Data Science & Artificial Intelligence

Summer Term 2022

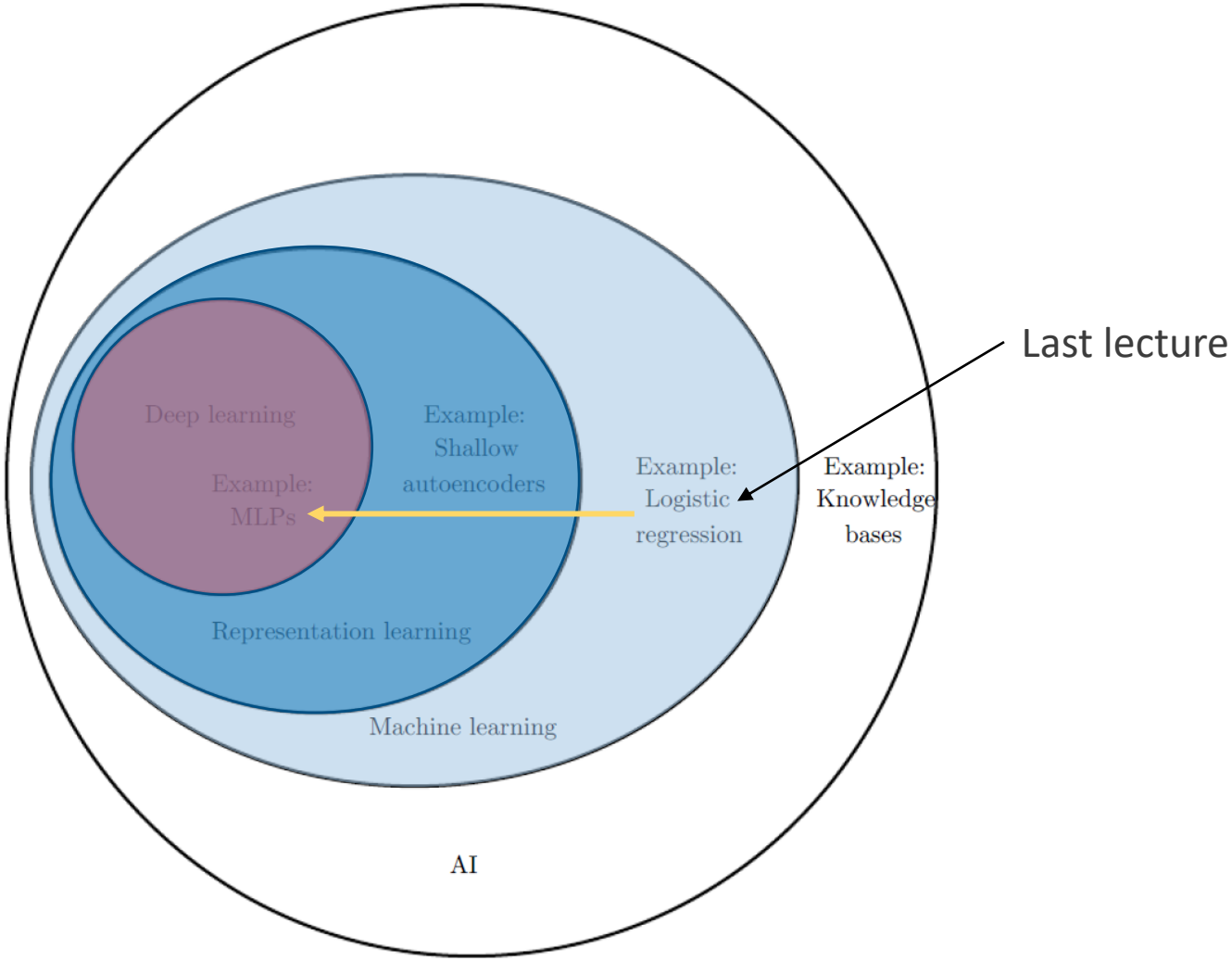
L07 Introduction to Deep Learning

B. Stuhr, D. Schneider, J. Haselberger

L07.1 Road to Deep Learning

Todays Trajectory

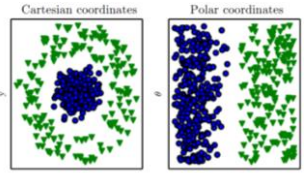
In this lecture we move from Machine Learning to Deep Learning



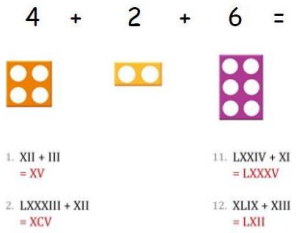
Deep Learning in the History of Learning

Abstract and Formal Tasks

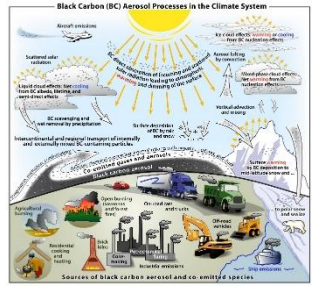
Informal Knowledge



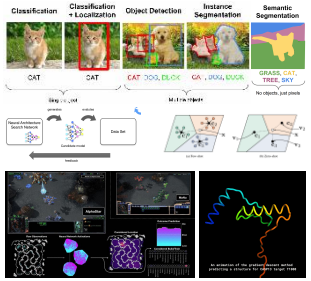
Machine Learning: Extracting own Knowledge from raw Data



Representation Learning: Learning to Represent Data in a different, useful way



Deep Learning: Disentangling Factors of Variation in the Representation

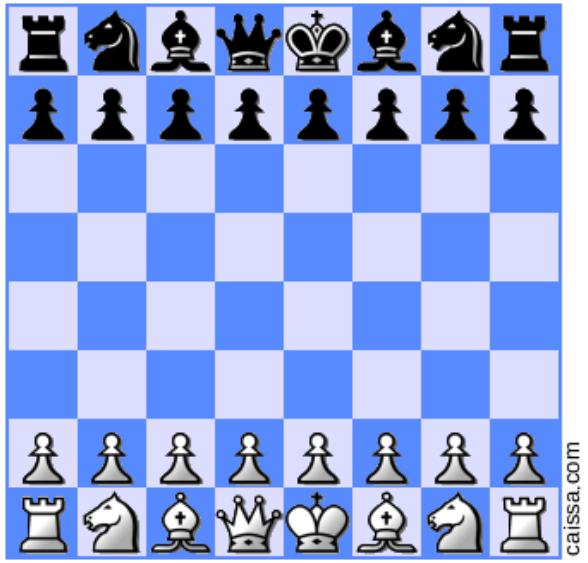


Deep Learning: Lets just learn everything we can

Early Days of AI: Abstract and Formal Tasks

- Early AI solves problems that can be described by formal, mathematical rules and formal environments (*e.g., Chess*)

➢ Ironically, abstract and formal tasks that are difficult for a human are among the easiest for a computer



Richard Greenblatt's „*Mac Hack*" – the first chess AI to participate in human tournaments (~1967, uses a transposition table) [Link]

True challenge: Solving tasks that are easy for people to perform but hard (for people) to describe formally

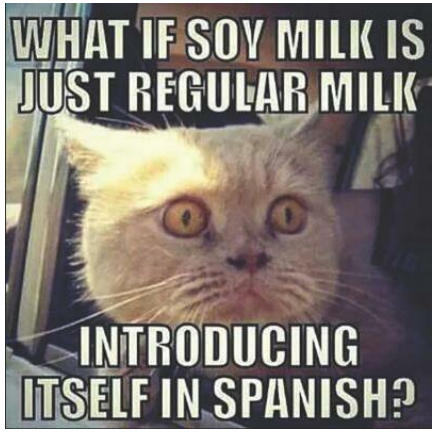
- Problems that we solve intuitively, that feel automatic
  - E.g., recognizing spoken words, or faces in images

Source: [Goodfellow2016]

Informal Knowledge

- Hard-code knowledge about the world in formal languages (e.g., *Knowledge bases*)

Different Meanings of the word "*cheater*" over context (and time)



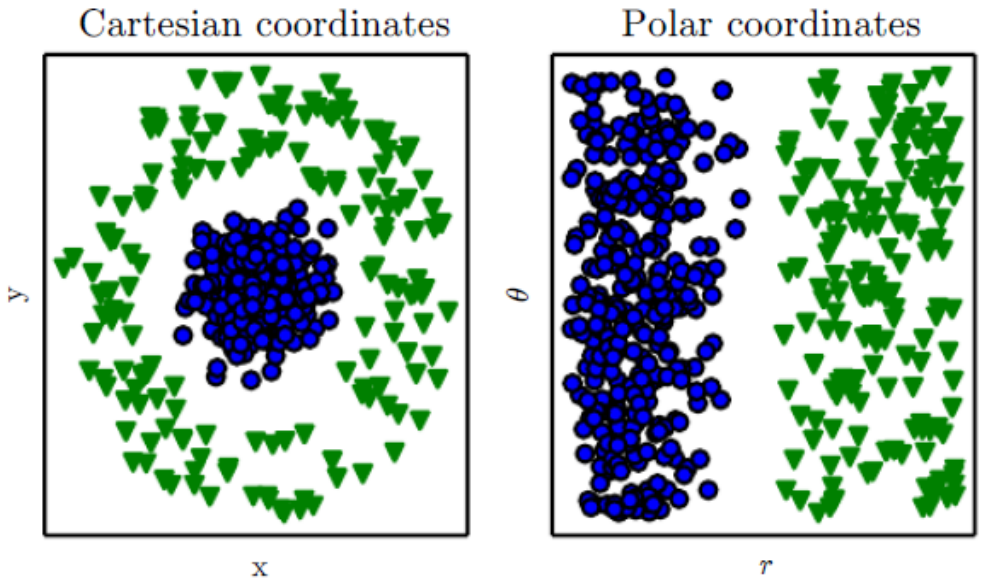
One of the key challenges in AI is how to get this *informal knowledge* into an agent

- Subjective, intuitive knowledge about the world is difficult to articulate in a formal way

Source: [Goodfellow2016]

Machine Learning: Extracting own Knowledge from raw Data

Machine Learning extract own knowledge from data instead of relying on a person as extractor
➢Therefore, Machine Learning needs raw data



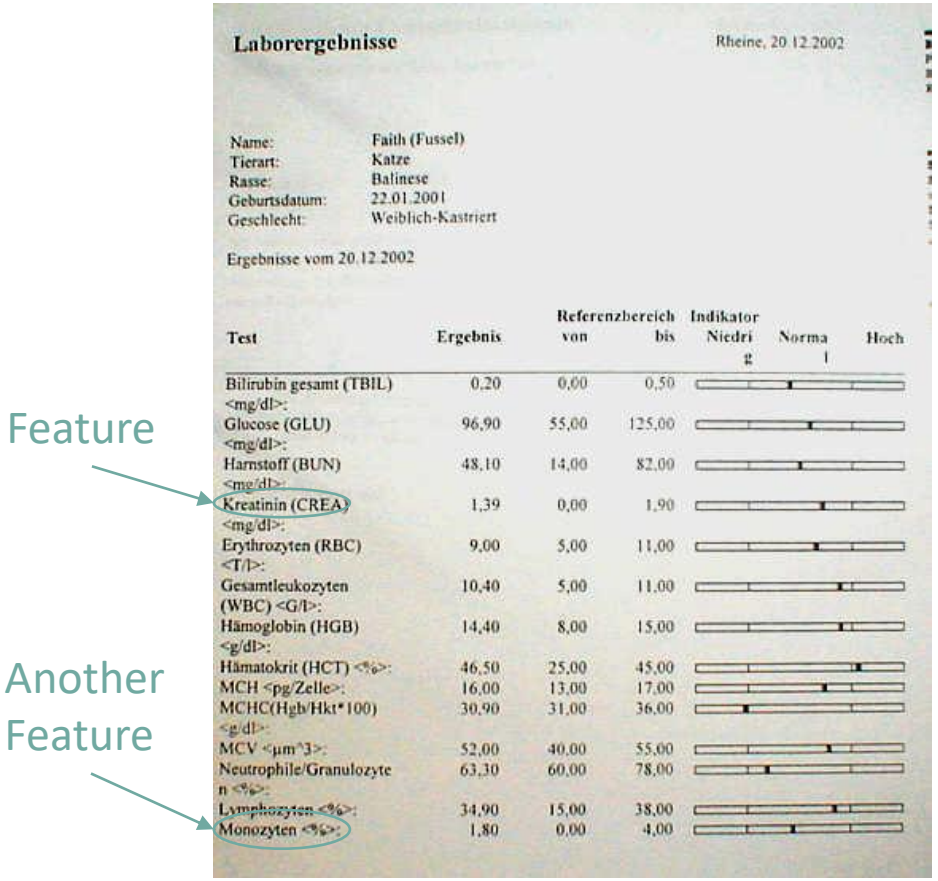
Cartesian and Polar coordinates

ML algorithms learn from raw data
→ performance depends heavily on the *representation* of the given "raw data"

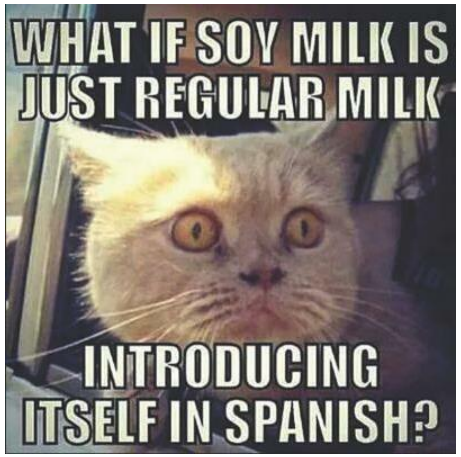
Source: [Goodfellow2016]

A *representation* can be seen as relevant pieces of information about the problem to be solved.

A *feature* is generally a recognizable property, that distinguishes one person, thing, or abstract context from another. It is each piece of distinct information included in the representation.



Representation of a cat to decide about a treatment [Link]



Representation of a cat as a Meme

Source: [Goodfellow2016]

Note: Dependence on Representation is a general Phenomenon

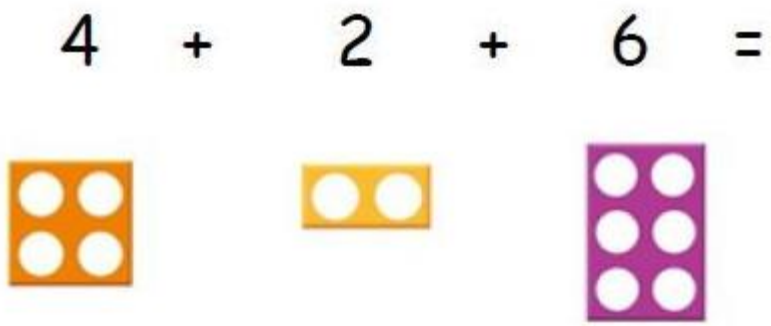
Which representation is easier for you to add the numbers?

1. XII + III
   = XV

2. LXXXIII + XII
   = XCV

11. LXXIV + XI
    = LXXXV

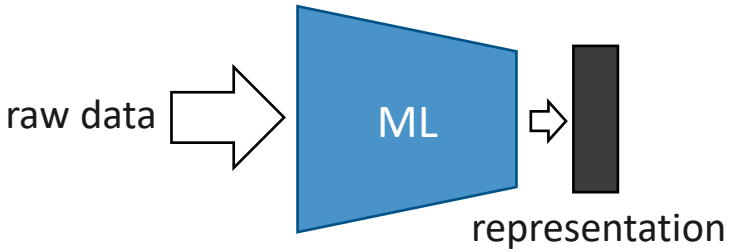
12. XLIX + XIII
    = LXII



Adding numbers in different representations. Images from [Link] [Link]

Representation Learning: Learning to represent Data in a different way

Representation Learning: Use ML to discover not only the mapping from representation to output, but also the representation itself



ML for Representation Learning

It can be very difficult to extract high-level, abstract features from raw data (complex *Factors of Variation*)

Source: [Goodfellow2016]

Factors of Variation

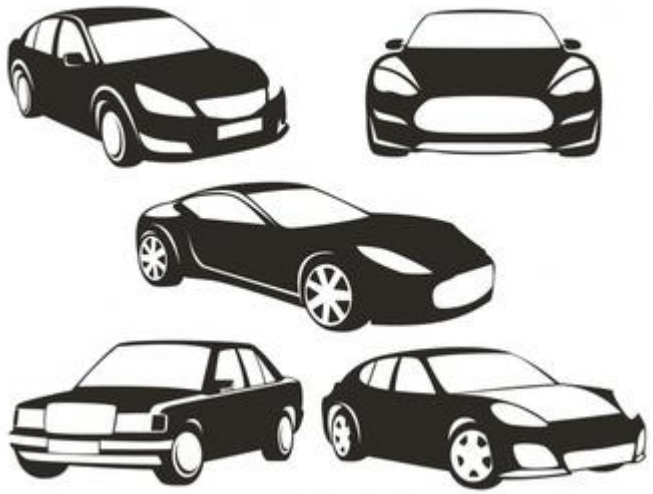
The word "*factors*" refers to separate sources of influence.

- E.g.: speech recording:
  - Speaker's age, sex, accent, the words that they are speaking, …
- E.g.: image of a car:
  - Position and color of the car, the angle and brightness of the sun, …
- Are often not quantities that are directly observed
  - Can be unobserved objects or forces in the physical world that affect observable quantities
  - May exist as constructs in the human mind that provide useful simplifying explanations or inferred causes of the observed data
  - Can be thought of as concepts or abstractions that help to make sense of the rich variability in the data

Source: [Goodfellow2016]

Disentangled Factors of Variation: Problem 1

Many of the factors of variation influence every single piece of observable data.



The shape of the car's silhouette depends on the viewing angle. Image from [Link]

Often it is required to *disentangle* useful factors of variation and *discard* the ones that we do not care about.

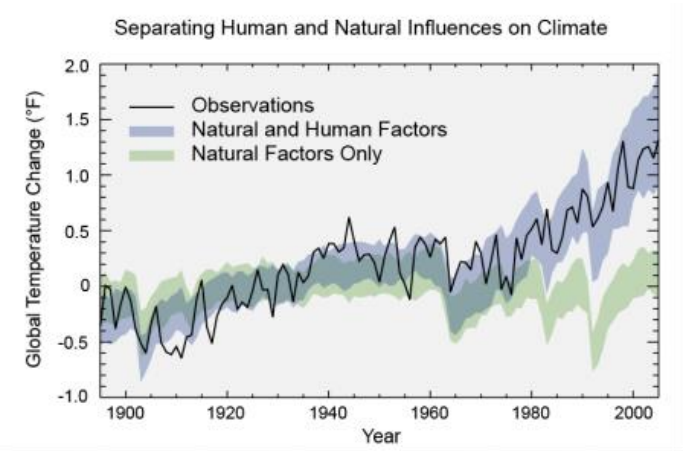
Factors of variation of a street scene. Image from [Link]

Source: [Goodfellow2016]

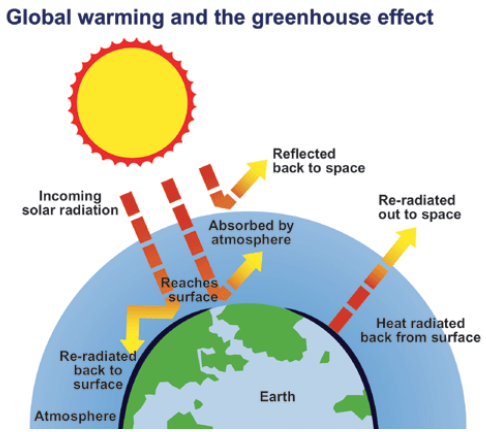
Disentangled Factors of Variation: Problem 2

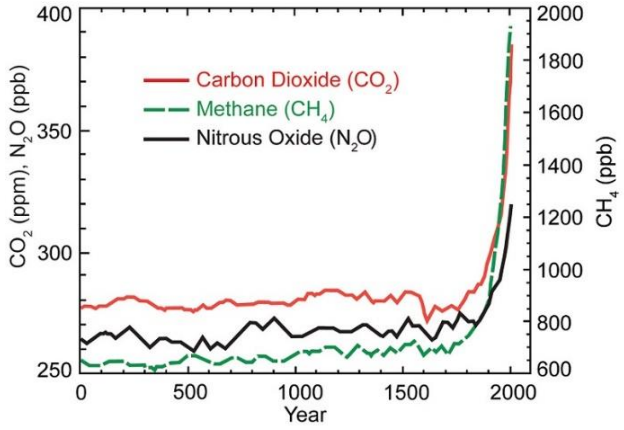
Many of these factors of variation can be identified only by using sophisticated, nearly human-level understanding of the data.

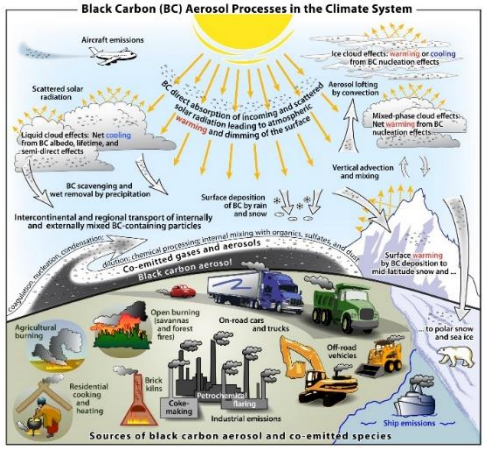
Observed Data



Knowledge needed, to be able to disentangle the Factors of Variation





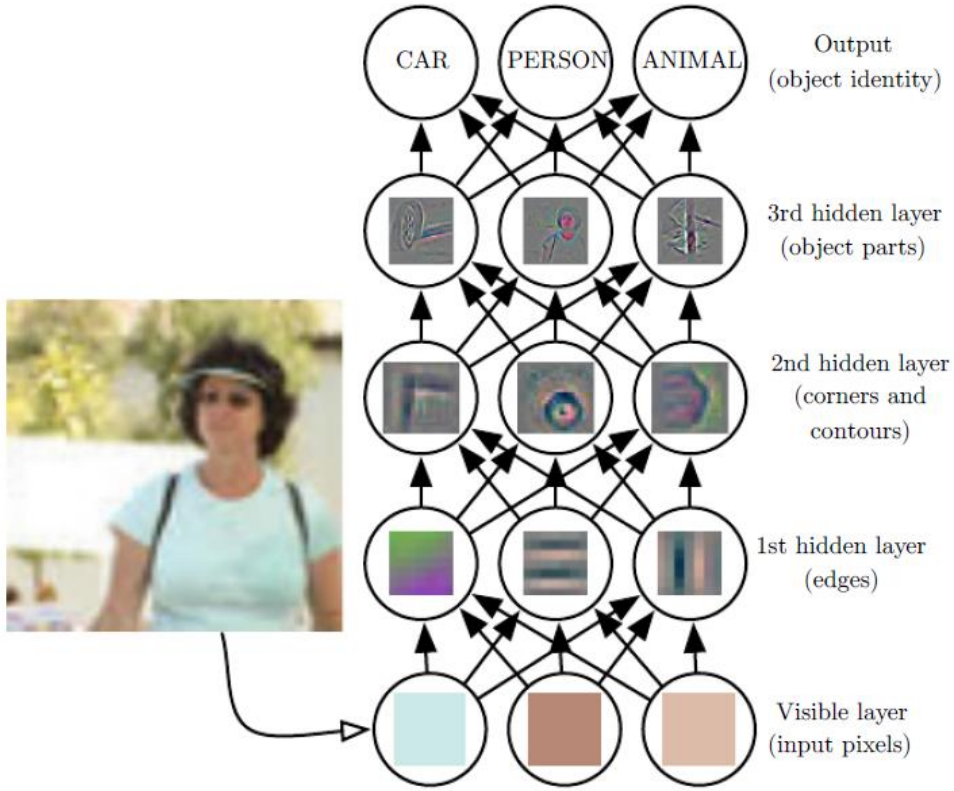


Disentangle factors of variation for climate change

Source: [Goodfellow2016]

Deep Learning: Disentangling Factors of Variation for a better Representation

DL solves this central problem in representation learning by introducing representations that are expressed in terms of other, simpler representations.

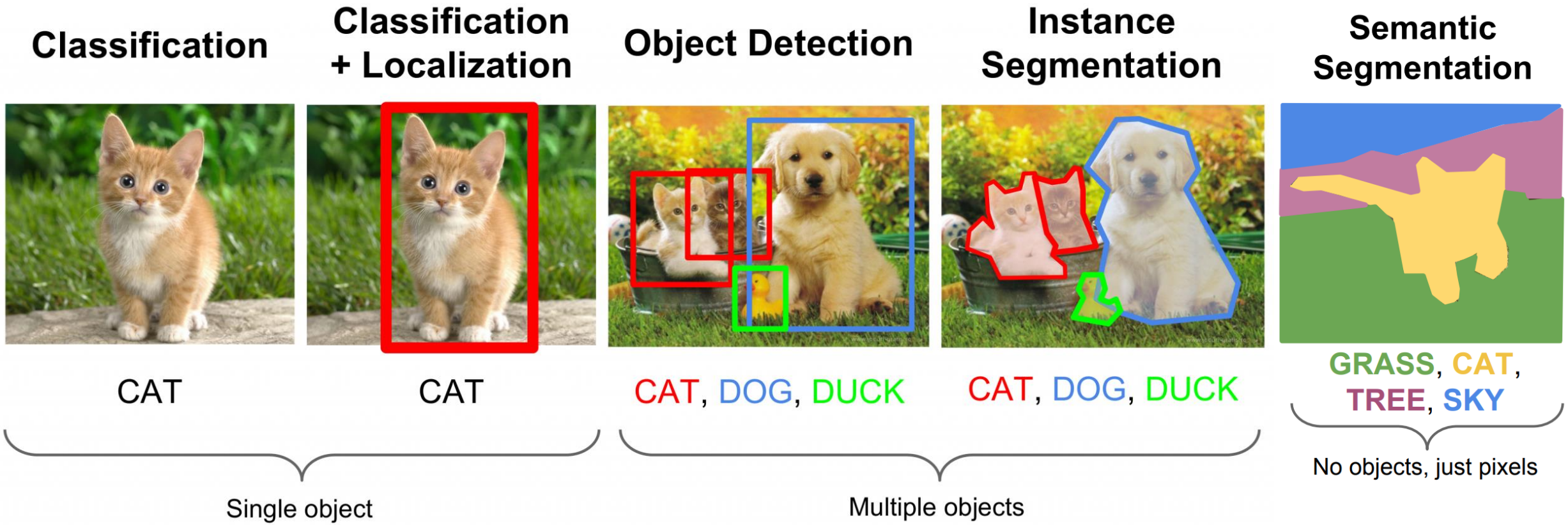


Deep Learning models can represent the concept of an image from a person by combining simpler concepts (e.g. ,corners and contours, which are defined in terms of edges)

Currently, this concept is very promising: Deep Learning methods improve performance on more and more problems

Source: [Goodfellow2016]

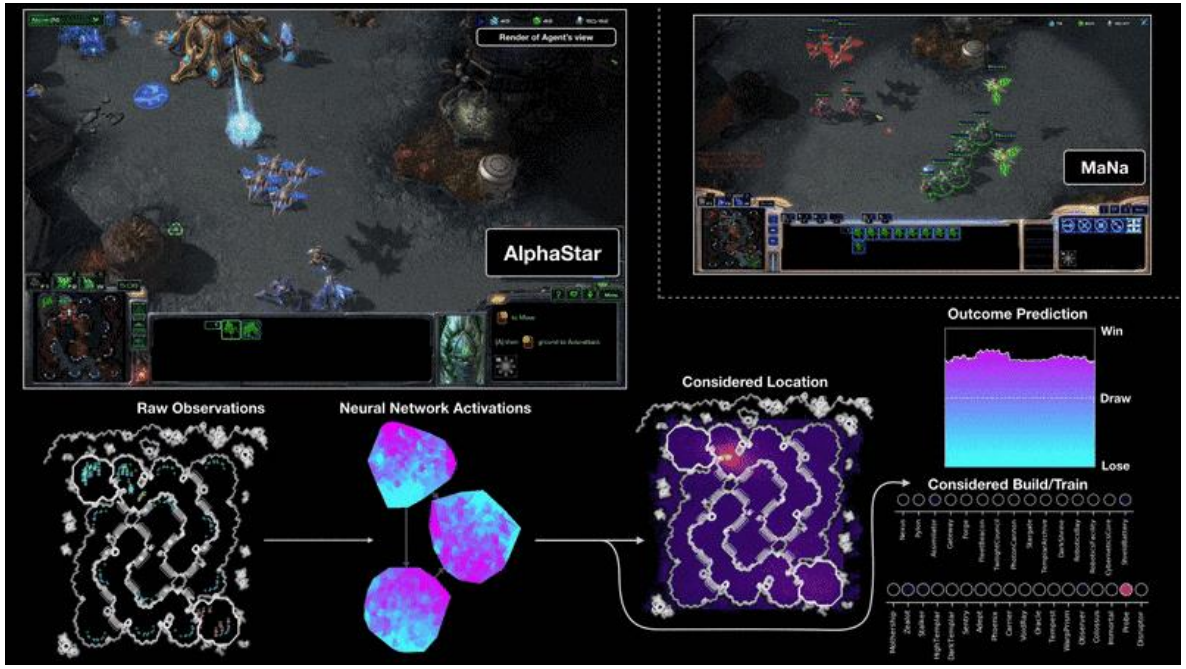
Deep Learning: Lets just learn everything we can (Motivation)



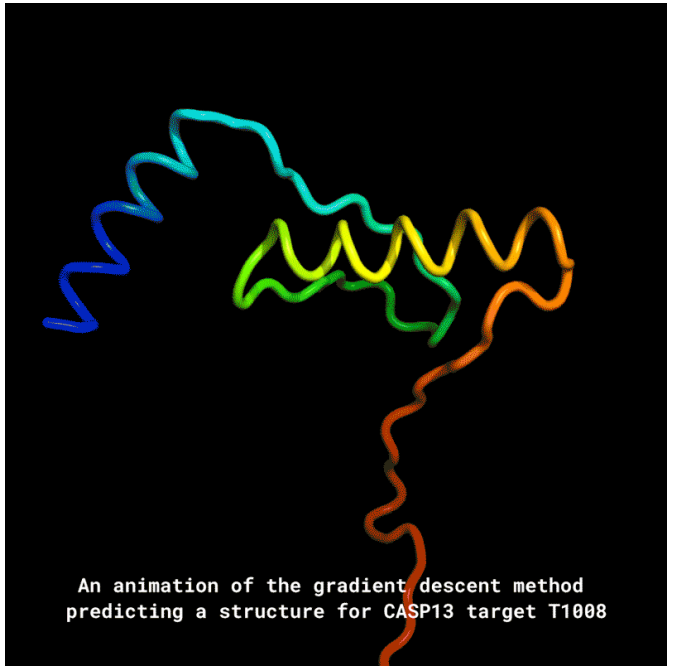
Deep Learning: Lets just learn everything we can (Motivation)

Pose Tracking in crowded scenes [Link]

Deep Learning: Lets just learn everything we can (Motivation)

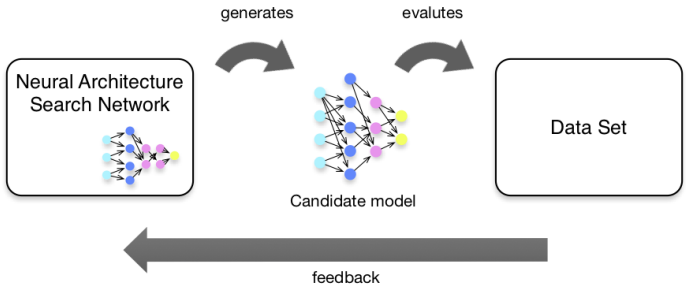


AlphaStar plays Starcraft [Link]

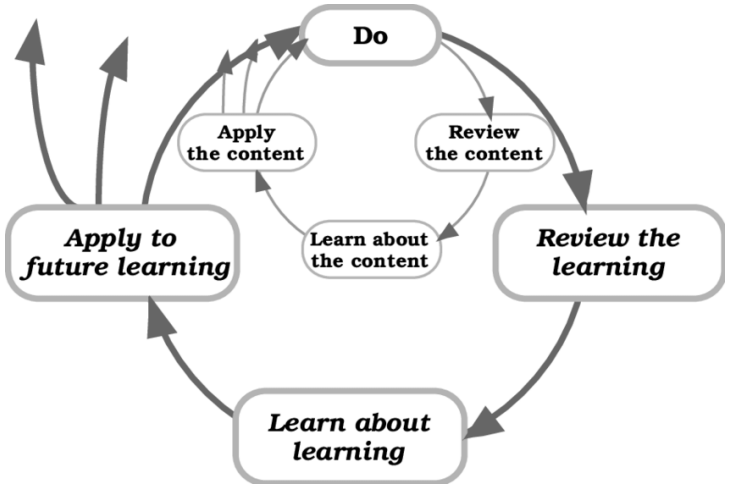


AlphaFold folds protein [Link]

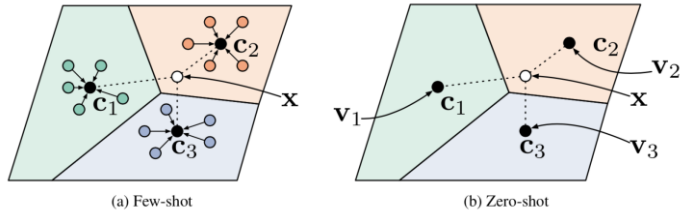
Deep Learning: Lets just learn everything we can (Motivation)



AutoML trains the model for you [Link]



DL models are learning to learn via Meta Learning [Link]

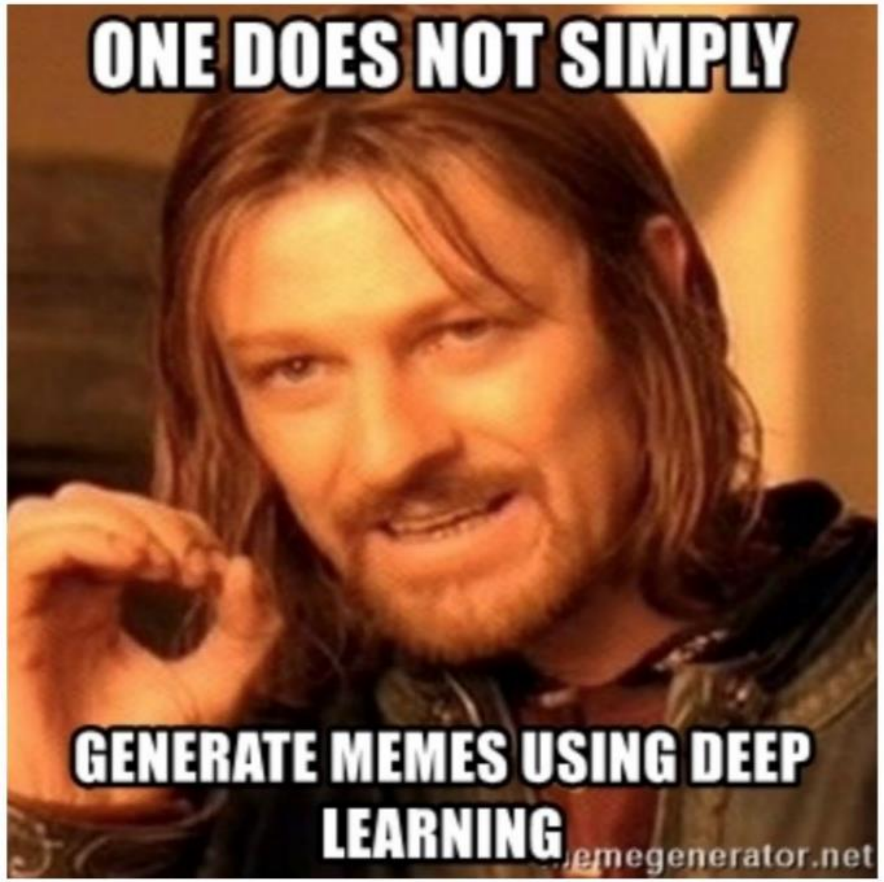


DL models learn for few examples with few-shot learning [Link]

Deep Learning: Lets just learn everything we can (Motivation)

StyleGanV2 creates realistic humans and more (create you own person under https://thispersondoesnotexist.com)

Deep Learning: Lets just learn everything we can (Motivation)



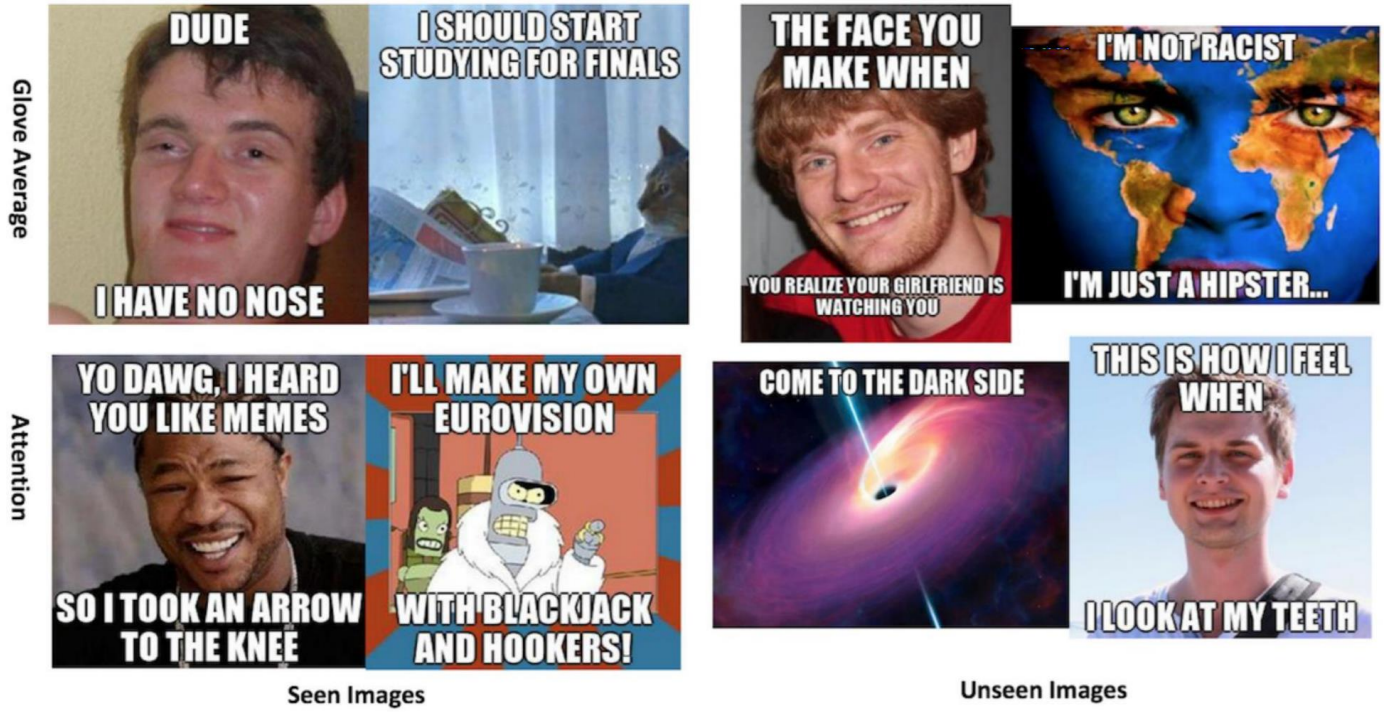


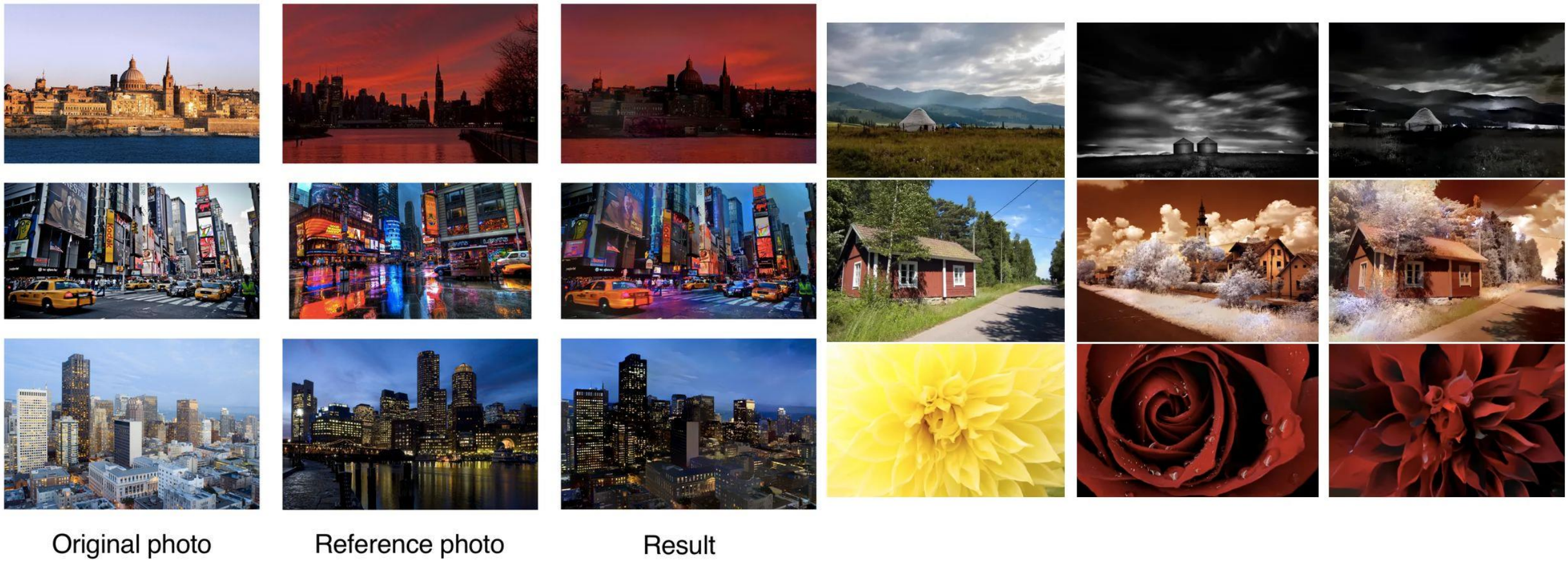
Figure 4: Original memes generated by both model variants. The input labels for the seen images are their labels from the training set while for the unseen image we use 'AI is the new electricity' for all 4.

Deep Learning Meme Generation [Peirson2018]

Deep Learning: Lets just learn everything we can (Motivation)

World-Consistent Video-to-Video Synthesis creates realistic looking street scenes [Mallya2020]

Deep Learning: Lets just learn everything we can (Motivation)



Neural Style Transfer [Luan2017]

Deep Learning: Lets just learn everything we can (Motivation)

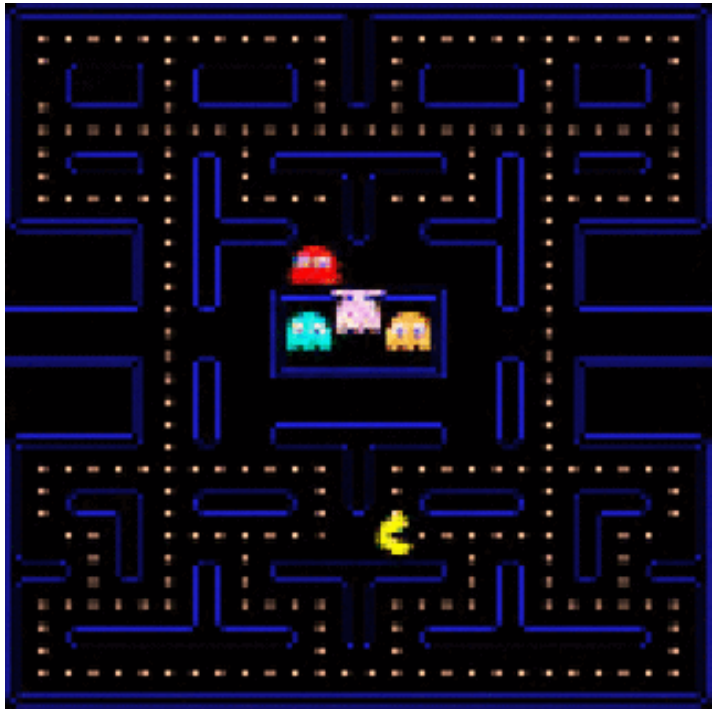
Neural Style Transfer [Link]

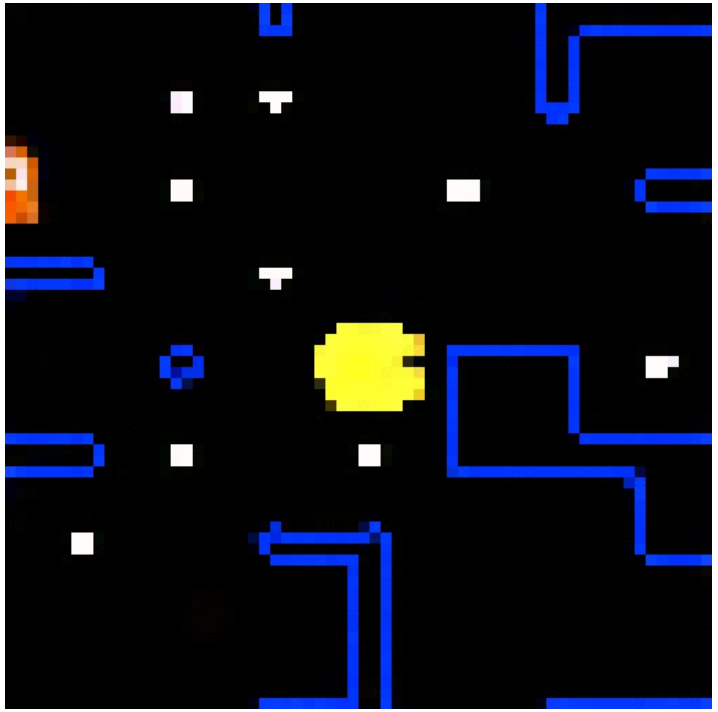
Deep Learning: Lets just learn everything we can (Motivation)

SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez. Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns. While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic." Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America. While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, "In South America, such incidents seem to be quite common." However, Pérez also pointed out that it is likely that the only way of knowing for sure if unicorns are indeed the descendants of a lost alien race is through DNA. "But they seem to be able to communicate in English quite well, which I believe is a sign of evolution, or at least a change in social organization," said the scientist.

Transformers (GPT2 in this case) create realistic text in a certain style [Link]

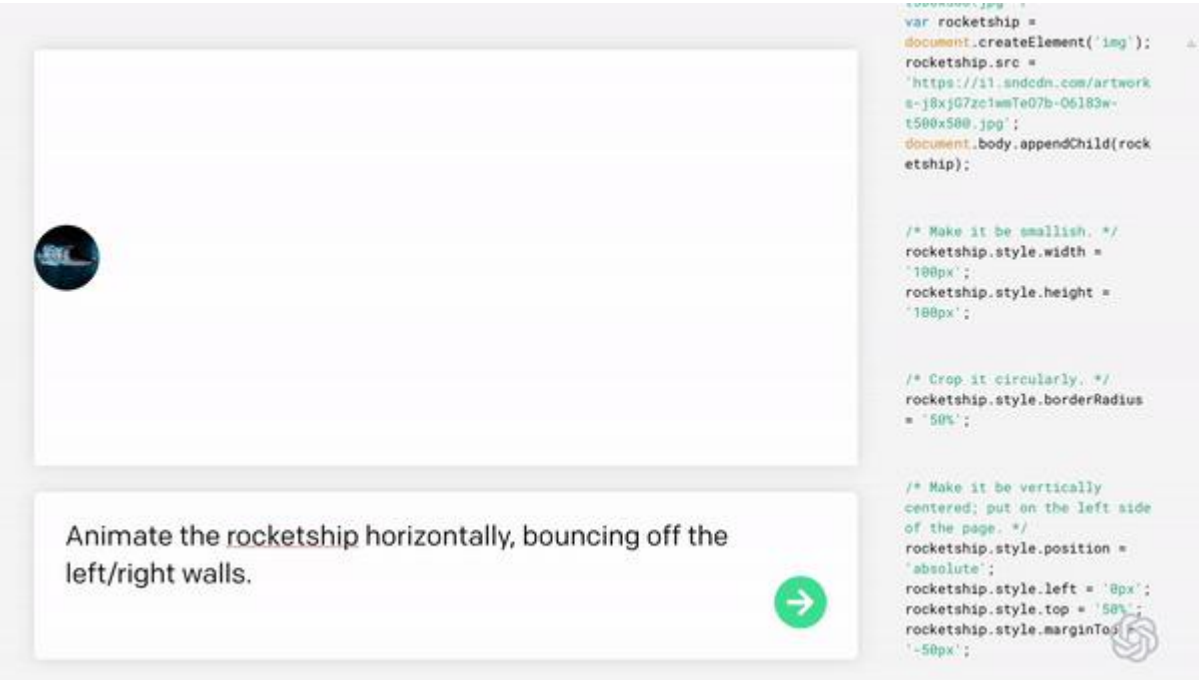
Deep Learning: Lets just learn everything we can (Motivation)

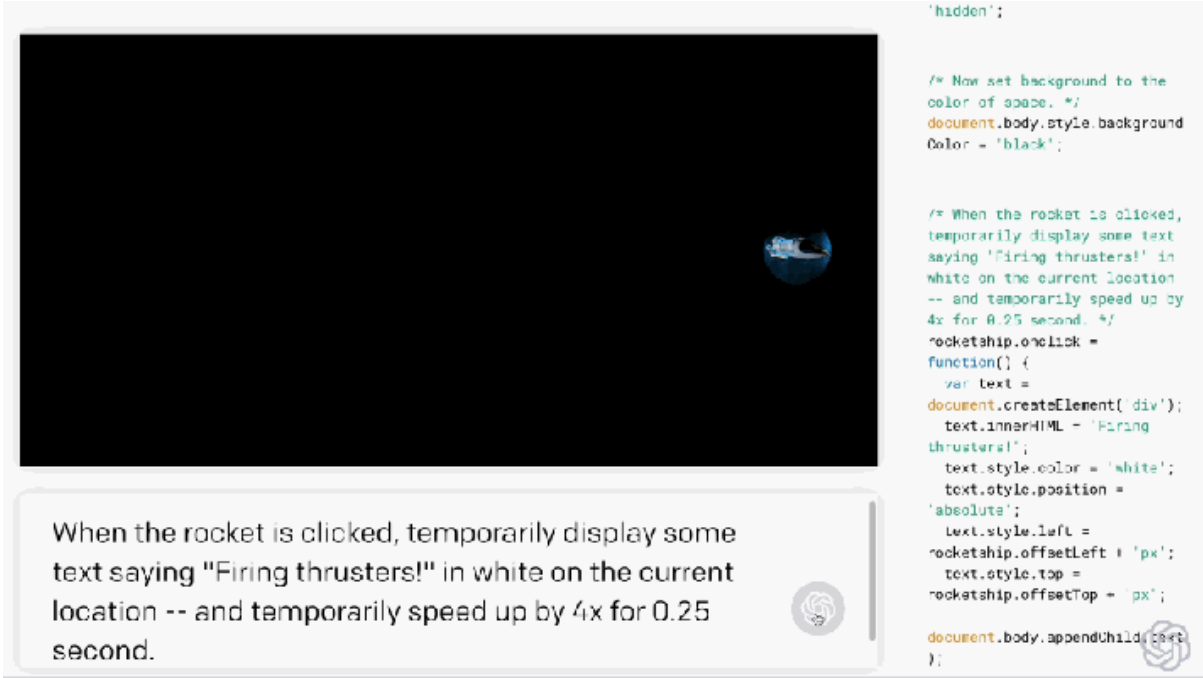




GameGAN creates an entire game (only inputs for the controller are given to the model) [Link]

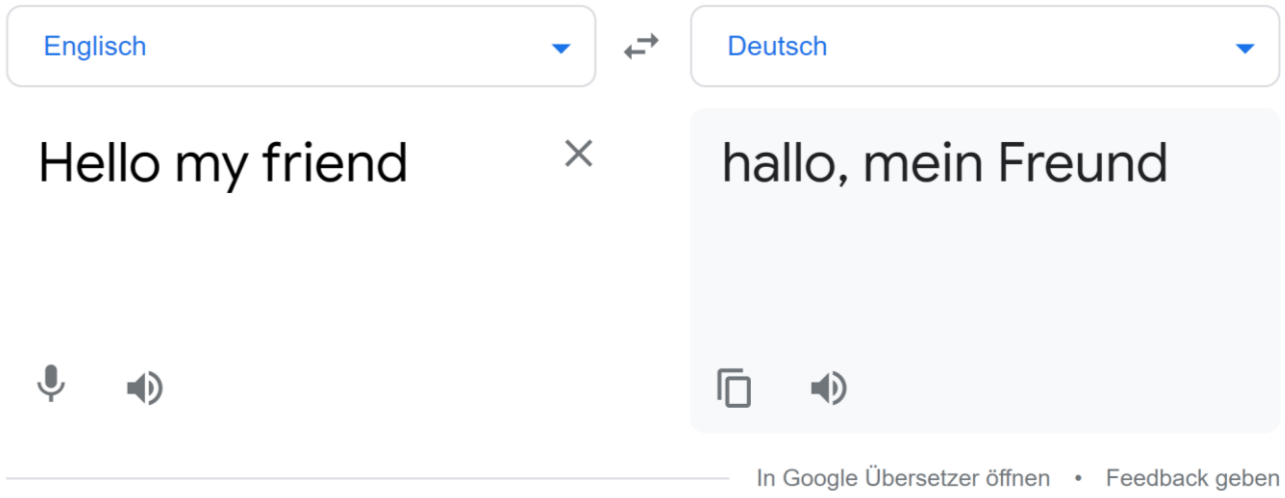
Deep Learning: Lets just learn everything we can (Motivation)



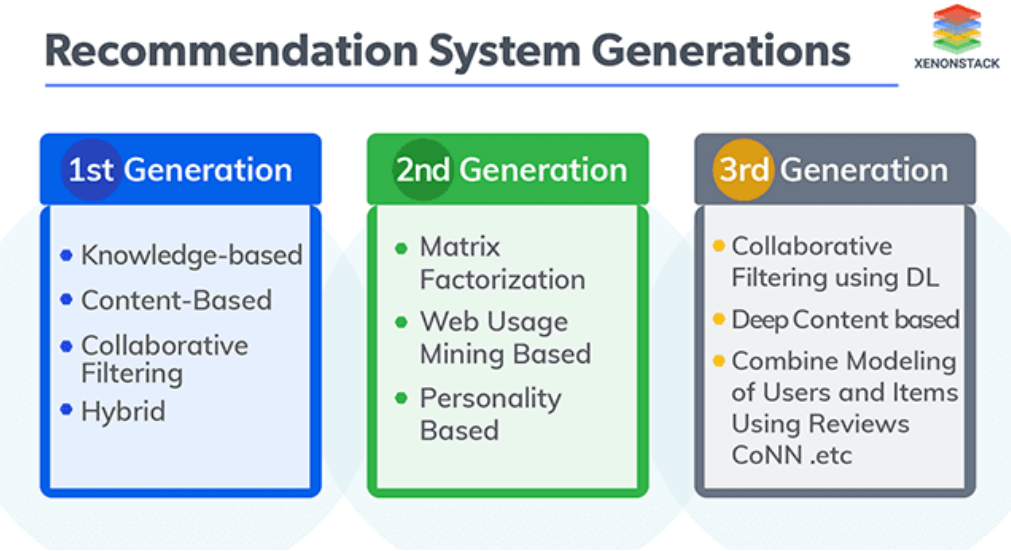


OpenAI Codex generates Code from a Description [Link]

Deep Learning: Lets just learn everything we can (Motivation)



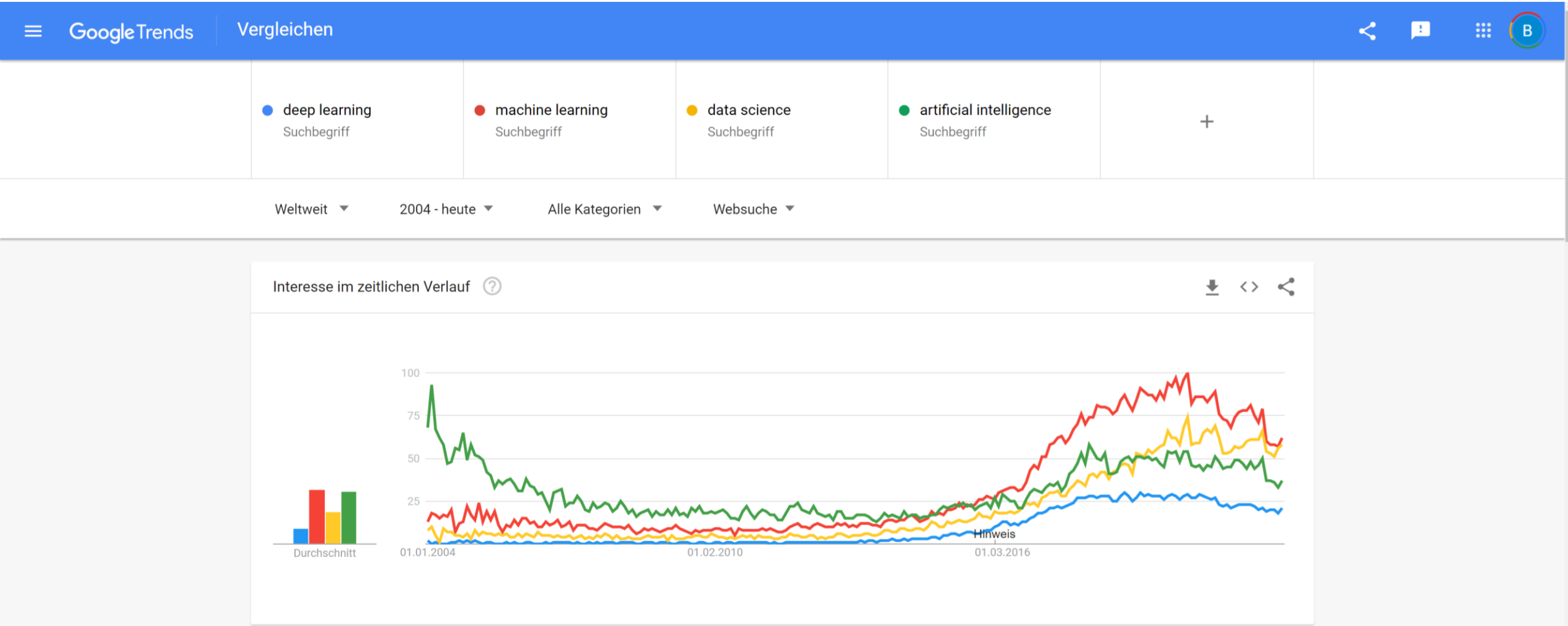
Every sentence that is translated to you by Google, DeepL, … [Link]



And of course, every content that is recommended to you in YouTube, Netflix, Amazon, Facebook, Instagram, TikTok, … [Link]

And much, much more … current trend: **Smart Everything**

The Deep Learning Boom and why it is happening (now)



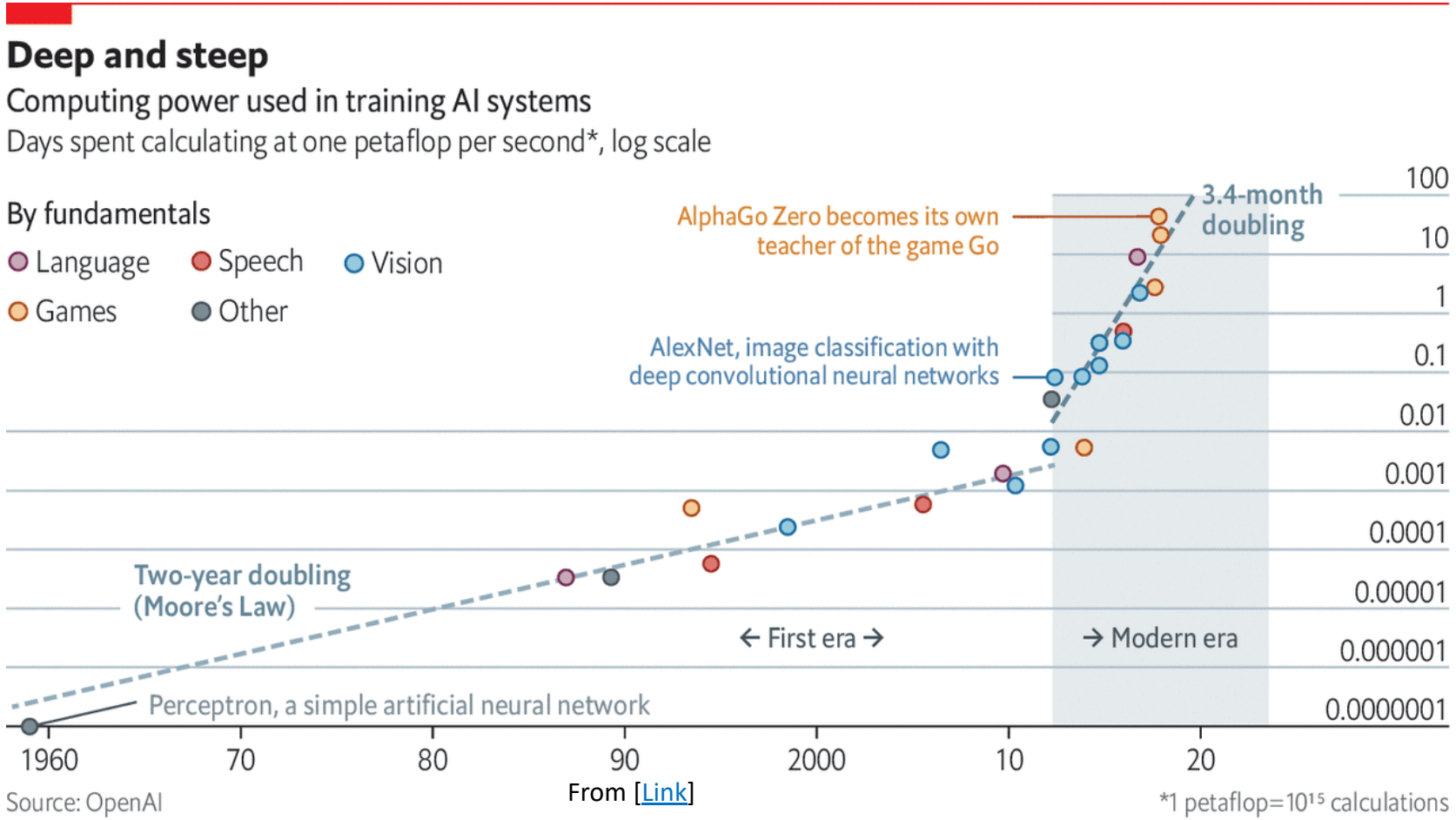
[Link]

The Deep Learning Boom and why it is happening (now)

- Main reason for the boom: More data, more computational power to process it (GPUs, TPUs …)
- Led to significant performance increase shown by the AlexNet model (explained in future lecture)

Image from [Link]

- SOTA (state-of-the-art) Deep Learning models are often hungry for computational power and data!

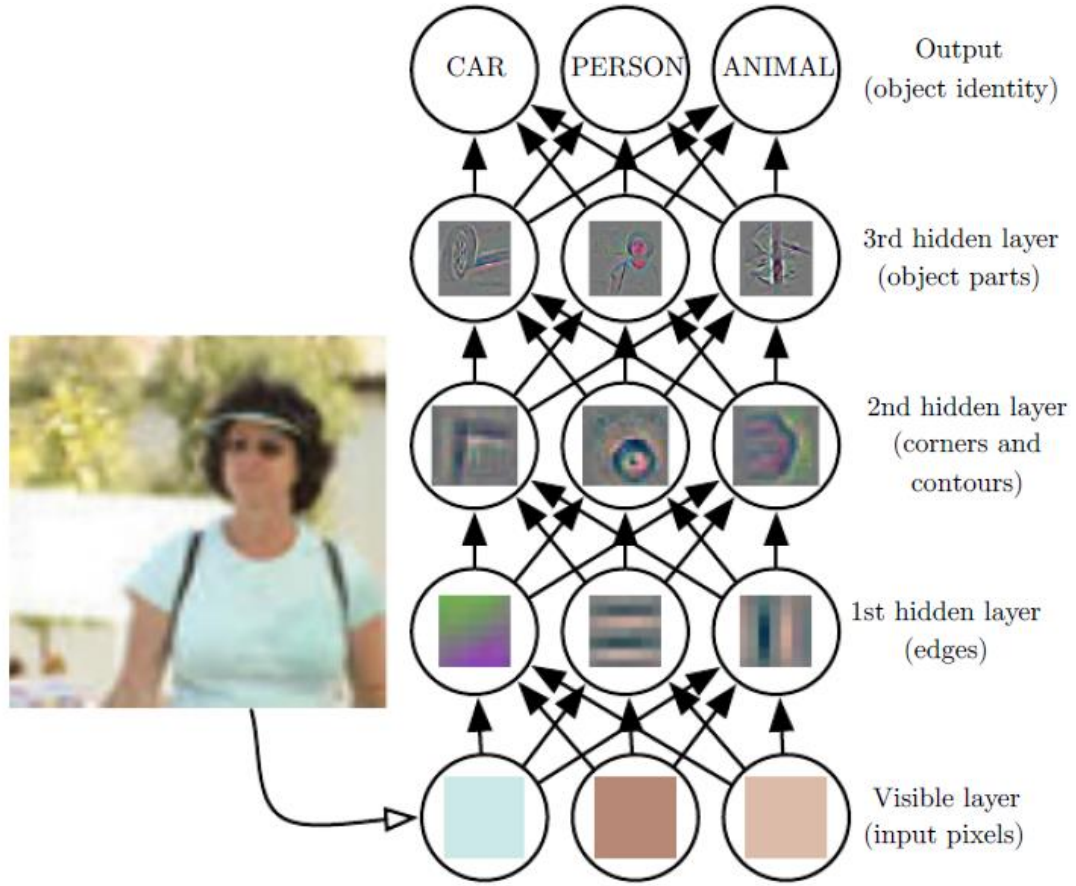


From [Link]

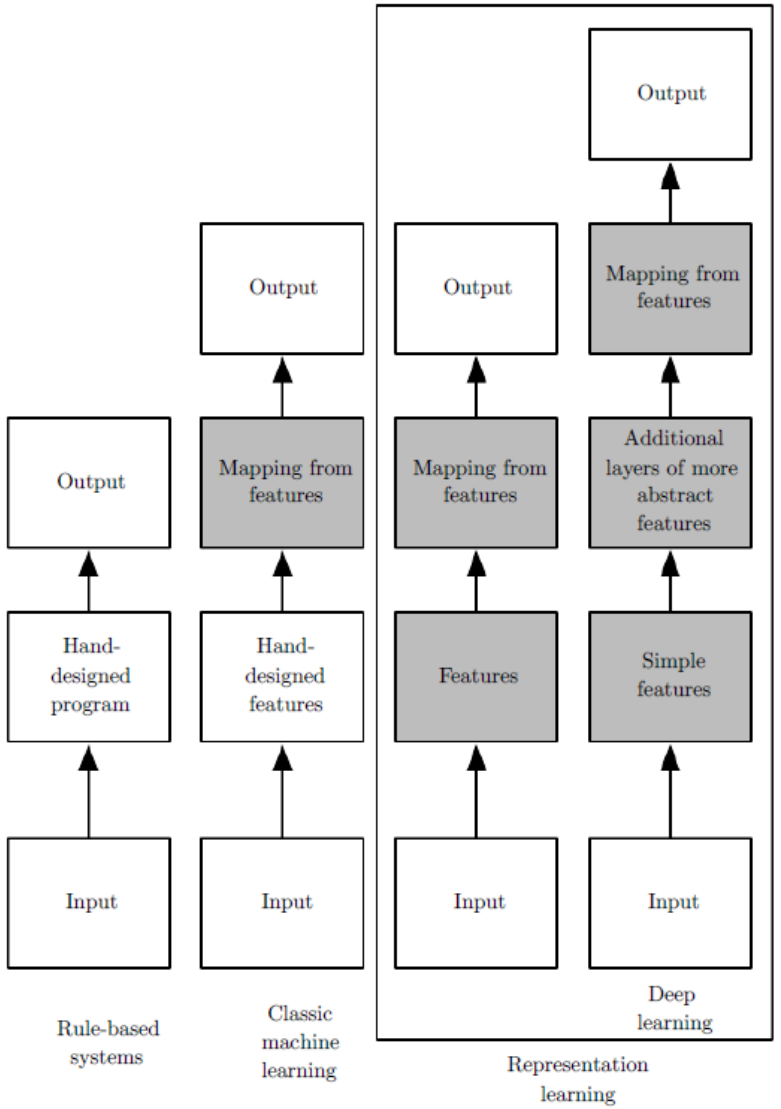
One petaflop corresponds to one quadrillion or 1,000 trillion floating-point operations per second!

Summary

Deep Learning is a particular kind of ML that learns to represent the world as a nested hierarchy of concepts. Each concept is defined in relation to simpler concepts. More abstract representations are computed in terms of less abstract representations.

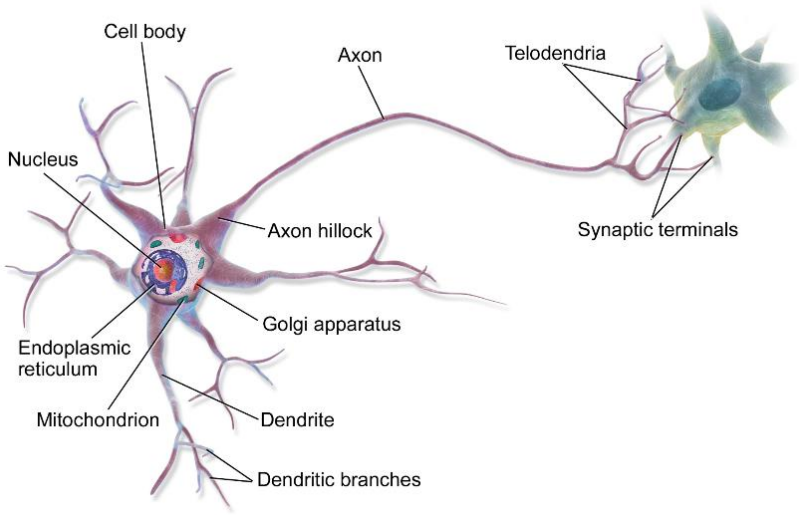


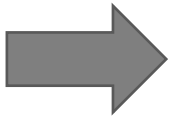
Source: [Goodfellow2016]

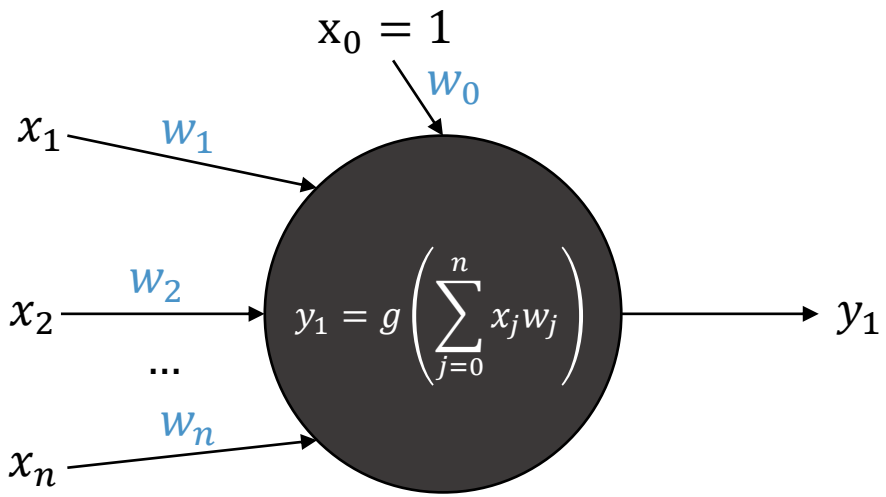


Flowcharts showing how the different parts of an AI system relate to each other within different AI disciplines. Shaded boxes indicate components that can learn from data

Source: [Goodfellow2016]







L07.2 Constructing an artificial Neuron

Inspiration: A oversimplified biological Neuron

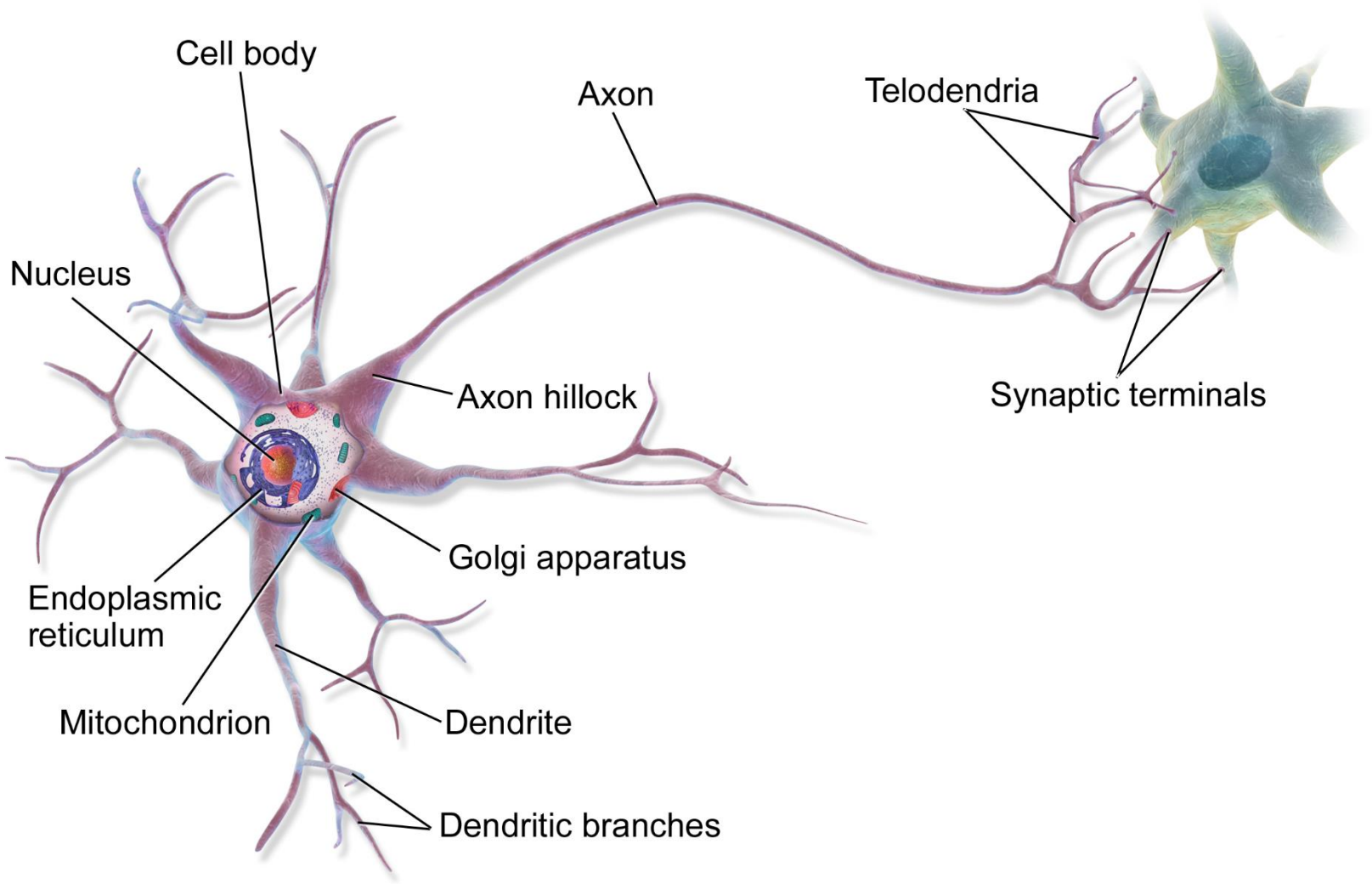
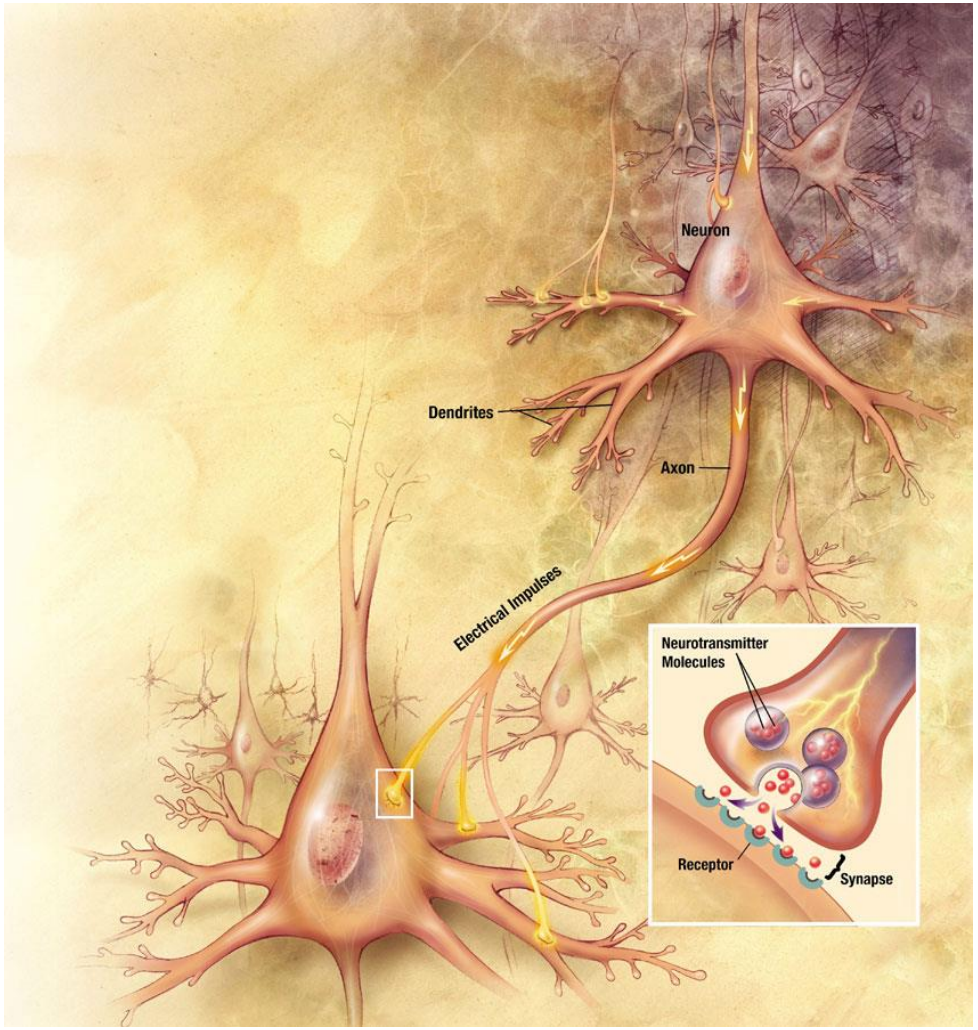
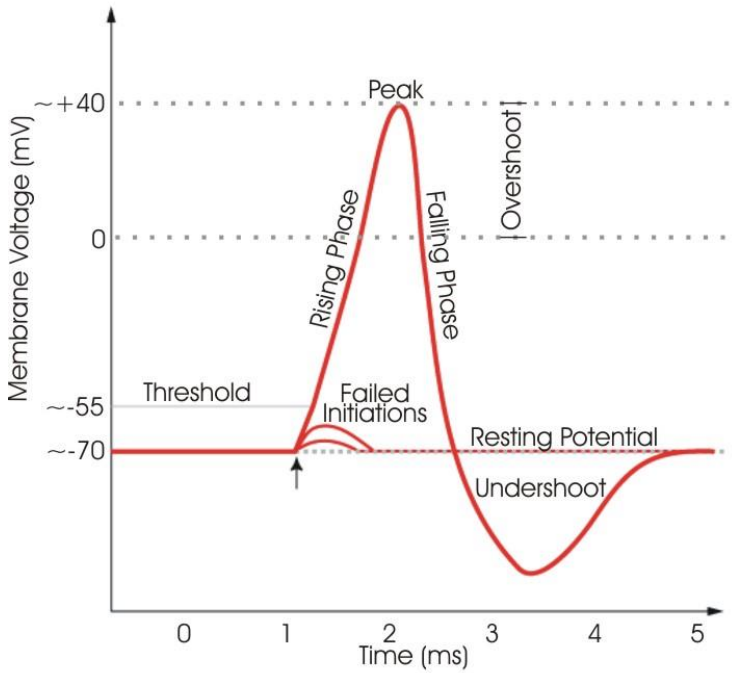


Image from: [Link]

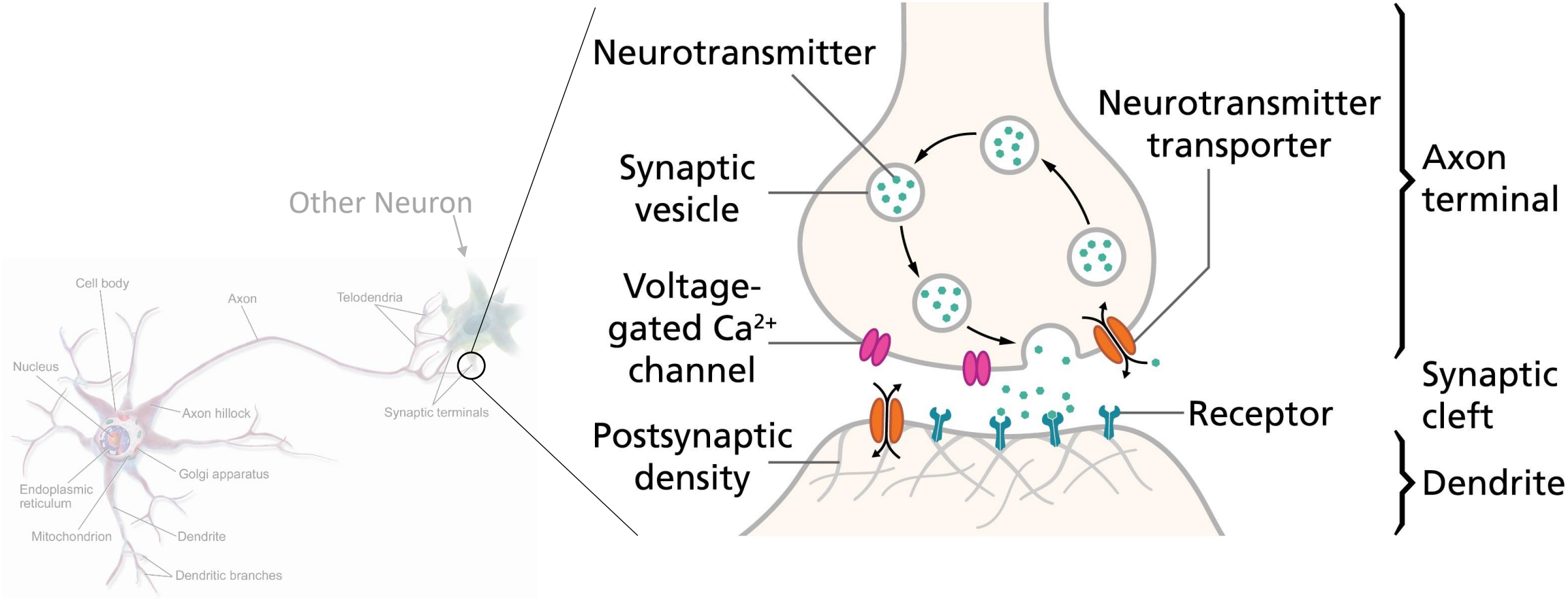
Inspiration: Spikes





Images from: [Link]

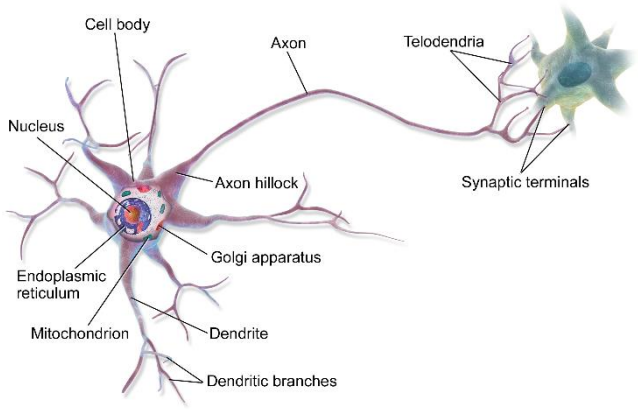
Inspiration: At the Synapses, the Network learns (oversimplified)

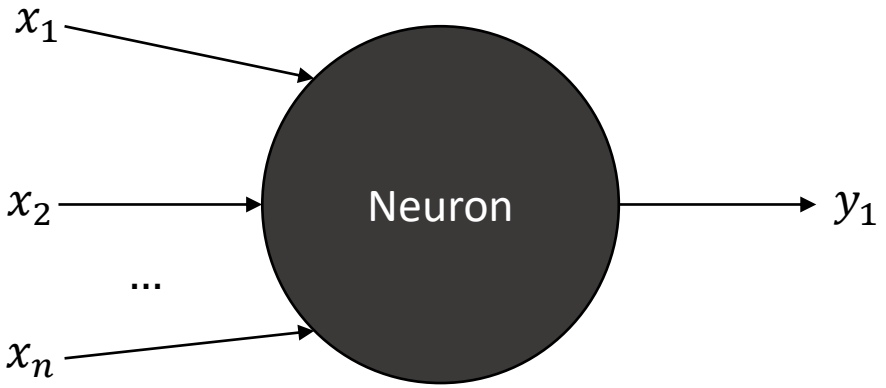


Altering properties of the connections (synapses) between neurons is one way to adapt the neural network. When the adaption is chosen well, **the network learns**

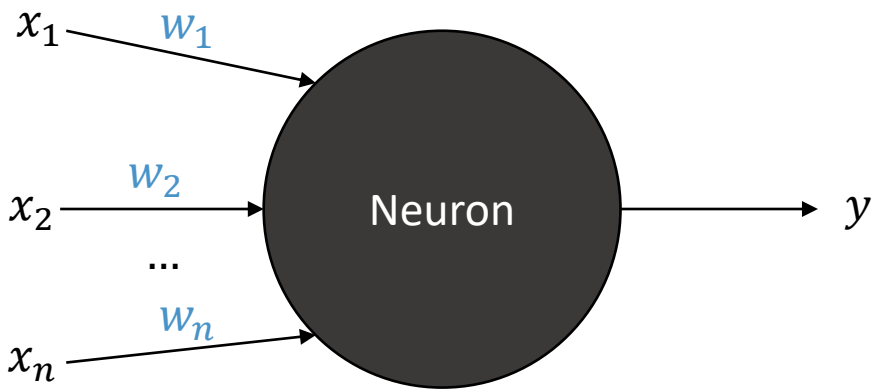
Images from: [Link]

Let‘s create a Model of our Neuron



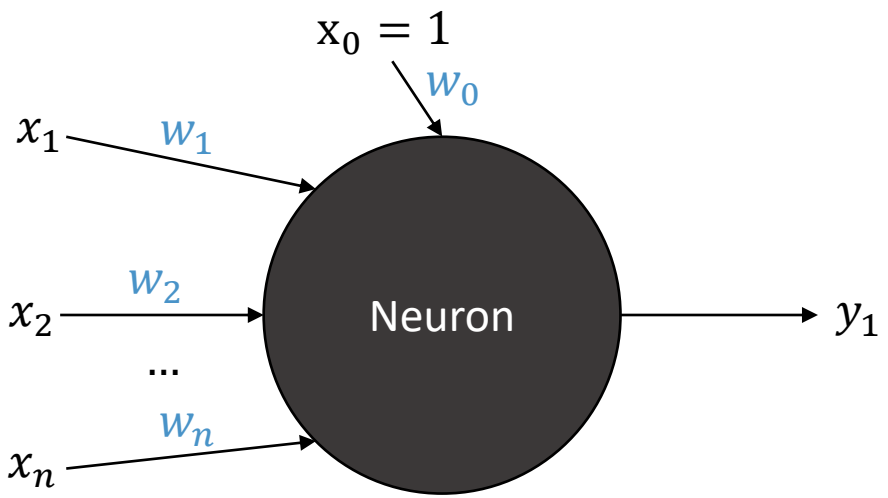


Let’s model the Synapses: Weights



1. We model the inputs from the *dendrites* with x_n
   - Each input is a single number.
2. We model *synapses* through *weights* w.
   - Weights are single numbers which we can change.
   - Weights can be changed by a learning procedure, which we explain later.
3. We model the neurons output through the *axon* with y

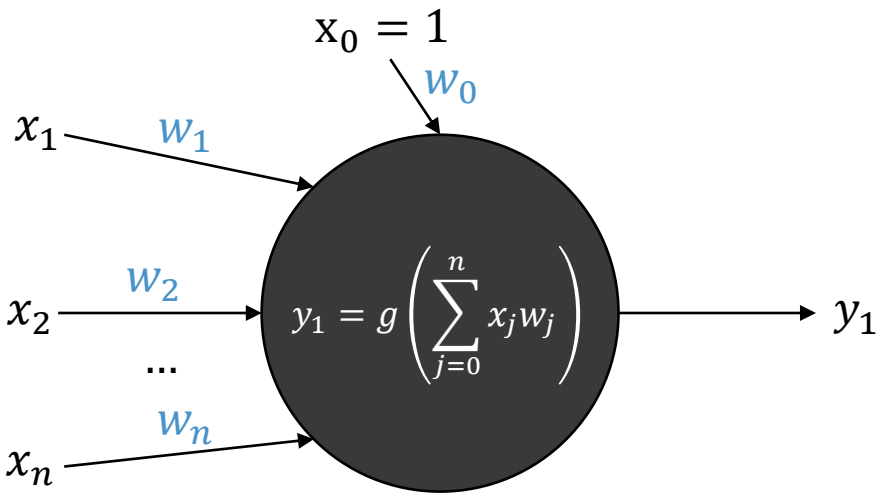
Lets model the Resting Potential: Bias

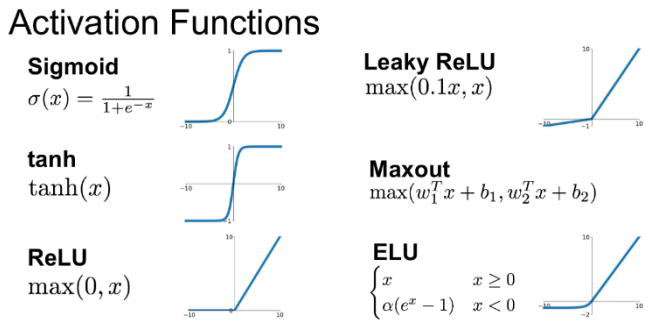


We add a *bias* input w_0.

- This can be seen as the neurons *resting potential*.
- The bias is important for mathematical reasons as well, which we explain later.

Let's model the Axon Hillock and Threshold: Summation + Activation Function

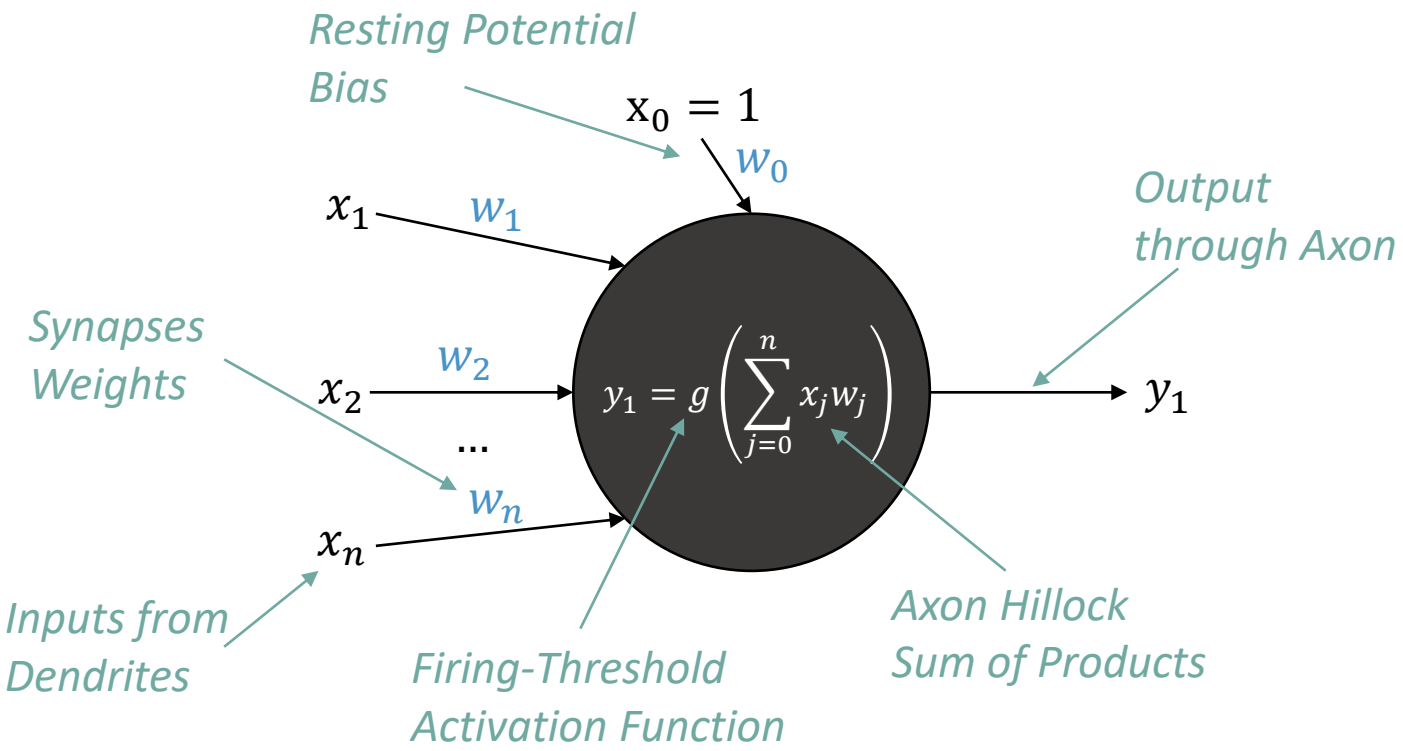




1. In our *axon hillock*, we *multiply each input* with its weight, then we take the *sum of all products*.
2. This sum is passed through an *activation function* g, which **can** model the *firing threshold* of the neuron.
   - In Deep Learning we can choose from a variety of activation functions, the right figure shows some commonly used functions.

```
neuron = torch.nn.ReLU(torch.nn.Linear(in_features=x, out_features=1, bias=True))
```

Final Artificial Neuron: **The Perceptron**



Artificial Neuron: Some Math

Neuron:

$$y = g\left(b_0 + \sum_{j=1}^{n} x_j w_j\right)$$

with $b_0 = x_0 w_{k0}$ *and* $x_0 = 1$:

$$y = g\left(\sum_{j=0}^{n} x_j w_j\right)$$

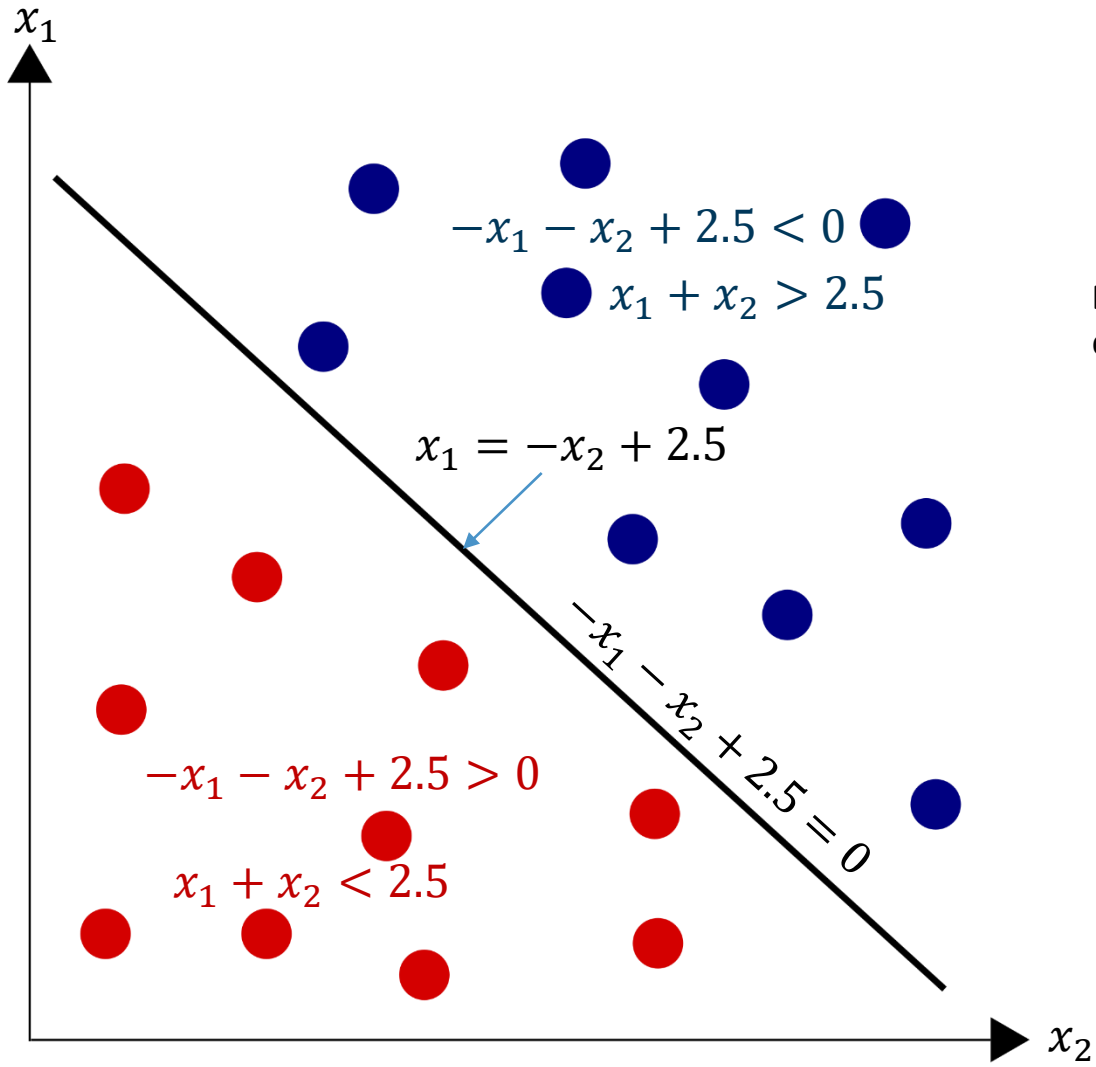
- b: bias
- x: input to neuron
- w: weights
- n: the number of inputs (from the incoming layer)

in vector (or matrix) form:

$y = g(\boldsymbol{x}\boldsymbol{w}^T)$, where $\boldsymbol{x} = \begin{bmatrix} x_0 \\ \dots \\ x_n \end{bmatrix}$ and $\boldsymbol{w} = \begin{bmatrix} w_0 \\ \dots \\ w_n \end{bmatrix}$

$$\boldsymbol{x}\boldsymbol{w}^T = \begin{bmatrix} x_0 \\ \dots \\ x_n \end{bmatrix} [w_0, \dots, w_n] = x_0 w_0 + \cdots + x_n w_n = \sum_{j=0}^{n} x_j w_j$$

Linear Separation + Example



Without bias the neuron is not able to represent every line!

$$y = g\left(b_0 + \sum_{j=1}^{n} x_j w_j\right)$$

without g:

Equation of a straight line in n-dimensional space

$$y = b_0 + \sum_{j=1}^{n} x_j w_j$$

$$y = x_1 w_1 + \cdots + x_n w_n + b_0$$

$$0 = x_1 w_1 + \cdots + x_n w_n + b_0$$

Equation of a straight line in two-dimensional space

"$\boldsymbol{y = mx + b}$"

let's plug in numbers ($n = 2$):

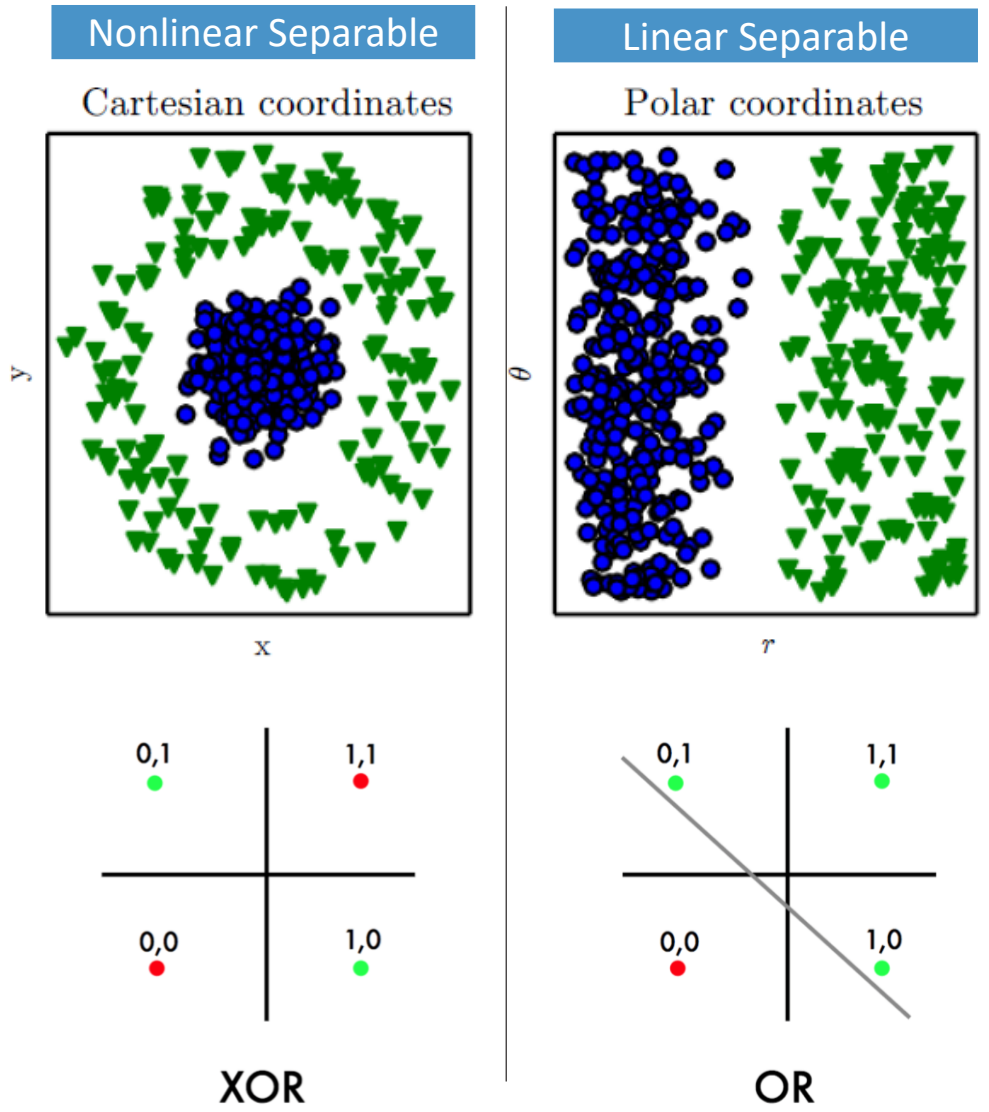
$$0 = w_1 x_1 + x_2 w_2 + b_0$$

$$-w_1 x_1 = x_2 w_2 + b_0$$

let's plug in more numbers:

$$x_1 = -x_2 + 2.5$$

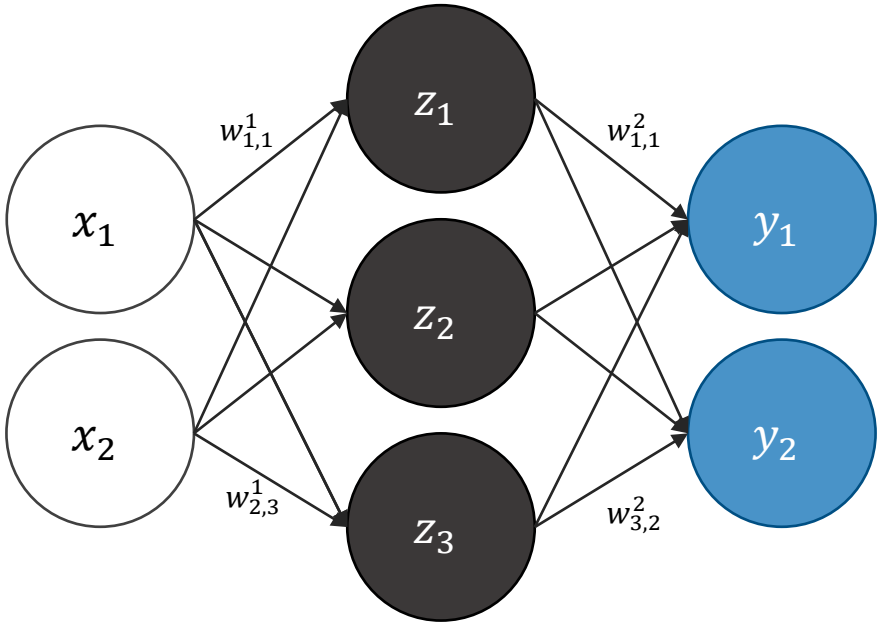
Non Linear Seperable → MLP



A simple neuron can not solve nonlinear problems!

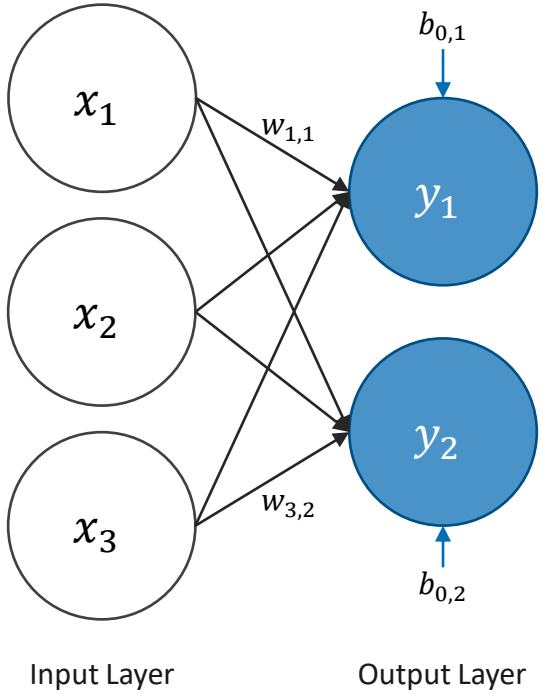
- Stacking layers of neurons to a MLP (MultiLayer Perceptron)

Source: [Goodfellow2016]



L07.3 Connecting neurons to a MultiLayer Perceptron (MLP)

The Multi**output** Perceptron



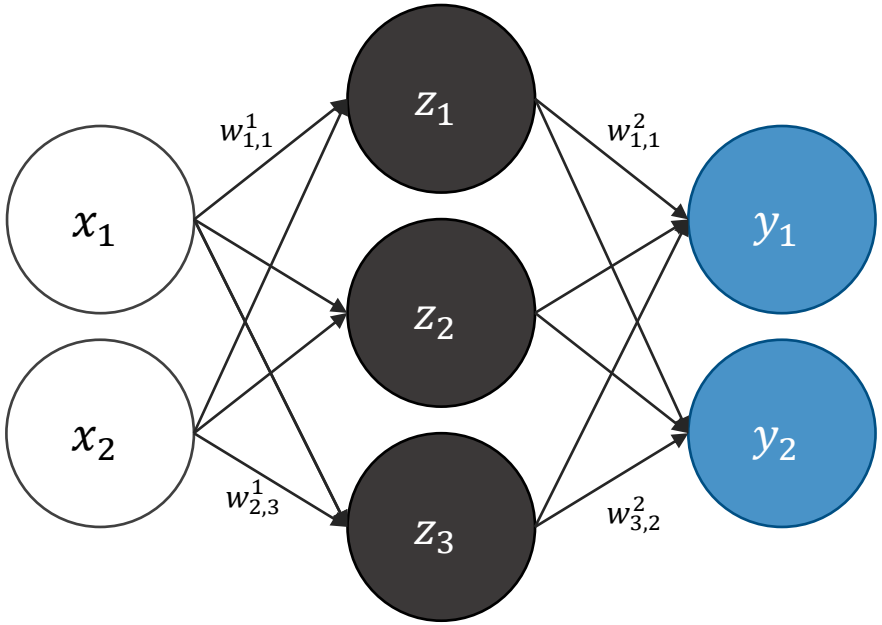
We define different Layers of the Neural Network

1. *Input Layer*: Represents the input two the Neural Network
2. *Output Layer*: Represents the output of the neurons in this layer (two in this case)

- A two-layer Neural Network is also called *Multioutput Perceptron*
- Each output of this network is still the result of a *linear transformation of the input*

$$y_i = g\left(b_{0,i} + \sum_{j=1}^{n} x_j w_{j,i}\right)$$

Single Hidden-Layer Neural Network (MLP)



*Bias not shown

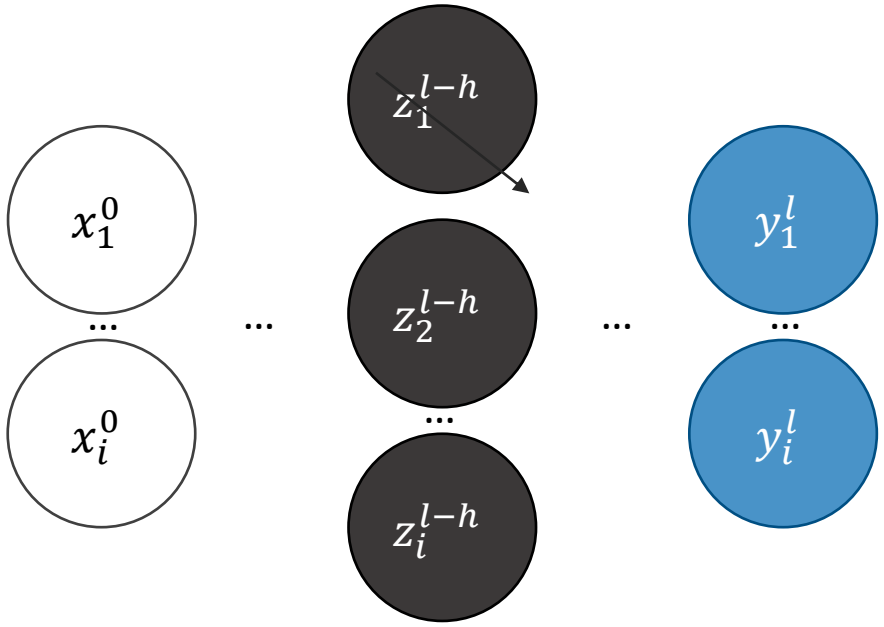
We need to stack neurons with nonlinear activation functions to achieve nonlinearity

1. *Hidden Layer:* Each layer between output and input layer is called hidden layer

- ➢ Each neuron in each layer gets the output from every other neuron in the previous layer → the network is called ***fully-connected***
- ➢ A fully-connected neural network with at least one hidden layer is called ***Multilayer Perceptron***
- ❖ Without nonlinear activation functions the neural network would still represent a linear transformation (see last Slide)

$$z_i = g^1\left(b^1_{0,i} + \sum_{j=1}^{n^0} x_j w^1_{j,i}\right) \qquad y_i = g^2\left(b^2_{0,i} + \sum_{j=1}^{n^1} z_j w^2_{j,i}\right)$$

The **Multilayer Perceptron** (MLP)



*Bias not shown

$$y_i^l = g^l\left(b_{0,i}^l + \sum_{j=1}^{n^l} x_j^l w_{j,i}^l\right)$$

*x^l is x^0 for the first hidden layer and z^{l-1} for all other layers

Multilayer Perceptron: Math

Output off one output neuron:

$$y_i^l = g^l\left(b_{0,i}^l + \sum_{j=1}^{n^l} w_{j,i}^l x_j^l\right)$$

Output of one layer in matrix form:

$$\boldsymbol{y}^l = g^l\left(\boldsymbol{b}_0^l + W^l \boldsymbol{x}^l\right)$$

- $\boldsymbol{y}^{l-1}$: input vector of layer l
- W^l: weight matrix of layer l
- $\boldsymbol{y}^l$: output vector of layer l

with $\boldsymbol{x}^l = \boldsymbol{y}^{l-1}$:

$$\boldsymbol{y}^l = g^l\left(\boldsymbol{b}_0^l + W^l \boldsymbol{y}^{l-1}\right)$$

with bias in W:

$$\boldsymbol{y}^l = g^l\left(W^l \boldsymbol{y}^{l-1}\right)$$

Each row contains the weights of one neuron!

$$W^l \boldsymbol{y}^{l-1} = \begin{bmatrix} w_{0,0}^l & \cdots & w_{n,0}^l \\ \vdots & \ddots & \vdots \\ w_{0,i}^l & \cdots & w_{n,i}^l \end{bmatrix} \begin{bmatrix} y_0^{l-1} \\ \dots \\ y_n^{l-1} \end{bmatrix} = \begin{bmatrix} w_{0,0}^l y_0^{l-1} + & \cdots & + w_{n,0}^l y_n^{l-1} \\ \vdots & \ddots & \vdots \\ w_{0,i}^l y_0^{l-1} + & \cdots & + w_{n,i}^l y_n^{l-1} \end{bmatrix} = \begin{bmatrix} y_0^l \\ \dots \\ y_i^l \end{bmatrix}$$

unrolled, $\boldsymbol{x}^1$ *is the input to the first layer:*

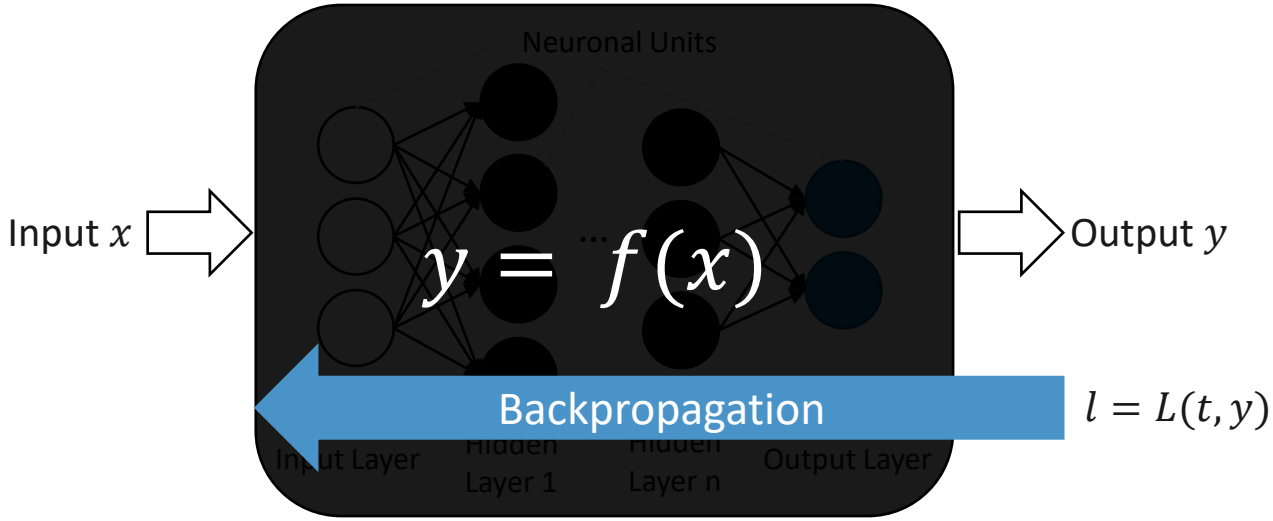
$$\boldsymbol{y}^l = g^l\left(W^l g^{l-1}(\dots W^{l-n}\, g^{l-n} \dots g^1\,(W^1 \boldsymbol{x}^1))\right)$$

➢ **The MLP can be realized with simple matrix multiplications! This is perfect for a GPU.**

Multilayer Perceptron: Code

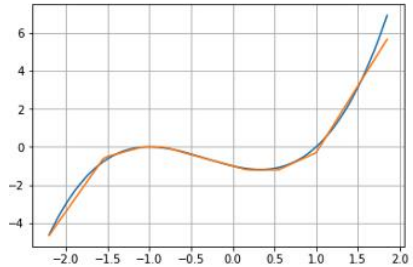
```
import torch.nn as nn

class MLP(nn.Module):
    def __init__(self, input_size, output_size):
        super(MLP, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(input_size, 128),
            nn.ReLU(),
            nn.Linear(128, output_size)
        )

    def forward(self, x):
        x = self.layers(x)
    return x
```

Neural Nets are learnable Functions - The MLP is a **Universal Approximator**



Universal Approximator

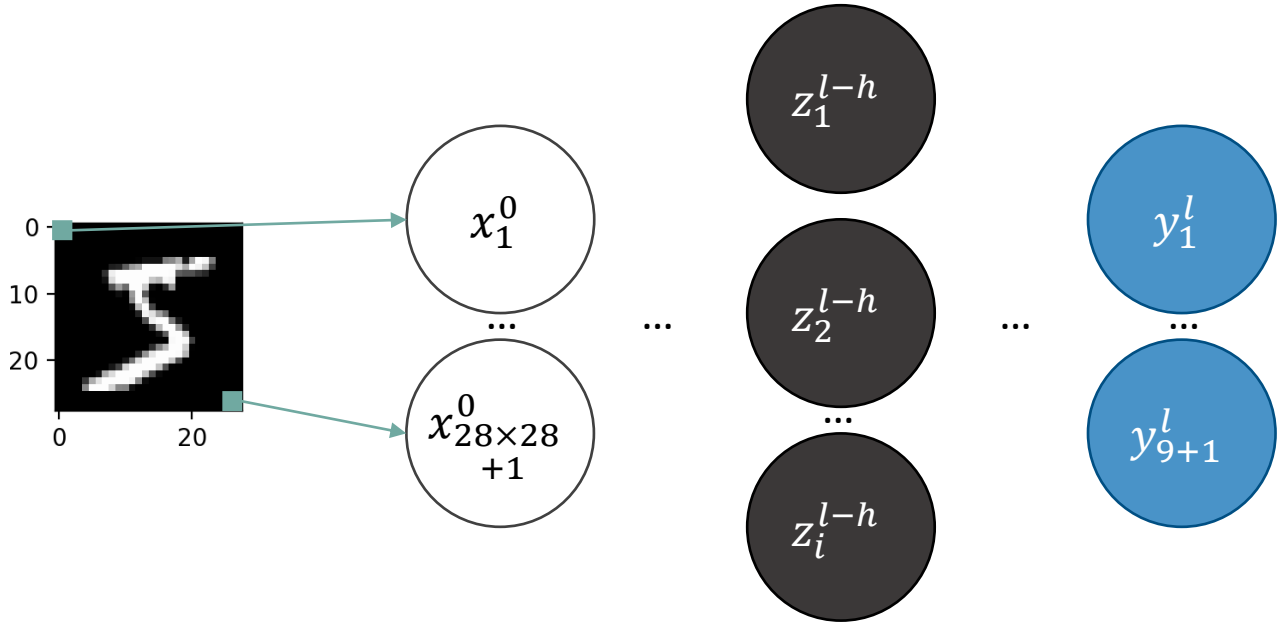
- Theoretically, there is a neural network that can approximate any (continuous) function [Cybenko1989] [Hornik1991]
- Theoretical foundation for why neural networks work




```
def feed_forward(input):
    Zh = np.dot(input, Wh)
    Ah = relu(Zh + Bh)
    Zo = np.dot(Ah, Wo)
    Ao = Zo + Bo
    return Ao
```

Approximating a Function with a Neural Network [Link]

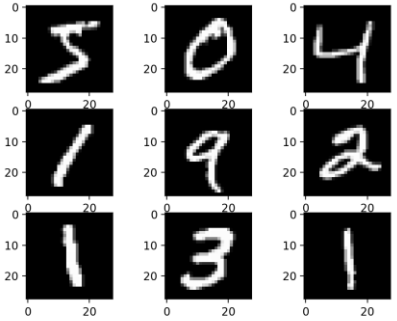
Multilayer Perceptron: MNIST Classification Example



Each output neuron represents one digit class, if neuron 1 has the highest output y_1^l the MLP classifies the input image as digit 0

Each x_i^0 represents one pixel

In the exercises of this and the following lecture we will build and train a MLP for MNIST!



Samples for the MNIST Dataset (Modified National Institute of Standards and Technology)

What happens if we remove the Activation Function?

unrolled, $\boldsymbol{x}^1$ is the input to the first layer:

$$\boldsymbol{y}^l = g^l\left(W^l g^{l-1}(\dots W^{l-n}\, g^{l-n} \dots g^1\,(W^1\boldsymbol{x}^1))\right)$$

$removing\ the\ activation\ function:$

$$\boldsymbol{y}^l = W^l \dots W^{l-n} \dots W^1\boldsymbol{x}^1$$

$now\ we\ multiply\ the\ matrices\ (in\ this\ simplyfied\ exampel\ all\ layers\ have\ i\ neurons)$:

$$W^l W^{l-1} = \begin{bmatrix} w_{0,0}^l & \cdots & w_{n,0}^l \\ \vdots & \ddots & \vdots \\ w_{0,i}^l & \cdots & w_{n,i}^l \end{bmatrix} \begin{bmatrix} w_{0,0}^{l-1} & \cdots & w_{m,0}^{l-1} \\ \vdots & \ddots & \vdots \\ w_{0,i}^{l-1} & \cdots & w_{m,i}^{l-1} \end{bmatrix} = \begin{bmatrix} w_{0,0}^l w_{0,0}^{l-1} + \dots + w_{n,0}^l w_{0,i}^l & \cdots & w_{0,0}^l w_{m,0}^{l-1} + \dots + w_{0,i}^l\, w_{m,i}^{l-1} \\ \vdots & \ddots & \vdots \\ w_{0,i}^l w_{0,0}^{l-1} + \dots + w_{n,i}^l w_{0,i}^{l-1} & \cdots & w_{0,i}^l w_{m,0}^{l-1} + \dots + w_{n,i}^l\, w_{m,i}^{l-1} \end{bmatrix} = \begin{bmatrix} w_{0,0}^{comb} & \cdots & w_{m,0}^{comb} \\ \vdots & \ddots & \vdots \\ w_{0,i}^{comb} & \cdots & w_{m,i}^{comb} \end{bmatrix}$$

$when\ we\ multiply\ all\ weights\ W^l\ with\ each\ other, we\ simply\ get\ another\ weight\ matrix!$

$$\boldsymbol{y}^l = W^{comb}\boldsymbol{x}^1$$

This is just a multiplication!
Therefore, the MLP has collapsed into a simple linear function!
Instead of learning all weights W^l we could simply learn the weights W^{comb}

Further Reading

- Deep Learning Book: https://www.deeplearningbook.org/
- The newest Papers (there are way too many)
- YouTube Lectures of other Unis, e.g.:
  - MIT: https://www.youtube.com/watch?v=5tvmMX8r_OM&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI
  - Stanford: https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk
  - TUM: https://www.youtube.com/watch?v=QLOocPbztuc&list=PLQ8Y4kIIbzy_OaXv86lfbQwPHSomk2o2e
  - TUM (advanced): https://www.youtube.com/watch?v=Bt5O1HjT9cI&list=PLog3nOPCjKBkngkkF552-Hiwa5t_ZeDnh
  - Tübingen: https://www.youtube.com/watch?v=OCHbm88xUGU&list=PL05umP7R6ij3NTWIdtMbfvX7Z-4WEXRqD
  - ….

www.hs-kempten.de/ifm