Do it, like we do

13: NEURAL NETS & DEEP LEARNING

Moritz Nipshagen mnipshagen@uos.de
Antonia Hain ahain@uos.de

Structure

- Week 1: Introduction
- Week 2: Syntax, Variables & Functions
- Week 3: Control Structures
- Week 4: Lists & Collections
- Week 5: RegEx & Strings
- Week 6: Sorting & I/O
- Week 7:Debugging, Errors & Strategies
- Week 8: 4P: Packages, Practices and Patterns
- Week 9: Object Oriented Programming
- Week 10: Time, Space & Documentation
- Week 11: Numpy & Matplotlib
- Week 12: Python & Science
- **Week 13: Neural Nets & Deep Learning**
- Week 14: Honorable Mentions & Wrap Up

Last Week's Homework

Last Week's Homework

- Documentation was a bit lacklustre
- Do not call get_x_y() twice. Many times we saw
  x = get_x_y()[0]
  y = get_x_y()[1]
- This calls the function twice! Meaning it is fully calculated twice and each time half of the results are discarded
- Saving the data and writing to a file, seemed hard
  - Maybe have a look again at the solutions to get an idea of how it can work
- A mistake on our sides: the circle was meant to disappear after pressing a key

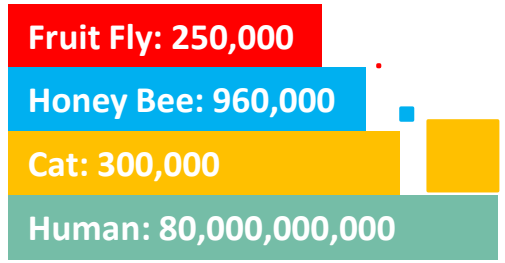
Why Neural Networks

Why Neural Networks?

- Biological neural networks are pretty amazing, general problem solvers
  - We take ~0.1 seconds for scene recognition, are able to operate our body parts at high precision, can process (different) language(s) in ~0.6 seconds, learn and adapt to our environment with analogous reasoning to solve novel tasks, memorise a lot, …….
    - We are pretty bad at symbolic computing though
- This is pretty much opposite of what computers are
  - Not entirely true, but computers are mostly good at symbolic computing and not that great at analogous reasoning and plasticity
- The idea is to create software that emulates biological properties
  - **Artificial Neural Networks**

The catch

- The brain is a fairly complex organ
- On average a neuron has several thousand connections
- This means an immense level of parallel processing
  - E.g. scene recognition takes around 100 steps to do, but is highly parallelised[4]



[1]

A comparison

- A fruit fly has 250,000 neurons
- It can
  - Navigate flying and walking
    - That already involves scene perception and motor control
  - Recognise and find food
  - Find mates & reproduce
  - Learn and retain information

www.youtube.com/watch?v=TxobtWAFh8o

[1]

The Transition

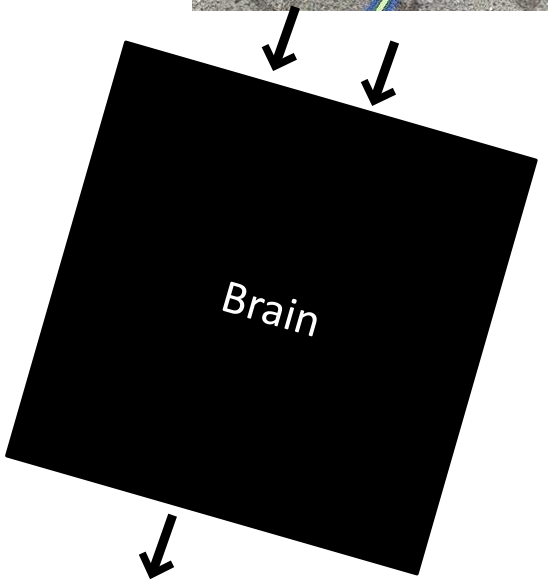
Biological to Artificial: The Takeaway

- The rate in which a neuron fires encodes its activity
- Inputs from other neurons influence the activity of the neuron
- Inputs can be weighted
- The activation of a neuron is non-linear
  - Meaning below a certain threshold there is nearly no activity
  - Once the threshold is reached, the activity increases (a lot)
  - But there is not much more, the rate of fire quickly saturates
- All neurons essentially share the same cell structure

Biological to Artificial: Modelling

- A neural network gets a certain set of inputs
  - E.g. light entering our eye, resulting in activity through the optic nerve
- Those inputs then are processed
  - In some way
  - It is a black box. We do not know *exactly* what is happening inside
- Output is produced
  - E.g. the knowledge that those patterns of light waves are reflected by a cute dog.
- The brain thus is some function that maps inputs to outputs

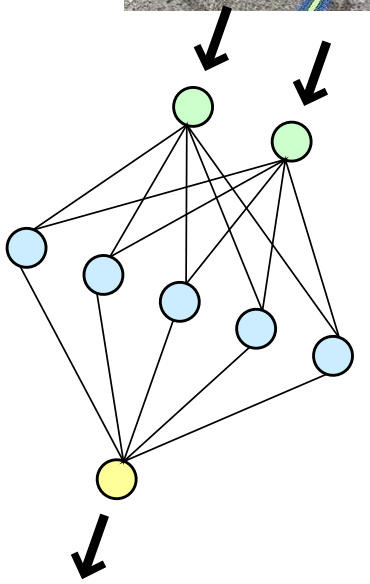
f() = “A cute dog!”



“A cute dog!”

Biological to Artificial: Modelling

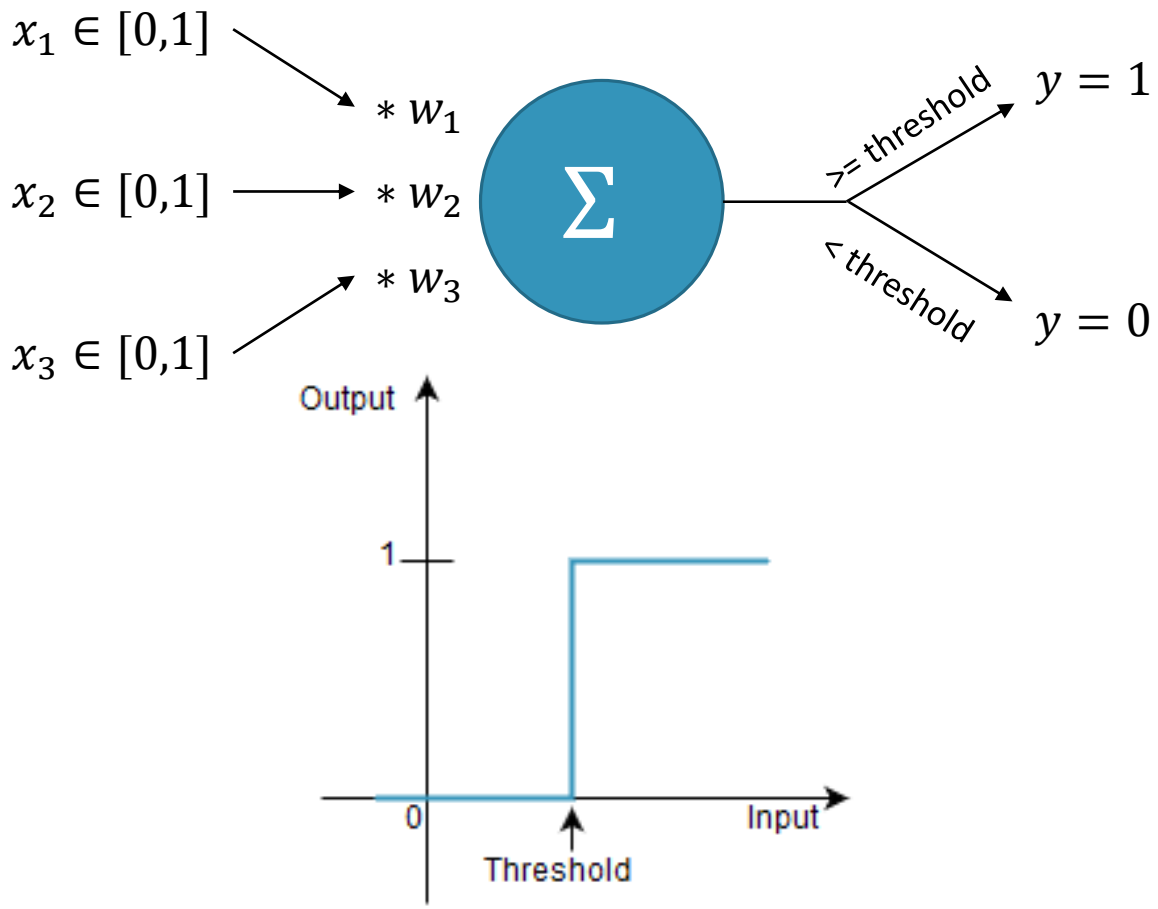
- **Artificial Neural Networks** abstract the biological black box to an artificial black box
- There is one kind of neuron. They all work the same.
- It is not as interconnected
- It is not as dynamic as the biological model
  - Look up reservoir computing for a more biological approach
- The data is processed sequentially, instead of highly parallelised



“A cute dog!”

Biological to Artificial: The Perceptron

- One of the fundamental building blocks
- Its purpose is to recreate biological neurons
- Most basic neurons in ANNs work in a similar fashion
- It gets inputs that are either 0 or 1
- The inputs are then weighted and summed up
- If they exceed a certain threshold the neuron “fires” and outputs a 1, otherwise a 0
  - This is called a step function



Artificial Neural Networks

A Good Start

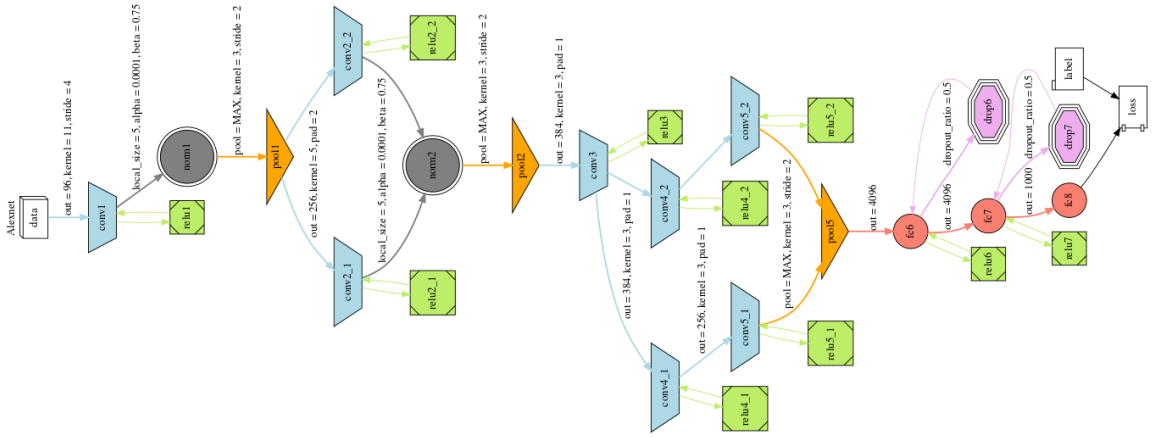
“At some fundamental level, no one understands machine learning.”
- Christopher Olah, 2014[3]

What are Artificial Neural Networks?

- Artificial Neural Networks (ANNs) are a *machine learning* approach
- Like with most machine learning algorithms the idea is simple:
  ANNs learn from the data presented to them and figure out a solution
- ANNs have come up as early as the 50's
- But since 2006 the field exploded
  - This is mostly due to a collection of techniques which are now known as *deep learning*
  - Having far more computing power and large amounts of data sets certainly helped, and probably are the two main reasons for the spike in usage since 2012

What is Deep Learning?

- It is a family of algorithms used to extract, model and predict structure in data
- The **deep** comes from the structure of those models
  - Deep learning models form from simple structures more and more complex structures
  - It is thus a **layered** approach. A deep learning model consists of many layers
- *Neural Networks* are probably the most prominent example of deep learning



AlexNet, Wikimedia commons

What is Deep Learning?

- So far we have written problem specific algorithms
  - Small precise tasks that together solve a bigger problem
  - They were tailor made to fit one specific problem and solve that
- This form of problem solving works well if we can formulate the way to the goal in a structured, unambiguous manner
  - Surprise: We cannot always do this
- In Deep Learning we structure an ANN – the **architecture**
  - This is basically the outline for the problem solving strategy the computer should use
- The computer then learns from the data and figures out the solution (or a good approximation) to the given problem

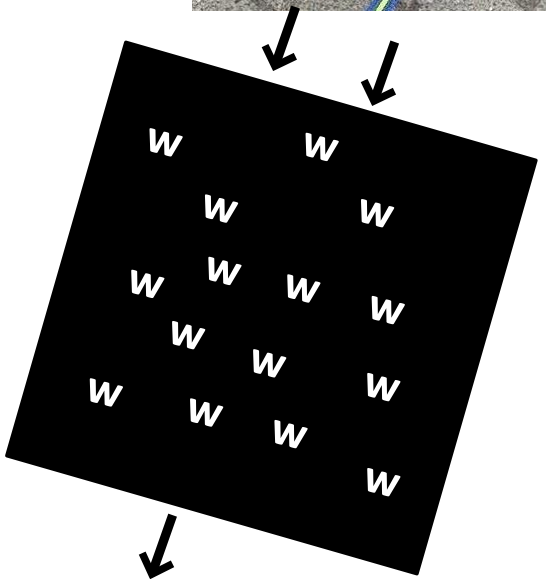
Learning & Behind the Scene

Blackbox Stays Blackbox

- Finding a solution hence is the search for the right weights
- With the right weights we will find a function that maps the input to the correct output

$$y = f(x, w)$$

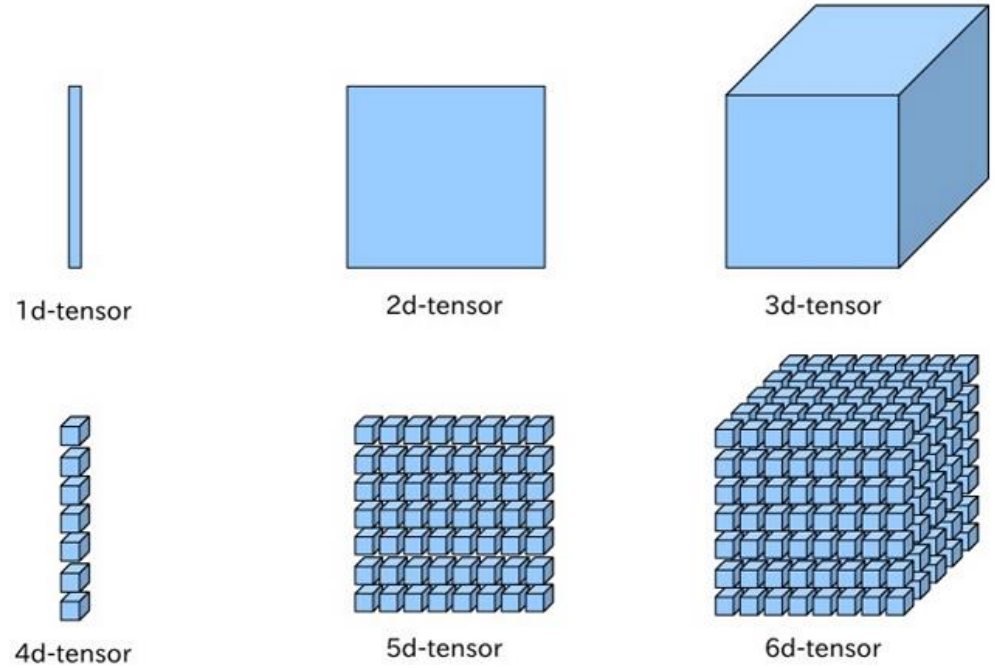
- **ANNs will still be black boxes**
- We do not understand how we get from input to output
- Our ANNs approximate the underlying function
  - But in the end we will still not know the function



“A cute dog!”

Tensors

- **Tensors** are the generalisation of vectors and matrices (an nd-array)
- “0”d would be a scalar
  - a single number
- 1d tensors would be vectors
  - A list/group of numbers
- 2d tensors would be matrices
  - So many vectors grouped together
- Everything above is just a tensor
  - 3d is still fairly imaginable: a group of matrices



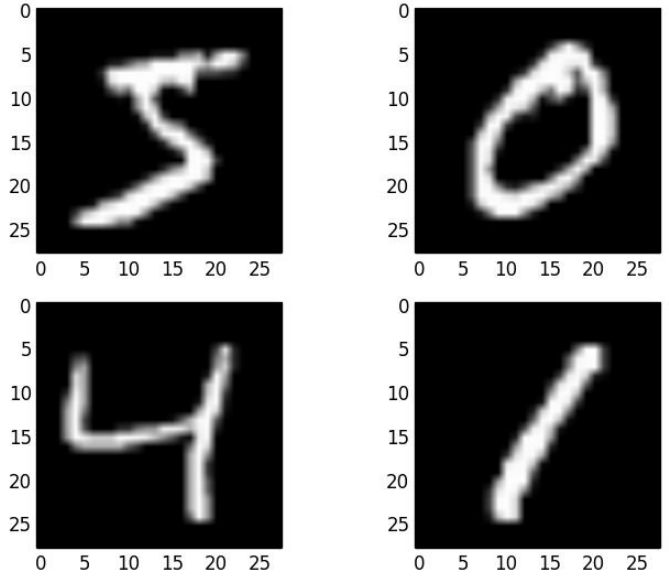
https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/linear_algebra.html

The Input

- We can feed a lot of things into our network, as long as they are numbers
  - We can convert a lot of things into numbers though
  - As said last week: pictures, audio and video are already just numbers anyway
  - Text is a bit more complicated and there are several solutions (e.g. word2vec)
- So the input is a *tensor* with one or many dimensions (often it is 1d, a vector)
  - A picture, for example, can be converted into a vector by “flattening” it into one row
  - The input vector than has a dimensionality of pixel-width * pixel-height * colour depth
    - Colour depth is how many values are used to represent the colour of the pixel
      - This is often 1 (grayscale), 3 (rgb, hsv, etc.) or 4 (rgba)

The Usual Start: MNIST

- MNIST is a data set of handwritten numbers and often used in introductions to programming neural networks
- Each number is 28 x 28 pixels and grayscale
- Converting this into a single vector yields $28 * 28 * 1 = 784$ as dimensionality of the input vector
  - Each dimension represents one pixel of the input



Weights and Biases

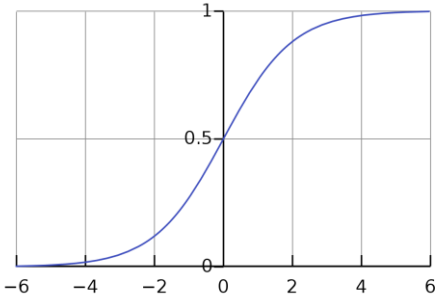
- **Weights** are also a tensor which transform the input to the layer via tensor multiplication
  - Which is (kind of) a generalisation of matrix multiplication
  - The input gets multiplied with the weights resulting in a new tensor, which can either be fed into the next layer or be the output
- **Biases** are an optional part to the layer which are additive to the *output* of the tensor multiplication

$$y = xW + b$$

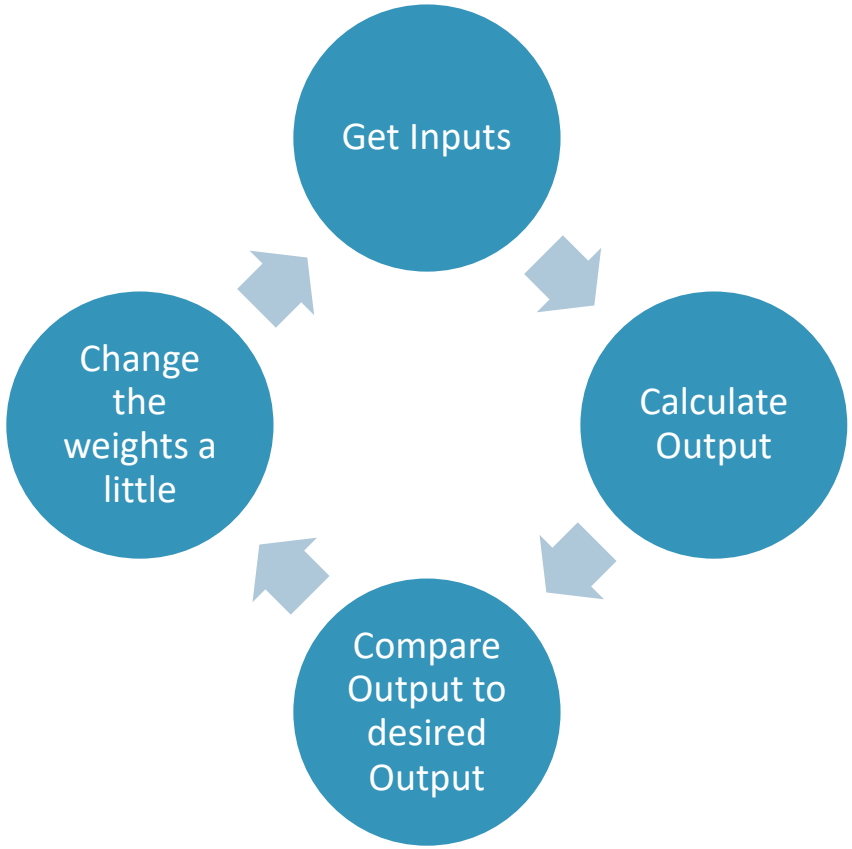
Where y is the output, x the input, w the weight and b the bias tensor

The Activation

- The last part of the layer is the **activation function σ**
- The output of the neuron after applying weights and biases is called **drive**
- Before the drive is propagated further, an activation function can be applied
- There are quite a few activation functions
  - Not altering the drive, would be to apply a linear activation (called identity) $\sigma(y) = y$
  - For the perceptron we used a step function, mapping the drive onto $[0,1] \in \mathbb{N}_0$
  - The *sigmoidian function* (or logistic) is more similar to a bio-physical neuron
  - There are many, many more
    - https://en.wikipedia.org/wiki/Activation_function



Learning



- The basic process for a network to learn
  - You feed the inputs in
  - The network produces some output
  - That output is (not manually) compared to the desired output
    - The desired output is a label attached to the input we fed into the network
  - Use that result to change the weights a little to better predict inputs in the future
    - This is called **backpropagation**
- The big caveat: We need to have labelled samples for our network to learn

Learning

- There are three major types of learning
- **Supervised Learning**
  - Example from before. Data is labelled by a “teacher”.
  - Task is to map the input to the given label as an output
- **Unsupervised Learning**
  - There is no “teacher”. Data is unlabelled
  - No straightforward way to measure accuracy. The effect of learning is coded in the rules.
- **Reinforcement Learning**
  - Learning by doing
  - A “weak teacher” is present telling when the goal or fail state was reached
  - The network needs to find the way by itself

Learning

- Another learning method that gains more popularity
- **Semi-supervised learning**
  - Start with unsupervised learning and generate a form of pre-structure
  - Use some labelled data sparingly after unsupervised learning to improve result
- It stands somewhere between supervised and unsupervised learning

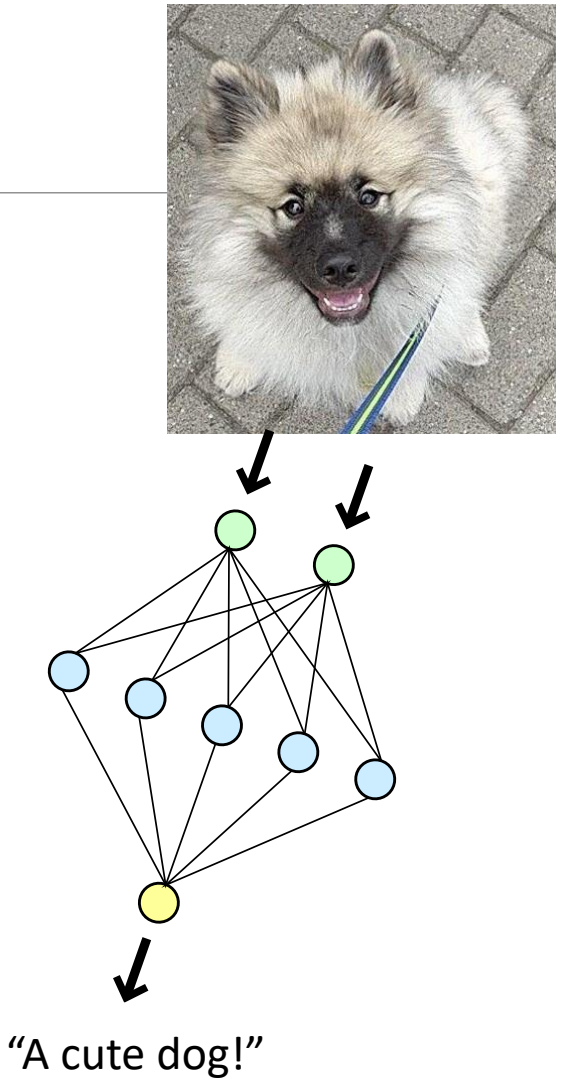
Training

- Letting a network learn is called training
- In this process the data is fed in, the weights are adjusted and data fed in again
- This is repeated for a while
- Often we do not have enough data to train the network with unique samples, and we need to show the same data twice
- Iterating through the whole data set once, is called an *epoch*
- Training can take a long, long while
- Training is also a fairly complex process with several strategies and subtasks

(A Part of) The ANN Family

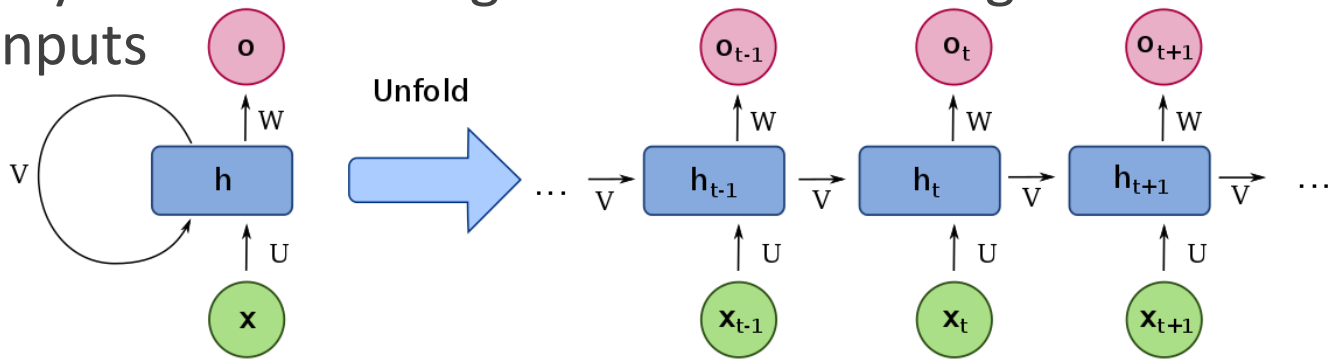
Feed Forward Networks

- Feed Forward Networks (FFN) the first and simplest type of ANNs
  - The perceptron is a FFN
- The input moves through each layer directly through the output layer without cycles and loops



Recurrent Neural Networks

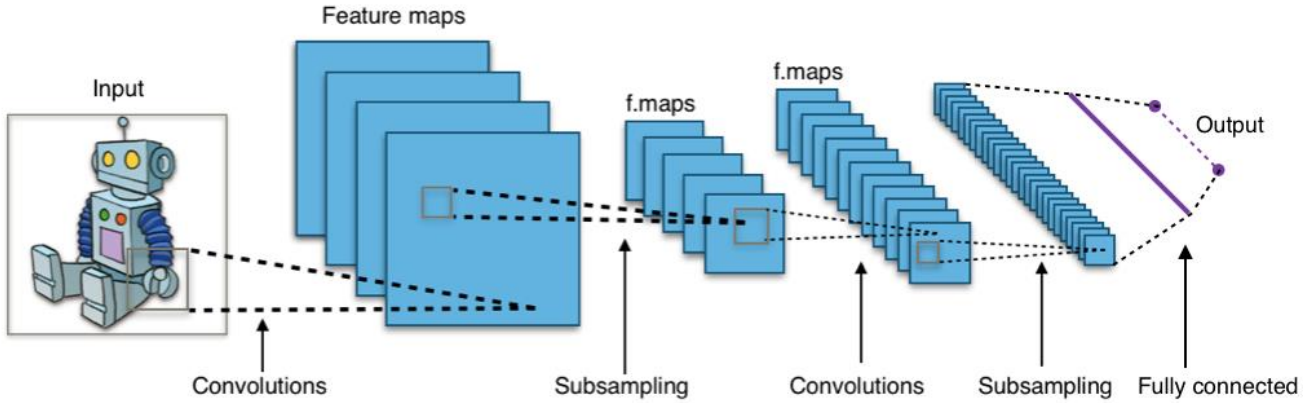
- In contrast to FFNs, data is not just propagated forward in Recurrent Neural Networks (RNNs)
- Data is also moving backwards from later to earlier layers
- In FFNs a sequence of inputs is handled each entry on its own, with no regard for earlier and following inputs
- RNNs can deal with sequential data, by interconnecting neurons and saving some weighted data from previous inputs



François Deloche, Wikimedia commons

Convolutional Neural Networks

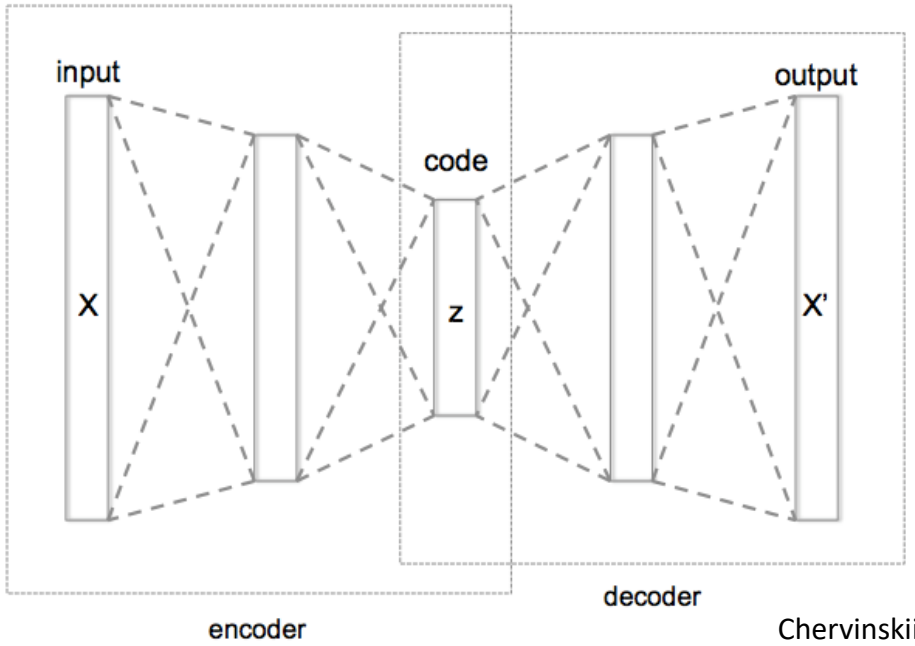
- A convolutional neural network (CNN) is inspired by the organisation of the visual cortex
- Each neuron responds to certain stimuli in a certain area of the input
- CNNs have also been used in e.g. audio recognition



Aphex34, Wikimedia Commons

Autoencoder

- Autoencoders are used to create efficient data codings
  - So to represent the same data, but with less memory
- They are (usually) used with unsupervised learning



Chervinskii, Wikimedia commons

Frameworks

Frameworks


Caffe
Chainer
theano
PYTORCH
Caffe2
mxnet
Microsoft
Cognitive
Toolkit
Microsoft
TensorFlow

Frameworks

Framework	Written In	Interfaces	Initial Release	Maintainer
Caffe	C++	Command Line, Python, C++	March 2014	Berkley Vision and Learning Center
Caffe2	C++	C++, Python	April 2017	Facebook
CNTK (Microsoft Cognitive Toolkit)	C++	C++, Python, Command Line	January 2016	Microsoft Research
Matlab	C, C++, Java, Matlab	Matlab	?	Mathworks
Chainer	Python, C	Python	January 2015	Chainer.org, Community
Torch	C, Lua	Lua	October 2002	Community Based
Pytorch (beta)	Python	Python	January 2018 (soon)	Facebook
MxNet	C++	C++, Python, Julia, Matlabl JS, Go, R, Scala, Perl	December 2015	Community Based (Apache Incubator)
Theano	Python	Python	August 2011	Université de Montréal
TensorFlow	C++	Python, C++	November 2015	Google

Frameworks

- Keras is a high level, object oriented API for fast experimentation
- It works on top of different other frameworks
- That makes it nice to switch and test different frameworks
- It is also fairly easy to get started with simple models
- Difficult to expand however

Python Frameworks

- There are a lot of frameworks with Python interfaces
- The most recognisable are probably TensorFlow & Pytorch
  - Pytorch is still in beta but has one significant advantage
    - It allows for dynamic data structures, whereas most other frameworks use static graphs
    - This is awesome especially for dynamic systems and prototyping
  - Runnerup: CNTK
- They are frequently updated and improved upon, popular, and grow fast
- Fairly low level, allowing to be expanded and built upon for custom models
- **In the end, most frameworks are extremely similar**
  - It is often fairly simple to switch between them

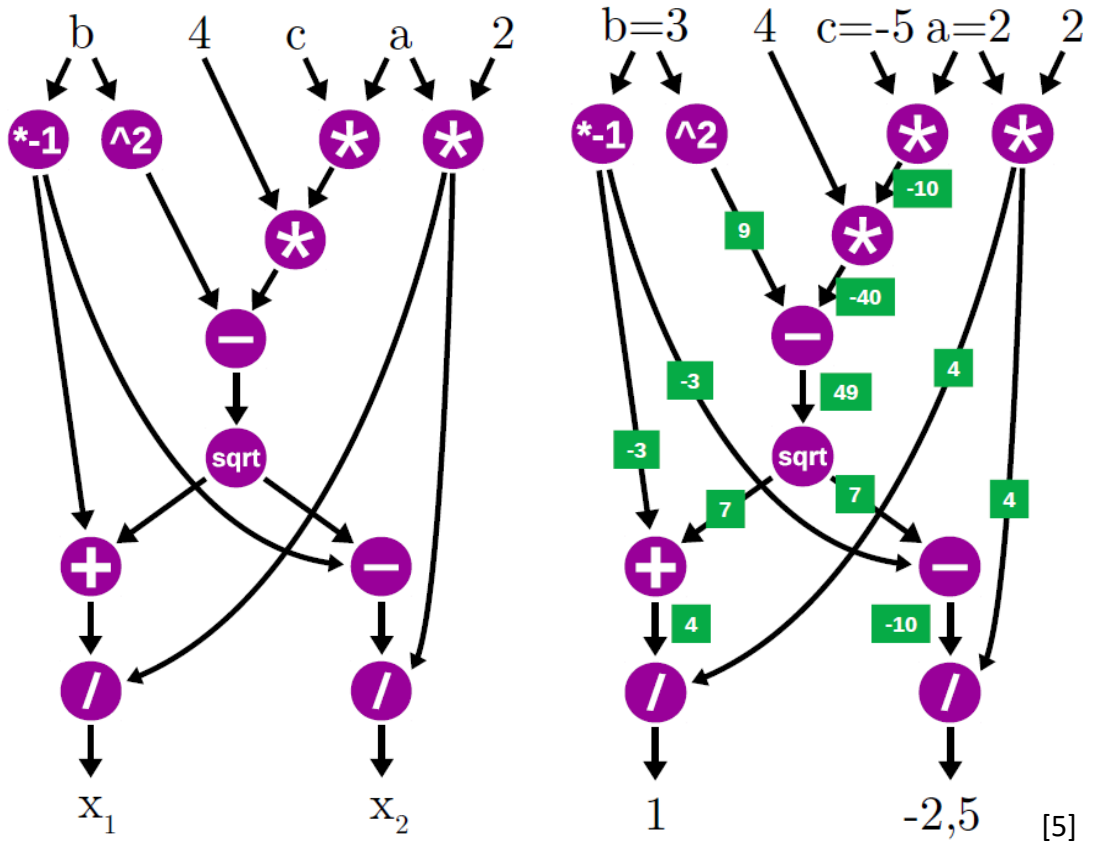
But how?

Data Flow Graph

- The **graph** is a representation of how **nodes** are connected
- Each node represents one **operation** (e.g. an activation function, an summation, …)
- The **edges** (connections between nodes) represents the flow of numbers
- The rights is the dataflow graph for

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- These graphs are great for parallelisation!



[5]

Homework

Floppy Ears, Fluffy Fur

- This week is a repetition of sorts
- You will need to implement a Tamagotchi-like program
- The player takes care of a little bunny that can be fed, played with or ignored
- If you play with it often, the bunny will grow to be a big happy bunny
- Take care though if you don’t!

See you all next week!

References

[1] Introduction to Artificial Neural Networks with TensorFlow, Lukas Braun, 2017

[2] http://neuralnetworksanddeeplearning.com, Michael Nielsen, 2017

[3] https://colah.github.io/, Christopher Olah, 2018

[4] Machine Learning, Gunther Heidemann, 2018