Dream Generator
Secure
https://deepdreamgenerator.com/u/449029/my-dreams

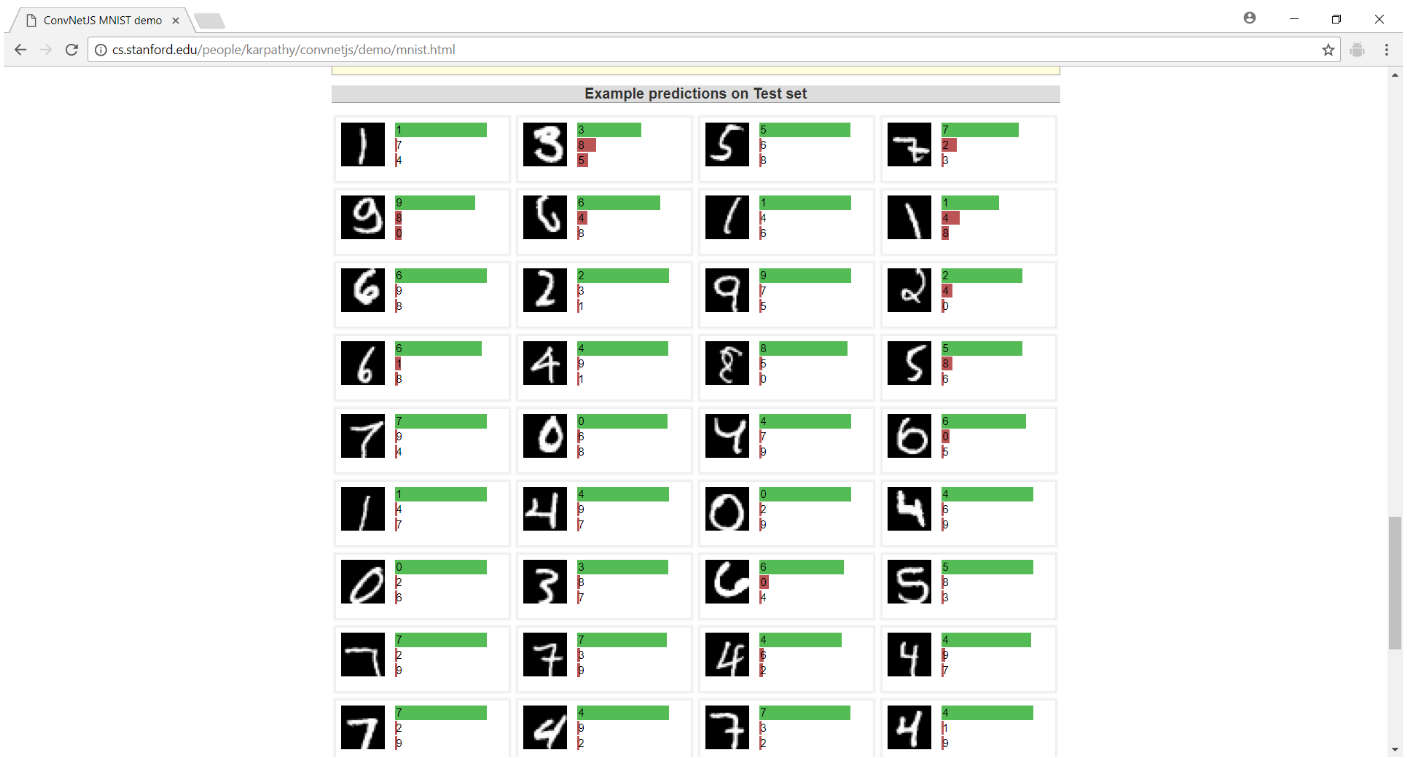
Activity

http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html



Activity

http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html



source: http://cs.stanford.edu/people/karpathy/convnetjs/intro.html

Summary

What we have learnt:

- Concepts behind Neural Networks
- Learning data representations
- Variety of Deep Learning Neural Network architectures
- Applications for Deep Learning