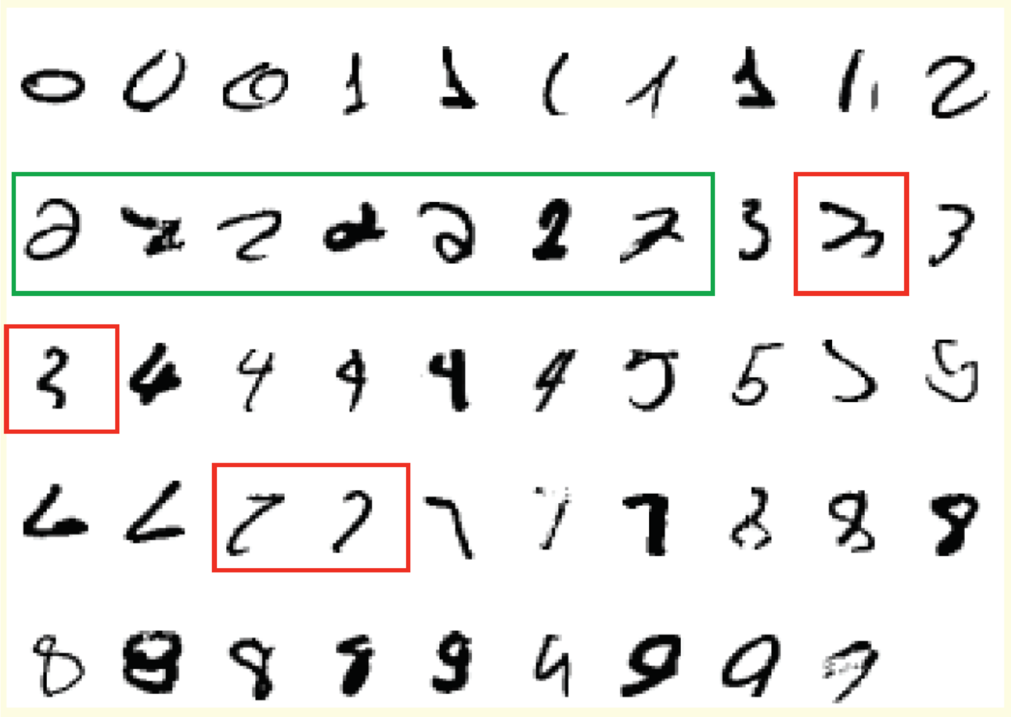
Neural Networks

Knut Hinkelmann

Motivation: Recognizing Numbers

- It is very hard to specify what makes a «2»



- It is nearly impossible to create or learn symbolic rules.

Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

History of Artificial Neural Networks

- Creation:
  - 1890: William James - defined a neuronal process of learning
- Promising Technology:
  - 1943: McCulloch and Pitts - earliest mathematical models
  - 1954: Donald Hebb and IBM research group - earliest simulations
  - 1958: Frank Rosenblatt - The Perceptron
- Disenchantment:
  - 1969: Minsky and Papert - perceptrons have severe limitations
- Re-emergence:
  - 1985: Multi-layer nets that use back-propagation
  - 1986: PDP Research Group - multi-disciplined approach

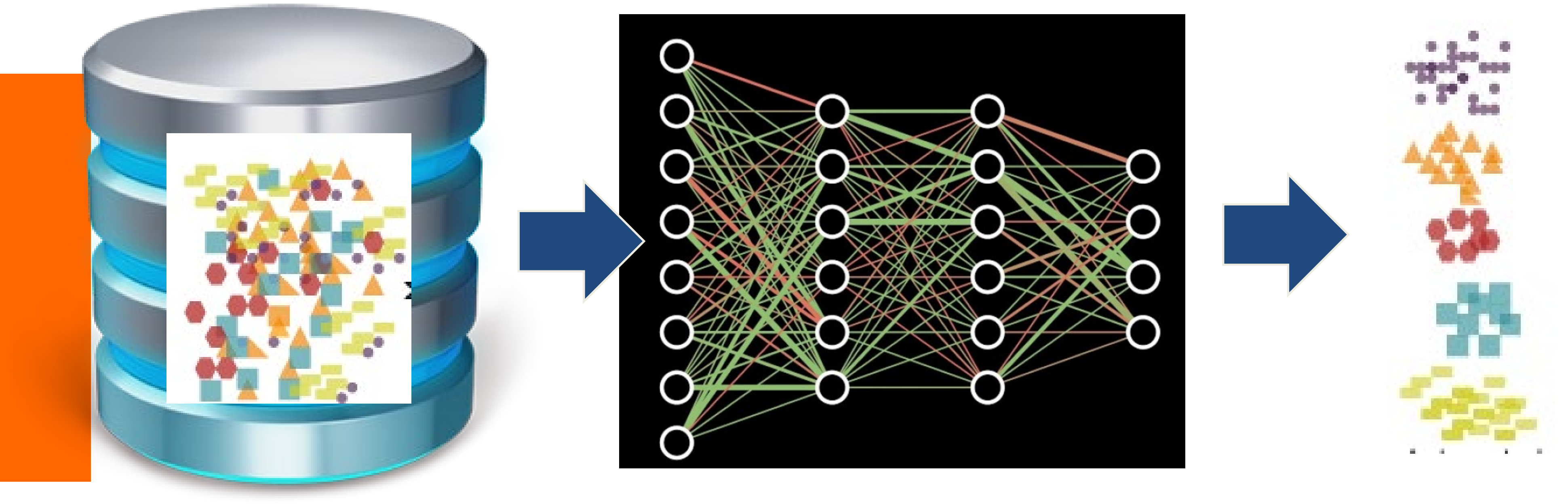
© plato.acadiau.ca/courses/comp/dsilver/5013/Slides/ANN_ml.ppt [04.06.2018]

ANN application areas ...

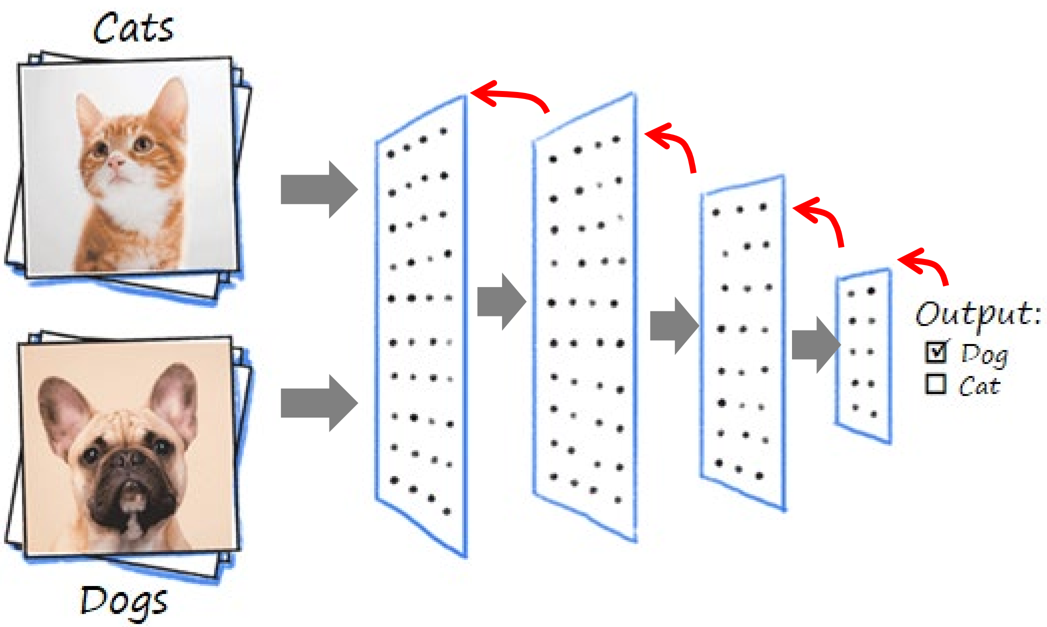
- Science and medicine: modeling, prediction, diagnosis, pattern recognition
- Manufacturing: process modeling and analysis
- Marketing and Sales: analysis, classification, customer targeting
- Finance: portfolio trading, investment support
- Banking & Insurance: credit and policy approval
- Security: bomb, iceberg, fraud detection
- Engineering: dynamic load schedding, pattern recognition

© plato.acadiau.ca/courses/comp/dsilver/5013/Slides/ANN_ml.ppt [04.06.2018]

Learning with Neural Networks

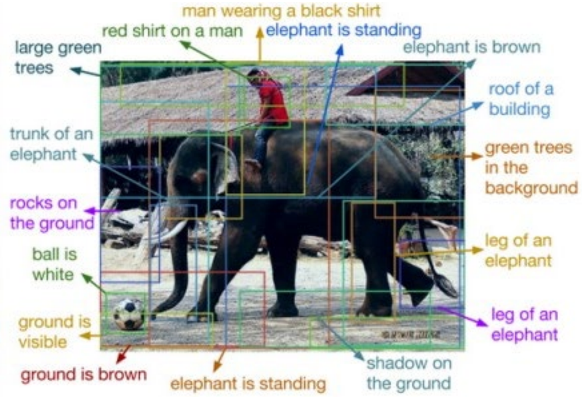


Principle of Neural Network: Learning from Data

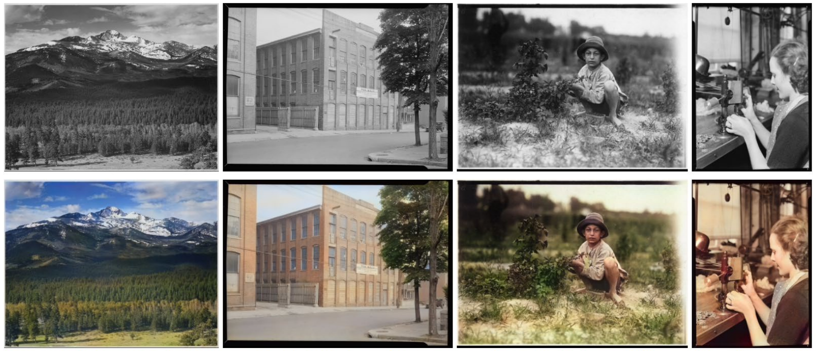


Application: Cat or Dog?

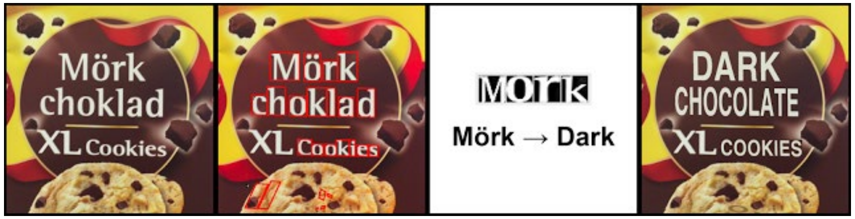
Applications of Deep Learners



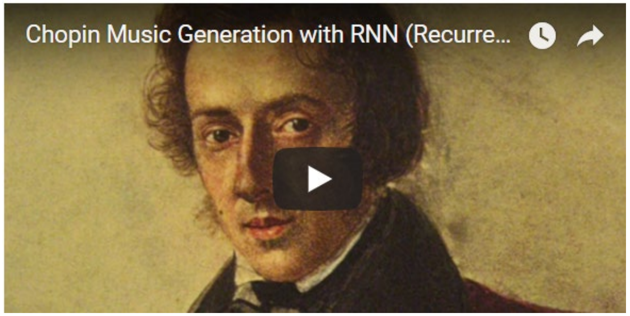
Describing photos


Colorado National Park, 1941 Textile Mill, June 1937 Berry Field, June 1909 Hamilton, 1936

Picture coloring



Translation



Music composition

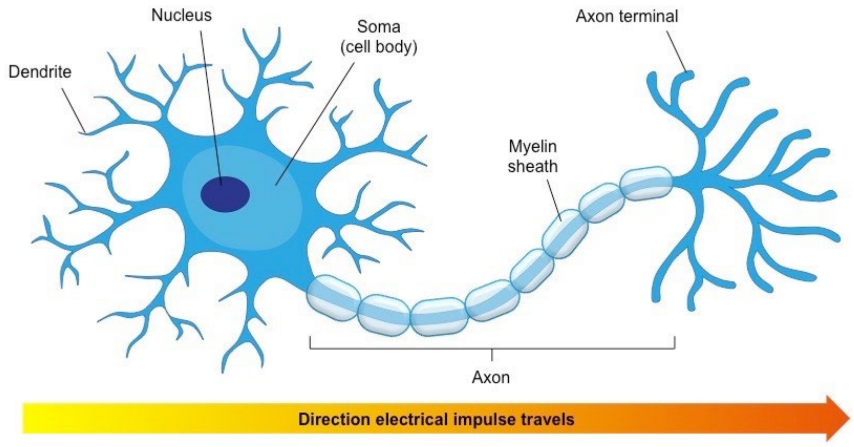
Self-driving Cars

© http://www.yaronhadad.com/deep-learning-most-amazing-applications/ [28.02.2018]

Artificial Neural Networks – Following the Brain

Biological Neuron

The basic computational unit of the brain is a **neuron**. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14}—10^{15} **synapses**.

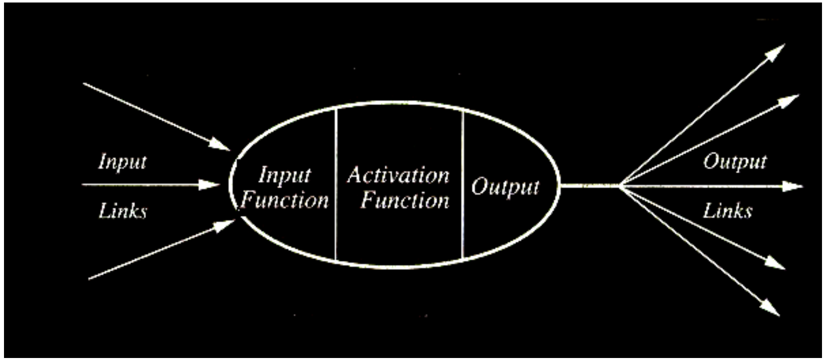


- dentrites - the receivers
- soma - neuron cell body (sums input signals)
- axon - the transmitter
- synapse - point of transmission
- neuron activates after a certain threshold is met
- Learning occurs via electro-chemical changes in effectiveness of synaptic junction.

© plato.acadiau.ca/courses/comp/dsilver/5013/Slides/ANN_ml.ppt [04.06.2018]

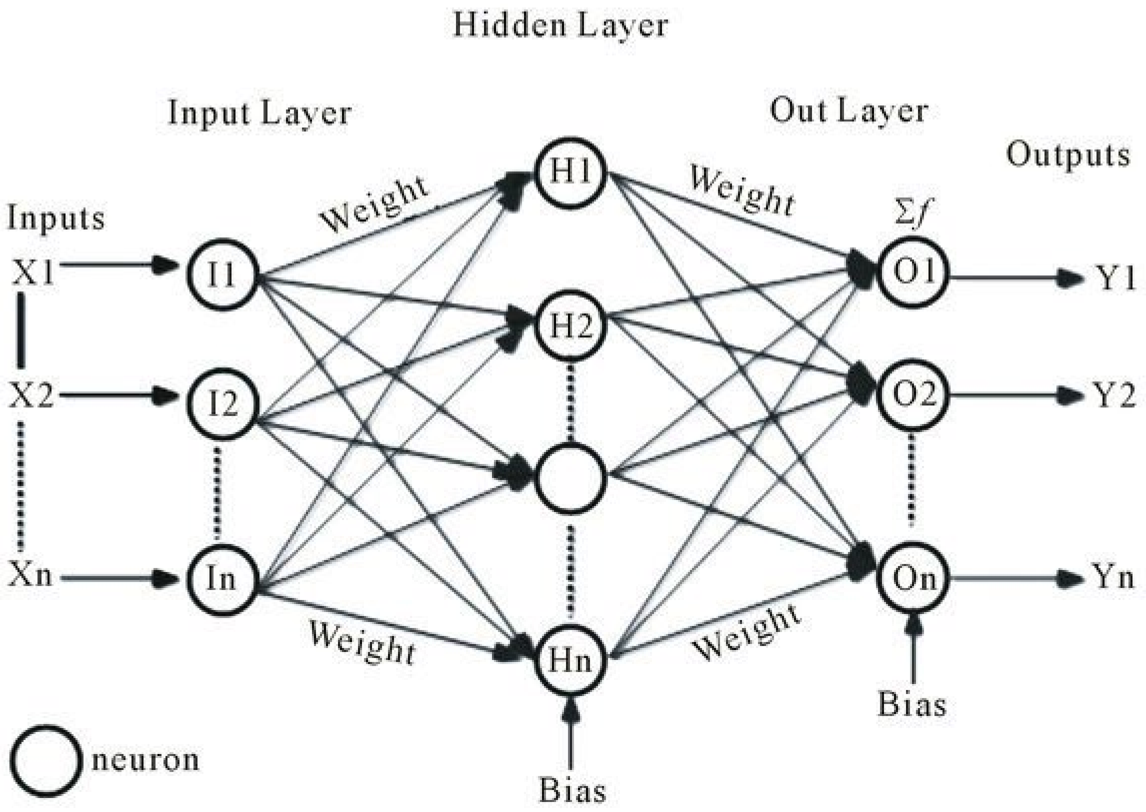
Mathematical Model of a Neuron

- The basic unit of computation in a neural network is the neuron (often called a node or unit).
- It receives input from some other nodes or from an external source. Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs.
- The node computes an output. It applies a function to the weighted sum of its inputs.



Source: David Fumo: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

Multi-layer Perceptron



Source: David Fumo: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

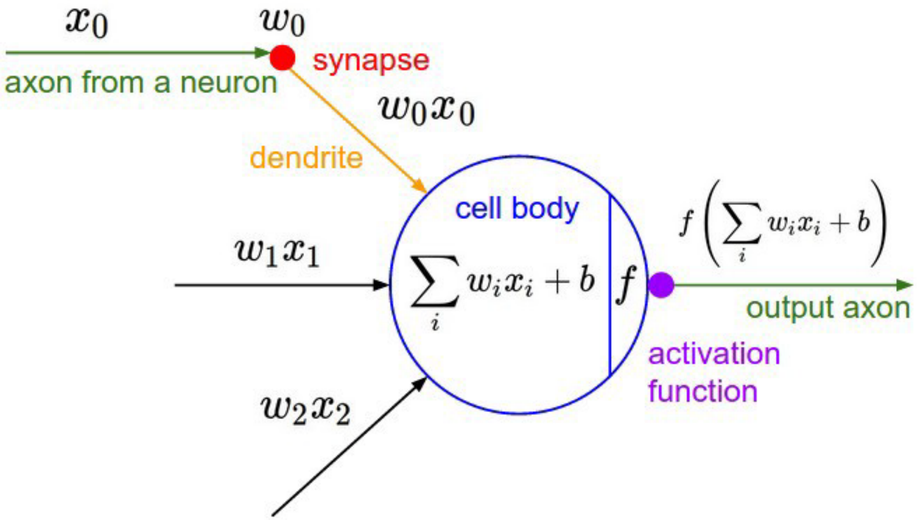
Basic Concepts of Neural Networks

- **Input Nodes (input layer):** No computation is done here within this layer, they just pass the information to the next layer.
- **Output Nodes (output layer):** Here we finally use an activation function that maps to the desired output format.
- **Hidden nodes (hidden layer):** Hidden nodes transfer the weights from the input layer to the following layer (another hidden layer or to the output layer).
- **Connections and weights:** The *network* consists of connections, each connection transferring the output of a neuron i to the input of a neuron *j*. Each connection is assigned a weight *Wij.*
- **Activation function:** Defines the output of that node given an input or set of inputs. *Nonlinear* activation functions allows such networks to compute nontrivial problems using only a small number of nodes.
- **Learning rule:** The *learning rule* modifies weights and thresholds of the neural network, in order for a given input to the network to produce a favored output.

Source: David Fumo: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

An Artificial Neuron - The Perceptron

- input connections - the receivers
- *node*, *unit*, or PE simulates neuron body
- output connection - the transmitter
- *activation function* – when is the neuron activated
  - e.g. $f(x) = \begin{cases} 1 & if\ x > \varphi \\ 0 & otherwise \end{cases}$
- *connection weights* act as synaptic junctions
- Learning occurs via changes in value of the connection weights.

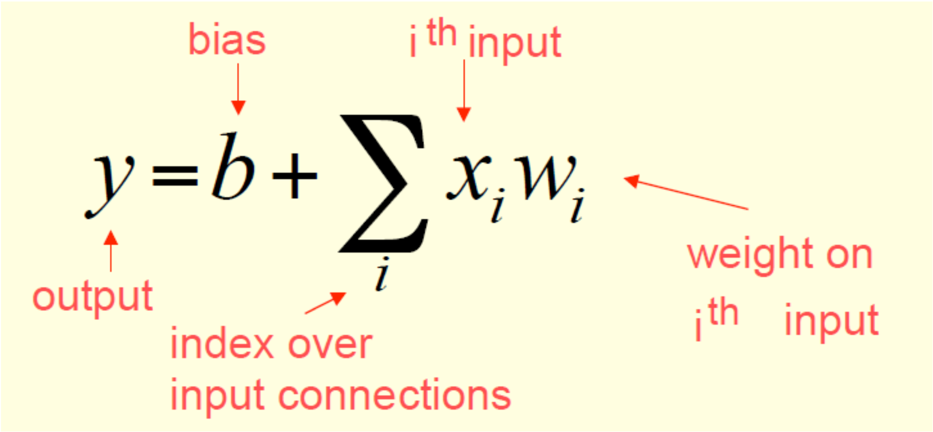


© David Fumo: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

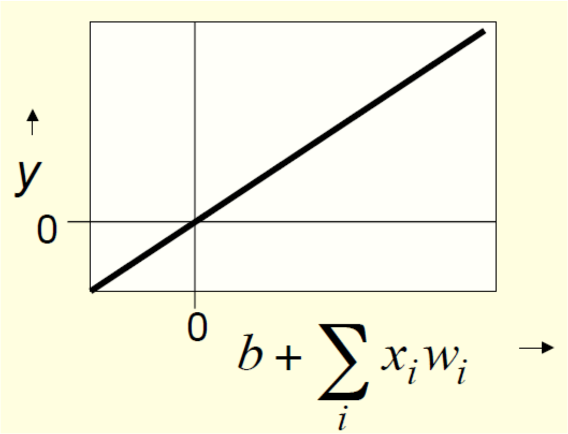
© plato.acadiau.ca/courses/comp/dsilver/5013/Slides/ANN_ml.ppt [04.06.2018]

Simple Type of Neuron: Linear Neuron

- Simple but computationally limited



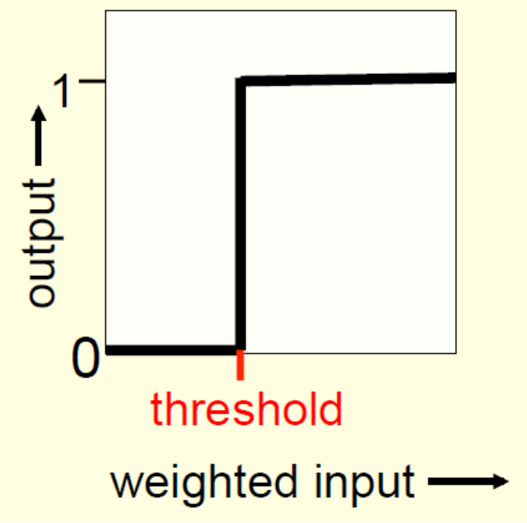
$$y = b + \sum_i x_i w_i$$



Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

Binary Threshold Neurons

- McCulloch-Pitts (1943)
  - First compute a weighted sum of the inputs.
  - Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.
- There are two equivalent ways to write the equations for a binary threshold neuron:



$$z = \sum_i x_i w_i \qquad y = \begin{cases} 1 \text{ if } z \geq \theta \\ 0 \text{ otherwise} \end{cases}$$

$$\boxed{\theta = -b}$$

$$z = b + \sum_i x_i w_i \qquad y = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

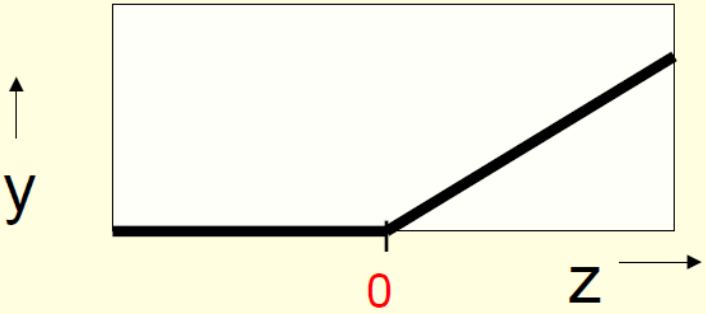
Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

Rectified Linear Neurons

- They compute a *linear* weighted sum of their inputs.
- The output is a ***non-linear*** function of the total input.

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$



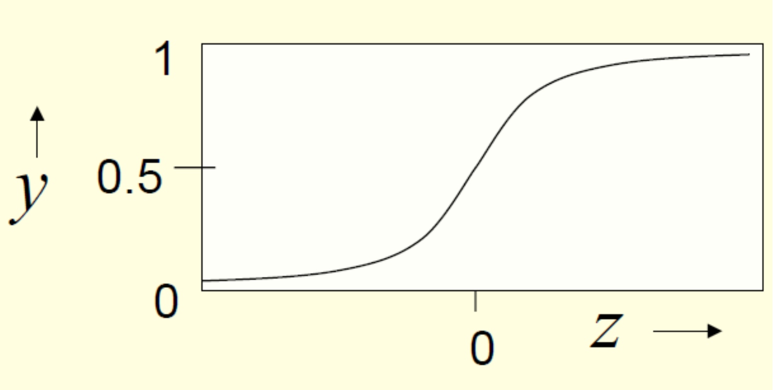
Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

Sigmoid Neuron

- These give a real-valued output that is a smooth and bounded function of their total input.
- – Typically they use the logistic function

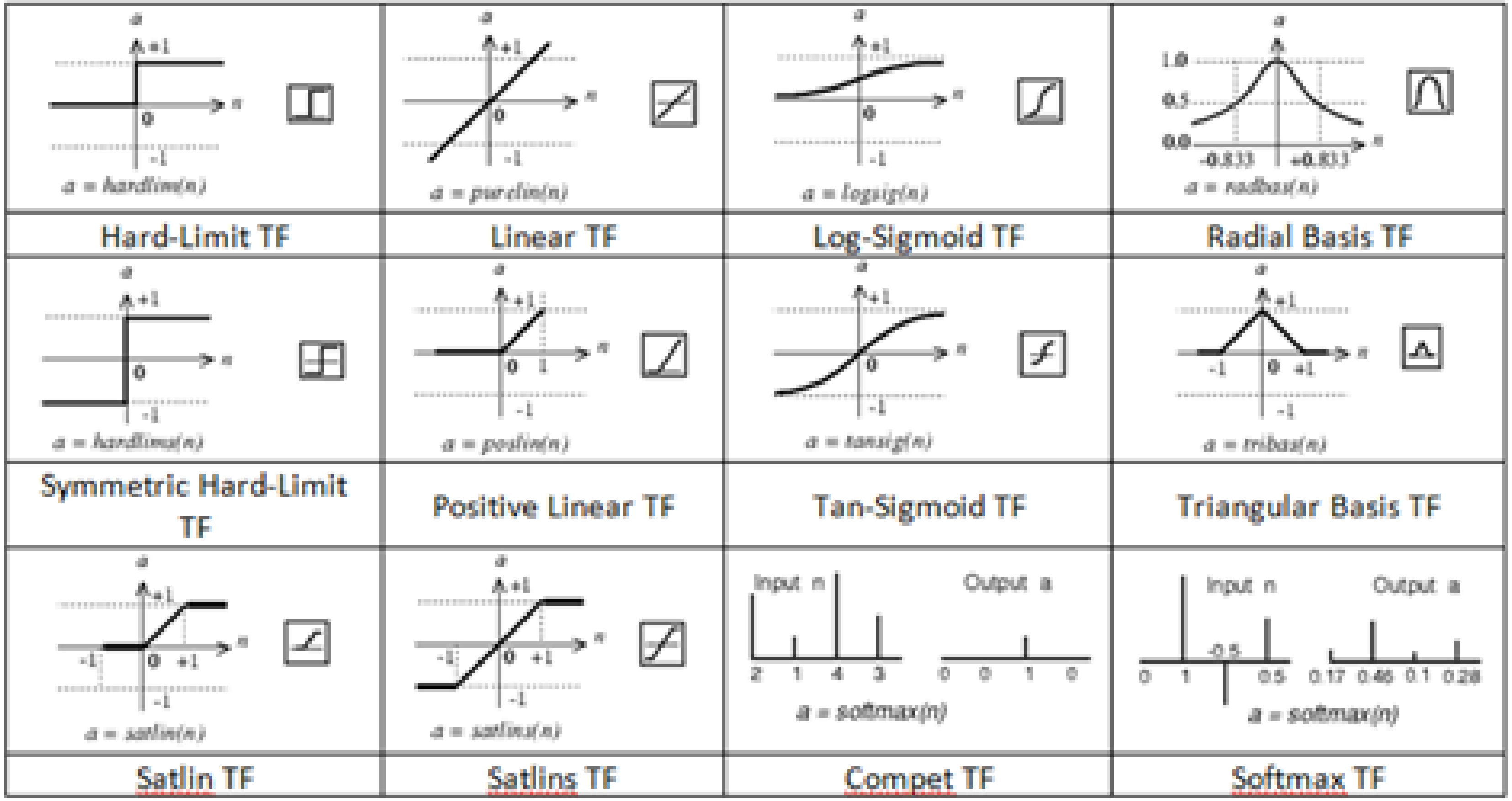
$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1 + e^{-z}}$$



Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

Other activation functions



Hard-Limit TF	Linear TF	Log-Sigmoid TF	Radial Basis TF
Symmetric Hard-Limit TF	Positive Linear TF	Tan-Sigmoid TF	Triangular Basis TF
Satlin TF	Satlins TF	Compet TF	Softmax TF

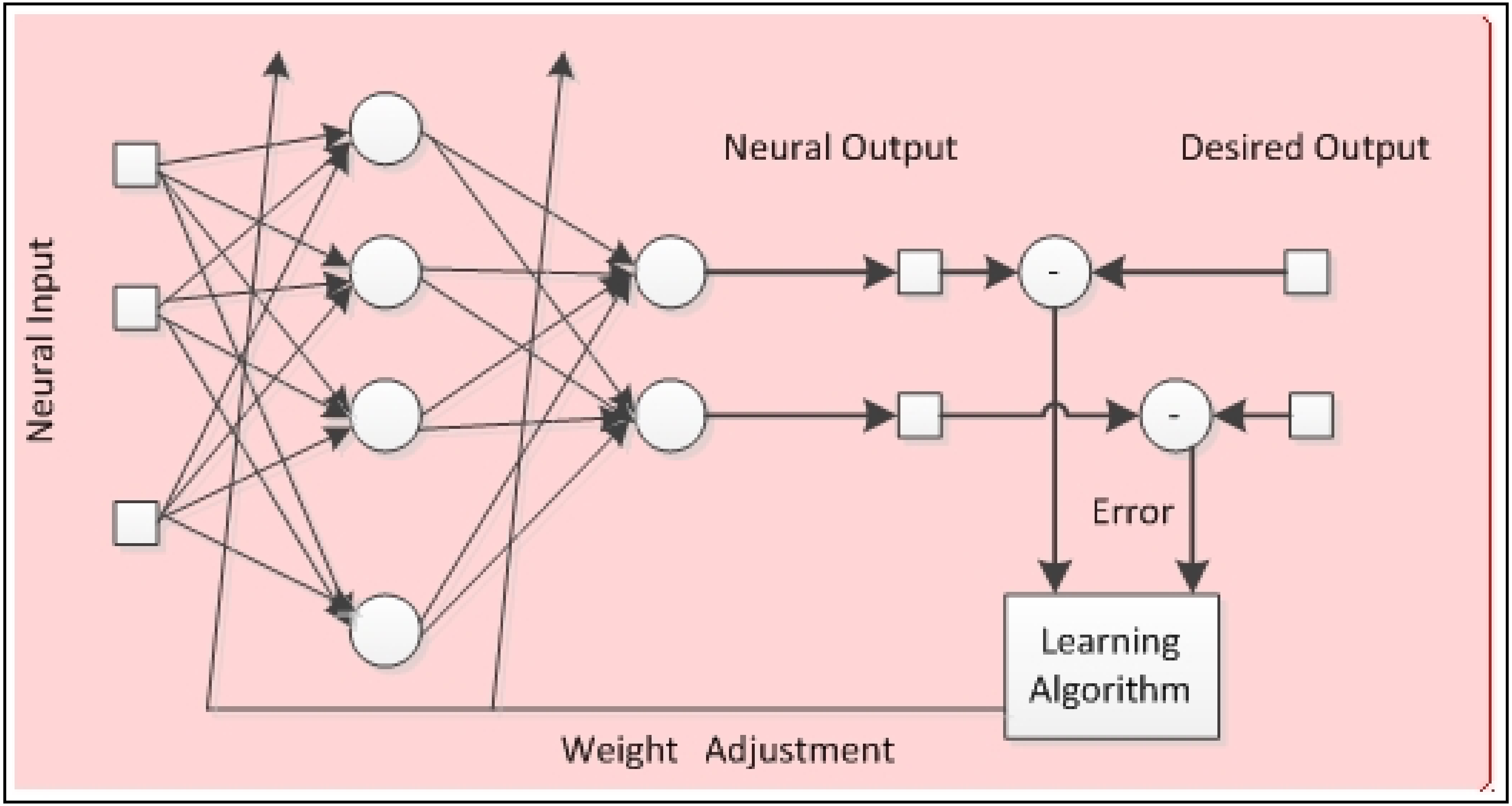
© https://www.quora.com/Do-people-use-power-functions-like-x-3-or-x-5-as-activation-functions-in-artificial-neural-networks. [19.05.2020]

Exercise

- Design a Perceptron with McCulloch-Pitts Neuron with 2 inputs and 1 output neuron(s) which operates an
  - AND
  - OR
  - XOR

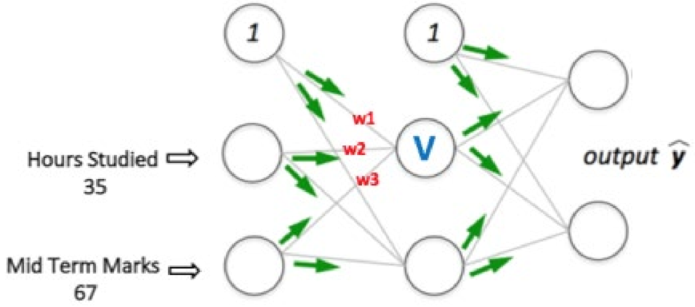
LEARNING

Supervised Learning



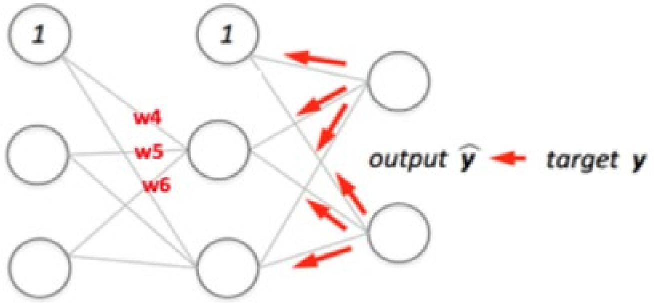
Principle of Application and (Supervised) Learning

- Application: The input values are **propagated forward** to the output



- Learning: Unexpected results at the output nodes are **propagated back** through the network leading to new weights

Backpropagation
+
Weights Adjusted



https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Learning: Backpropagation

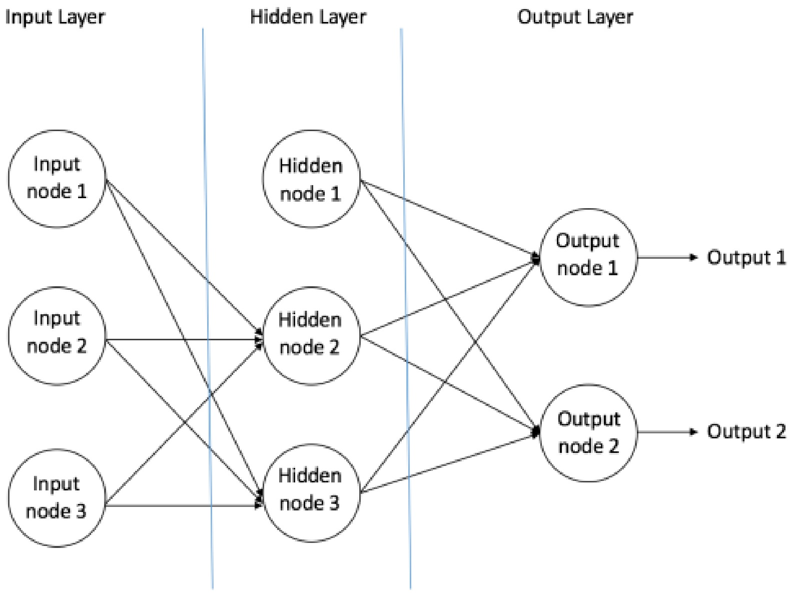
- **Backward Propagation of Errors,** often abbreviated as BackProp is one of the several ways in which an artificial neural network (ANN) can be trained.
- It is a supervised training scheme, which means, it learns from labeled training data.
- To put in simple terms, BackProp is like "**learning from mistakes**". The supervisor ***corrects*** the ANN whenever it makes mistakes.

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Feedforward Network

- In a feedforward network, the information moves in only one direction – forward – from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network

- Two examples of feedforward networks are given below:
  - **Single Layer Perceptron** – This is the simplest feedforward neural network and does not contain any hidden layer.
  - **Multi Layer Perceptron** – A Multi Layer Perceptron has one or more hidden layers.



https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

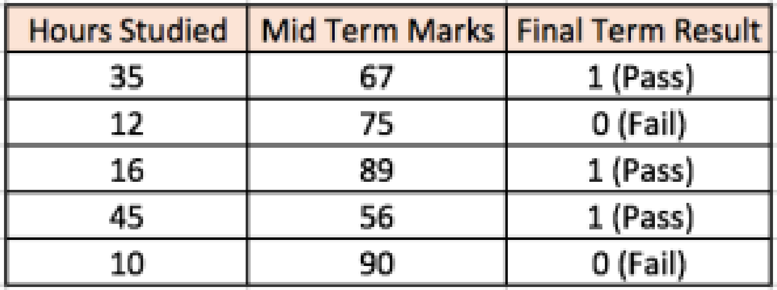
Backprop Algorithm

- Initially all the edge weights are randomly assigned.
- For every input in the training dataset, the ANN is activated and its output is observed. This output is compared with the desired output that we already know, and the error is "propagated" back to the previous layer. This error is noted and the weights are "adjusted" accordingly.
- This process is repeated until the output error is below a predetermined threshold.
- Once the above algorithm terminates, we have a "learned" ANN.

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Backprop Example (1)

- We want to learn the probability of a student to pass or fail an exam
- Training set:



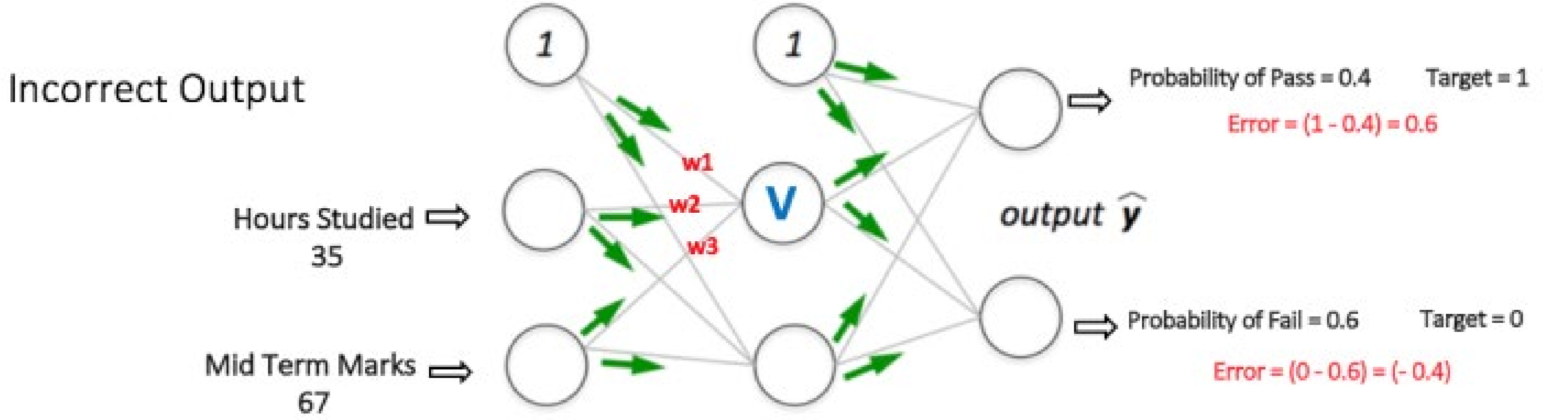
Hours Studied	Mid Term Marks	Final Term Result
35	67	1 (Pass)
12	75	0 (Fail)
16	89	1 (Pass)
45	56	1 (Pass)
10	90	0 (Fail)

- The two input columns show the number of hours the student has studied and the mid term marks obtained by the student. The Final Result column can have two values 1 or 0 indicating whether the student passed in the final term.

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Backprop Example (2)

- Step1: forward propagation step in a multi layer perceptron

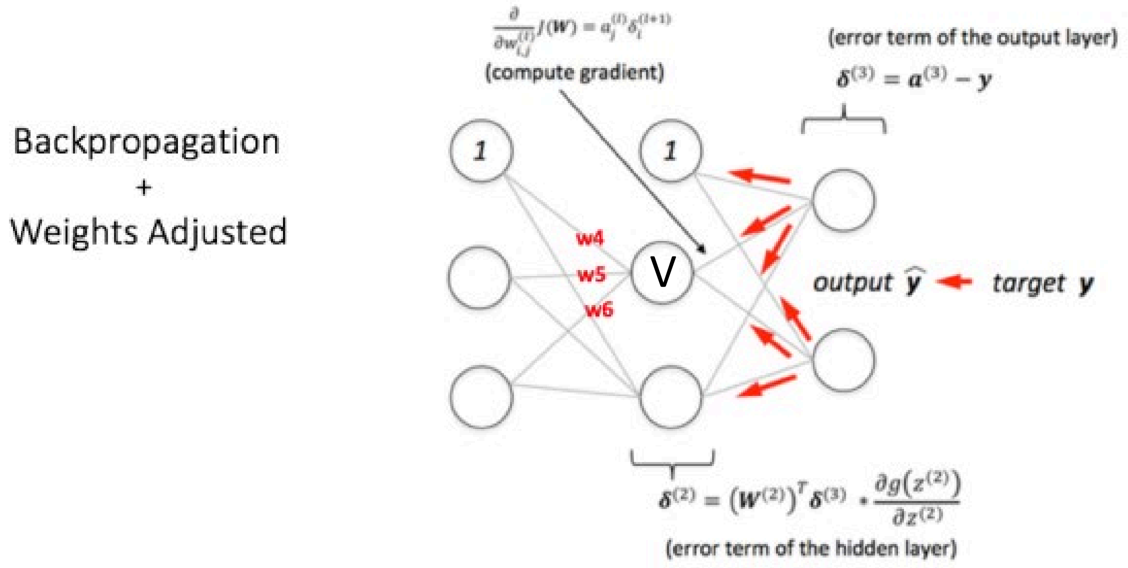


In this example the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0 respectively)

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Backprop Example (3)

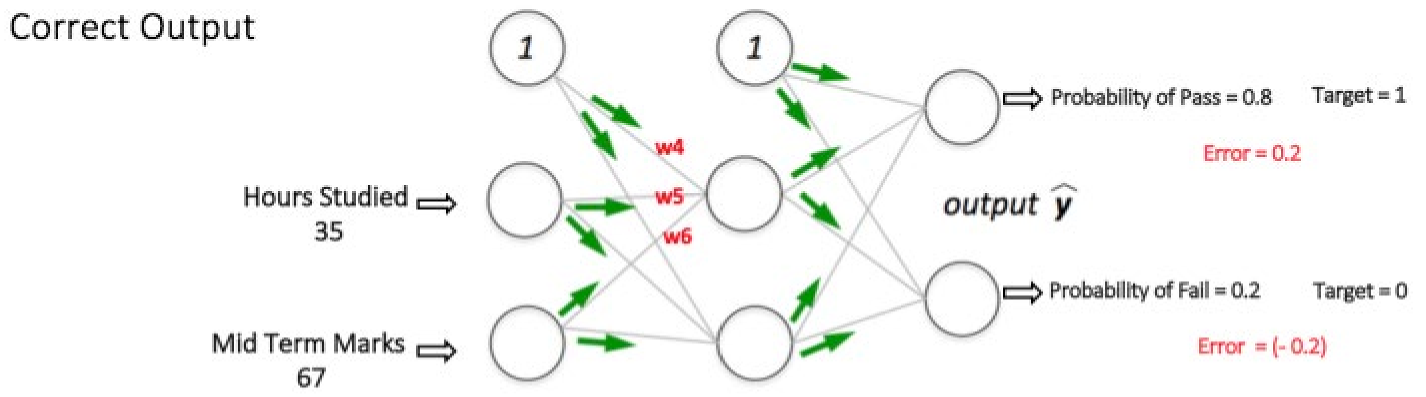
- We calculate the total error at the output nodes and propagate these errors back through the network using Backpropagation (we ignore the mathematical equations in the figure for now).
- This leads to new weights w4, w5 and w6 for node V


Backpropagation
+
Weights Adjusted
$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$
(compute gradient)
(error term of the output layer)
$\delta^{(3)} = a^{(3)} - y$
1
1
w4
w5
w6
V
output $\hat{y}$
target y
$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} \cdot \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$
(error term of the hidden layer)

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

Backprop Example (4)

- Using the same input data for the network with the adjusted weights leads to much better results



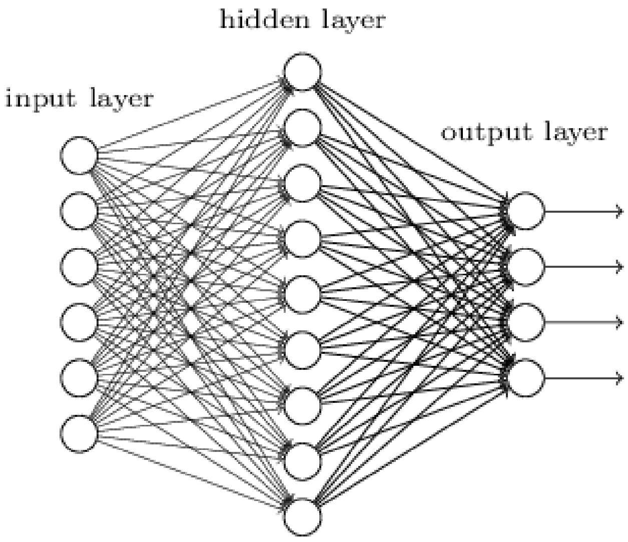
- This process is repeated with all other training data sets.
- When the network has *learnt* these examples, it can find the output probabilities for Pass and Fail also for new input data through the forward propagation step and.

https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

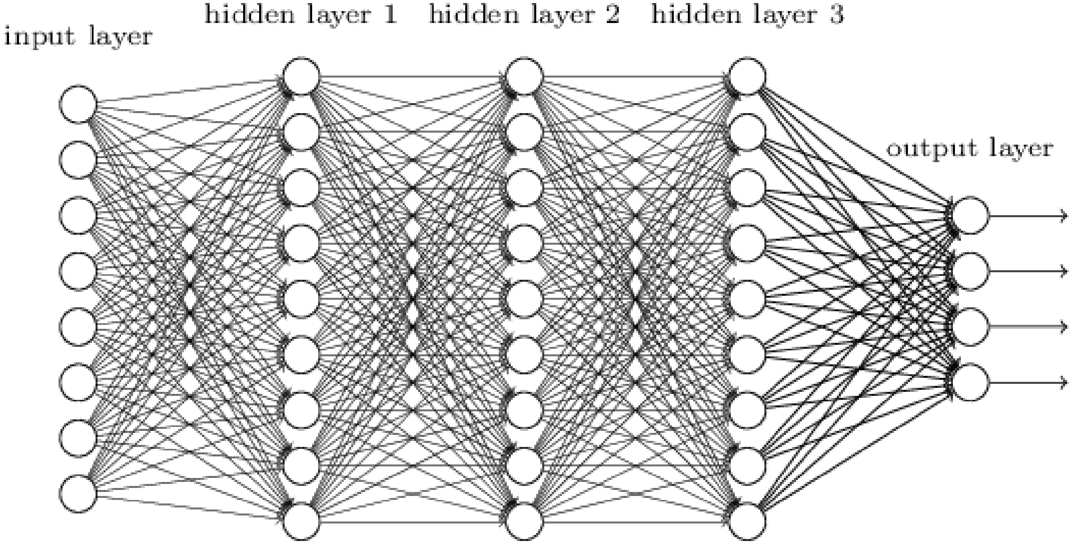
NETWORK TYPES

Deep Neural Networks

"Non-deep" feedforward neural network

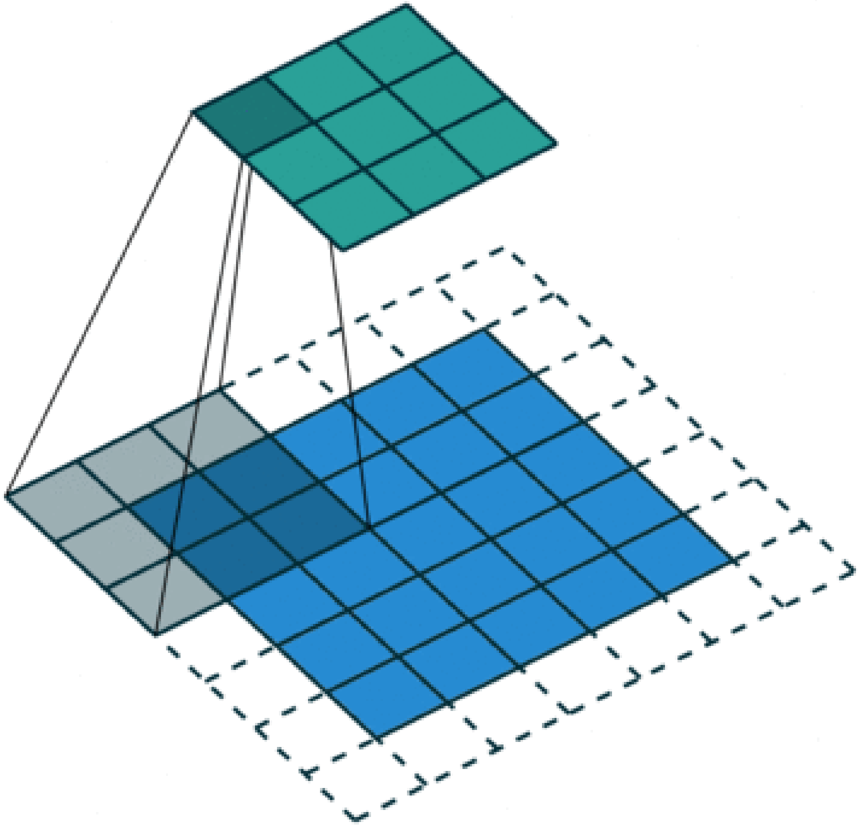


Deep neural network



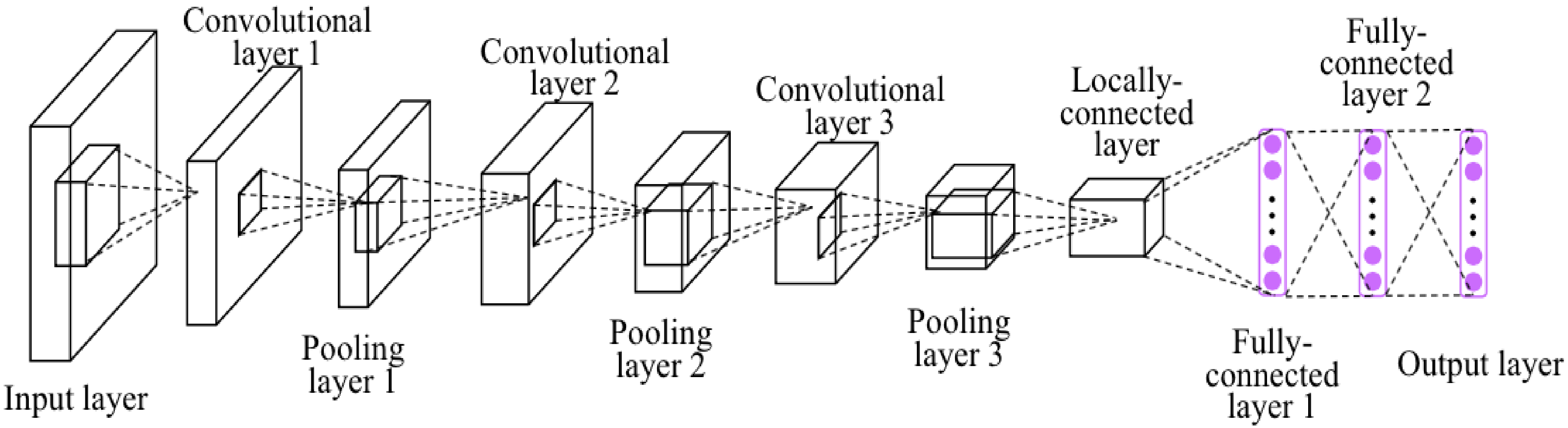
Convolutional Neural Networks (CNN)

Most commonly applied to analyze visual imagery



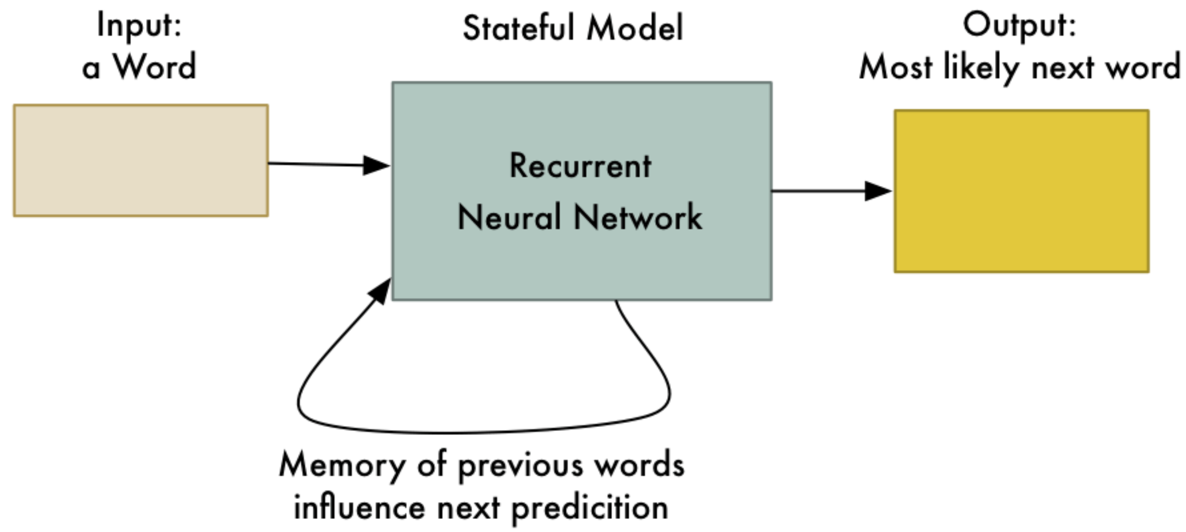
© https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology/ [04.06.2018]

Convolutional Neural Networks (CNN)



© http://personal.ie.cuhk.edu.hk/~ccloy/project_target_code/images/fig3.png [19.05.2020]

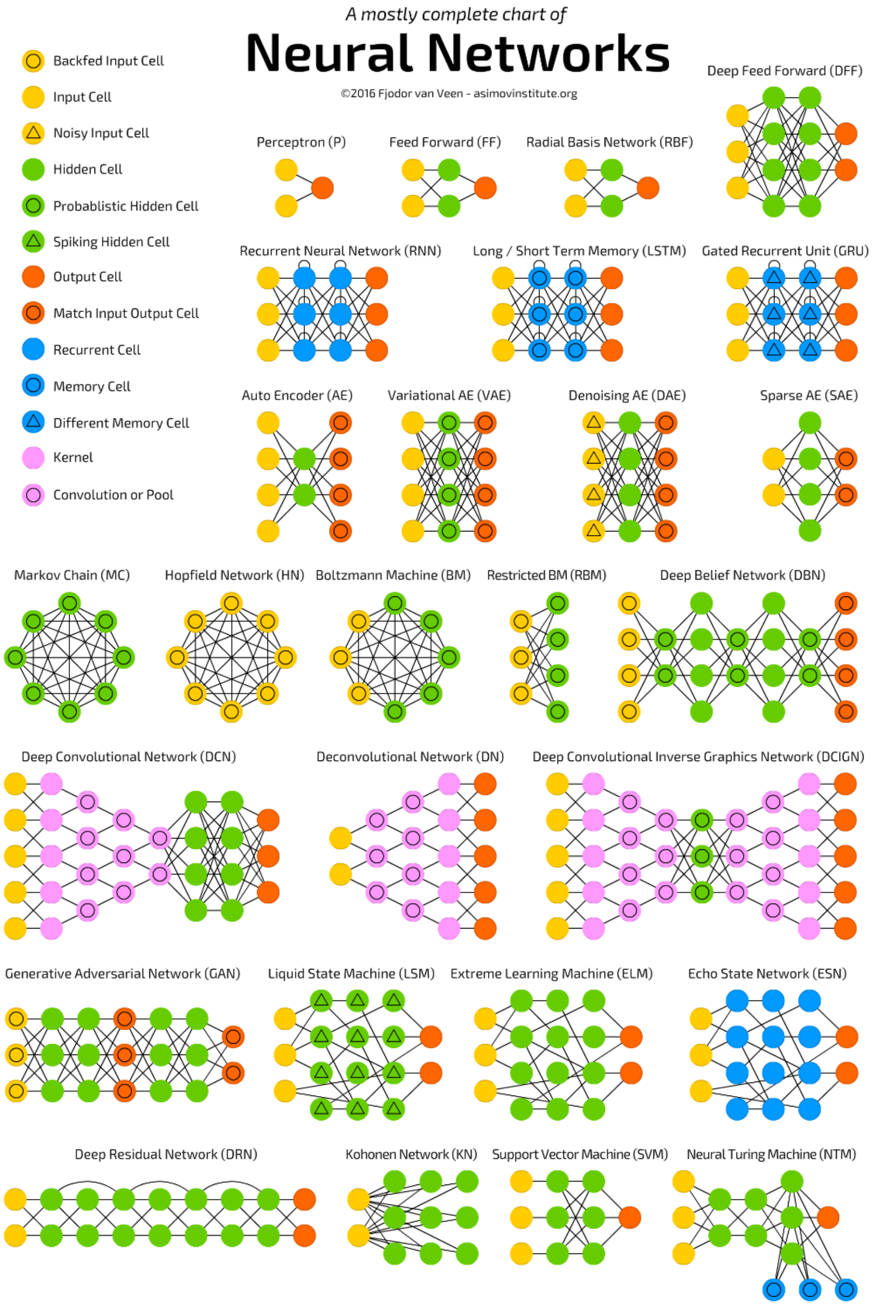
Recurrent Neuronal Network (RNN)



Output so far:

Machine

© https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology/ [04.06.2018]


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

© https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464 [04.06.2018]

Neural Networks vs. Decision Trees/Rules

- Classification with Neural Networks is very good
- Decisions with neural networks are not comprehensible
- Decision Trees and Rules are often more trustworthy
  - Decisions with trees and rules are comprehensible and explainable
  - One can see which rules are applied to make a decision
- In applications, in which trust in the decision or explainability is important, people prefer decision trees or rules