School of Management, Organizational Development and Technology / Universität Klagenfurt

DIHSÜD

ULG Daten- und KI-Management

Modul 2: Methodische GL

Dietmar Millinger, Juli 2022

Artificial Intelligence

1 Introduction

Notice

These slides are intended for teaching and studying the introduction to Artificial Intelligence and Machine Learning. The information contained is carefully compiled, however, it should be noted that due to the limited time only an overview is included and many detailed information can be found only in the sources accessed through the provided links. Since the area evolves very quickly, some information may already be out of date.
Images belong to the corresponding authors. Links to the source of the images can be found on the slides, however, some images do not contain correct referencing of the sources.

Please don't distribute those slides in the general public

*automatically translated with google translate

Dr. Dietmar Millinger

1994-1998: Assistent am Institut für Technische Informatik, TU Wien

1998: Doktorat am Institut für Technische Informatik der TU Wien

1999-2007: Gründer und CTO der DECOMSYS GmbH, Automobilindustrie

2008-2011: Consulting im Bereich Functional Safety in der Automobilindustrie

Seit 2011: Miteigentümer der TWINGZ Development GmbH

Schadensvermeidung und Energieeffizienz in Industrieanlagen und Privathaushalten

Seit 2016: GREX Professional Makers

Prototypen innovativer Ideen

Seit 2018: Gründungsmitglied von AI Austria

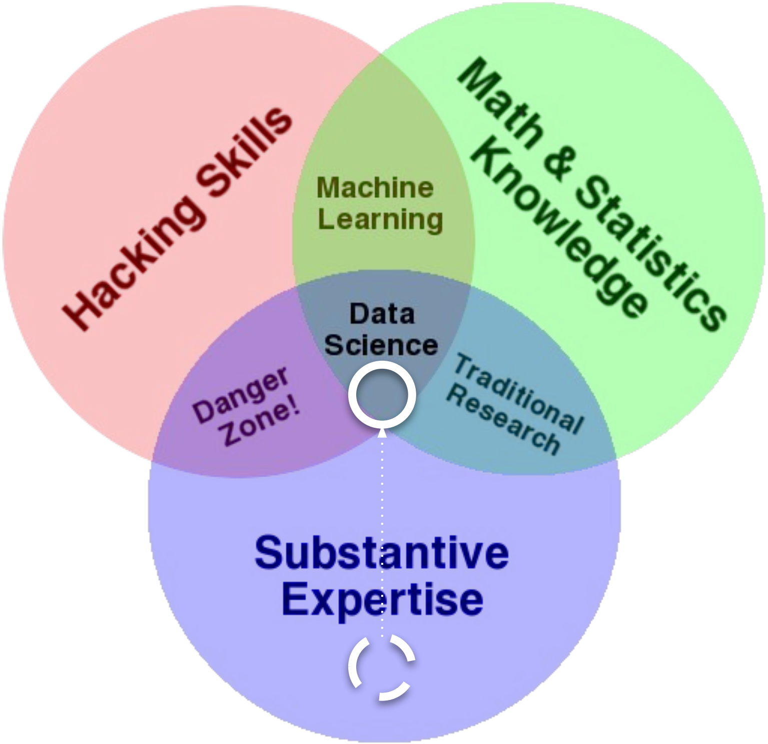
Interessen: AI, Windsurfen, Laufen, das Bewusstsein

dietmar@grex-app.com

Knowledge gained in this class

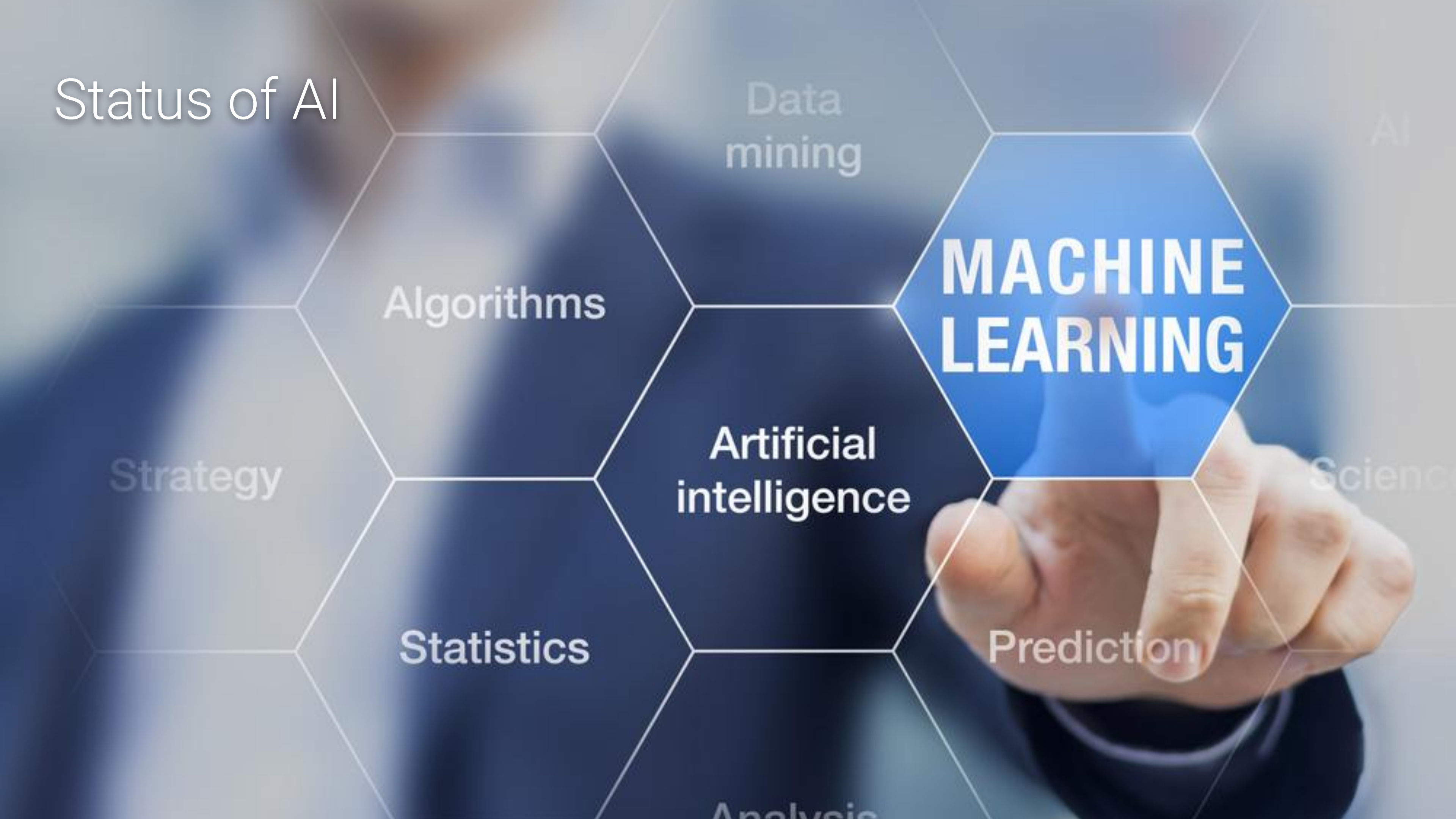
- AI and ML are **relatively new** fields in science and technology
- Think about where you would position yourself in the venn-diagram on the right side
- Basically this class attempts to strengthen your position near the white circle

The slides are in English language, since most literature and information about AI is in English.



source: http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

Status of AI


Data
mining
AI
MACHINE
LEARNING
Algorithms
Strategy
Artificial
intelligence
Statistics
Prediction

https://www.youtube.com/watch?v=mzZWPcgcRD0

Autonomous vehicles

Machine learning is driving the technologies behind self-driving vehicles.
The most advanced cases are autonomous trucks in the US.

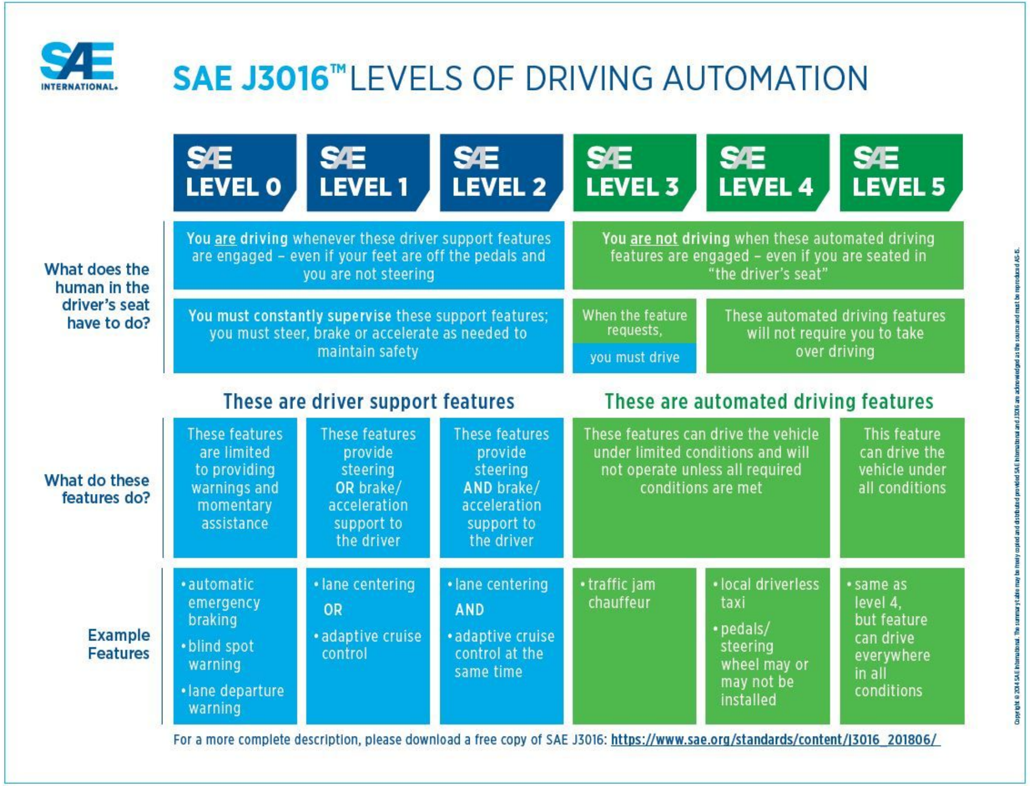
They are operated by humans in complex areas (e.g. cities) and drive independently in reduced complexity scenarios (e.g. motorway).

This is happening **today** (e.g. UPS).

Waymo has already 20 million miles of experience with self-driving taxis on public roads.

https://www.theverge.com/2019/8/15/20805994/ups-self-driving-trucks-autonomous-delivery-tusimple
https://www.youtube.com/watch?v=yjztvddhZmI

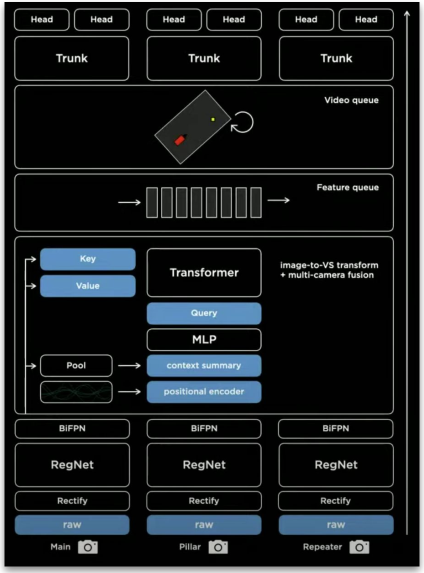
Levels of autonomous driving

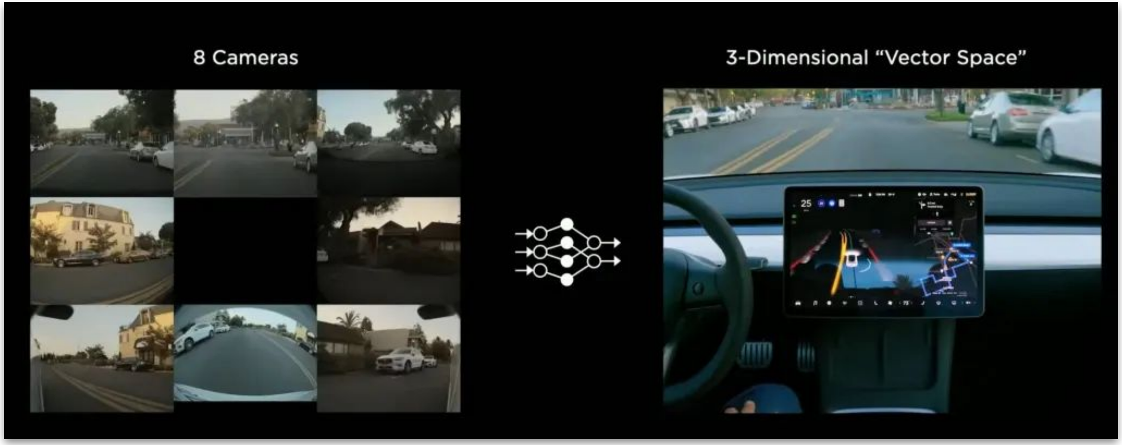


source: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic

Tesla Hydra Architecture

➢ Neural network architecture for full self-driving (FSD) function in Tesla cars
➢ Sensor fusion of video streams from eight cameras into one unified 3D vector space
➢ Based on transformer architecture





Boston Dynamics robot handle

Very fast progress in robotics in the last years due to **better control algorithms** for sensory data processing, control optimization and planning.

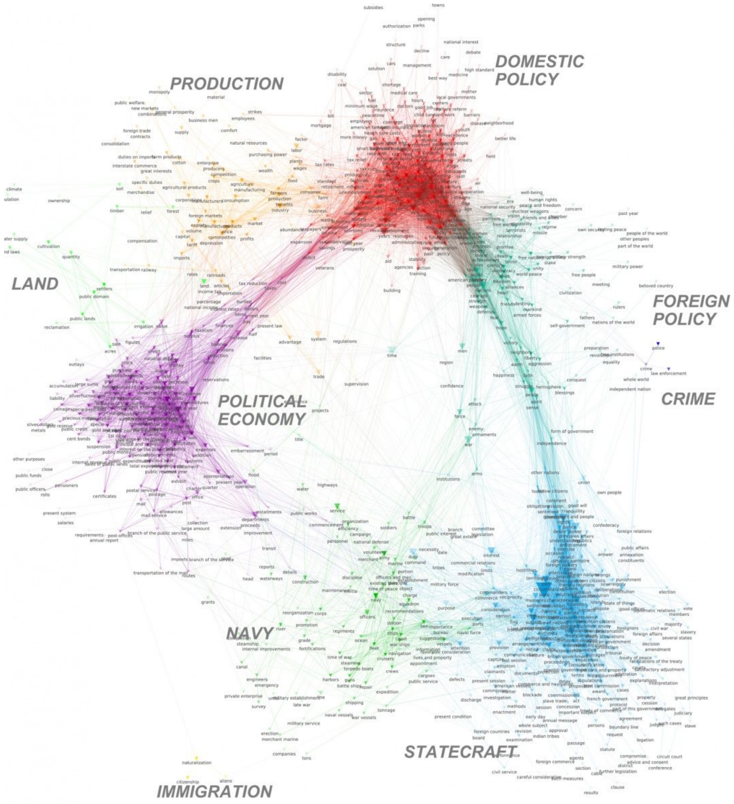
Reinforcement learning for robots still a challenge, since learning by experimenting does not work for robots. However, progress is fast.

https://arxiv.org/pdf/1707.07217.pdf

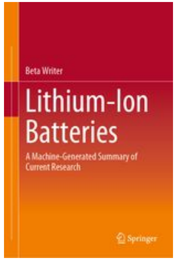
New understanding of natural language

Today, AIs can learn to understand text content. This is based on a statistical learning process that uses a **high-dimensional vector space**. Every word gets a position in this space. In this space arithmetic operations can be performed.

- ➢ comparison of documents
- ➢ knowledge extraction
- ➢ document summary
- ➢ translation of texts



In 2018 Springer Nature published the first book created with machine learning methods. It compiles abstracts of many thousands of publications in the field of Lithium-Ion batteries.

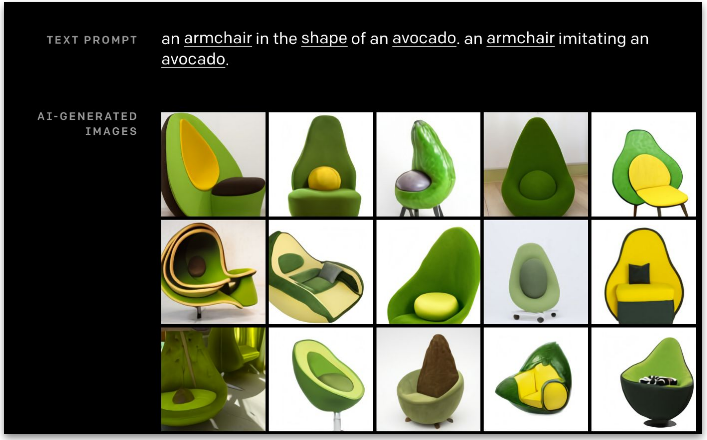


GPT-3

Autoregressive **language model** trained on 6 million text articles

GPT-3's full version has a capacity of **175 billion** machine learning **parameters.** It shows signs of a **deeper language understanding**

DALL-E, a variant of GPT-3 was trained on a combination of **text and images**. It delivers impressive results for text to image tasks.

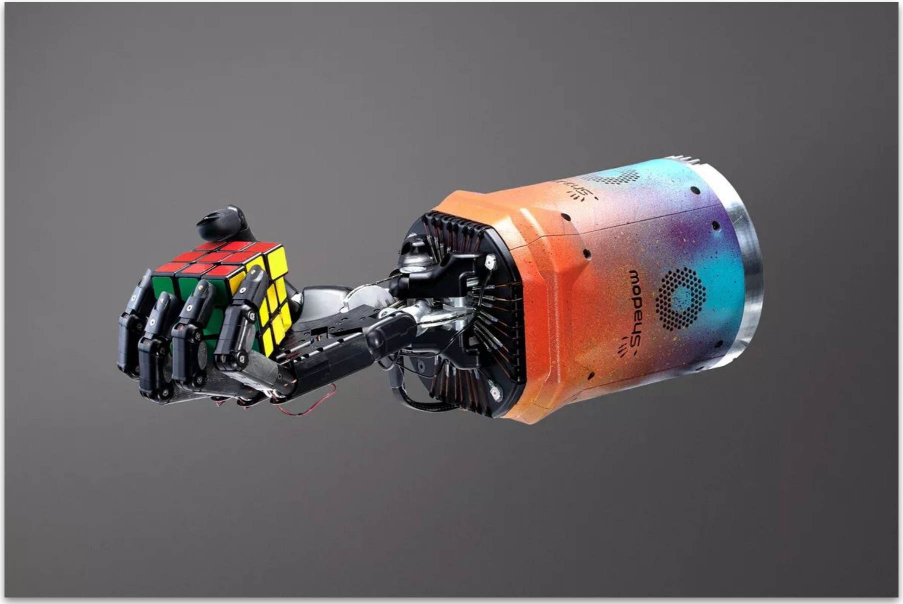


https://www.theverge.com/21346343/gpt-3-explainer-openai-examples-errors-agi-potential
https://towardsdatascience.com/have-you-seen-this-ai-avocado-chair-b8ee36b8aea
https://arxiv.org/pdf/2102.12092.pdf

Reinforcement learning

Learning by **experimenting** and feedback

- ➢ instead of training data we need a **reward function** that delivers a signal for success and failure
- ➢ e.g. robot learns to solve Rubic's cube. Experiments can only be executed in simulation, not in physical world.
- ➢ e.g. agent learns to control the flow of product pieces in a smart factory
- ➢ requires a precise **simulation** of the physical environment



https://www.marutitech.com/businesses-reinforcement-learning/
https://towardsdatascience.com/7-real-world-applications-of-reinforcement-learning-f80955cefcd5
https://www.youtube.com/watch?v=x4O8pojMF0w

What are the drivers of the current AI hype?

Moore's Law

- ➢ CPU power still grows at exponential rate

Data

- ➢ available data double every two years

Algorithms

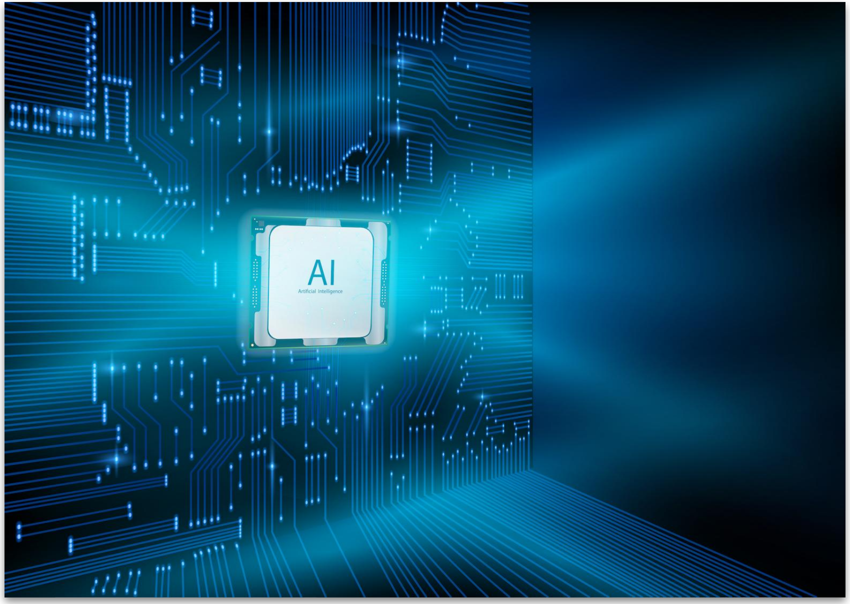
- ➢ major improvements in the last decade

Funding

- ➢ highest funding rate for AI in history

Open Source

- ➢ latest technology available open source



https://www.youtube.com/watch?v=gLeuCj0ZFo4

What are hindering factors?

Lack of **skilled experts**

Lack of **knowledge** about technology

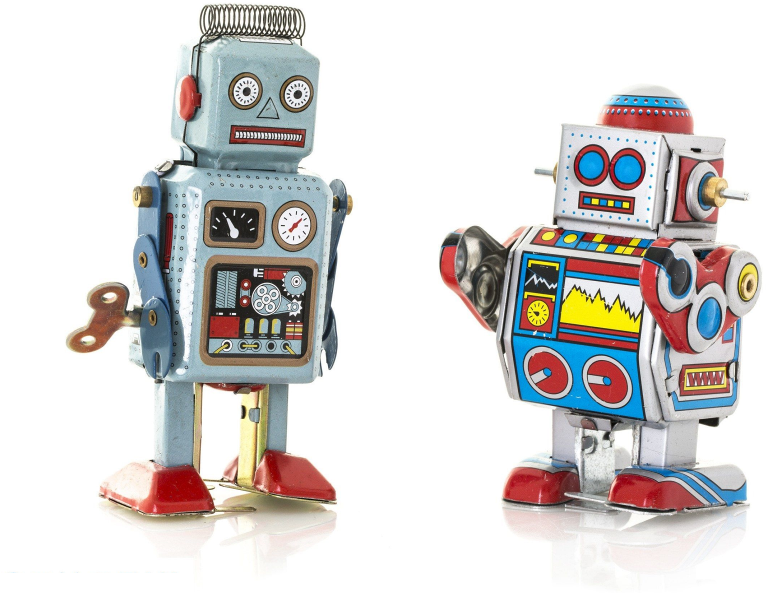
Missing **trust** in technology

High effort to create **data for training**

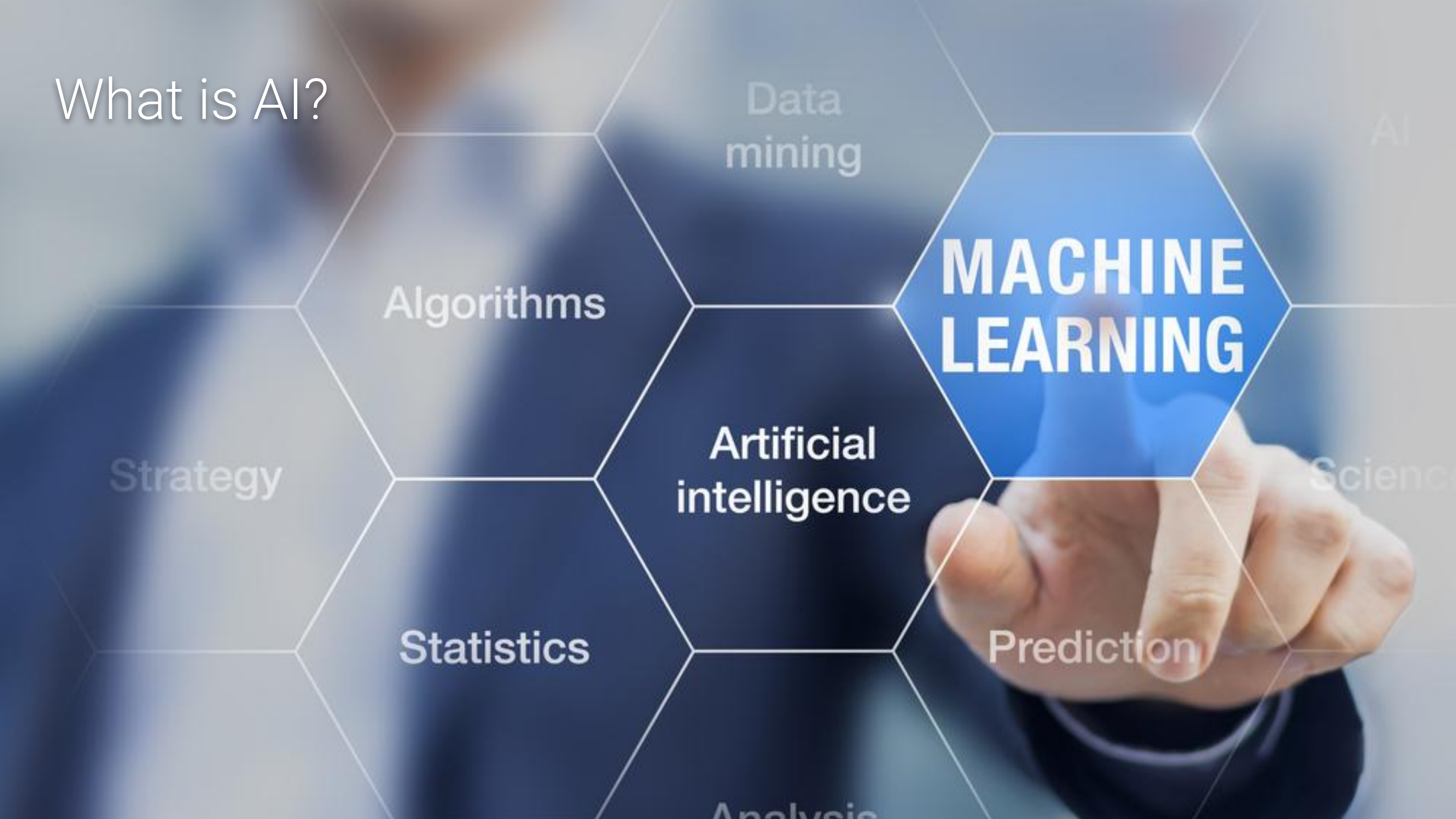
High **energy demand** for training and inference

Open **legal** questions

Open **ethical** questions



What is AI?


Data
mining
Algorithms
MACHINE
LEARNING
Strategy
Artificial
intelligence
Statistics
Prediction

Natural intelligence

From Wikipedia

Intelligence:

“...the ability to perceive or infer information, and to retain it as knowledge to be applied towards adaptive behaviors within an environment or context”

Cognition:

“...the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses...”

Cognitive abilities of most humans:

- ➢ perception
- ➢ attention
- ➢ memory
- ➢ learning
- ➢ problem solving
- ➢ creativity
- ➢ planning
- ➢ orientation
- ➢ imagination
- ➢ argumentation
- ➢ introspection
- ➢ will/intent
- ➢ model building

...

Artificial Intelligence

Wikipedia: *"In computer science AI research is defined as the study of "intelligent agents": any device that **perceives its environment and takes actions** that maximize its **chance of success** at some goal."*

UN Paper states: *"AI is the field of study devoted to developing computational technologies that **automate aspects of human activity** conventionally understood to require intelligence"*

My favorite: *"Looking at things and know what to do..."*

Levels of AI

AI-

weak AI systems solve isolated problems better than humans

AI+

strong AI systems solve complete tasks by combining many weak AI systems

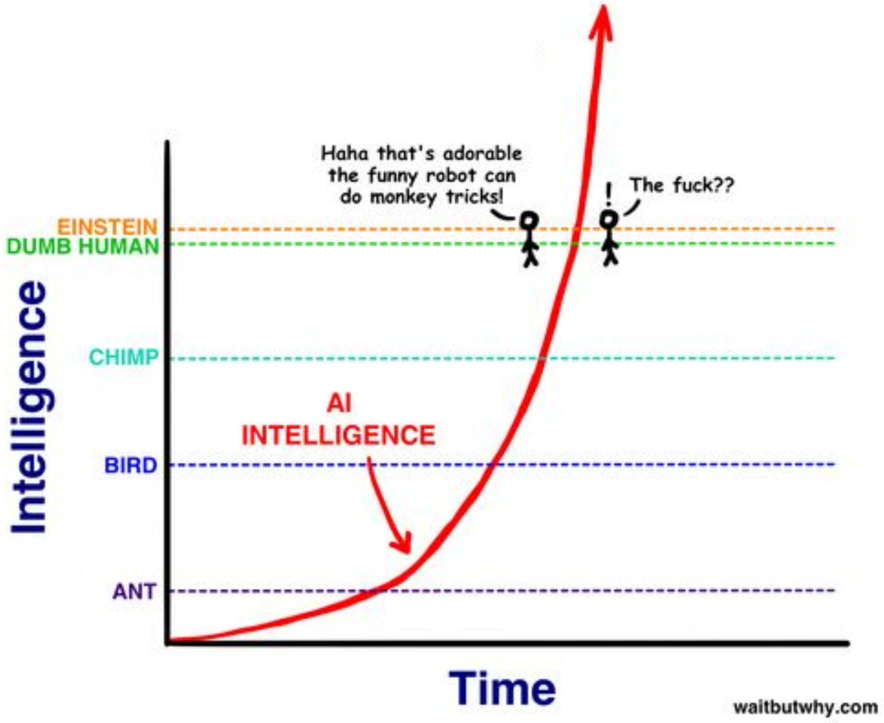
AGI (artificial general intelligence)

AGI systems beat humans in **all aspects** of cognitive abilities

ASI (artificial superintelligence)

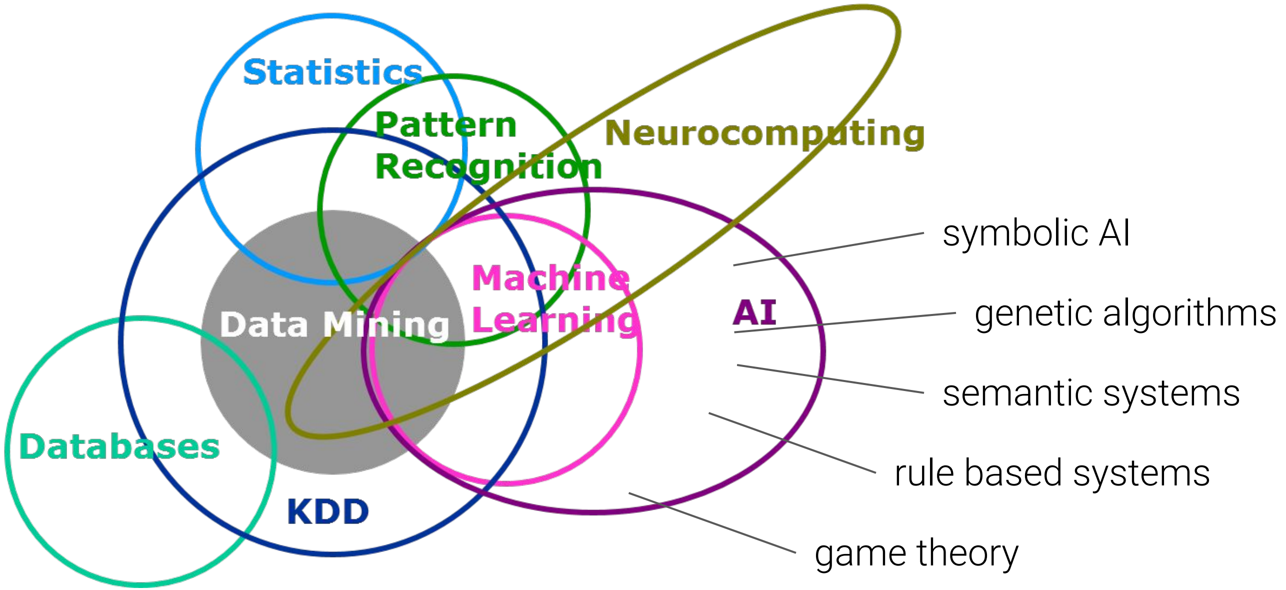
AI system with intelligence level that is incomprehensible for humans

DISCUSSION



https://waitbutwhy.com/2015/01/artificial-intelligence-revolution-2.html

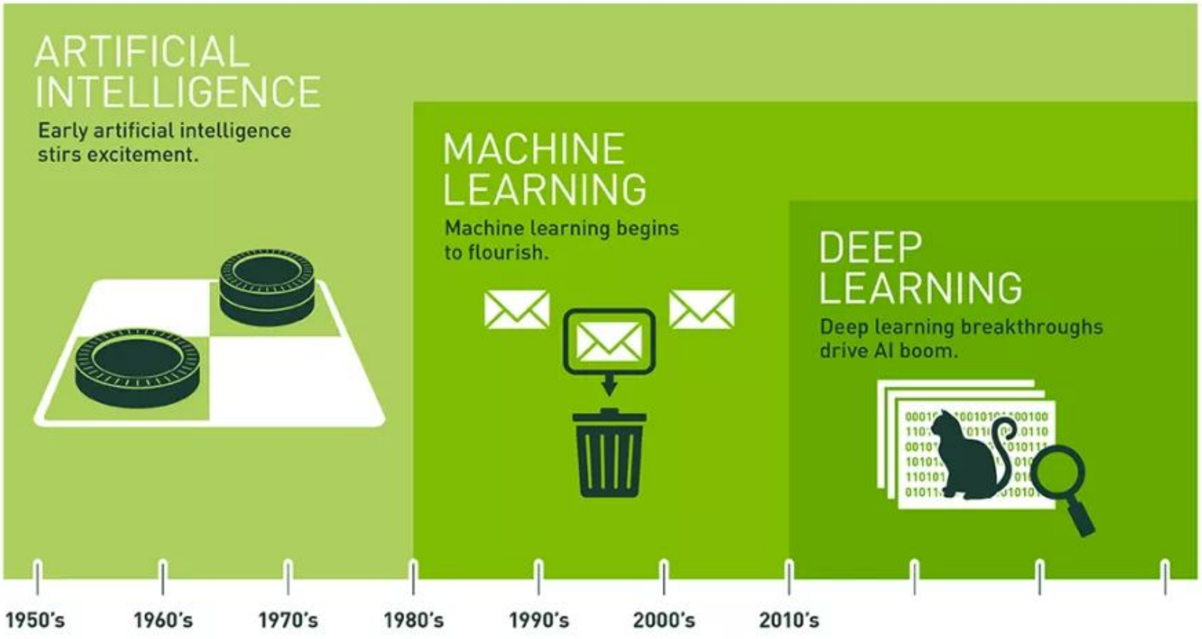
AI in the context of data science



KDD: Knowledge Discovery and Data Mining

https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/

Machine learning in the context of AI



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.

AI in the context of digitalization

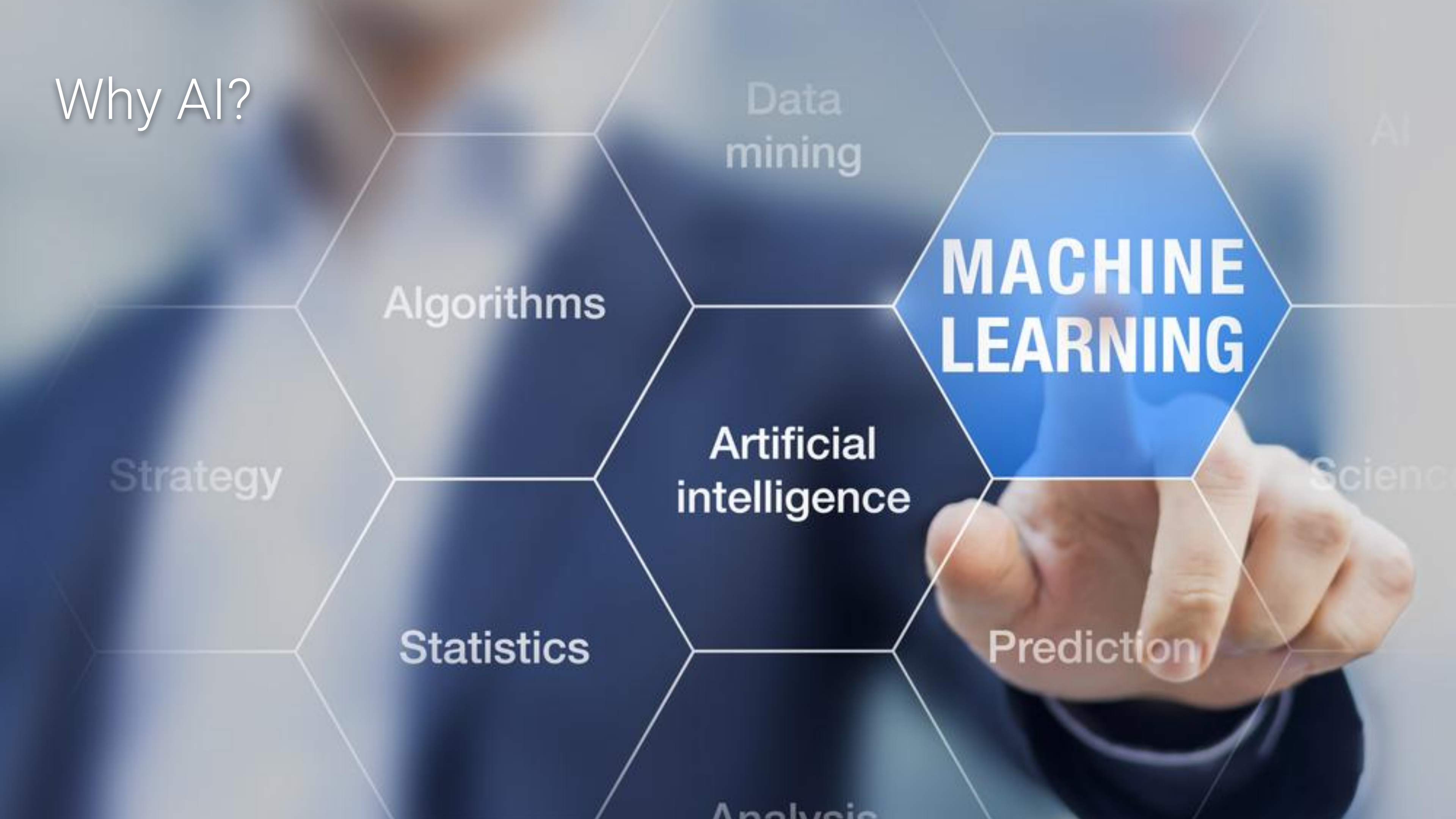
AI requires data to work.

AI requires some form of digitalization to work.

Many promises of digitalization will not work without AI. Some examples are:

- ➢ **Industry 4.0** (resource planning, robot control, transformation of planning data, ...)
- ➢ **IoT** (data without intelligent analysis do not provide value)
- ➢ **Customer service** (chatbots, processing of customer inquiries, ...)
- ➢ **Accounting** (automatic recording and classification of invoices, ...)
- ➢ **Supply chain management** (demand prediction, dynamic routing, ...)

Why AI?


Data mining
MACHINE LEARNING
Algorithms
Artificial intelligence
Strategy
Statistics
Prediction

Why do we build AIs?

Why did evolution favour intelligent systems?
Because intelligence is a major ***advantage*** *in the fight for scarce resources.*

We build AIs, since the owner and operator of the AI system expects an ***advantage*** *in the fight for scarce resources of our economy.*

And maybe AI will also help us to survive and to understand ourselves.

Which benefits do we expect from AI?

New business models

- ➢ self driving taxis

Improvement of existing value chains

- ➢ productivity by automatization

Scientific progress

- ➢ material science for batteries
- ➢ better understanding of emerging phenomena

Progress in medicine

- ➢ faster development of medications

Expected economic impact of AI

PWC forecasts a possible **increase in GDP** for Germany of 11% until 2030 with the targeted use of AI.

This is an increase of about 1% per year.

- ➢ Great potential is seen in the healthcare and automotive sectors.
- ➢ Products can get more user orientation through AI.

https://www.pwc.de/de/digitale-transformation/business-analytics/kuenstliche-intelligenz-sorgt-fuer-wachstumsschub.html

Applied machine learning today

AI in smart manufacturing and supply chain

Demand prediction

Integrated planning

Flexibilisation of production lines

Increase of production rate

Reduction of scrap production and quality issues

Optimization of logistics

Improvement of control systems

Judgement on improvement measures

https://www.techemergence.com/machine-learning-in-manufacturing/
http://www.industryweek.com/leadership/whats-stopping-smart-factory-revolution

Predictive maintenance

Classification of the status of a machine (OK/NOK)

Prediction of time until next failure

Detection of anomalies

Damage prevention

https://www.infoq.com/articles/machine-learning-techniques-predictive-maintenance

Energy management

Energy demand prediction

Avoidance of too high energy demand

Automatic energy trading

Intelligent control of cooling and heating

Damage prevention

https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/
https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/42542.pdf

Quality control

Find correlations between operational data of a factory and the quality of the produced products

Visual quality checks

Fusion of IoT data

Prediction of production quality long before the product is ready (avoid scraping)

https://blog.deepsense.ai/spot-the-flaw-visual-quality-control-in-manufacturing/

Autonomous cars and robots

Automatic transport systems in logistics

Cooperation between humans and robots

Flexible and self-learning robots

Robots with no programming

Self-driving cars and trucks

https://automotivelogistics.media/news/bmw-tests-autonomous-vehicles-in-factories

AI in financial industry

Customer service automation

Credit scores

Trading and money management

Prediction of prices

Compliance checks

Fraud detection

https://www.netguru.co/blog/ai-and-machine-learning-in-fintech.-five-areas-which-artificial-intelligence-will-change-for-good

AI in medicine and healthcare

Drug development

Toxicity estimation

Diagnostics and personalized treatment

Monitoring of infections in hospitals

Robotics supported surgery

AI in tourism

Aggregation of information for tourists

Customer service (booking, traveling)

Personalized travel information (recommendations)

Online translation

Travel assistant (knows your context)

Route planning

Cost forecasting

https://www.altexsoft.com/blog/datascience/data-science-and-ai-in-the-travel-industry-9-real-life-use-cases/

AI in business, marketing and sales

Cash flow estimation

Recommendation systems

Invoice and document processing

Customer service (booking, traveling)

Sentiment analysis and customer review analysis

Lead generation

Customer segmentation

Demand prediction

Replenishment

AI in agriculture

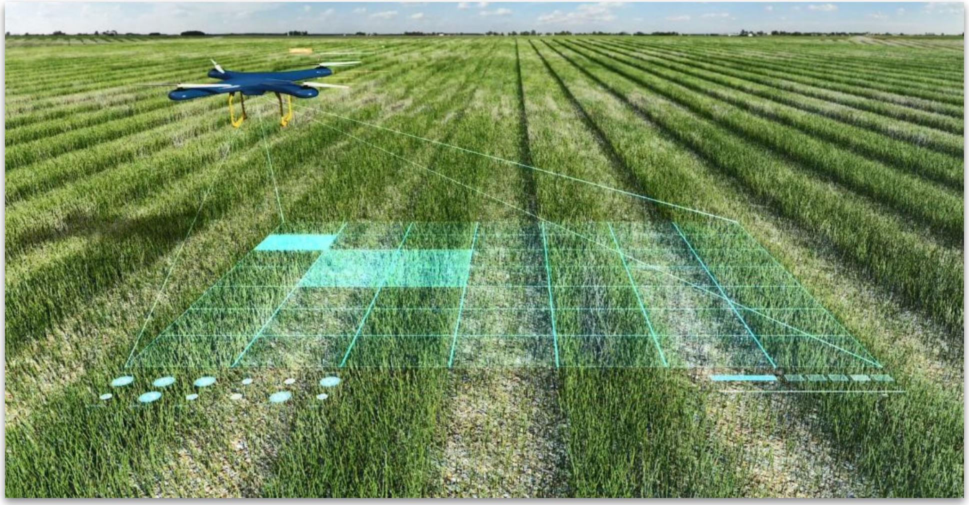
Improved planning of crop cycles

Selective irrigation and spraying

Improvement of plants

Monitoring of health of forests

Autonomous machinery



https://www.cambridgeconsultants.com/insights/video-ai-and-future-agriculture

AI applications in public administration

Support legal work by natural language processing

Automatize legal processes

Chatbots for citizen service

Compliance checks

http://www.icdk.us/aai/public_administration

Ethics and legal issues

The trolley problem

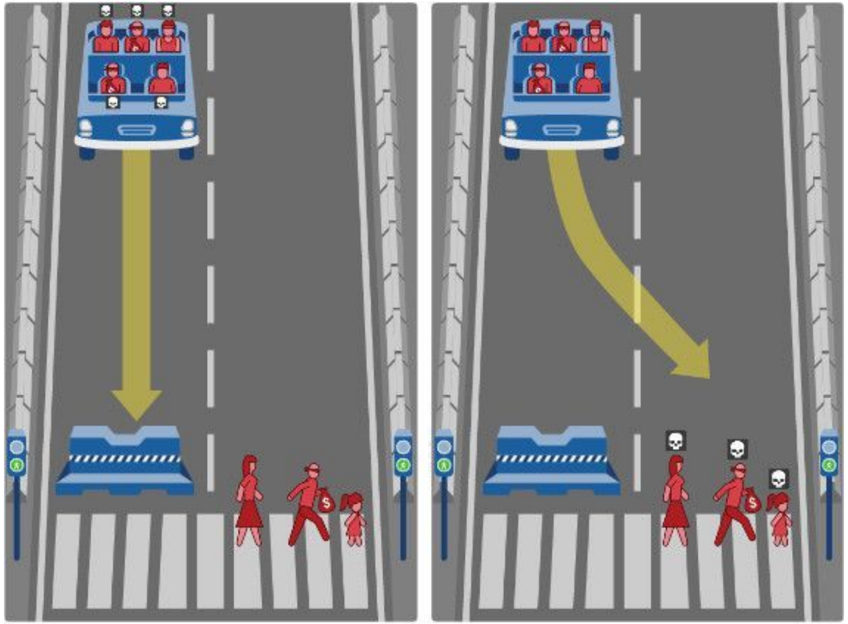
Can a machine decide over the fate of humans?

Long ongoing discussion misleading into utilitaristic arguments...

Finally some clarity arises.

"An autonomous car shall not decide over the fate of humans" (recommendation by German council related to ethics and computers).

The discussion about vehicles that have to decide who to kill is misguiding. **A vehicle must always be able to stop in time**.

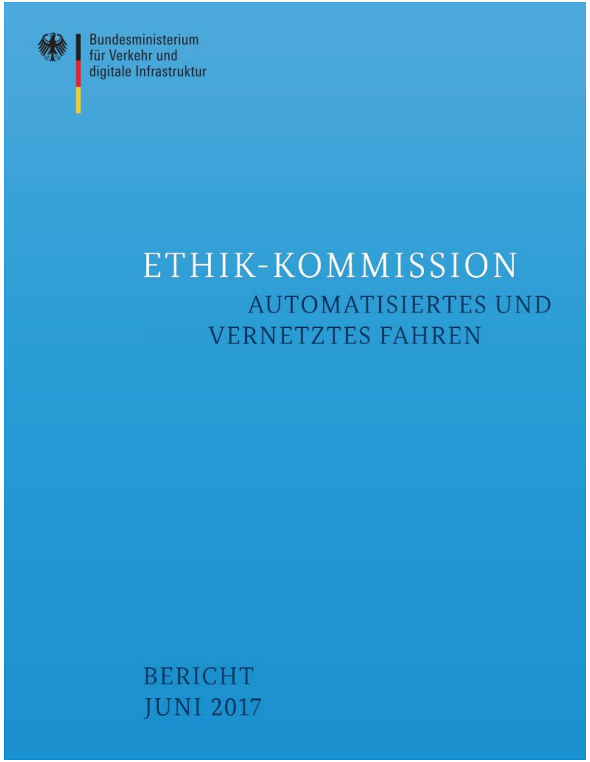


http://moralmachine.mit.edu/

Progress for autonomous vehicles

In June 2017 the german ministry for traffic published the recommendations of a commission for autonomous driving

- ➢ 20 ethical rules
- ➢ autonomous cars have to increase traffic safety
- ➢ in case of unavoidable accidents, the car must not favor any group of persons



https://www.bmvi.de/
https://factorynet.at/a/20-ethische-regeln-fuer-autonomes-fahren

Selection of ethical issues with AI

- do we want machines that **evaluate and judge** people?
- should we build machines that can potentially become **independent and threaten** us humans?
- should we build machines that can have **emotions** or a **consciousness** of their own?
- how do we **distribute the responsibility** in dealing with AIs?
- how do we deal with the fact that many **jobs** of humans disappear?

Specific AI criticism

Jobs

AI will steal all our jobs.

This is a real risk. We as a society have to decide which jobs shall never be automatized. Otherwise only low profile jobs, which are not worth to automatize, will be left to us.

Legal Status

AI systems have no clear legal status.

This needs to be solved. E.g. it is unclear if an AI trained by the user is in the responsibility of the manufacturer.

Black Box Argument

AIs make decisions using intransparent algorithms. We want to know how a system decides about us.

This argument is very valid. We need explainability as a new technology function. However, we also need to work for more transparency in other fields (banks, administration)

Selection of open legal issues

- how do we deal with the fact that a machine **learns new things from an owner** and problems can arise as a result?
- who is responsible if problems arise due to the **unplanned interaction** of two AIs?
- do we allow machines to **represent humans**?
- which legal status do we give intelligent machine?
- which data may be used for the training of AI models?
- what degree of anonymization is sufficient for different applications?

Trustworthy AI

The EU promotes the concept of **trustworthy AI** based on the following principles

- **lawful** - respecting all applicable laws and regulations
- **ethical** - respecting ethical principles and values
- **robust** - both from a technical perspective while taking into account its social environment

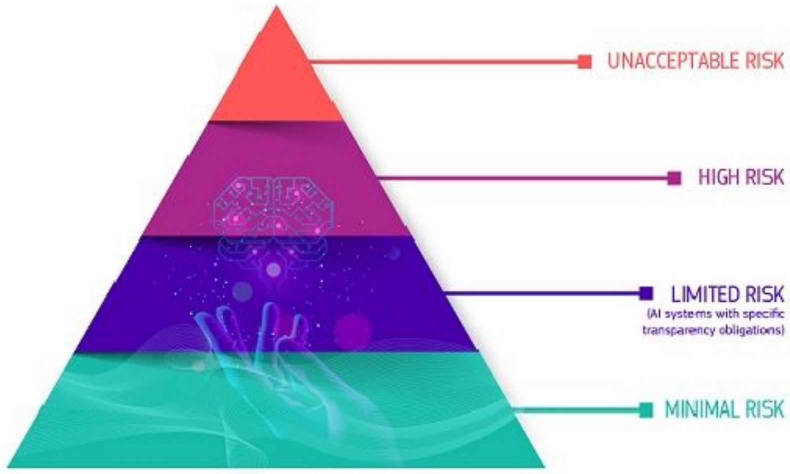
Legal work to **regulate** use of AI is underway in the EU.

https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai
https://algorithmwatch.org/en/project/ai-ethics-guidelines-global-inventory/

AI Regulation

The EU strategy for AI contains a legal initiative that will contribute to building trustworthy AI. A first step is a

- ➢ **Regulatory framework proposal on artificial intelligence**.
- ➢ The framework establishes a hierarchy of risk levels for AI systems ranging from unacceptable risk to minimal risk
- ➢ Risky AI systems require strict certification
- ➢ Following the Commission’s proposal in April 2021, the regulation could enter into force in the second half of 2022 in a transitional period.



https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence
https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

How to build safe AI

Functional safety

Previous methods for the validation of models make only limited statements about how the system behaves in unknown situations.

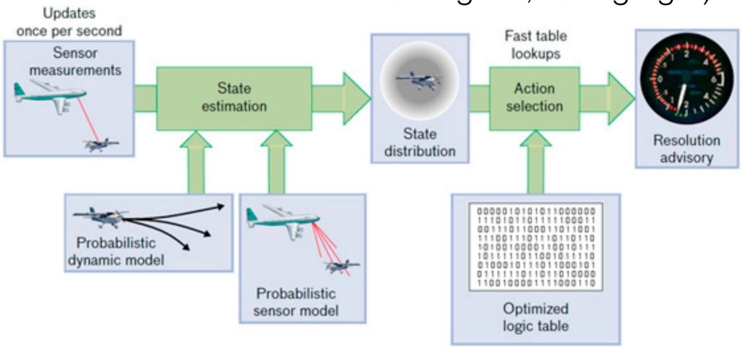
Problem: **unknown unknowns**

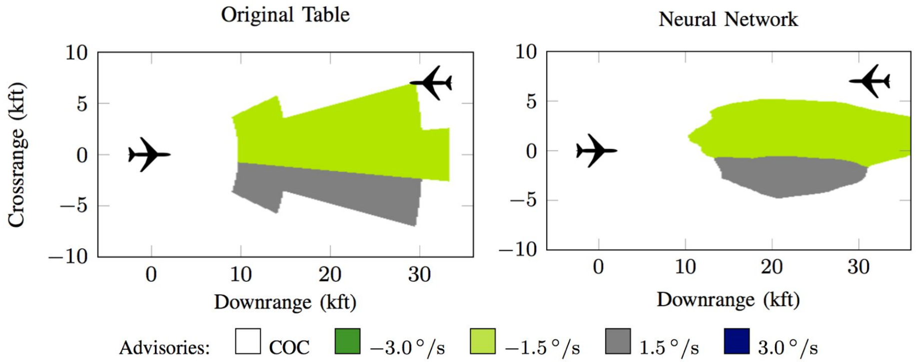
Functional safety

- ➢ Safety standards require proof that a system is **fit for purpose**.
- ➢ This is an active field of research (where you can earn good money :-)

Input: sensor data once per second

Output: one of 5 resolution advisories (COC, weak left, weak right, strong left, strong right)





http://composition.al/blog/2017/05/30/proving-that-safety-critical-neural-networks-do-what-theyre-supposed-to-where-we-are-where-were-going-part-1-of-2/

Build safe AIs

Goal setting for AIs
precise specification of goals for AIs is very difficulty, since many situations in the future cannot be anticipated and goals may outdate.

Implant values
Give AIs values and hope the AI will follow those values. However, our value systems are inconsistent.

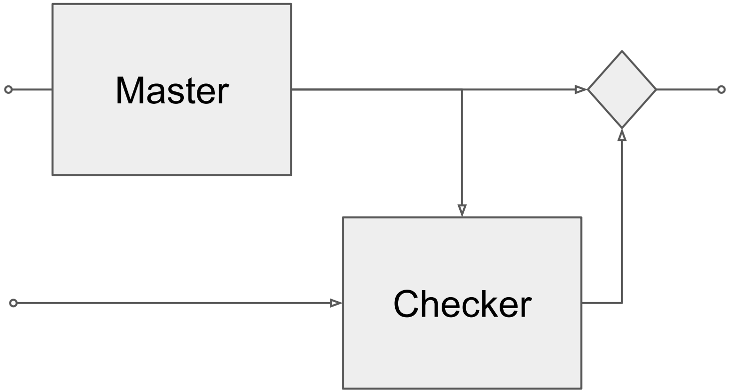
Open loop systems
an open loop machine is unlikely to develop consciousness and may be easier to controls

Independent safety checker
an independent checking mechanism checks all outputs of the AI and blocks potentially dangerous actions. This concept is in use in many technologies.

Old but good idea: master-checker strategy

A **master** computer is responsible for strategy and optimization. It calculates the next action with the help of neural networks.

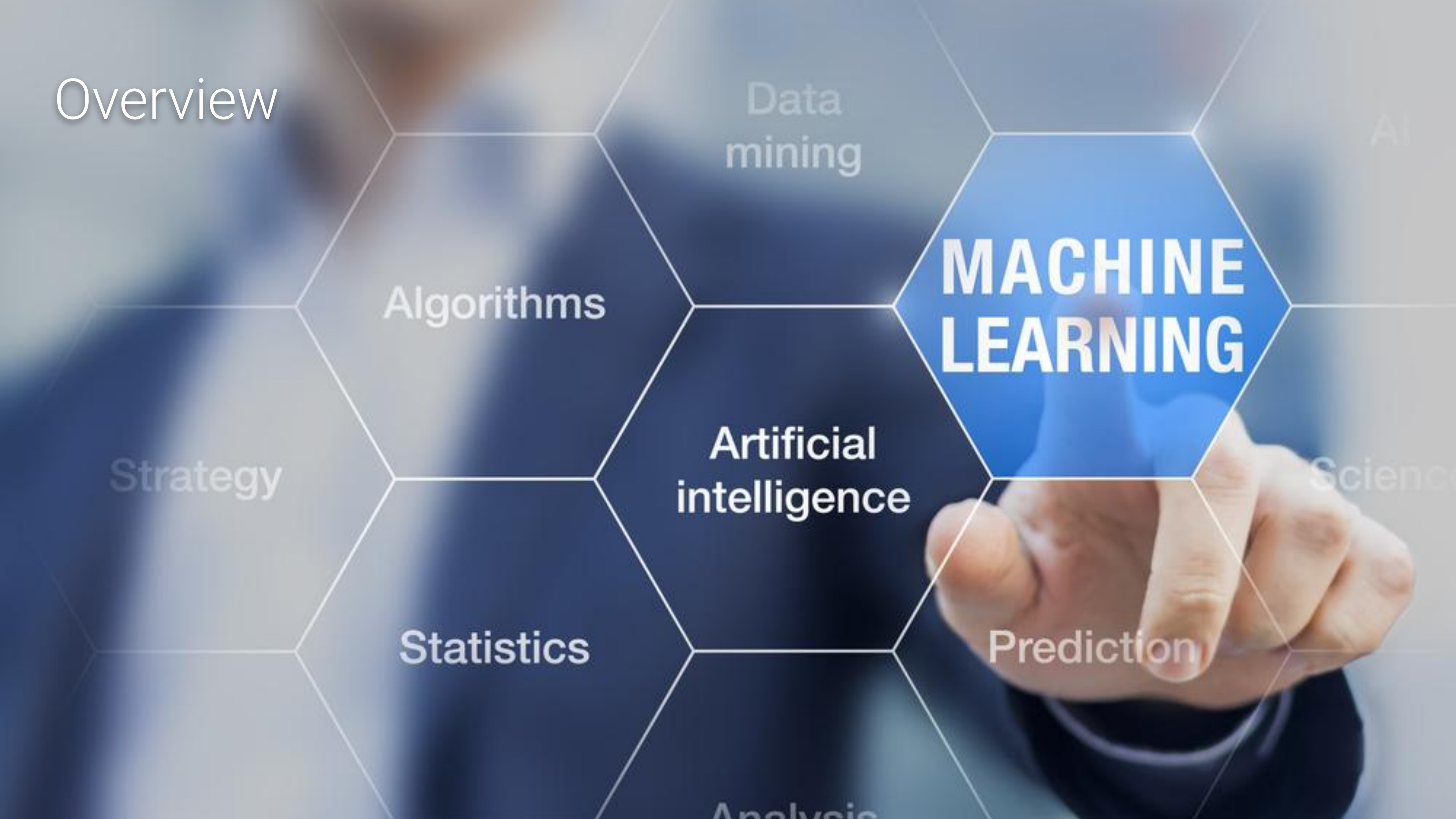
A **checker** computer independently checks whether the current action in the current state can lead to a dangerous situation. It is implemented and certified with a classic technology. In the event of an error, the checker brings the system into a **safe state**.



Artificial Intelligence

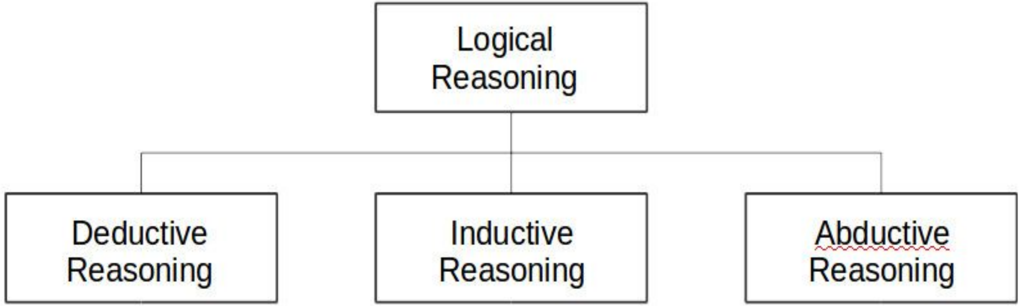
2 Machine Learning

Overview


Data mining
AI
Algorithms
MACHINE LEARNING
Strategy
Artificial intelligence
Statistics
Prediction

Reasoning in AI

Generate conclusions from existing information based on different **procedures**



Deduction: conclusions from facts and rules (top-down)

Induction: generate rules from specific cases (bottom-up)

Abduction: explain specific cases based on existing rules

https://www.quora.com/What-is-a-good-example-of-abductive-reasoning

learn from data

machine learning

What can machine learning do?

Regression
- ➢ estimation of output values from input values

Classification
- ➢ estimation of category from input values

Locating/masking
- ➢ locate and classify categories in data (e.g. object in image)

Clustering
- ➢ grouping of samples from unknown criteria

Dimensionality reduction
- ➢ extract the most important features

Anomaly detection
- ➢ find the difference between normal and unusual

Generation of data
- ➢ generate new examples from given data sets (e.g. faces of fictional stars)

Planning
- ➢ create strategies (e.g. for game play)

Transformation
- ➢ language translation
- ➢ question answering
- ➢ summary of text
- ➢ video to text

Methods grouped by learning type

Supervised learning

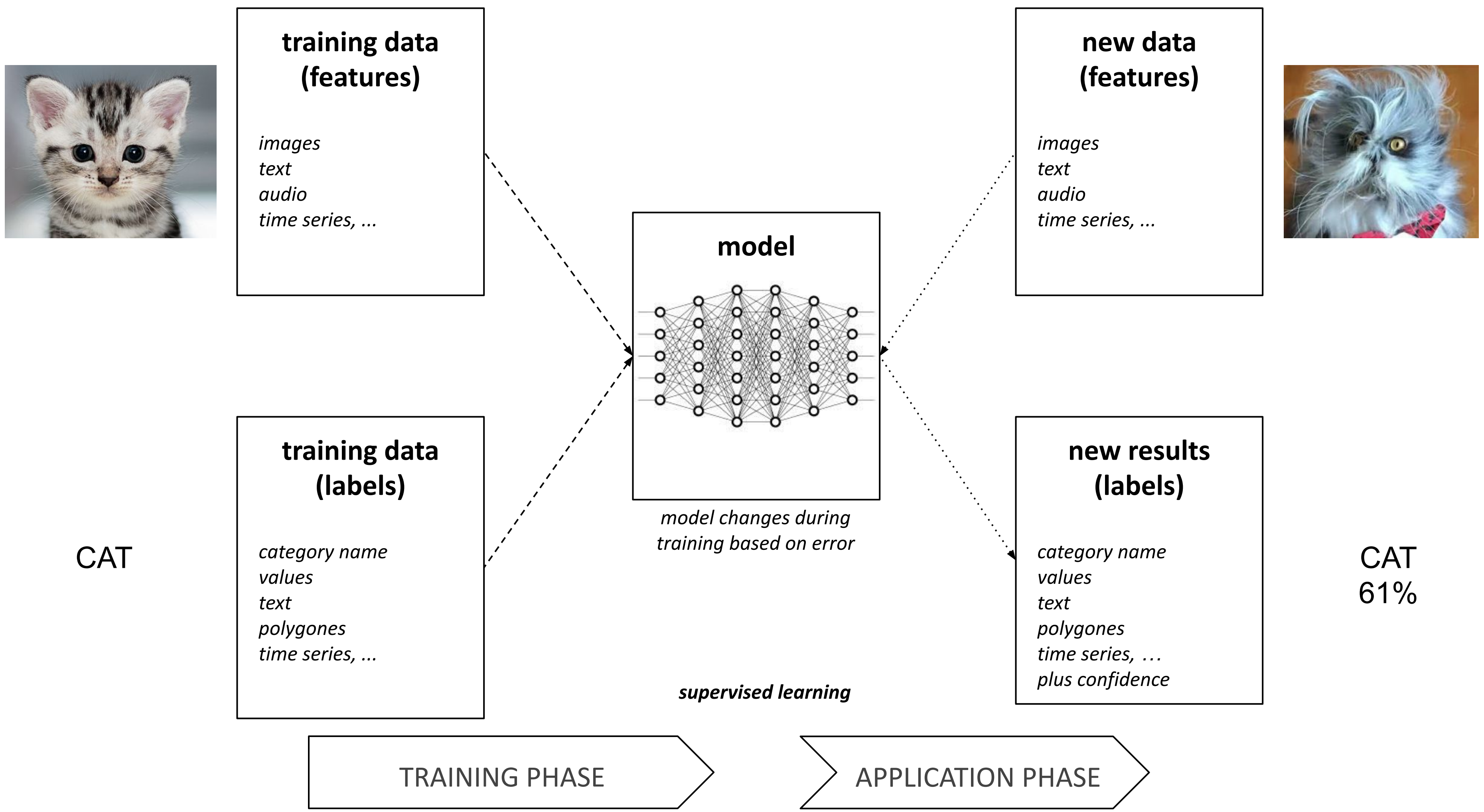
- ➢ data samples with the desired labels are available for the learning process (trainings data)
- ➢ regression
- ➢ classification
- ➢ locating/masking
- ➢ generation of data
- ➢ transformation

Unsupervised learning

- ➢ trainings data not available
- ➢ clustering
- ➢ dimensionality reduction
- ➢ anomaly detection

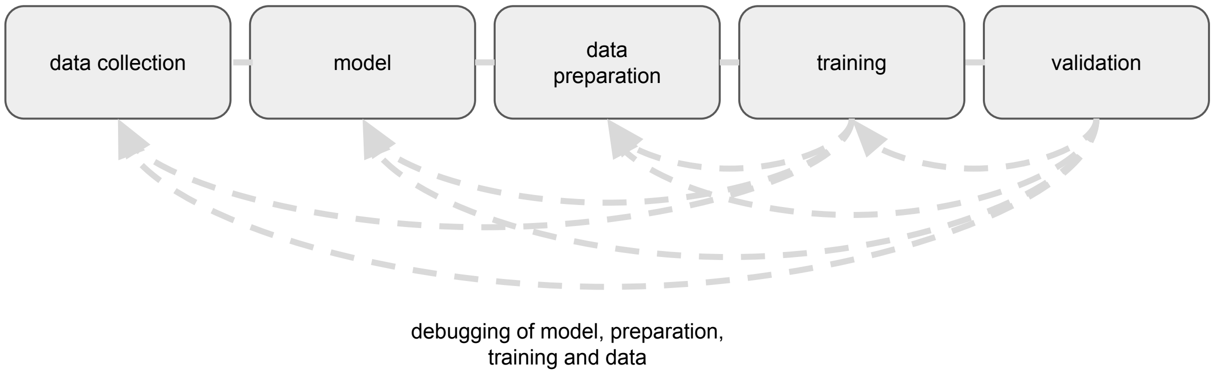
Reinforcement learning

- ➢ instead of using examples, the system learns by try-and-error and rewarding correct solutions
- ➢ planning
- ➢ advanced control

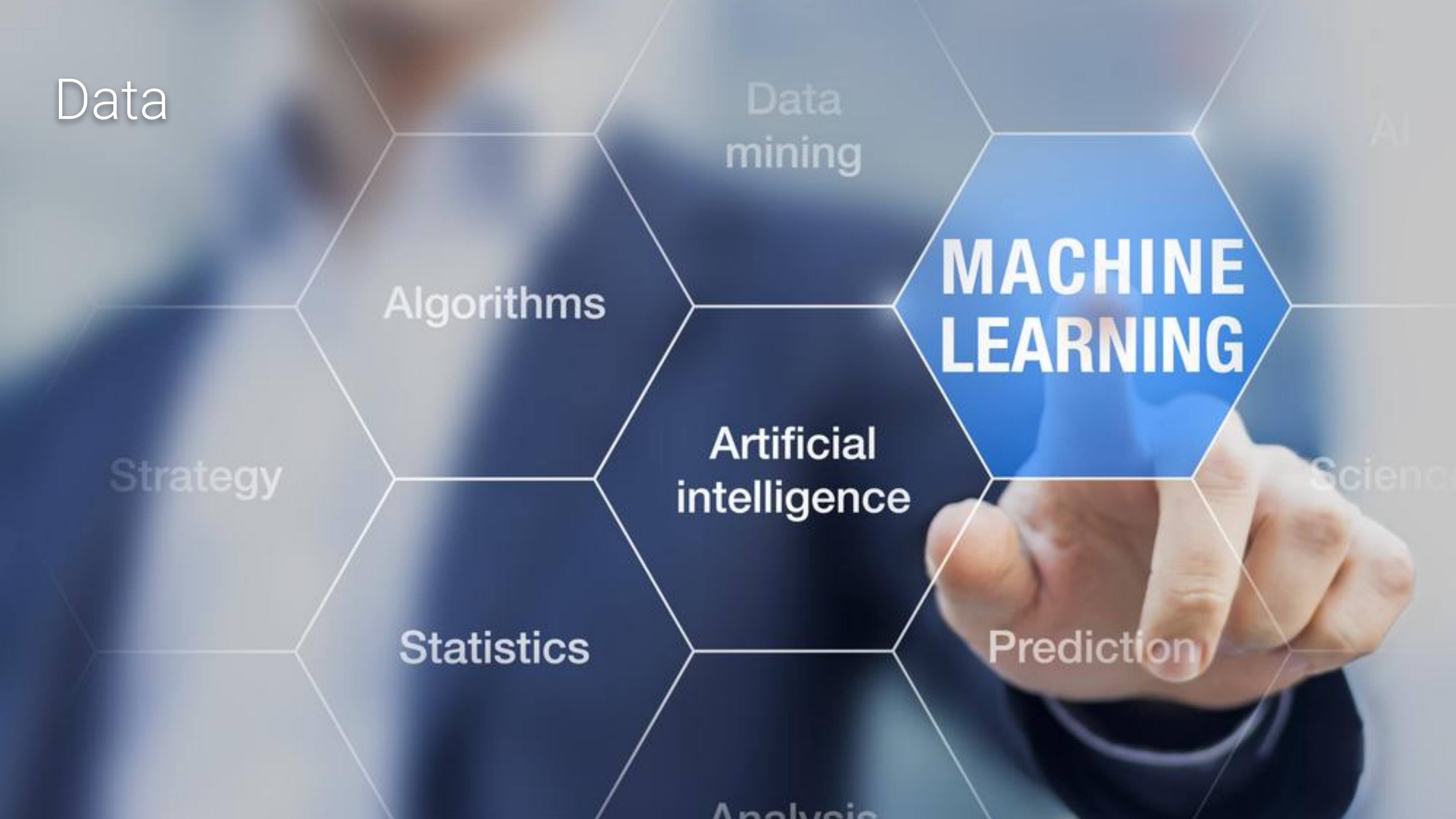


training data (features)
images
text
audio
time series, ...
CAT
training data (labels)
category name
values
text
polygones
time series, ...
model
model changes during training based on error
supervised learning
new data (features)
images
text
audio
time series, ...
new results (labels)
category name
values
text
polygones
time series, …
plus confidence
CAT
61%
TRAINING PHASE
APPLICATION PHASE

Simplified life cycle of ML projects



Data


Data mining
AI
MACHINE LEARNING
Algorithms
Artificial intelligence
Strategy
Statistics
Prediction

Let's talk about data

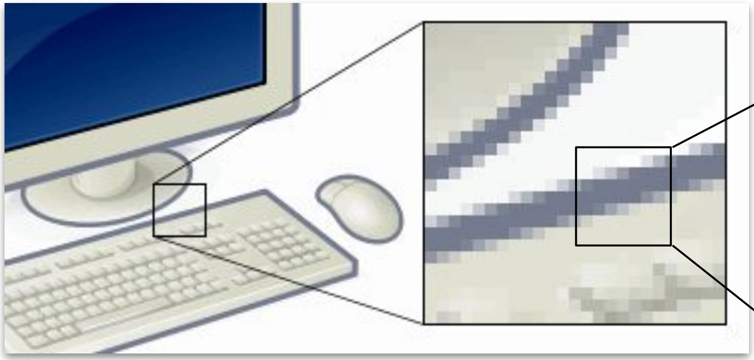
Some definitions that we will use:

- ➢ data is often subdivided into **datasets**
- ➢ a dataset contains many **samples**
- ➢ a sample is a tuple (vector)
  - ○ sample: $\{ y, x_0, x_1, \dots, x_{k-1} \}$
  - ○ k: dimension
  - ○ x : **feature**, predictor, input
  - ○ y : **label**, output, class

Example of IRIS data in pandas (k=4)

		x_0	x_1	x_2	x_3	y
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Image data



255	255	255	255	235
255	235	200	180	140
235	200	180	140	140
200	180	140	140	140
180	140	140	140	230
140	140	180	230	230
140	200	230	230	230
230	230	230	230	230

An image file consists of pixel data organized in rows and columns. Each pixel defines the brightness of the three primary colors red, green, blue in the RGB model. The brightness is for example a value from 0 to 255.

source: By ed g2s • talk - Example image is a rendering of Image:Personal computer, exploded 5.svg., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=807503

Image data

Image data are **features.**
Organization of the image data is in rows and columns. This results in **2 dimensions**.

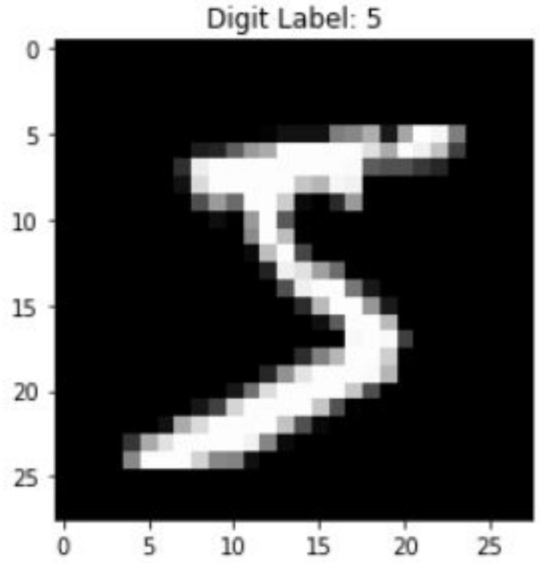
- ➢ 0 means black
- ➢ 1 means white
- ➢ Values in between represent grey tones

In case of color data, 3 or four channels of additional data represent red, green, blue and transparency as additional dimensions.

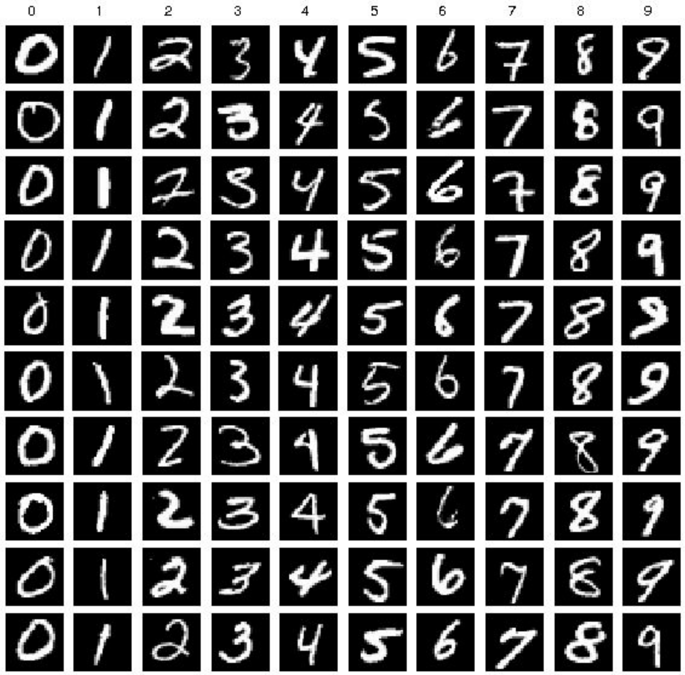
```
X.head(28)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
6	0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
7	0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
8	0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	14	1	154	253	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	252	253	187	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	249	253	249	64	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	23	66	213	253	253	253	253	198	81	2	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	18	171	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	55	172	226	253	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	136	253	253	253	212	135	132	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Image data



MNIST dataset: 60,000+10,000 scanned handwritten digits, produced by the National Institute for Standards and Technology.



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

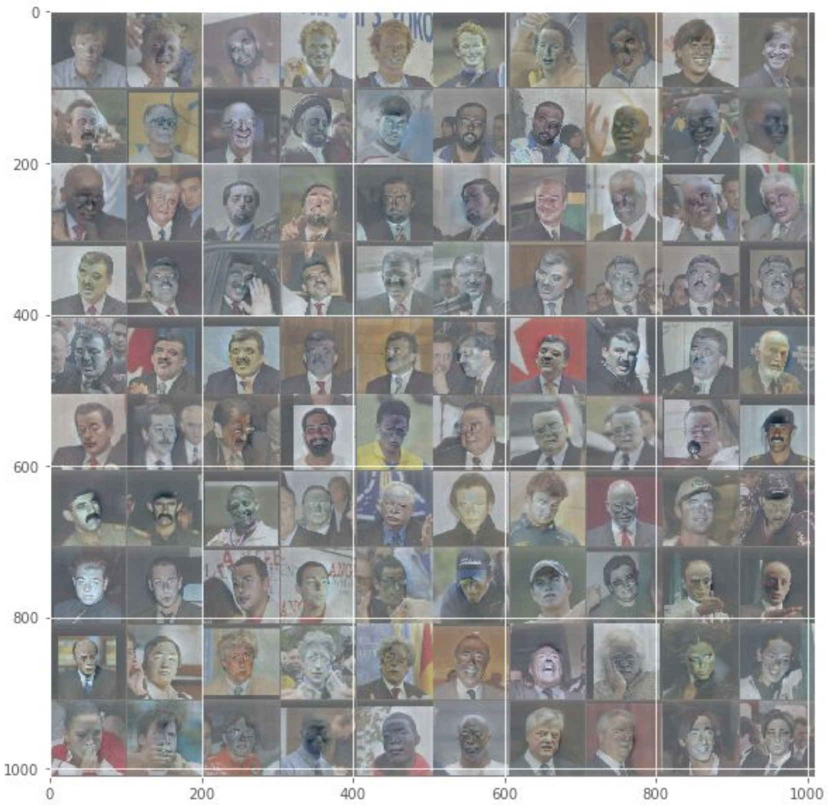
Processing image data

Image data tend to be large, there are several options to **store** image data

- ➢ in files
- ➢ in database blobs
- ➢ in memory

For **processing**, image data has to be

- ➢ **loaded** into memory
- ➢ **uncompressed**
- ➢ **cropped** and **resized**
- ➢ **improved** (e.g. higher contrast)
- ➢ **normalized**



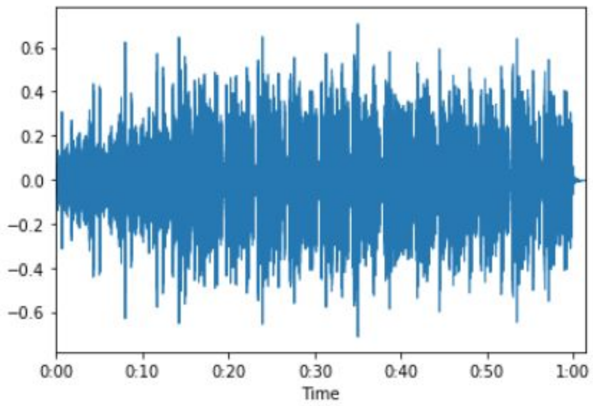
source: https://machinelearningmastery.com/best-practices-for-preparing-and-augmenting-image-data-for-convolutional-neural-networks/
https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258

Audio data

The features of audio data in uncompressed form are usually represented as **power values** (PCM: pulse code modulation). The values themselves are either integer values with 8 or 16 bit resolution or stored as floating point numbers.

The **time axis** is not explicitly stored in such a data set, since equidistant points are assumed. However, the sampling rate is defined (e.g. 22050 values per second).

	pcm
0	0.000223
1	0.000249
2	0.000243
3	0.000250
4	0.000274
5	0.000262
6	0.000194
7	0.000168
8	0.000184
9	0.000184
10	0.000186
11	0.000224
12	0.000222
13	0.000199
14	0.000221
15	0.000265
16	0.000274
17	0.000302
18	0.000303
19	0.000290
20	0.000314



Sensor data

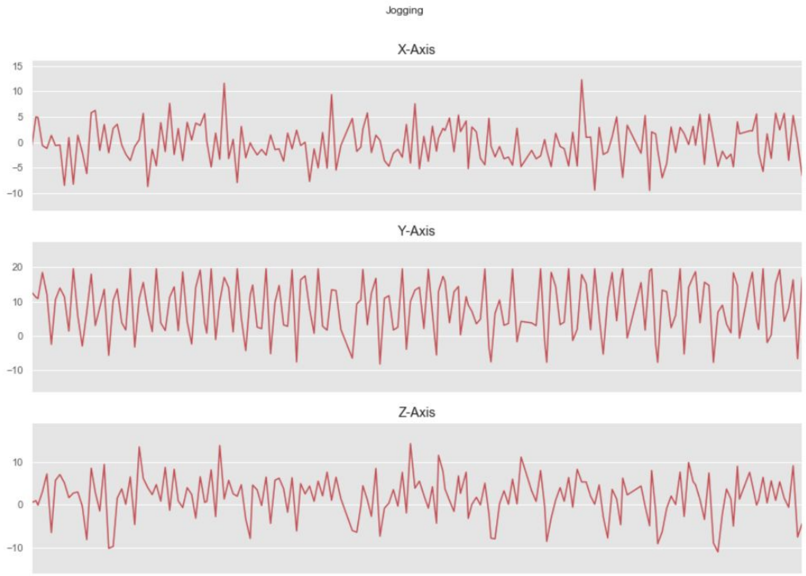
E.g. Human Activity Recognition data.

- ➢ Measured acceleration values during human activities.
- ➢ The values are divided into 3 axes and represent the acceleration as floating point values.

The time is represented in milliseconds since 1.1.1970 in UTC (unix epoch).

- ➢ Timestamps may be in arbitrary distances.
- ➢ It is important that all feature values relate to the same timestamp.

	user-id	activity	timestamp	x-axis	y-axis	z-axis
0	33	Jogging	49105962326000	-0.7	12.7	0.5
1	33	Jogging	49106062271000	5.0	11.3	1.0
2	33	Jogging	49106112167000	4.9	10.9	-0.1
3	33	Jogging	49106222305000	-0.6	18.5	3.0
4	33	Jogging	49106332290000	-1.2	12.1	7.2
5	33	Jogging	49106442306000	1.4	-2.5	-6.5
6	33	Jogging	49106542312000	-0.6	10.6	5.7
7	33	Jogging	49106652389000	-0.5	13.9	7.1
8	33	Jogging	49106762313000	-8.4	11.4	5.1
9	33	Jogging	49106872299000	1.0	1.4	1.6
10	33	Jogging	49106982315000	-8.2	19.6	2.7
11	33	Jogging	49107092330000	1.4	5.8	3.0



Stock market data

Stock market data is **time series** data with features that are topic-related and one or more features that define a point in time (timestamp). It has to be carefully decided if the timestamp is used just as **index**, or if the timestamp and derived values are **features**.
Special focus for time series data must be given to consistent representation of time.
Derived features of the timestamp can be hour, day, month, season, ...

	Open	High	Low	Close	Volume
Date					
2020-01-16	313.59	315.70	312.09	315.24	27207254
2020-01-15	311.85	315.50	309.55	311.34	30480882
2020-01-14	316.70	317.57	312.17	312.68	40653457
2020-01-13	311.64	317.07	311.15	316.96	30521722
2020-01-10	310.60	312.67	308.25	310.33	35217272

Text data

3 different ways to represent text data:

- ➢ as a sequence of **characters** in the form of 1-4 bytes (ascii, utf-8, ...)
- ➢ as words in **one-hot** encoding by reserving one dimension (e.g. 400,000 dimensions) for each word
- ➢ For each word a point in a **dense vector space**.

E.g. string and sequence of values

```
hello = "hello world"
```

```
'104, 101, 108, 108, 111, 32, 119, 111, 114, 108, 100'
```

E.g. representation as one-hot encoding

Rome Paris word V

```
Rome   = [1, 0, 0, 0, 0, 0, …, 0]
Paris  = [0, 1, 0, 0, 0, 0, …, 0]
Italy  = [0, 0, 1, 0, 0, 0, …, 0]
France = [0, 0, 0, 1, 0, 0, …, 0]
```

E.g. high dimensional representation of the word 'king'

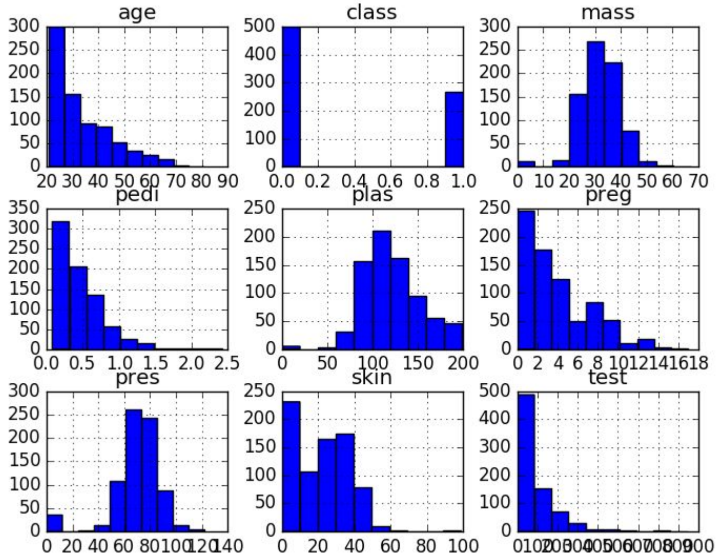
```
[ 0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 ,
-0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961
, -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 ,
-0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 ,
-1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042 ]
```

source: http://jalammar.github.io/illustrated-word2vec/

Data analysis

Data visualization is a key activity for working with data. It allows humans to get a **basic intuition** about the nature of the data.

Visualize data as they are going **into the model** to get a better understanding of the data. In many cases problems in the data processing pipeline lead to reduced quality of the model.



Example

➢ the project tried to detect manipulations of the images

➢ the images were downsampled during the preparation phase.

➢ the downsampling deleted all traces of manipulation.

source: https://machinelearningmastery.com/visualize-machine-learning-data-python-pandas/

Summarize data

Use specific and countable parameters to get an overview of the data

- **shape** of the data (how many dimensions)
- **features** (names and data type of columns)
- **rows** (count of samples)

Print a **small number of samples** of the data

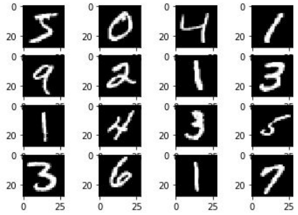
Anzeige der Anzahl der Samples

In [3]:
```
# Anzeige der Anzahl der Samples
print('Trainingsdaten: X=%s, y=%s' % (trainX.shape, trainY.shape))
print('Testdaten: X=%s, y=%s' % (testX.shape, testY.shape))
```

```
Trainingsdaten: X=(60000, 28, 28), y=(60000,)
Testdaten: X=(10000, 28, 28), y=(10000,)
```

Anzeige von Beispielen der Samples

In [4]:
```
# Anzeige von Beispielen der Daten
for i in range(16):
    plt.subplot(4,4,1 + i)
    plt.imshow(trainX[i], cmap=plt.get_cmap('gray'))
plt.show()
```



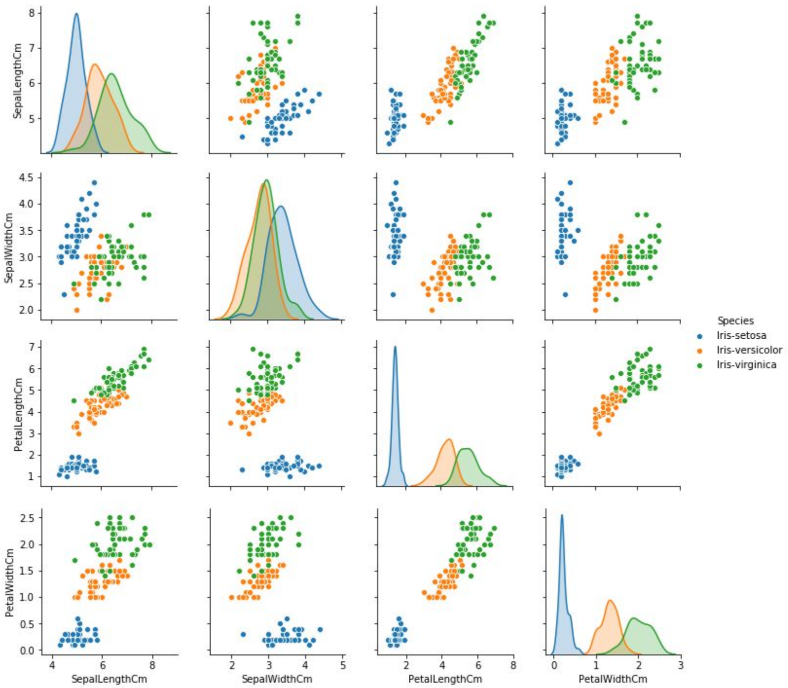
source: https://machinelearningmastery.com/quick-and-dirty-data-analysis-for-your-machine-learning-problem/

Statistical data analysis

Calculate and print statistics about the features and labels as well as relations between features.

- ➢ **histograms** of individual features
- ➢ **pairplots** of feature pairs
- ➢ **distribution** of labels

pairplots of IRIS dataset features



source: https://towardsdatascience.com/data-visualization-a6dccf643fbb

Origin of data

Data can have many different origins. The switch from hand collected data in prototyping and proof-of-concept projects to data from real sensors can pose a significant challenge.

Hand collected data in CSV files

- ➢ clean data
- ➢ usually less variation in data

Real-time **data from IoT sensors**

- ➢ sensor errors
- ➢ measurement errors and noise
- ➢ missing data
- ➢ duplicated data
- ➢ varying sampling rate
- ➢ distortions from harsh environment

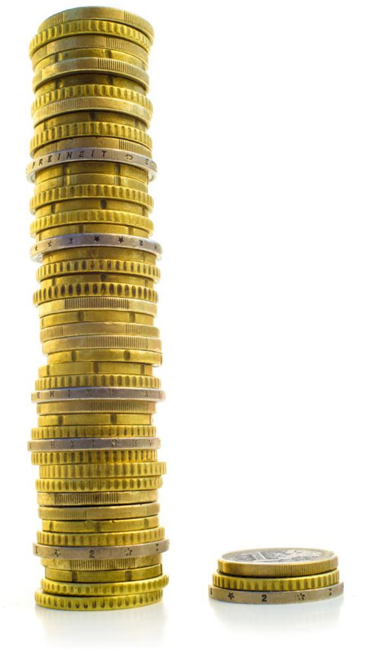
The value of data

Data ownership is increasingly recognized as a new value. New business models increasingly depend on the ownership of data.

Example

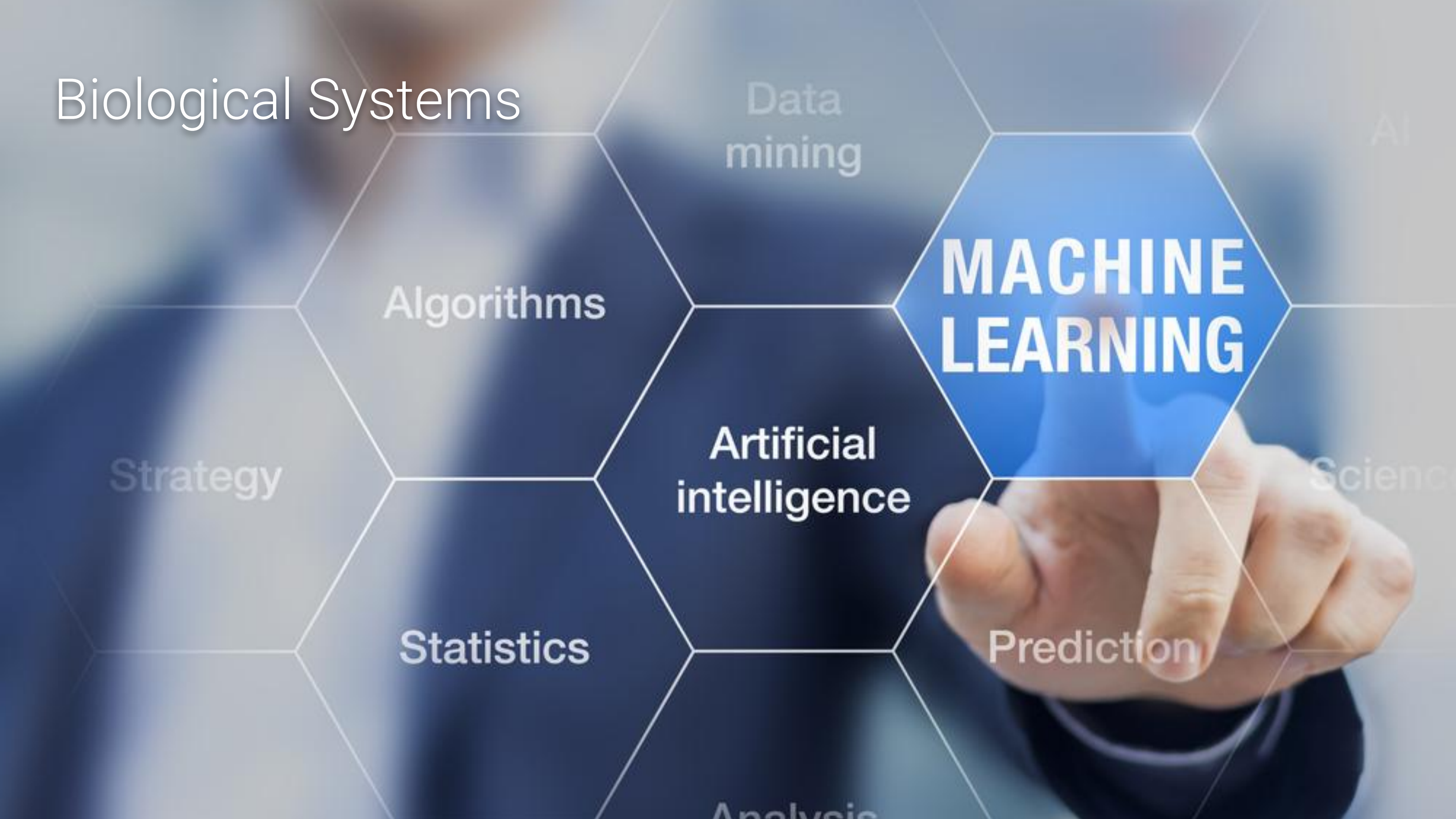
- Amazon vs Google/Mastercard
- Amazon owns the data on the purchasing behavior of millions of people
- Google pays Mastercard millions to get similar data.
- Google needs the data for targeting advertisements

Application: Marketing and Sales



https://www.bloomberg.com/news/articles/2018-08-30/google-and-mastercard-cut-a-secret-ad-deal-to-track-retail-sales

Biological Systems

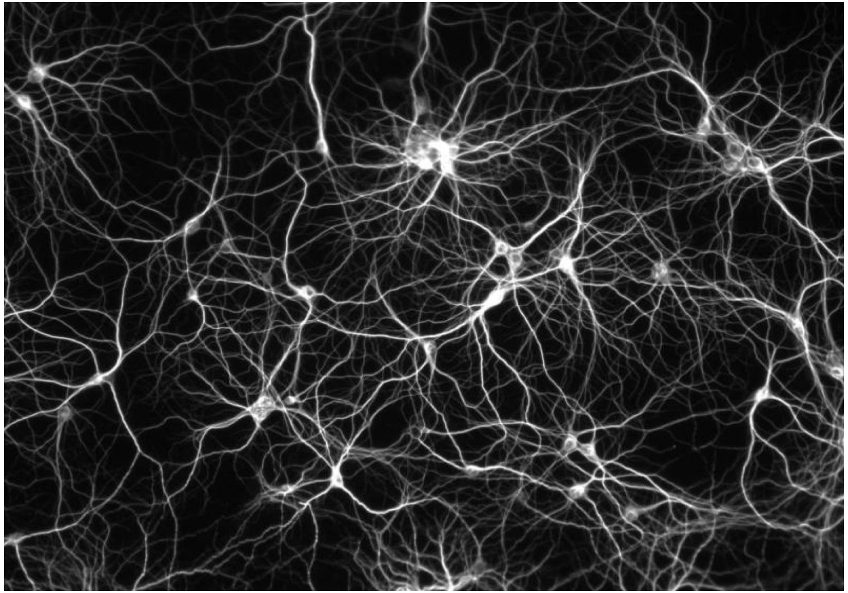


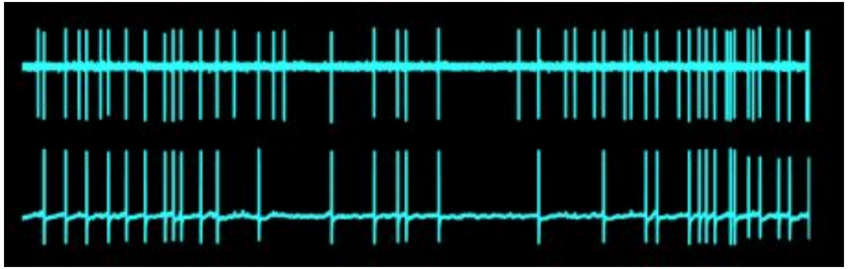
Neurons in a brain

The brain and nerve centres of biological organisms are composed of strongly networked neurons. Neurons are living cells.

A neuron is a biological information processing unit with an extremely **low energy** requirement.

A neuron processes information in the form of electrical impulses. The coding takes place via the **frequency of pulses**.





source: https://www.monell.org/researchoverviews/graeme_lowe.html

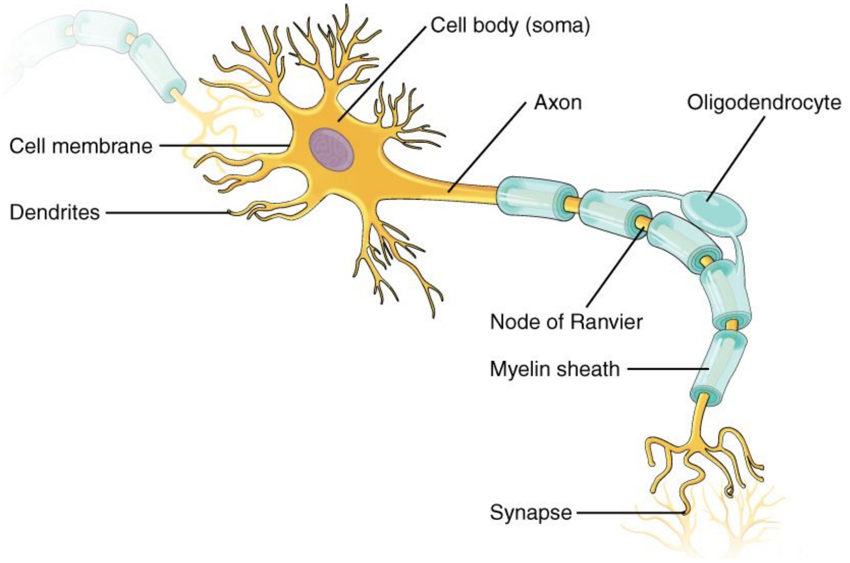
Setup of neurons

Cell Nucleus (Soma)
  life support and activation function

Dendrites
  receive impulses from other cells and pass them on to the cell nucleus.

Axons
  sends the output of the cell nucleus to other cells

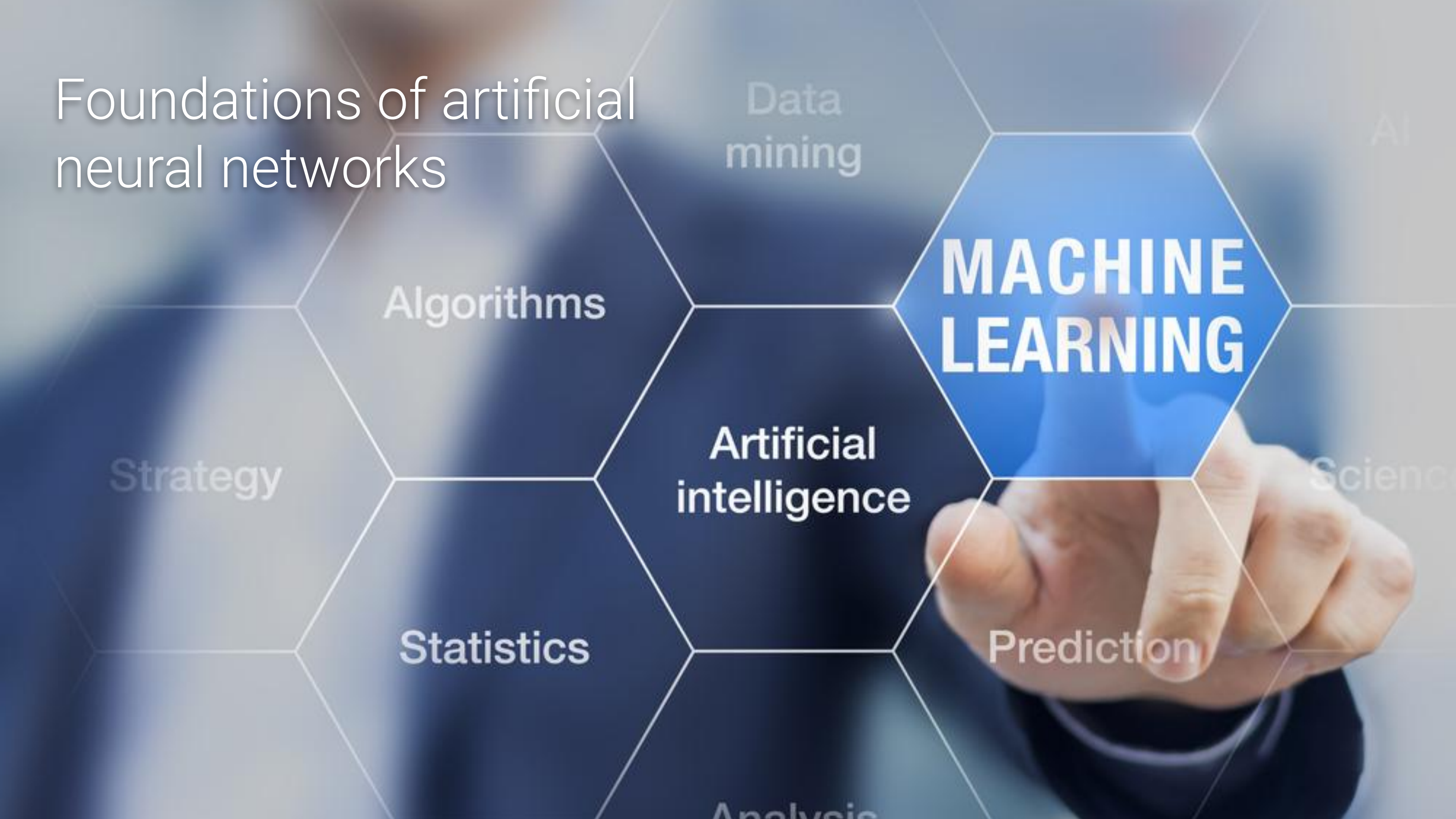
Synapses
  connect axons and dendrites with a weighted transfer function.



Neurons can do a form of **calculation**.

http://philschatz.com/anatomy-book/contents/m46509.html

Foundations of artificial neural networks



The perceptron

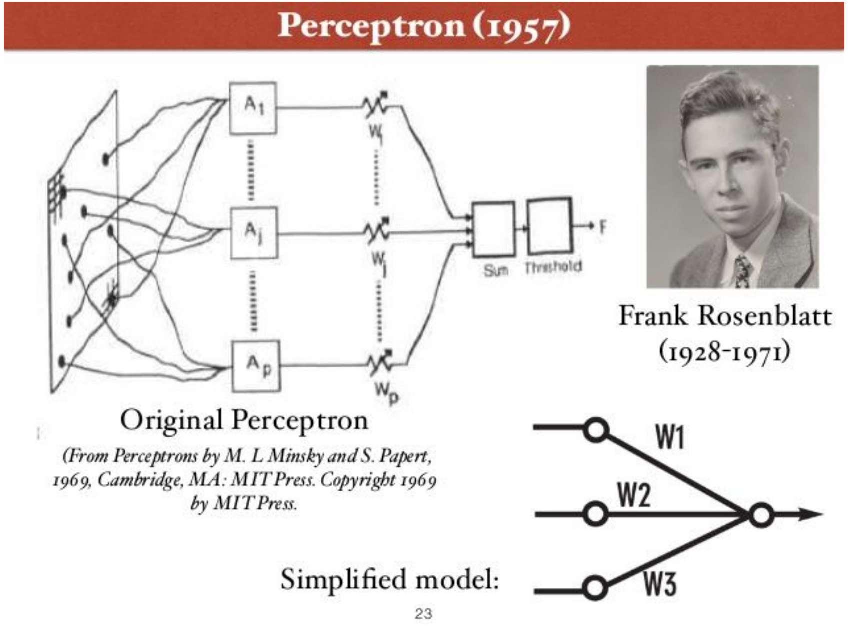
First publication of an artificial neural network by Frank Rosenblatt.

Each input value has a **weight**.

The perceptron forms the **sum** of the weighted inputs.

There is a **threshold** switch (0,1) at the output.

Simple linear classifier. Can also simulate logical operations (OR, AND, NOT) but **not XOR**.



http://slideplayer.com/slide/5209299/

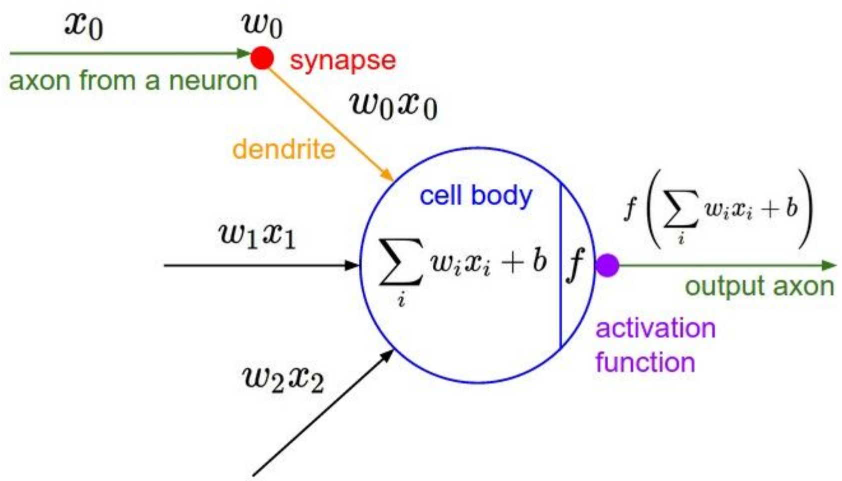
Simplified model of neurons

Input values **X** come from other neurons and are multiplied with a weight value **W**

The core **sums** up all weighted X values and applies a non-linear **activation function A**

The resulting value is sent out as Y value to other neuron inputs

Function is very similar to a logistic regression function if the activation function is the sigmoid function



$$\hat{y} = A\left(\Sigma_i w_i x_i + b\right)$$

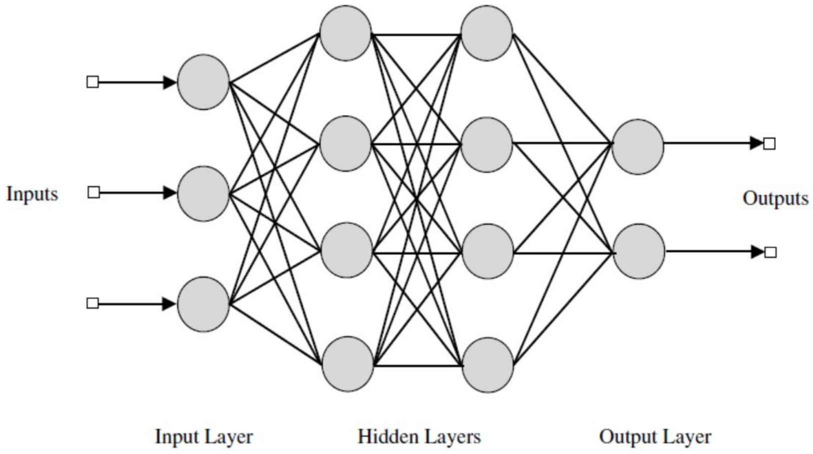
https://stackoverflow.com/questions/38850538/single-neuron-in-neural-network-using-c

Variants of networks

Which directions do the connections take?

If the connections go only in one direction from input to output it is a **feed forward network**. Resonance is prevented by avoiding circular connections.

Networks with specific circular connections are called **recurrent neural networks**.



Feed forward networks only have connections towards output.

Recurrent Neural Networks have selected loops and thus get a memory function which allows them to store context information.

More variants of networks

Layers

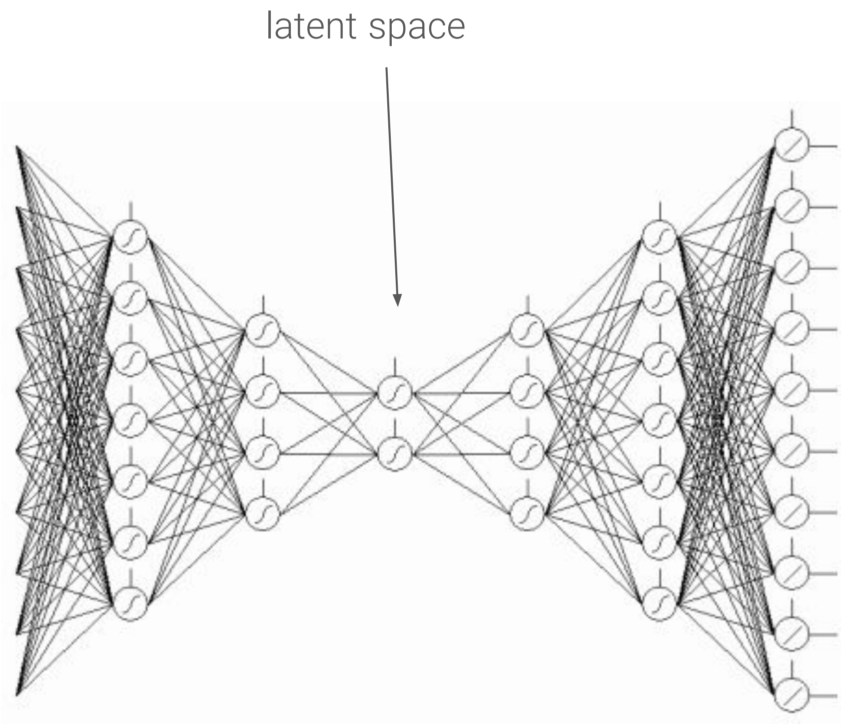
- ➢ many **layers** of neurons increase the capacity
- ➢ the number of neurons per layer can vary

Generalisation

- ➢ layers with fewer neurons than the previous layer promote **generalization**

Memorization

- ➢ wide layers can learn many features which can lead to **overfitting**



Latent Space: inner and hidden representation of data

Forward propagation

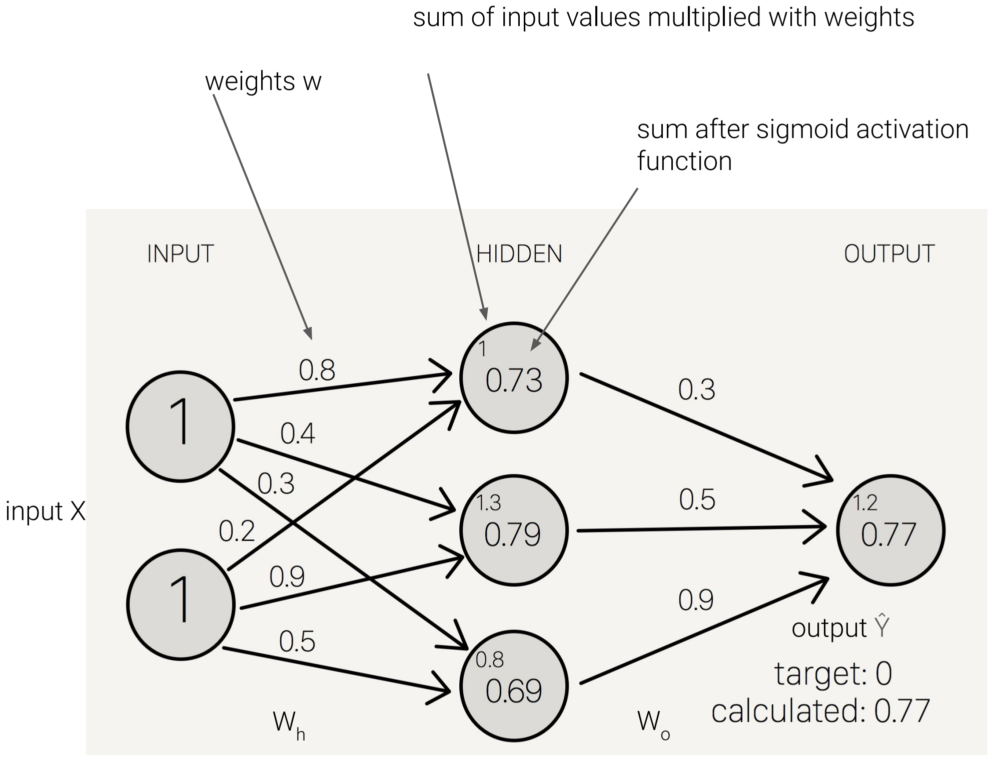
How does a neural network calculate?

1) the **input** feature values **X** are applied to the input neurons

2) for each hidden neuron, all input values are multiplied with their **weights**

3) the **activation function A** is applied to the sum. In this case the sigmoid function.

4) the resulting value is the **activation** of the neuron. This is the input for the next layer. The final **output** is the prediction **Ŷ**



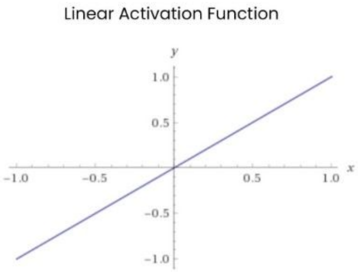
$$\hat{Y} = A\,(\,A\,(\,X * W_h) * W_o\,)$$

https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

Regression and classification with neurons

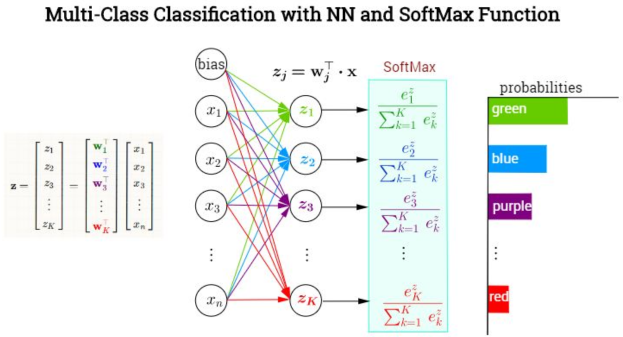
Regression

- ➢ for each label (output value) we use one output neuron
- ➢ the **activation function** of the output neuron has to be **linear**
- ➢ other activation functions are possible, but limit the possible output value range (e.g. sigmoid [0,1])

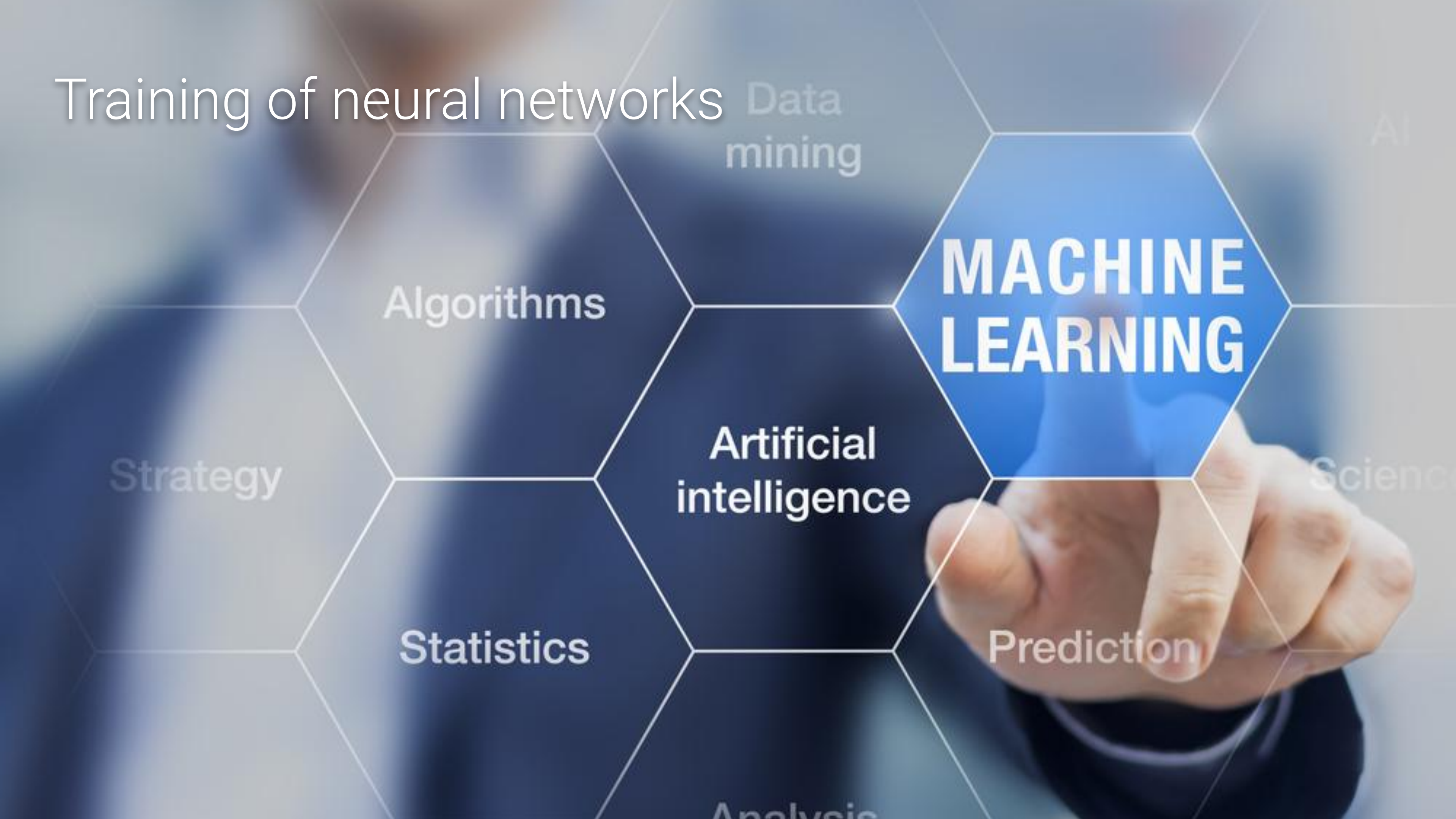


Classification

- ➢ for each class we use one output neuron
- ➢ for single label classification we use the softmax activation function. Each neuron delivers a probability value for one class.
- ➢ for multi-label classification we use the sigmoid activation function
- ➢ the number of classes cannot be changed without retraining of the model



Training of neural networks



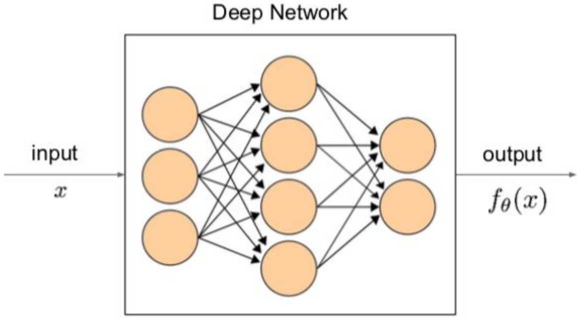
Loss functions (cost, objective, error)

How to evaluate the performance of a network?

Loss L is a function of **distance** between the expected label (Y) and the estimated label ($f_\theta(X)$)

loss is 0 if the output is correct

loss is not 0 if the output is wrong



For **training** of a network, the optimizer **minimizes** the loss function

labels (ground truth)

input

$$\mathcal{L}(w) = distance(f_\theta(x), y)$$

error

parameters (weights, biases)

https://www.kdnuggets.com/2018/04/right-metric-evaluating-machine-learning-models-1.html

https://telecombcn-dl.github.io/2017-dlai/

Loss functions

Different loss functions are available:

- mean absolute error
    - constant derivative
    - not suitable

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

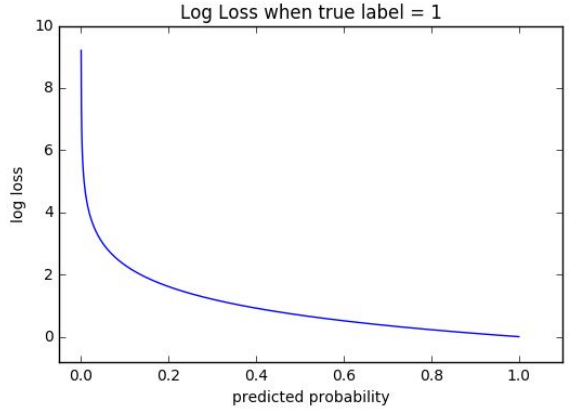
- mean squared error
    - derivative of 2*w
    - accelerated learning

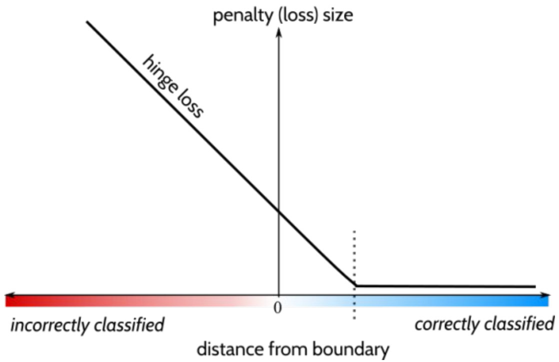
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

- cross entropy loss
    - supports fast learning
    - ignores almost wrong results

$$CE = -\sum_{x} p(x)\, \log q(x)$$

- hinge loss



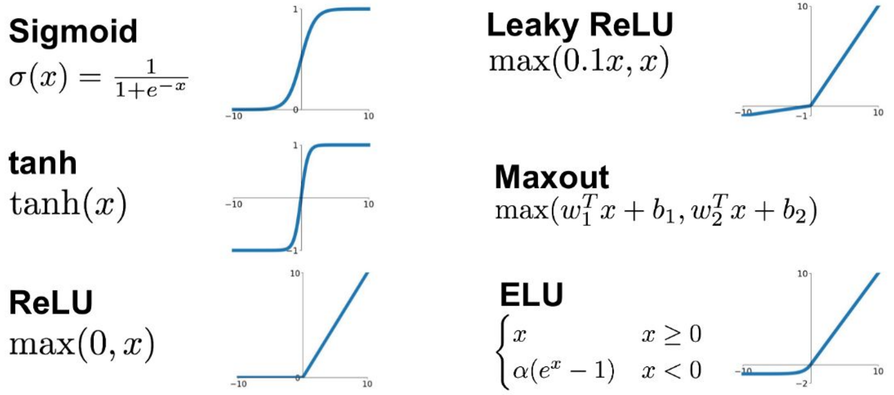


http://rohanvarma.me/Loss-Functions/
https://towardsdatascience.com/visualising-relationships-between-loss-activation-functions-and-gradient-descent-312a3963c9a5
https://math.stackexchange.com/questions/782586/how-do-you-minimize-hinge-loss

Activation functions

Activation functions are an essential part of a neural network. Without a non-linear activation function, neural networks could not learn anything reasonable.

- ➢ need to be differentiable
- ➢ define the output value range of a neuron
- ➢ influence the training process by computational cost
- ➢ influence the training process by flat derivations far out



https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

Train a network with gradient descent

Idea

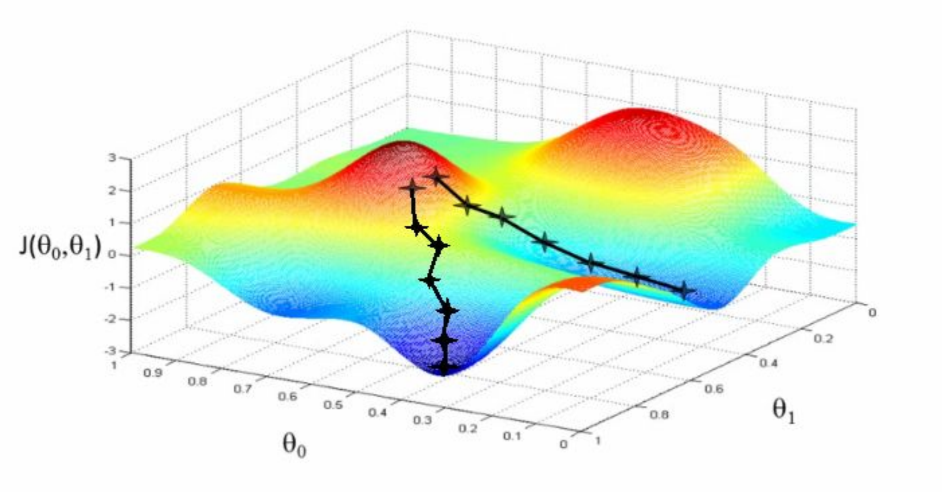
- ➢ change those weights the most, which contribute most to the error (loss)

What we need

- ➢ we need the **gradients** (derivation) of the loss function with respect to each **weight** in the network

How does it work

- ➢ calculate the gradient of loss with respect to weights for each layer and change each weight in the layer **weighted** by the **gradient** and a **learning rate** factor



Gradient Descent minimizes the loss by adapting the weights

Mathematics of gradient descent

We start with the loss function

$$L = L_f\ (\ A(\ sum(\ X * W\ )\ )\ ,\ Y\ )$$

We need the gradient (partial derivative) of loss with respect to each weight in the layer below (chain rules)

$$\frac{\vartheta L_f}{\vartheta w_i} = \frac{\vartheta L_f}{\vartheta A} * \frac{\vartheta A}{\vartheta sum} * \frac{\vartheta sum}{\vartheta w_i}$$

$$A = \frac{1}{1+e^{-sum}}\ (sigmoid) \qquad \frac{\vartheta A}{\vartheta sum} = A(sum)\ *\ (\ 1 - A(sum))$$

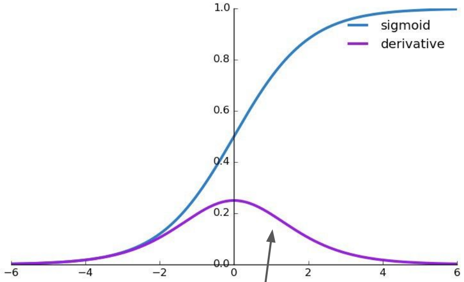
$$L_f\ =\ \Sigma \tfrac{1}{2}\ (y - A)^2 \qquad \frac{\vartheta L_f}{\vartheta A}\ =\ -\ (y - A)$$

$$\frac{\vartheta L_f}{\vartheta w_i} = (y - A) * A(sum) * (1 - A(sum)) * x_i$$

input x

error

derivation of activation function



derivation of activation function

More details on gradient descent

Once the gradients of the weights are calculated, the **weights are changed** by the **negative gradient** times a **learning rate** factor.

Gradient descent for **hidden layers**

- ➢ the previous formulas hold for weights of the output neurons
- ➢ for weights of hidden layer neurons it has to be taken into account that this weight influences **many outputs**. So the gradient depends on the error at all of the nodes this weighted connection can lead to.
- ➢ as an intuition it can be assumed that the loss (error) is scaled down by the weights of the upper layer (vanishing gradient)

Depending of the amount of samples we process in one step we distinguish:

- ➢ **batch gradient descent**: all available samples are injected at once. Risk of suboptimal solution. High memory demand.
- ➢ **stochastic gradient descent**: a single random sample is introduced on each iteration. Very slow.
- ➢ **mini-batch gradient descent**: instead of feeding the network with single samples, N random items are introduced on each iteration. Compromise. Mini-batch size needs to be optimized.

https://medium.com/yottabytes/everything-you-need-to-know-about-gradient-descent-applied-to-neural-networks-d70f85e0cc14
https://www.cs.swarthmore.edu/~meeden/cs81/s10/BackPropDeriv.pdf

Loss optimization methods

Vanilla stochastic gradient descent is not optimal. Several improved optimizers exist. Many ideas for optimization come from the concept of **momentum**.

Momentum is created by **adding** a **fraction** of the **weight updates** in the last round to the updates in the next round.

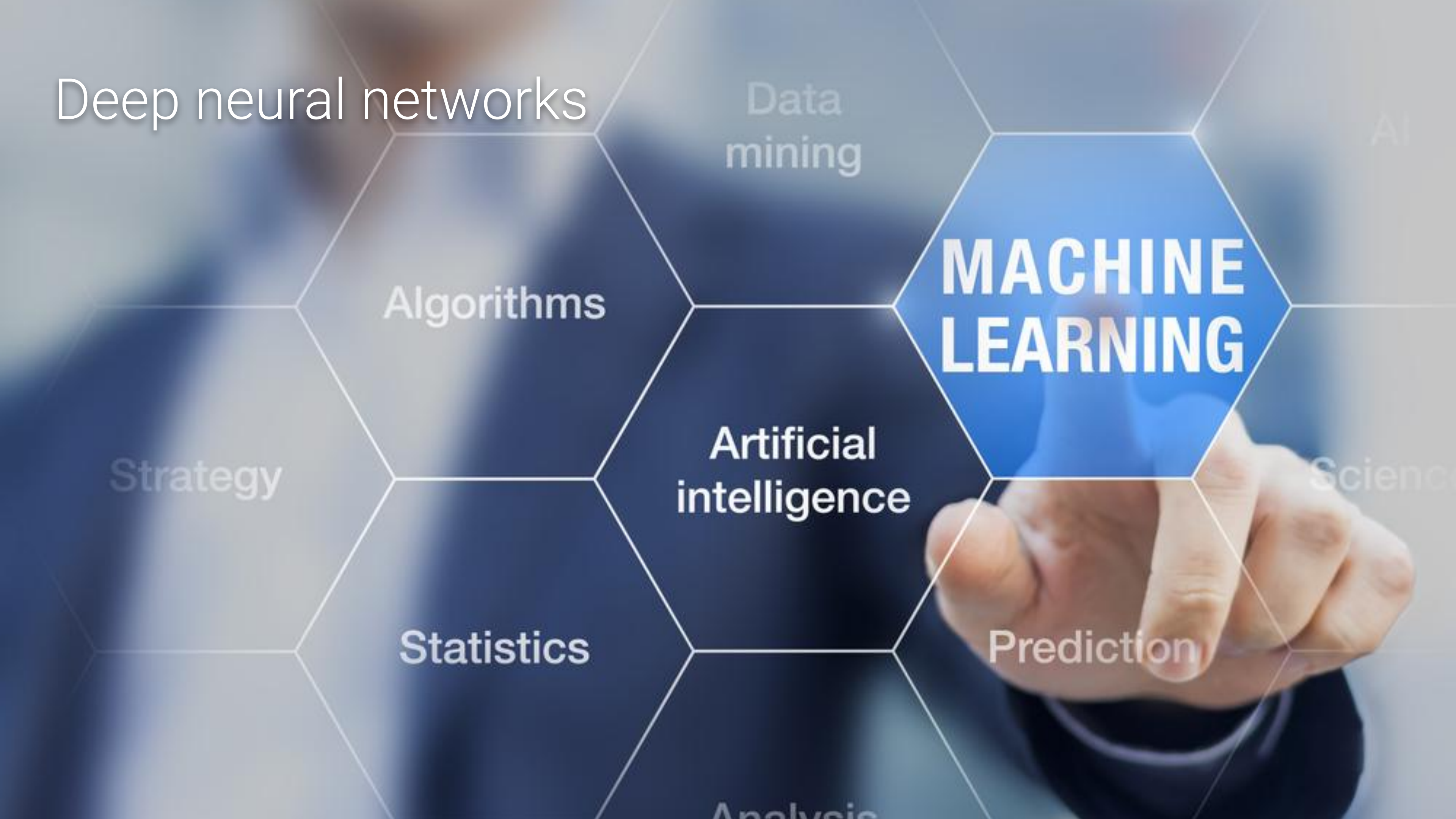
The momentum term **increases** for weights whose gradients point in the **same directions** and **reduces** updates for weights whose gradients **change directions**. As a result, we gain faster convergence and reduced oscillation.

Some examples of optimizers are:

- SGD: not optimal
- **Adagrad**: adaptation of learning rate on frequency of features.
- **RMSProp**: similar to Adagrad
- **Adam**: (Adaptive Moment Estimation). The larger the spread in the history of gradients for each weight, the smaller will be the learning rate.

source: https://ruder.io/optimizing-gradient-descent/

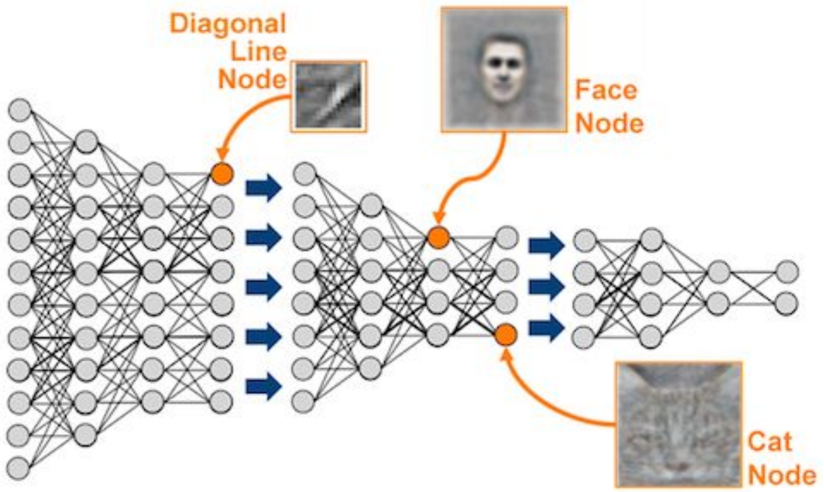
Deep neural networks



Deep neural networks

Idea

- ➢ learning a model of very complex functions requires a large number of neurons. If generalization is desired, then **more of narrow layers** are better suited than wider layers.
- ➢ deeper networks with higher number of layers represent more abstract concepts.
- ➢ training very deep networks requires special measures and tricks.



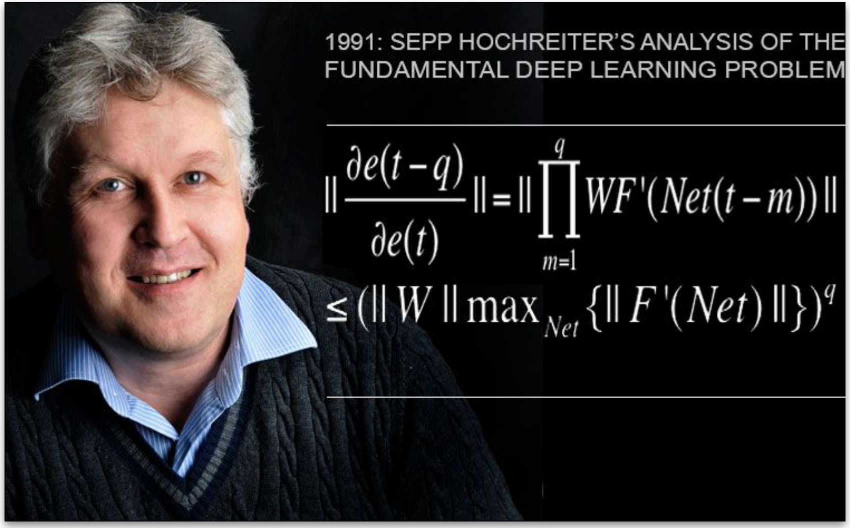
Deep Neural Networks contain **3** or more hidden layers.

https://www.kdnuggets.com/2015/10/top-arxiv-deep-learning-papers-explained.html

Training of deep networks

Problem

- Gradient descent requires the calculation of **gradients** throughout all layers of a network.
- With normal activation functions the error terms are getting exponentially smaller with each layer. This leads to a very slow learning process (**vanishing gradient problem**).



http://people.idsia.ch/~juergen/fundamentaldeeplearningproblem.html

Solutions to the vanishing gradient problem

Multilevel approach

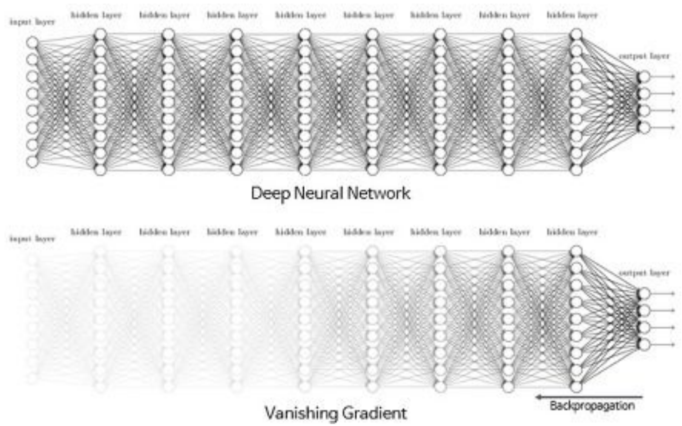
➢ each level of the network is independently pre-trained. Only the fine tuning is done with backpropagation.

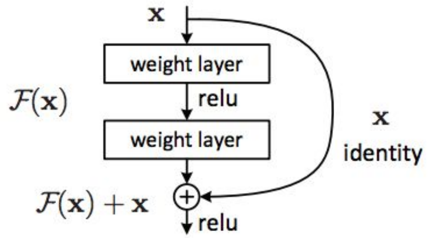
Activation function

➢ ReLu activation function reduces the problem

Residual networks

➢ groups of layers contain a shortcut to pass input information to the output. This allows the propagation of the error without being decreased.





https://chatbotslife.com/resnets-highwaynets-and-densenets-oh-my-9bb15918ee32
https://arxiv.org/pdf/1512.03385.pdf

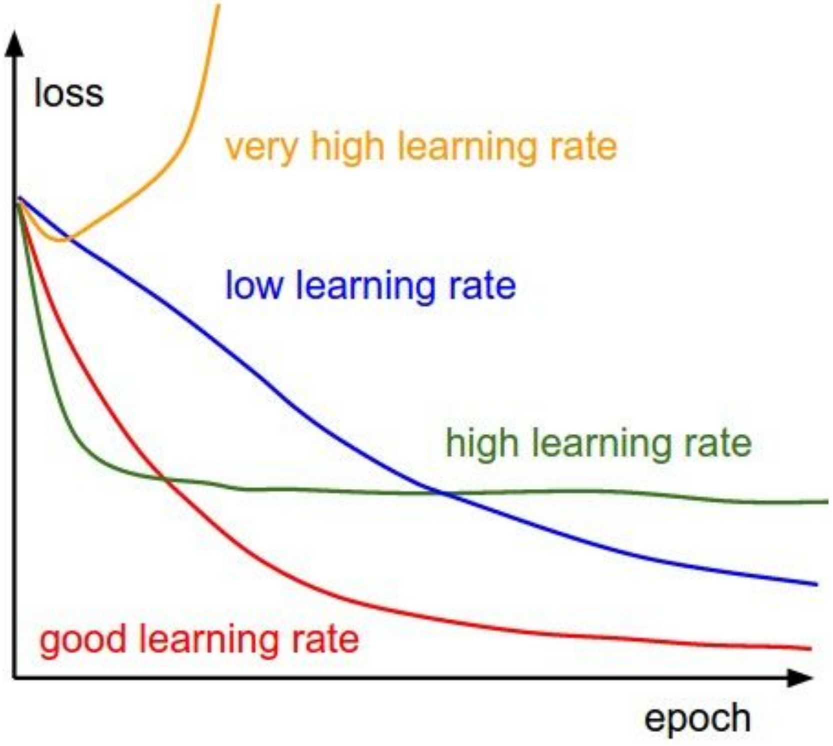
Learning Rate

Background

Gradient Descent works in small steps. The learning rate determines a factor for the applied change per training step.

Small learning rate improves finding the optimal model, but requires more computing time.

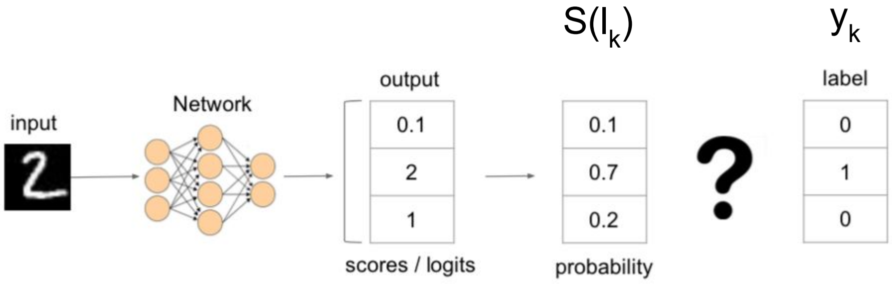
Large step size accelerates training, but can overlook optimal solutions.



Learning Rate can be reduced during a training.

http://cs231n.github.io/neural-networks-3/

Cross entropy loss



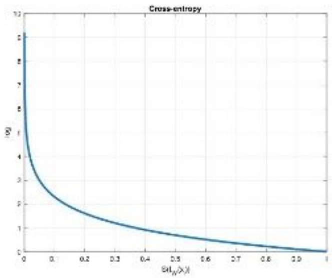
State of the art loss function for **classification**.

➢ Very strong focus on error of correct label.

➢ Error of almost wrong classification is not strongly penalized.

➢ Improved training, but less robust in production use.

$$\mathcal{L}_i = -\sum_k y_k \log(S(l_k)) = -\log(S(l))$$



https://telecombcn-dl.github.io/2017-dlai/

Online vs batched training

Online Training

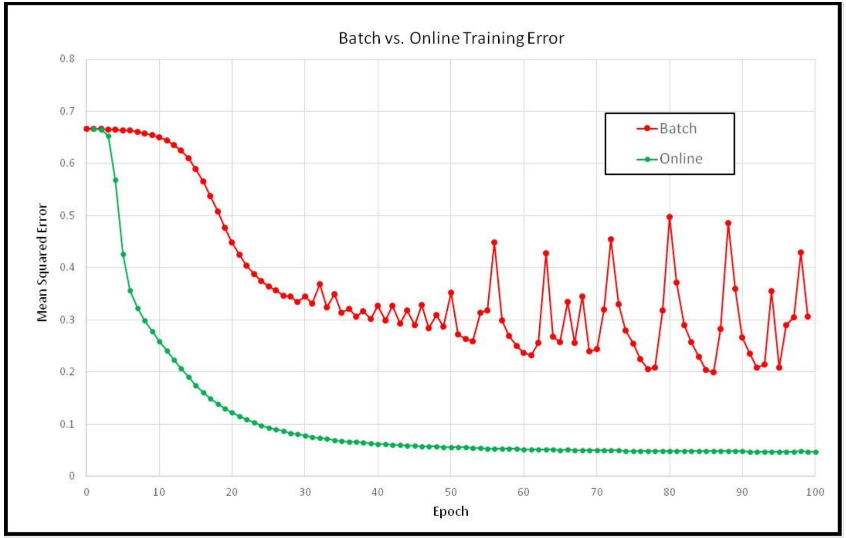
Weights are changed after backpropagation for **each sample**

Batched Training

Training data is divided into batches. Weights are collected (accumulated) after backpropagation and only changed at the **end of a batch**.

So why batched at all?

Support for **distributed training** on GPUs and in some for prevention of overfitting.



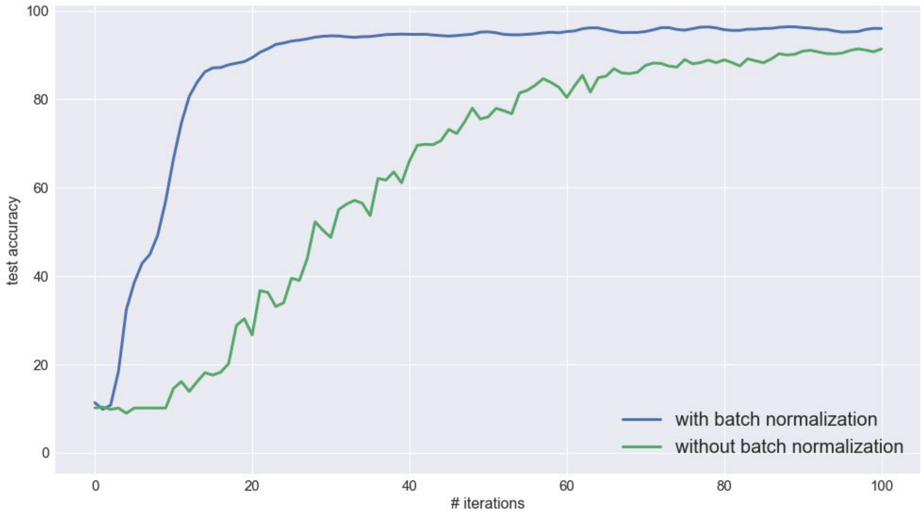
https://visualstudiomagazine.com/Articles/2014/08/01/Batch-Training.aspx?Page=2

Batch Normalization

Normalized features support an efficient training. Inside mini batches, the distribution of the feature values may be distorted, leading to a permanent change of the weights and a less efficient training.

A possible countermeasure is to add a batch normalization layer. This layer learns the distribution and corrects it to mean 0 and variance 1.

The learned parameters are stored and maintained as sliding average values. They have to be stored for use in prediction situation.



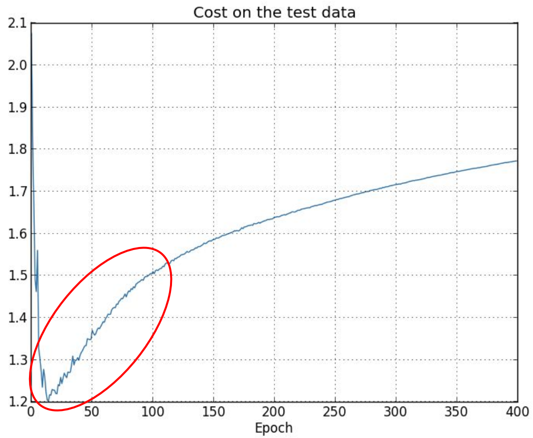
https://towardsdatascience.com/how-to-use-batch-normalization-with-tensorflow-and-tf-keras-to-train-deep-neural-networks-faster-60ba4d054b73
https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/

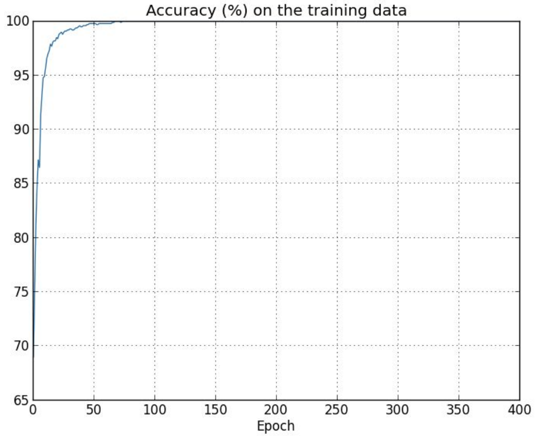
Overfitting

With overfitting the network learns to **reproduce** the trainings data, but does not learn generalization of properties.

Overfitting can be avoided by

- extension of training data
- early stopping
- dropout layer
- weight penalty L1 L2
- reduction of size of network



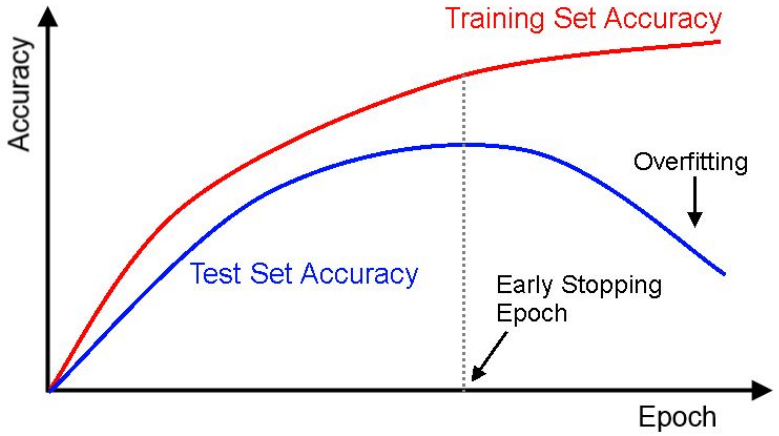


https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

Early Stopping

Idea

- ➢ Abort the training before an overfitting occurs. Requires a measurement criterion
- ➢ Separate the data record into a **training set** and a **test set**.
- ➢ Train exclusively with the training set.
- ➢ Periodically check the accuracy of the test set. When the accuracy starts to get worse, stop the training.



One **epoch** is one round of training using all training samples.

Dropout Layer

In order to avoid the excessive specialization of individual neurons, a certain number of neurons are **ignored** in each training run. The others are improved.

Dropout Rate

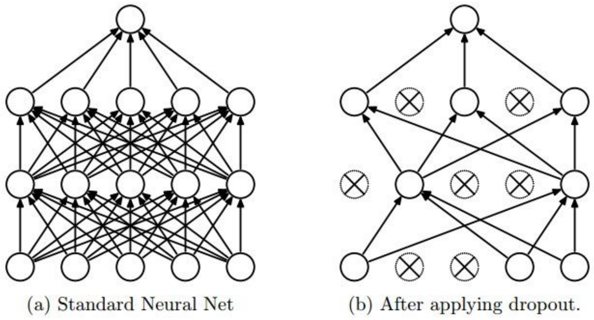
➢ determines the number of neurons that are ignored.

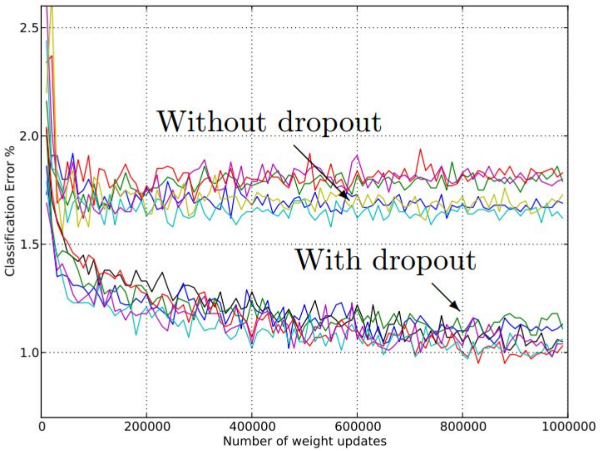
Dropout Layer

➢ Single layer on which dropout is performed.

Input layer

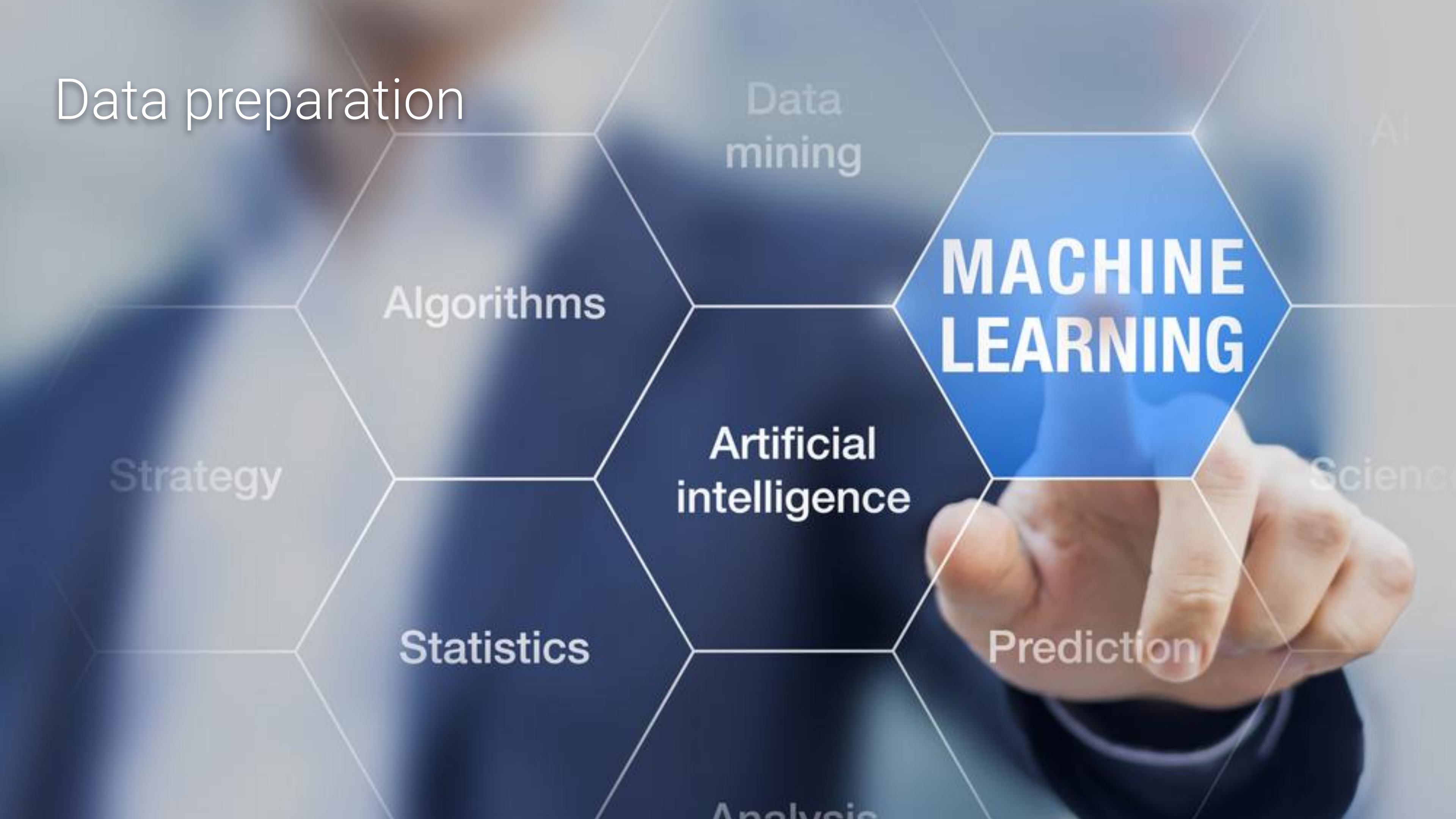
➢ Add noise rather than dropout





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

Data preparation



Normalization and standardization of samples

Background

➢ non-normalized feature values create a bias for features that have a wide range of values.

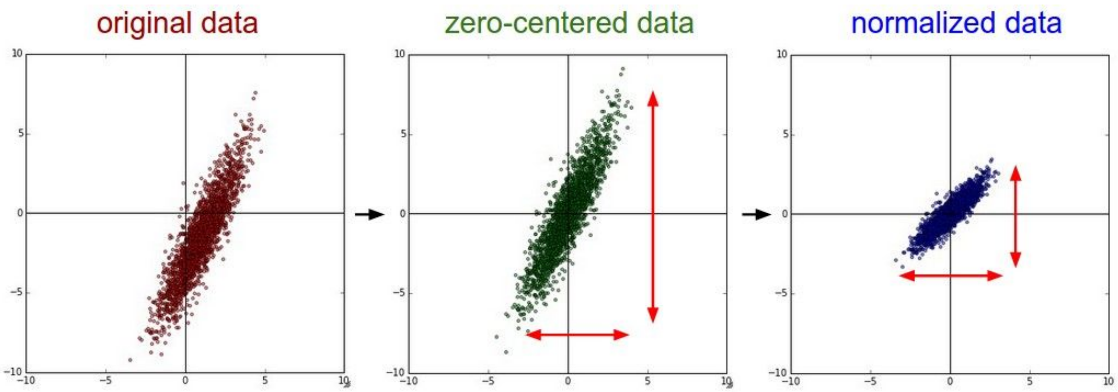
Linear Normalizations

➢ the value range is made uniform for all features. E.g. 0 .. 1

Standardization

➢ transformation to have mean value 0 and standard deviation of 1.

please note the confusion of terms



Standardization supports the learning process for many optimizers.

https://ahmedhanibrahim.wordpress.com/2014/10/10/data-normalization-and-standardization-for-neural-networks-output-classification/

Coding of features

Background

All input values (features) must be represented as real numbers. Class assignments must therefore be converted to vectors.

-1 +1 Coding

if class has only two values

Matrix Coding (**One-Hot**)

$$\begin{bmatrix} French \\ Italian \\ Russian \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation of features X

In some cases it can be helpful to transform an X value to achieve a better differentiation of samples.

e.g. in time series data it can be advantageous to use the difference to the X value of T_{n-1} instead of an X value at time T_n.

One-Hot Coding allows the representation of classes with many elements.

https://ahmedhanibrahim.wordpress.com/2014/10/10/data-normalization-and-standardization-for-neural-networks-output-classification/

Calculation of output classes (labels)

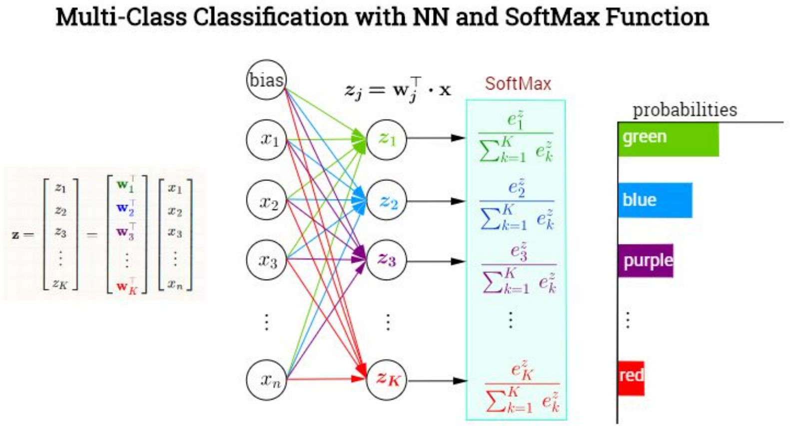
One output neuron is created per class

Multiclass

- ➢ **softmax** as activation function generates a probability distribution over all output neurons. Strong bias for best class.

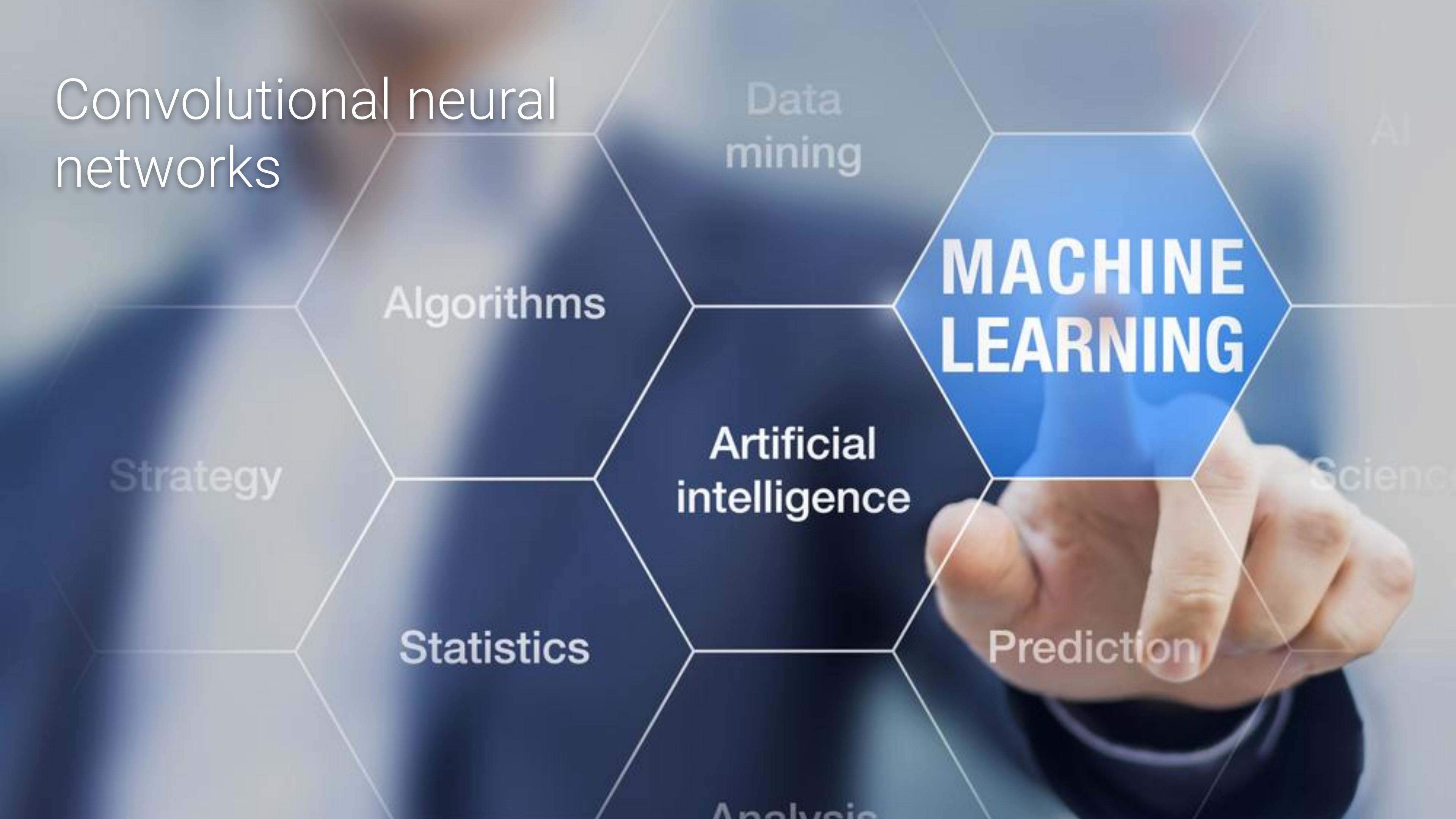
Multilabel

- ➢ **sigmoid** as an activation function generates an independent probability distribution per output.



https://www.dlology.com/blog/how-to-choose-last-layer-activation-and-loss-function/

Convolutional neural networks



Convolutional Neural Networks

Images contain many features that occur more than once, but are distributed over the image area. When using a DNN, you would have to learn each feature for each region of the image.

An object recognition should have **translation invariance**. This means it should detect a learned feature in any region of an image.

Classic feature detectors were written by hand. With a CNN, the optimal **feature detectors** are **learned** from the data in a training process.

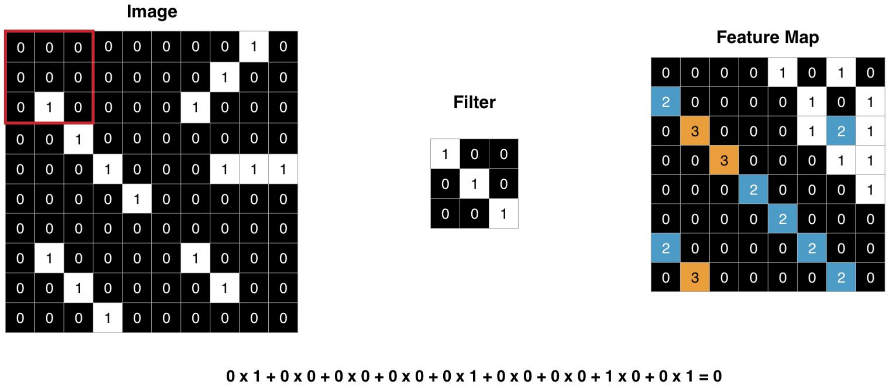
https://stats.stackexchange.com/questions/208936/what-is-translation-invariance-in-computer-vision-and-convolutional-neural-netwo

CNN feature detectors

A CNN learns the important features only once, but can then recognize this feature in any region of the image. To do this, **filters** are moved over all regions of the image (**convolution**) and the results are stored in a feature map.

For **each filter** a **feature map is created** as output. A feature map contains the similarity of the region with the filter values.

The number of filters should be higher in the higher layers of a CNN.



Filters are the feature detectors in a CNN.

https://hashrocket.com/blog/posts/a-friendly-introduction-to-convolutional-neural-networks

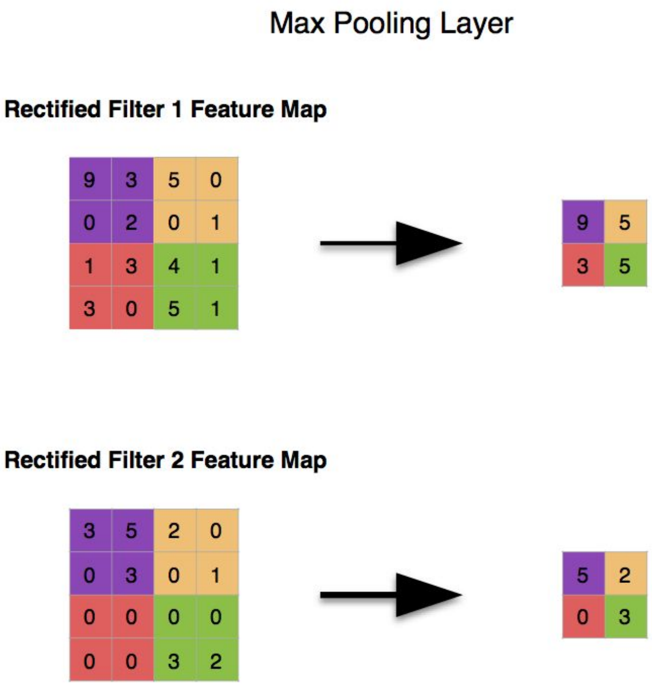
Activation and Pooling Layer

Activation Layer

- ➢ With current training methods, non-linear layers allow better generalization. Therefore a ReLu layer is introduced after each convolutional layer.

Pooling Layer

- ➢ The central method to learn abstract features is **downsampling**. This allows the network to look at different resolutions and find the same object in different sizes. The pooling layer performs this operation. Filter dimensions and stride determine the operation. e.g. 2x2 and Stride 2. Pooling can also go over all features (global pooling). Pooling is calculated for each channel (color/filter).

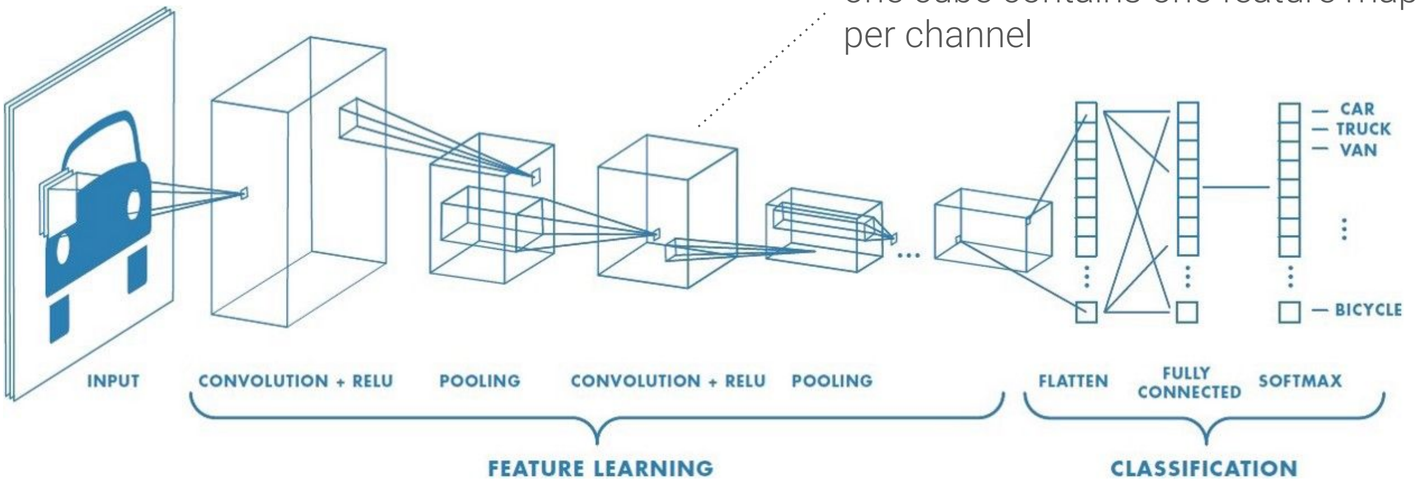


Fully connected layers for classification

Summary of the last feature map to perform a classification

- ➢ the number of output neurons determines the number of classes (fixed)
- ➢ multiple layers increase the capacity for abstraction and generalization.
- ➢ the concepts of a **deep neural network** are applied.
- ➢ for many cases 2 layers are sufficient.

one cube contains one feature map per channel

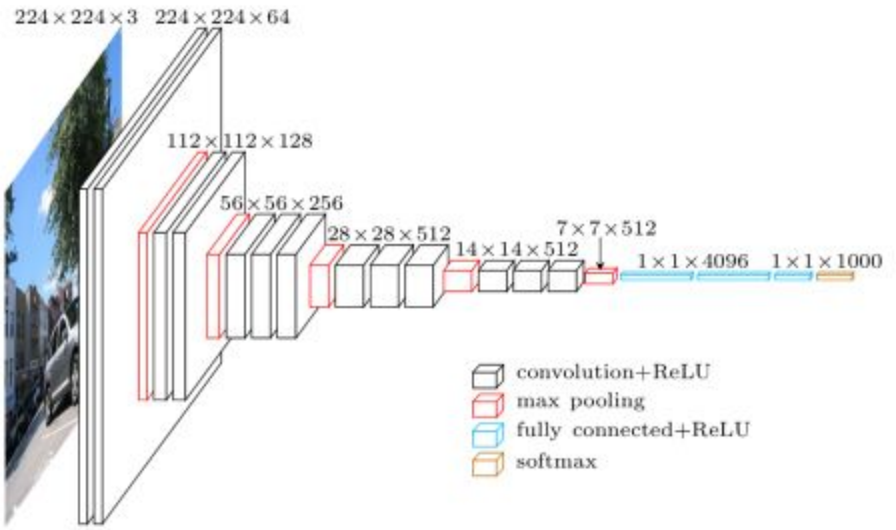


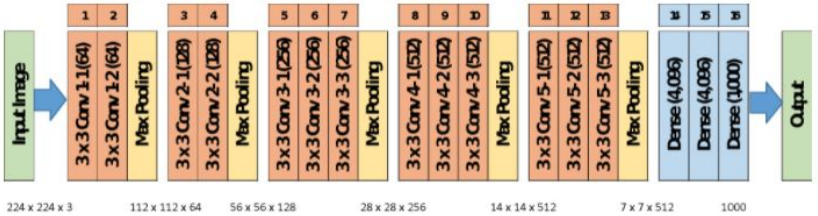
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

The VGG-16 module

This module is used for image recognition. It is widely used in visual classification tasks and as backend for object detection modules.

- about 138 million parameters
- 224x224 pixel input
- 19 weight layers
- introduced the idea of **stacked convolution layers** with **small filter size**. This provides a more discriminative decision function due to multiple stacked non-linear activations.



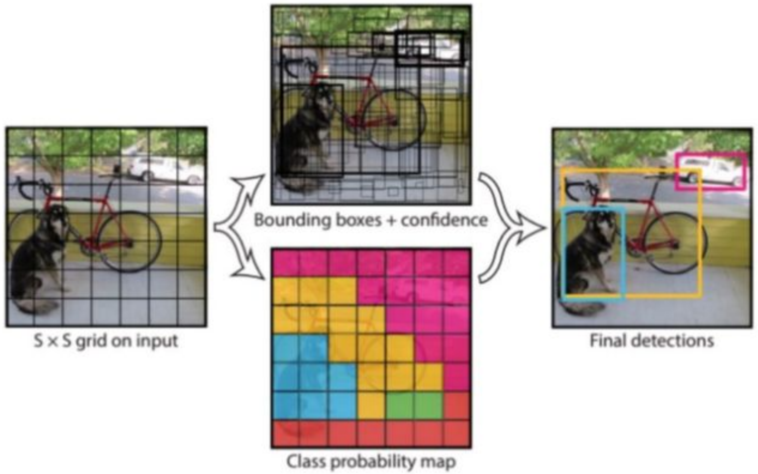


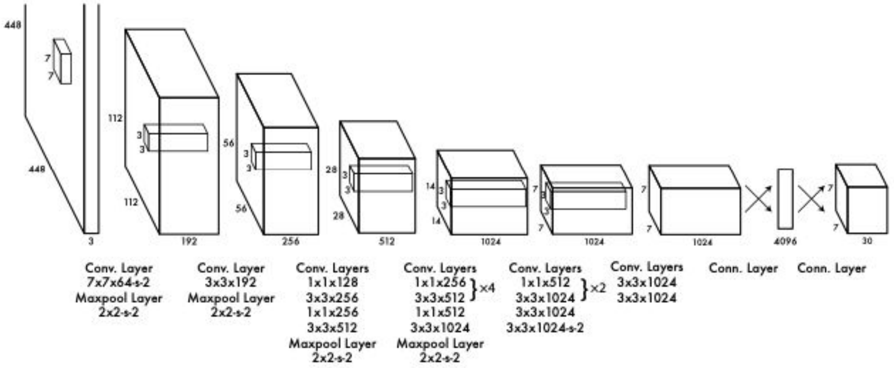
https://arxiv.org/pdf/1409.1556.pdf
https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615

Yolo object localization

Yolo (you only look once) is a unique approach for multiple object detection in one single step.

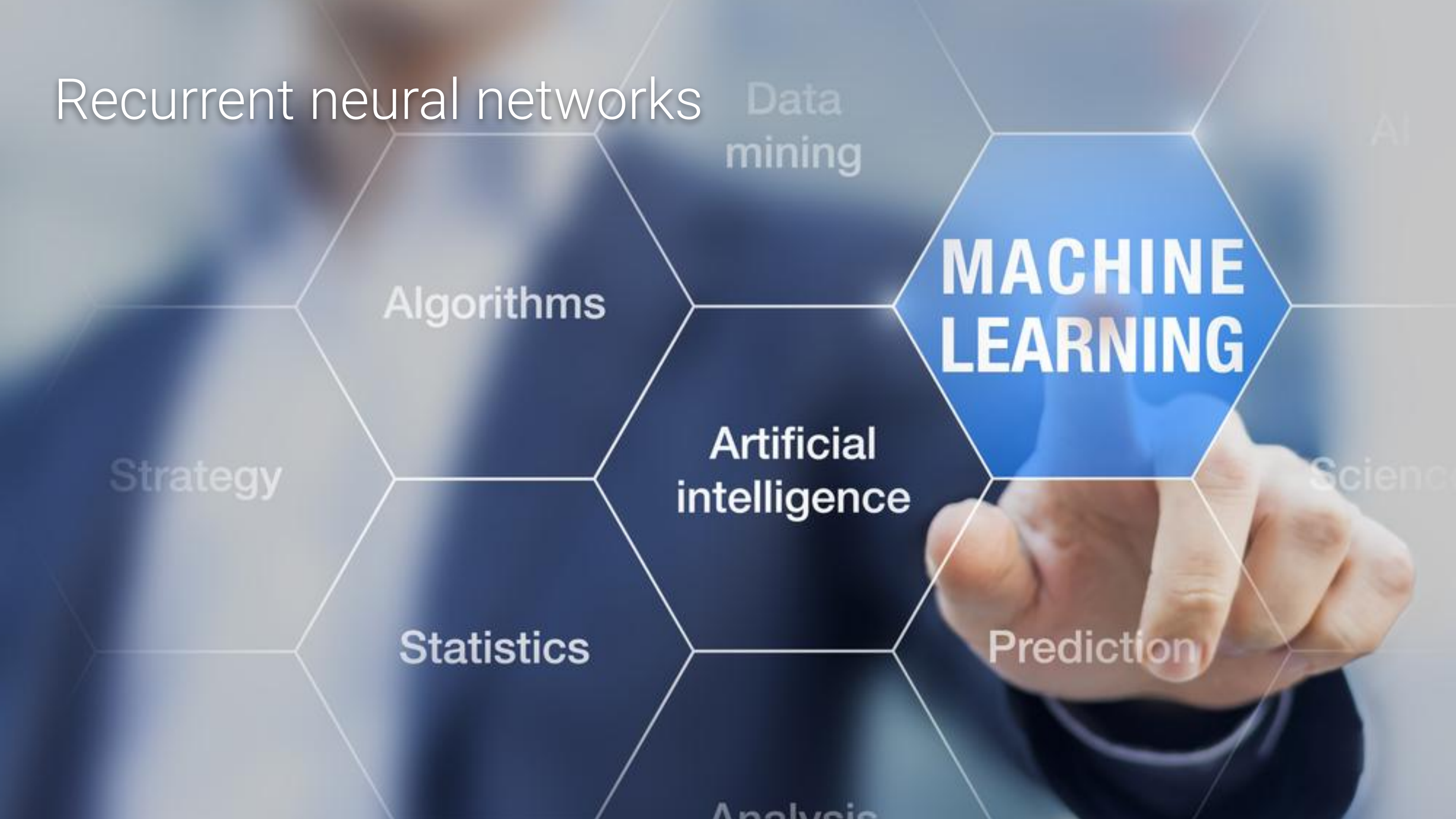
- image is subdivided in SxS **grid elements**
- for each grid element it predicts
    - a **bounding box**
    - a **class**
    - a **confidence** value collected in an output group
- multiple output groups are allocated per grid element to support multiple objects per element
- the CNN part contains 24 convolutional layers followed by 2 fully connected layers





https://arxiv.org/abs/1506.02640
https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

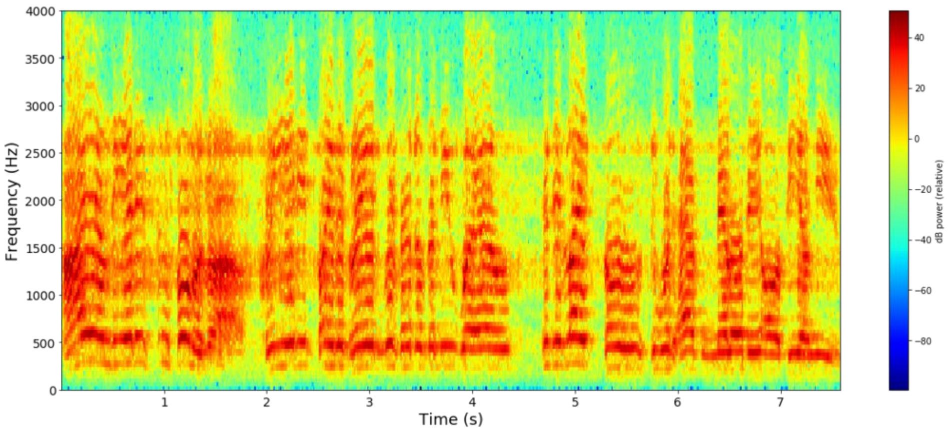
Recurrent neural networks

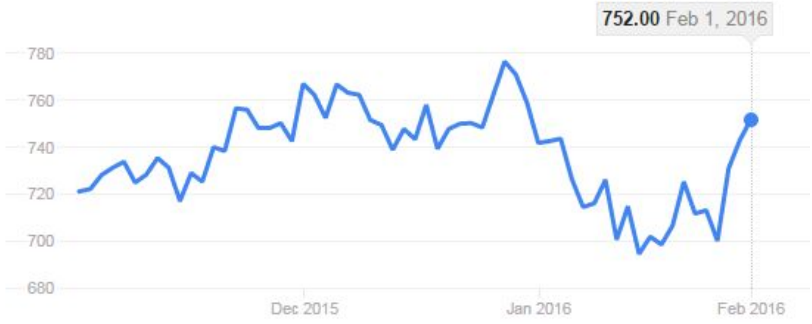


Recurrent Neural Networks

Issue

- ➢ Sequences of data are only learned inefficiently by perceptrons, since there is no memory for past values and therefore the input vector must have the width of all the relevant **context**.
- ➢ This width would be fixed by the architecture. Thus no sequences of variable length could be processed.
- ➢ Sequential data are e.g. audio data, transaction data, stock market prices, language, sensor data usually have variable length features





https://www.lexicon.ai/journal/2017/7/13/tensorflow-rnn-tutorial

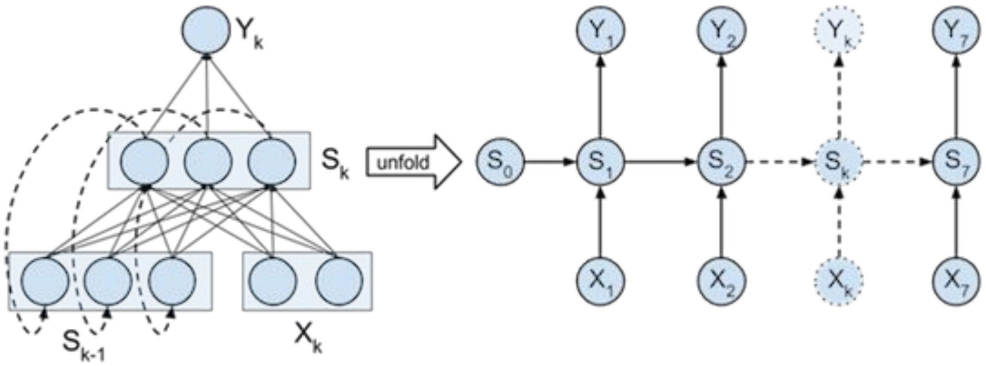
Recurrent Neural Networks

Solution

Networks get a **memory cell** for the previous state, where the state of the last data point is additionally available as an input vector to the normal input data point.

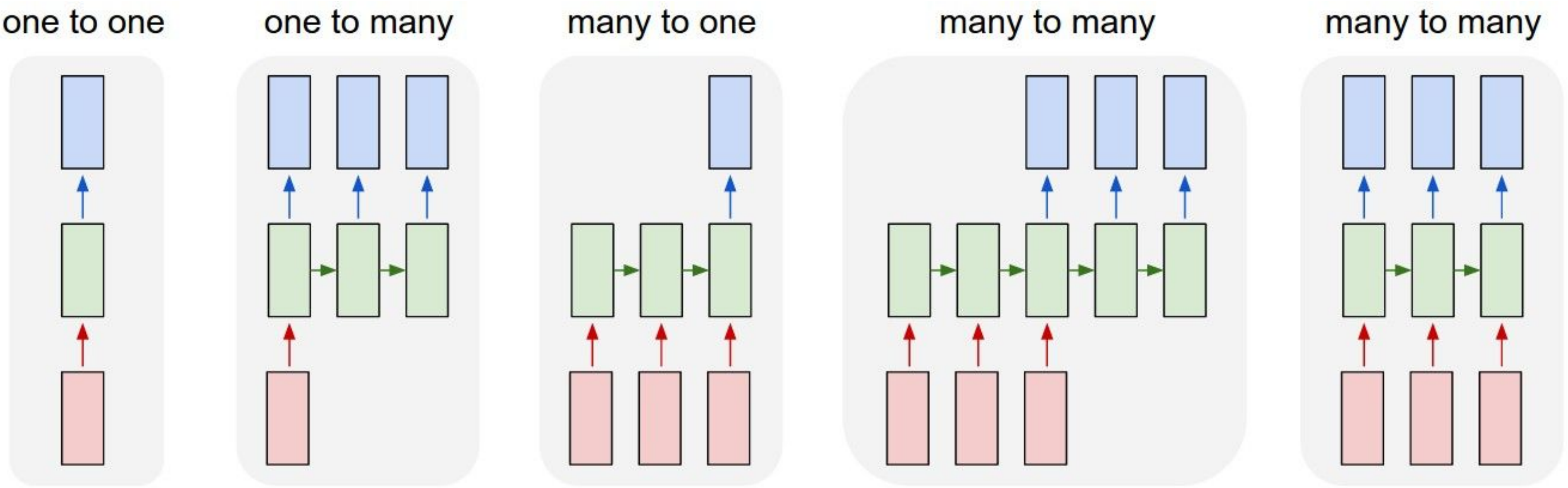
The learning algorithm then determines whether the **history context** plays a role or not.

This allows a network to include the historical context of a state and better learn or classify sequences.



source: http://peterroelants.github.io/posts/rnn_implementation_part01/

Applications of RNNs



classification (MNIST)

Image Capturing

Sentiment Analysis

NLP Translation

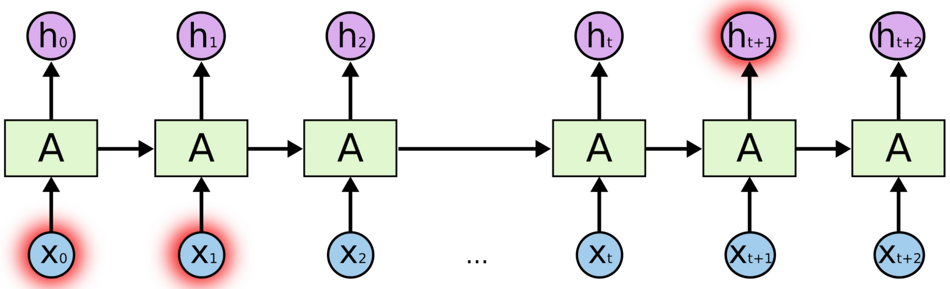
NLP Translation (sync)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

LSTM (by Sepp Hochreiter JKU Linz)

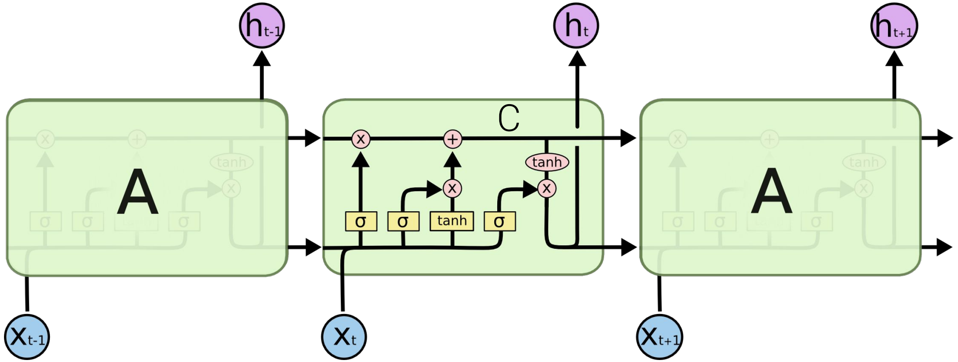
Idea

Extend RNN by gated context state in order to include history, but avoiding vanishing gradient problems. Adds internal state and function gates.



Context funktions

- ➢ **Forget** C
- ➢ **Update** C from Input
- ➢ **Output** h from C and input

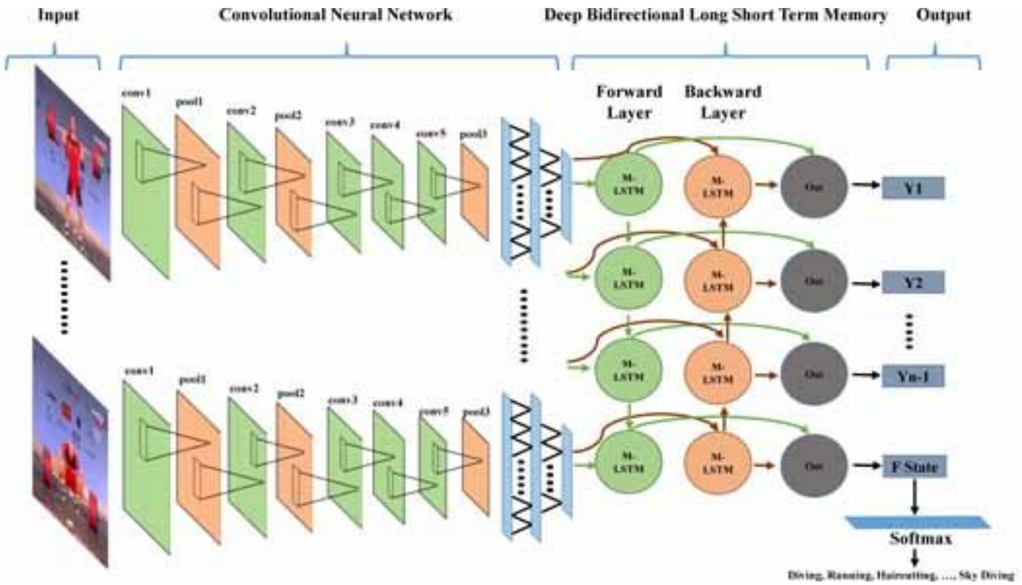


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Video classification using bidirectional LSTMs

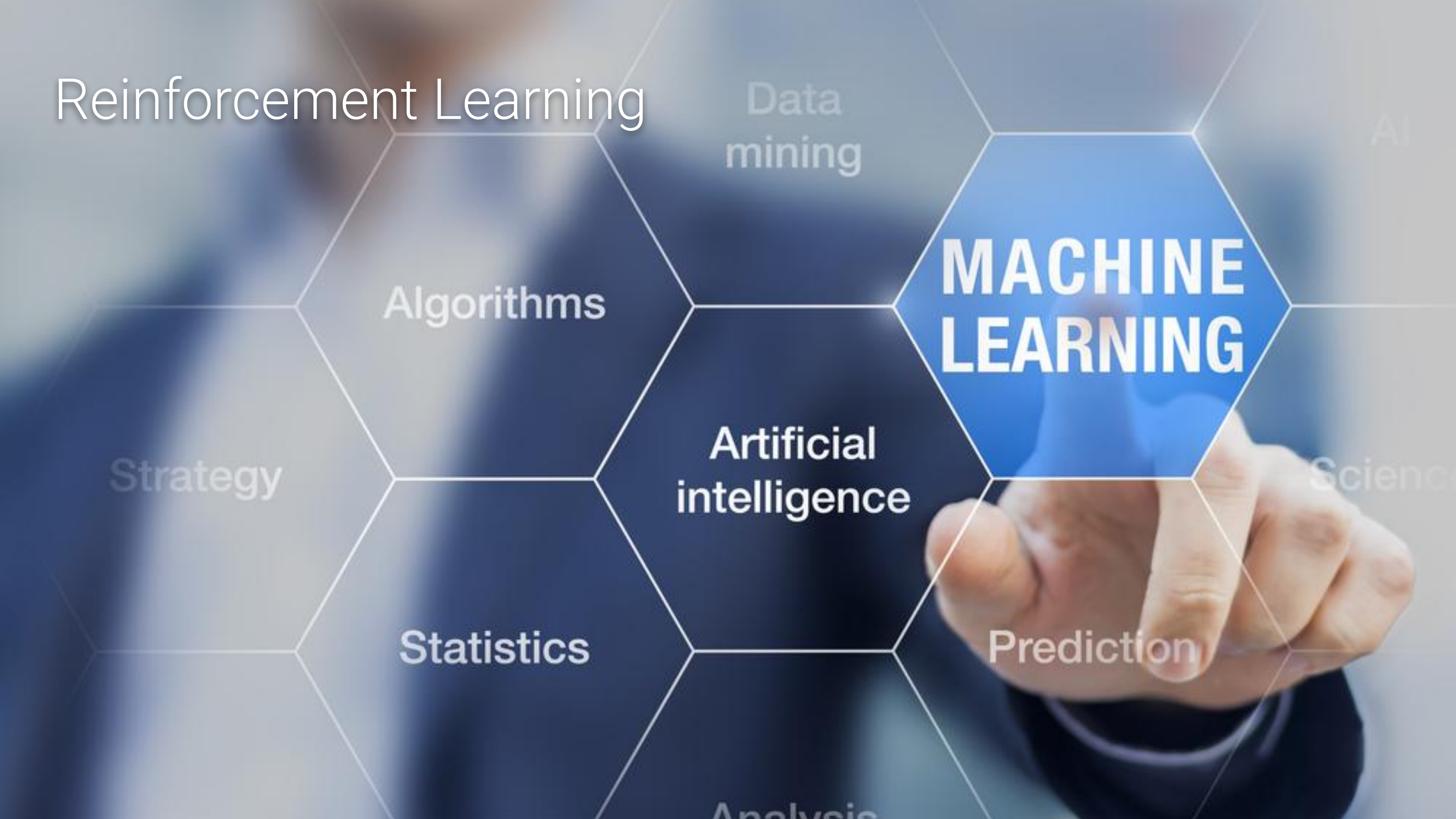
Classification of video sequences using a combination of CNNs and bidirectional LSTMs.

- ➢ deep features are extracted from every sixth frame of the videos
- ➢ the sequential information among frame features is learnt using DB-LSTM network, where multiple layers are stacked together in both forward pass and backward pass of DB-LSTM to increase its depth
- ➢ the method is capable of learning long term sequences and can process lengthy videos by analyzing features for a certain time interval



https://ieeexplore.ieee.org/abstract/document/8121994

Reinforcement Learning

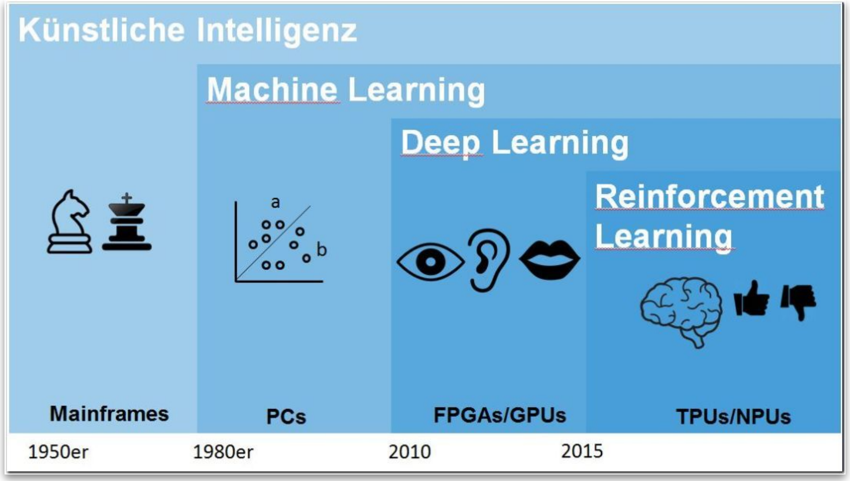


Reinforcement learning

Rather old segment of machine learning.
RL attempts to **optimize** how software **agents** take **actions** in an environment in order to maximize the cumulative **reward**.

Classical approaches regard a system as a **markov decision process** (MDP) assuming that the optimal future action depends only on the current state of the system (no history).

There are mathematical solutions for such MDP systems called **dynamic programming**. However, larger problems are better solved using more modern approaches.



https://www.sigs-datacom.de/ots/2017/ki/2-was-leisten-die-domaenen-der-kuenstlichen-intelligenz.html
https://en.wikipedia.org/wiki/Reinforcement_learning

Modern reinforcement learning

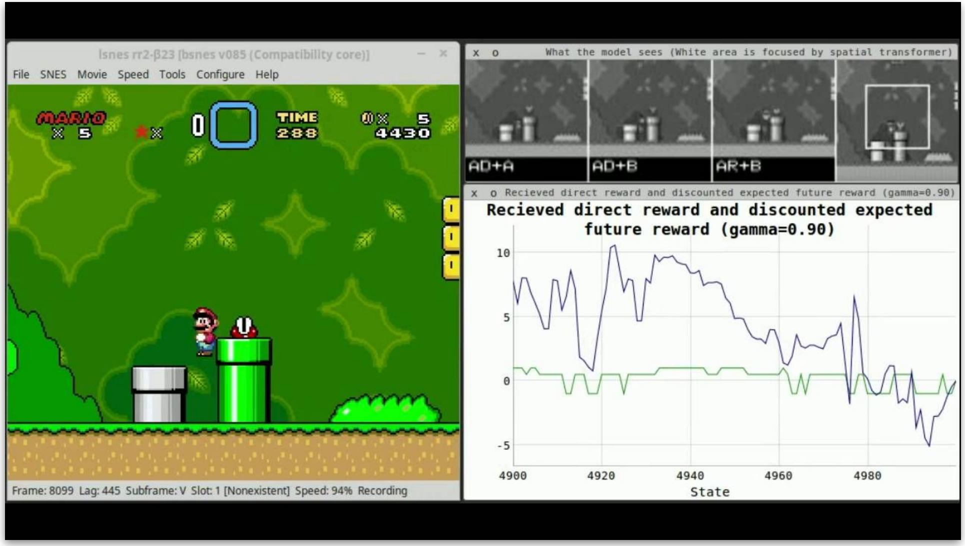
E.g. agent which controls a video game based on the video image pixels and the score count.

Applications

- **control of robots**
- **stock exchange trading**
- **conduct a conversation**
- **strategy in companies**
- **optimization of logistics**

Research

- analysis of the structures learned through RL. E.g. communication between agents.



https://www.youtube.com/watch?v=L4KBBAwF_bE

Elements of reinforcement learning problems

Agent
- ➢ observes state and decides next action

Environment
- ➢ system in which agent acts

State
- ➢ environment state (may be only partially visible). Discrete or continuous.

Action
- ➢ activity the agent can take in a state. Discrete or continuous.

Reward
- ➢ positive or negative feedback from environment. Sparse and time delayed.

Exploration vs exploitation
- ➢ trade off between exploring unknown spaces and optimizing known spaces

Value function V(s)
- ➢ accumulated highest returns of all actions starting from state s

Q function Q(a,s)
- ➢ accumulated highest returns of all actions starting from state s taking action a

Policy π
- ➢ state to action mapping which determines the **best action** in each state

Model
- ➢ model of environment

Discount factor γ
- ➢ used in value function to penalize rewards which are further in the future

Some types of optimization methods

Value iteration

➢ iterative approximation of value function. Calculation of policy after last iteration.

Policy iteration

➢ update policy with greedy estimate of value function after each iteration

Policy gradient

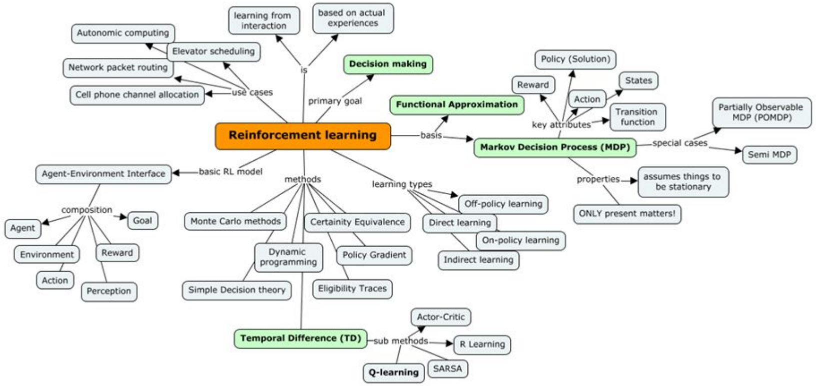
➢ maximize rewards by promoting actions with high rewards in policy by looking at the gradient

Q-Learning

➢ similar to value iteration without regarding the policy. Has extension for exploration.

Deep Q-learning

➢ transfer of the q-learning idea to use a neural network for approximation of the Q function.



https://towardsdatascience.com/self-learning-ai-agents-part-ii-deep-q-learning-b5ac60c3f47
https://medium.com/@jonathan_hui/rl-reinforcement-learning-algorithms-quick-overview-6bf69736694d
https://medium.com/datadriveninvestor/reinforcement-learning-rl-simplified-87b4aa74b85b
https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#what-is-policy-gradient

Bellman equation

The core idea is an **iterative update** of the value function $V_k(s)$ of each state in rounds by looking only **one step** into the future (**greedy** update).

Greedy update refers to the fact that only **one step into the future** is analysed (from s to s' via action a).

In each round the **value** of each **state s** is updated with

- for each **action a**, the highest value of expectation value over all next states of reward plus discounted value of next state
- expectation using probability of changing to state s' from s with action a

The iterative update is repeated in **episodes**.

value of next state s' times a discount factor

probability of change to state s' from state s with action a
(system level, not policy)

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

new value V_{k+1}

resulting reward when switching from state s to state s' with action a

iterate over all actions of state

Q-learning concept

Idea

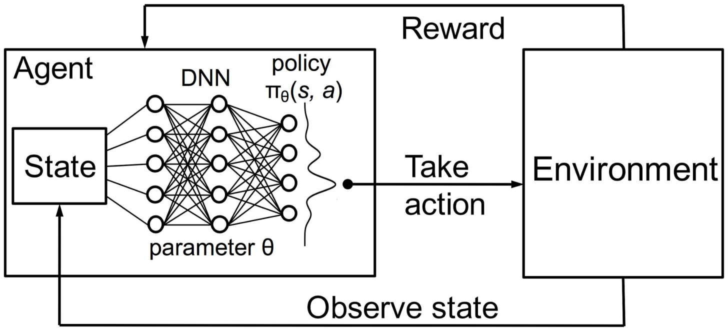
- **optimize** a **Q-function** based on repeated experiments in a system
- the Q value of a state-action pair is stepwise **adapted towards the best reward** value which can be reached from the given state.

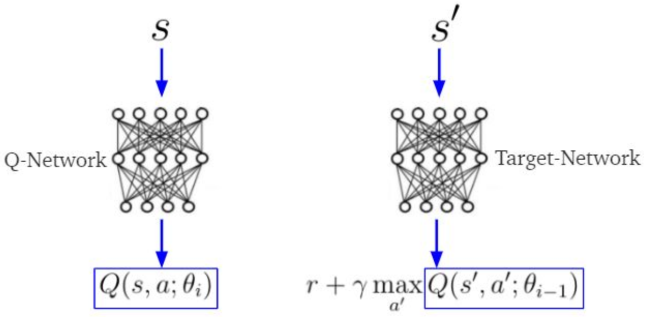
$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

source: https://en.wikipedia.org/wiki/Q-learning

Deep Q-learning

- a neural network (**Q-network**) is trained to predict the Q function Q(s,a).
- a second network (**target network**) is used for calculation of Q values during an experience phase (not changed during phase).
- experiments are run in episodes to collect experience tuples.
- observed experience **tuples of <s, s', a', r>** are stored for later use in training.
- the Q-network determines the action of a state.
- actions need to be **discrete**., otherwise the max operation to find the best action is not calculable.
- a special mechanism randomly selects other actions for a state to allow the agent to **explore** new states.





https://towardsdatascience.com/self-learning-ai-agents-part-ii-deep-q-learning-b5ac60c3f47
https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc
https://ai.intel.com/demystifying-deep-reinforcement-learning/
http://people.csail.mit.edu/hongzi/content/publications/DeepRM-HotNets16.pdf

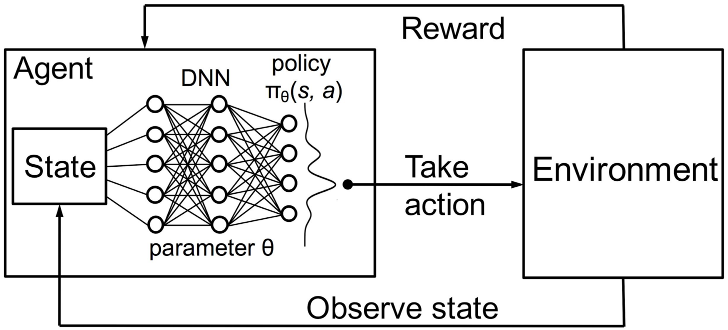
Deep Q-learning

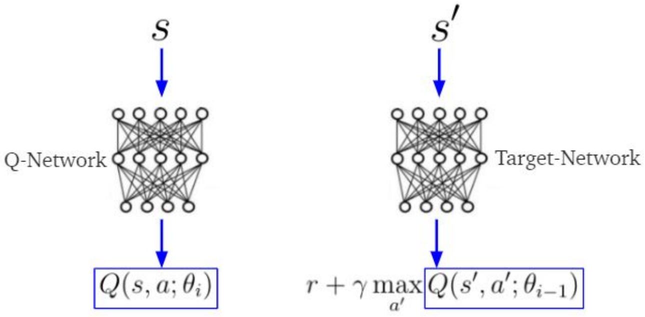
- the training of the Q-network
- the **loss function** is the squared error between the results of the Q-network and the next state Q-value estimation (from target network).

$$L_i(\theta_i) = \mathbb{E}_{a\sim\mu}\left[(y_i - Q(s,a;\theta_i))^2\right]$$

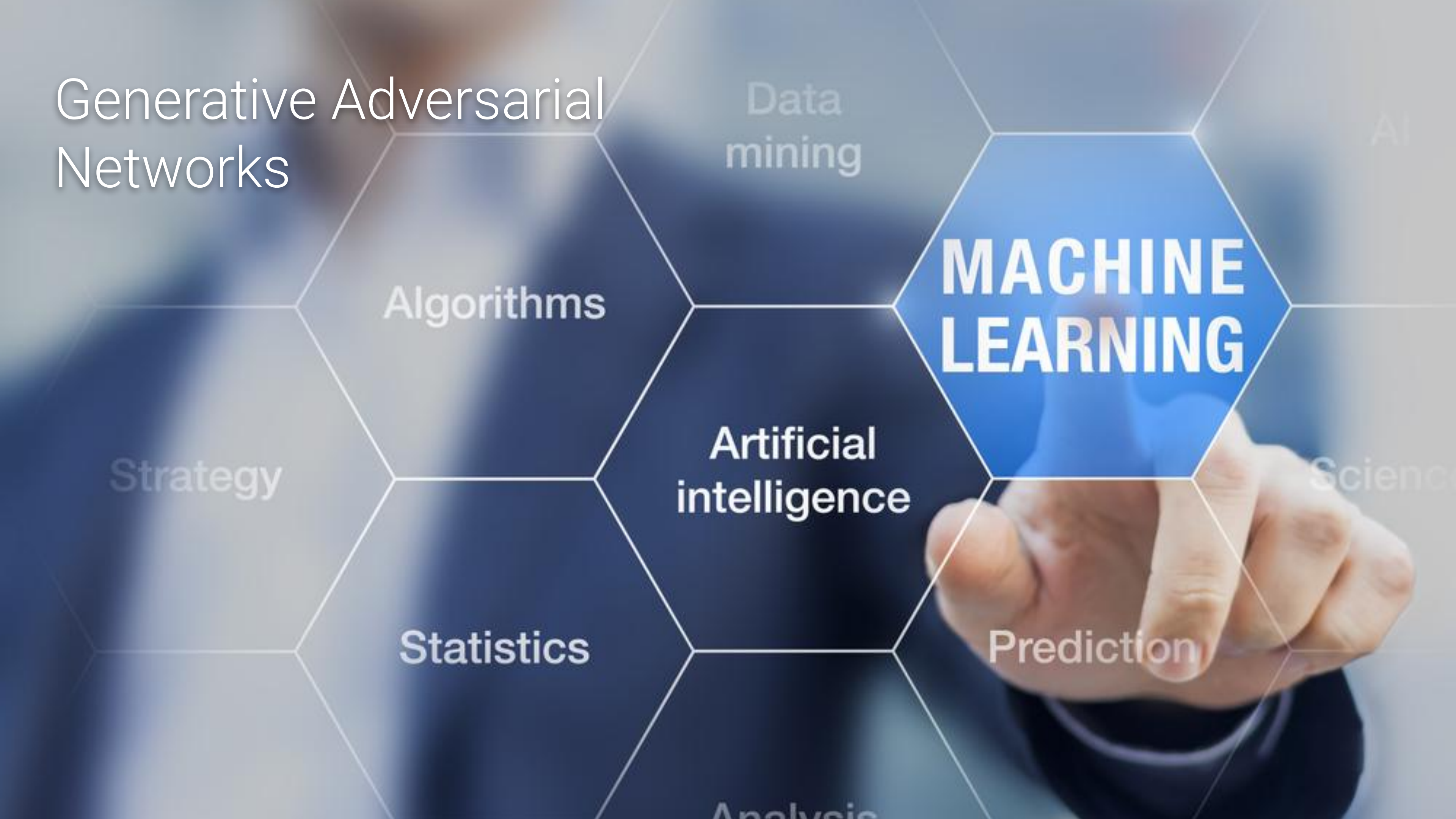
where $y_i := \mathbb{E}_{a'\sim\pi}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | S_t = s, A_t = a\right]$

- for a training step, the experience tuple delivers all values required to fill the loss function as well as the features (state).
- several experience tuples are sampled to build a minibatch for training





Generative Adversarial Networks



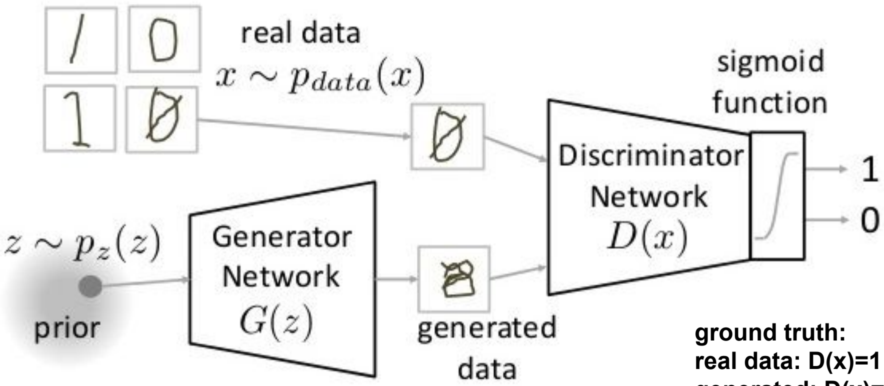
Generative Adversarial Networks (GAN)

Published 2014 by Ian Goodfellow et al.

- ➢ Let two networks learn in competition against each other to create an optimal synthetic output from real data.
- ➢ the generator network **generates** synthetic output,
- ➢ the discriminator network **predicts** the **probability** that the input is from the **real data**.

Training of the two networks

- ➢ the discriminator is trained to maximize the loss *log (D (x))* for real data (D(x) = 1) and *log (1- D(G(z)))* for generated data
- ➢ the generator is trained through the discriminator to minimize *log (1- D(G(z)))* for generated data



ground truth:
real data: D(x)=1
generated: D(x)=0

https://arxiv.org/pdf/1406.2661.pdf
https://www.analyticsvidhya.com/blog/2017/06/introductory-generative-adversarial-networks-gans/
https://medium.com/analytics-vidhya/understanding-gans-deriving-the-adversarial-loss-from-scratch-ccd8b683d7e2

Some types of GANs

DCGANs

➢ are deep convolutional GANs. Further development with better trainability and considerably better image quality.

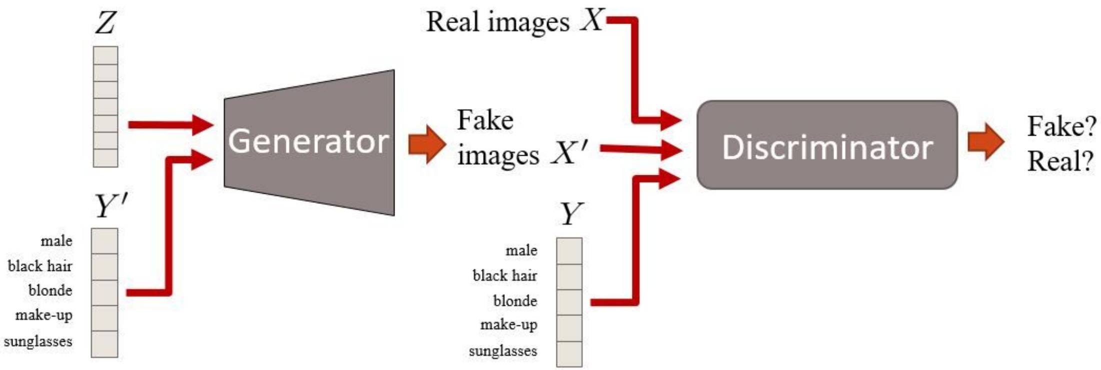
Conditional GANs

➢ use additional information to distinguish the generated image. E.g. hair color, eye color.

Info GANs

➢ automatically develop additional features from the data. These can then be used for generation.

conditional GAN

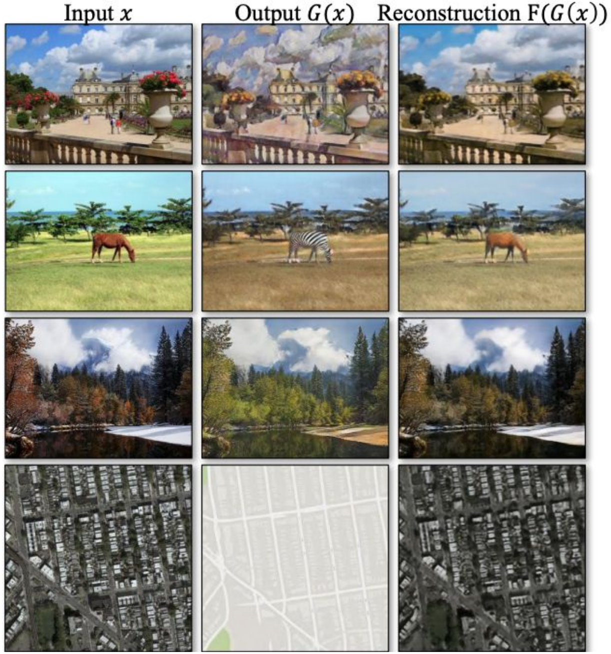


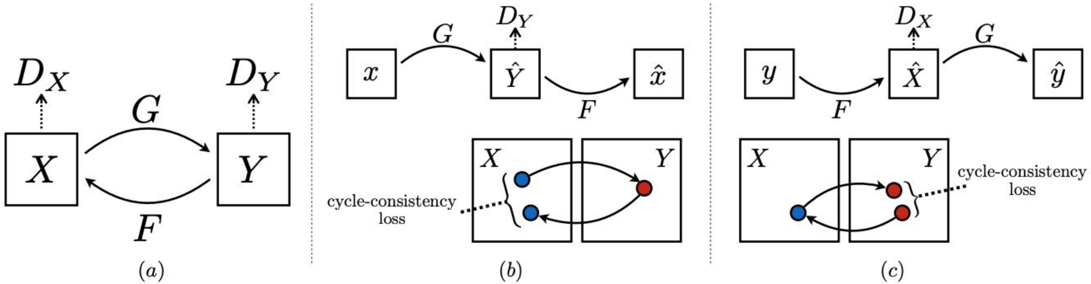
http://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them

CycleGAN

A cycleGAN can transform the style of an image. It uses two generator networks G and F which form a style cycle.

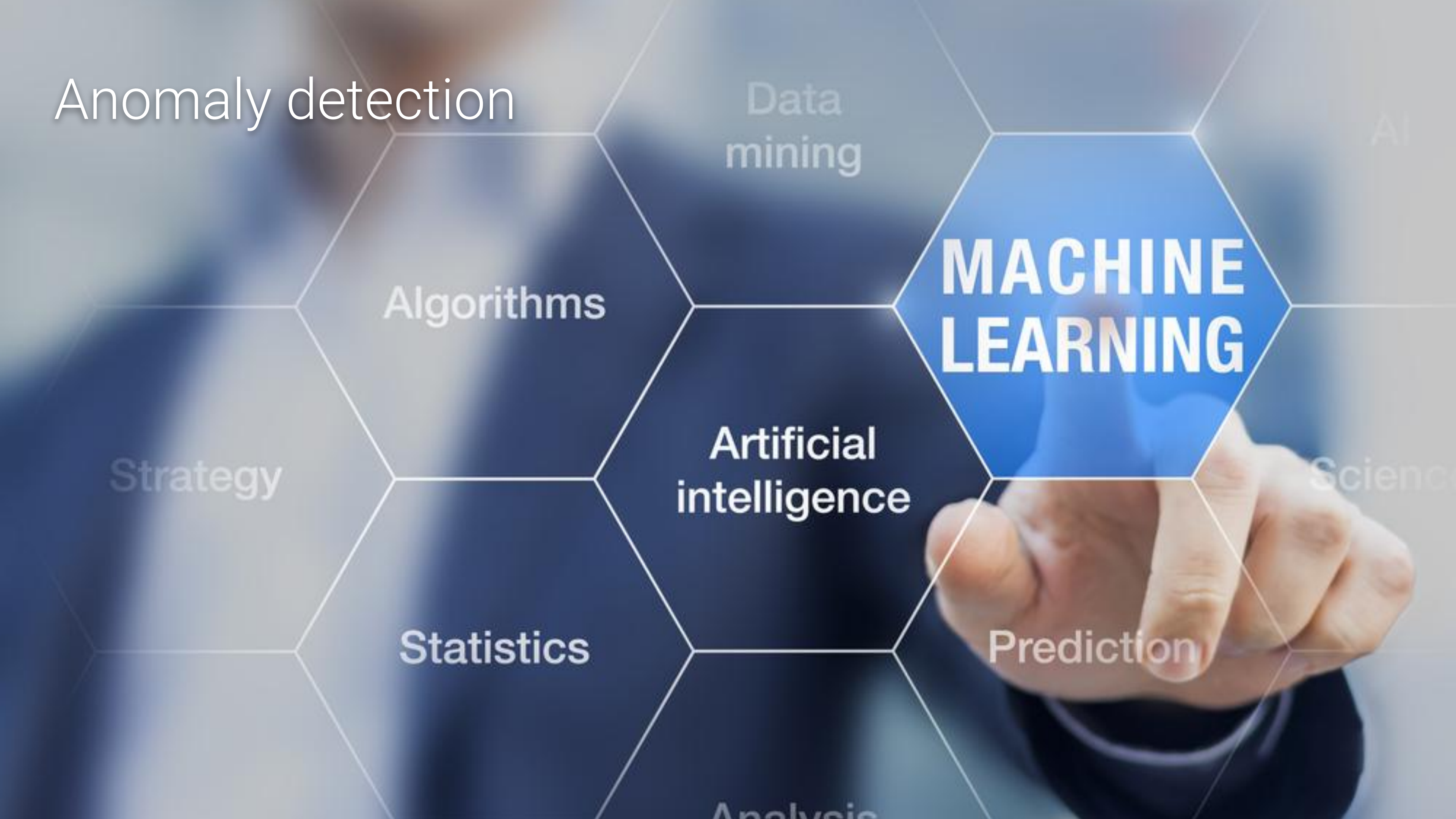
The loss for training is a combination of the GAN loss and a cycle-consistency loss, which ensures that the generated image contains the same content (but a different style).





https://arxiv.org/pdf/1703.10593v7.pdf

Anomaly detection



Anomaly detection

Anomaly detection is concerned with finding deviations from a 'normal' state in data. This normal state is usually not explicitly defined, but it's properties have to be learned from historical data.

There are several approaches for this goal.

- ➢ **outlier detection** (mostly statistical)
- ➢ **clustering** and centroid distance
- ➢ **reconstruction error** (autoencoder)

From perspective of time we distinguish between

- ➢ states deviating from normal
- ➢ sequences deviating from normal

The biggest challenge in AD is the reduction of **false positives** (they annoy the customer)

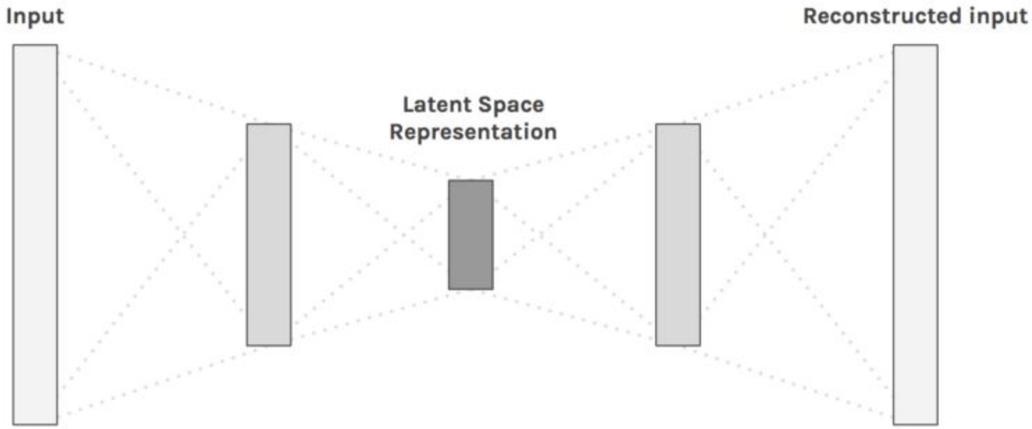
Autoencoder

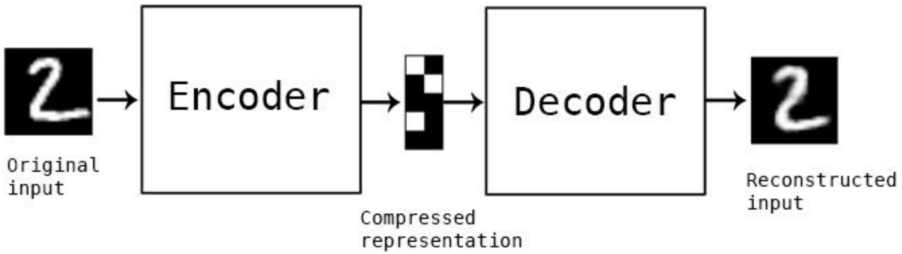
Certain methods require a compact optimal representation of a data set. This representation can be found with an auto-encoder.
Generation of an **compact representation** of the data (e.g. device profiles of consumers)

Generative models
e.g. create image from latent space vector

Reconstruction of input data

Feature learning





https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694
https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Reconstruction error

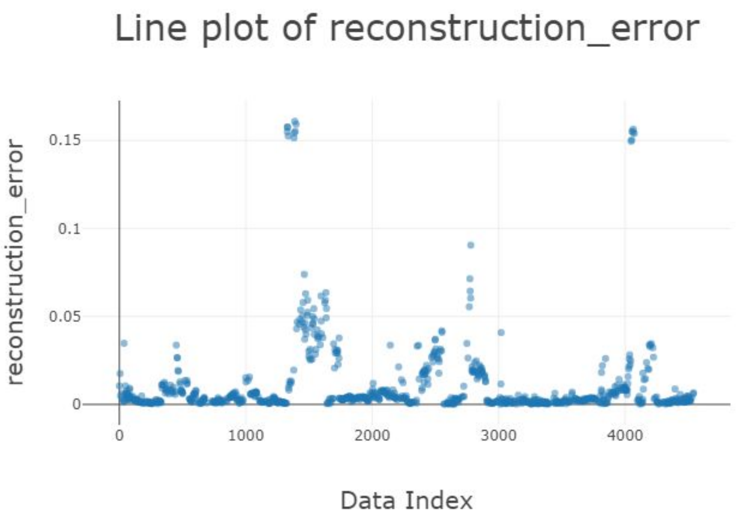
After training an autoencoder to reconstruct the input x, the construction error can be calculated as:

$$error_k = \sum_{i=1}^{m} (\mathbf{x}^i - \hat{\mathbf{x}}^i)^2$$

The error of reconstruction can be used for:

➢ **anomaly detection**: use the reconstruction error as a signal for anomalies. E.g. high error indicates **unseen** data.

➢ input reconstruction: use the reconstruction error to find **regions with distortions** (e.g. noise).



https://blog.goodaudience.com/neural-networks-for-anomaly-outliers-detection-a454e3fdaae8

Use cases of anomaly detection

Avoidance of delays in supply chain

➢ identification of deviations in supply chain KPIs

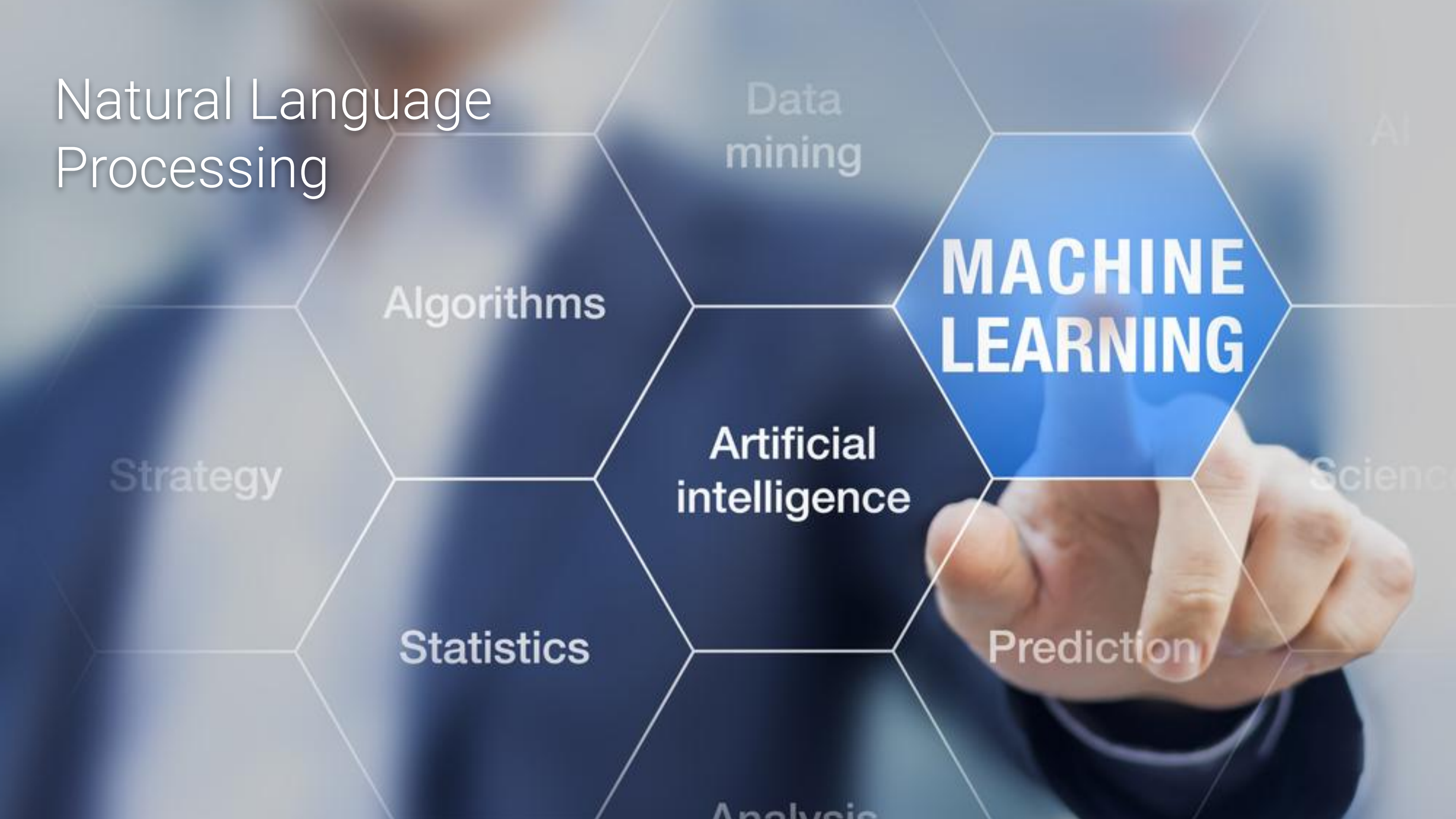
Fraud detection

➢ deviations in transactions

Change in customer behavior

➢ improvement of demand prediction

Natural Language Processing

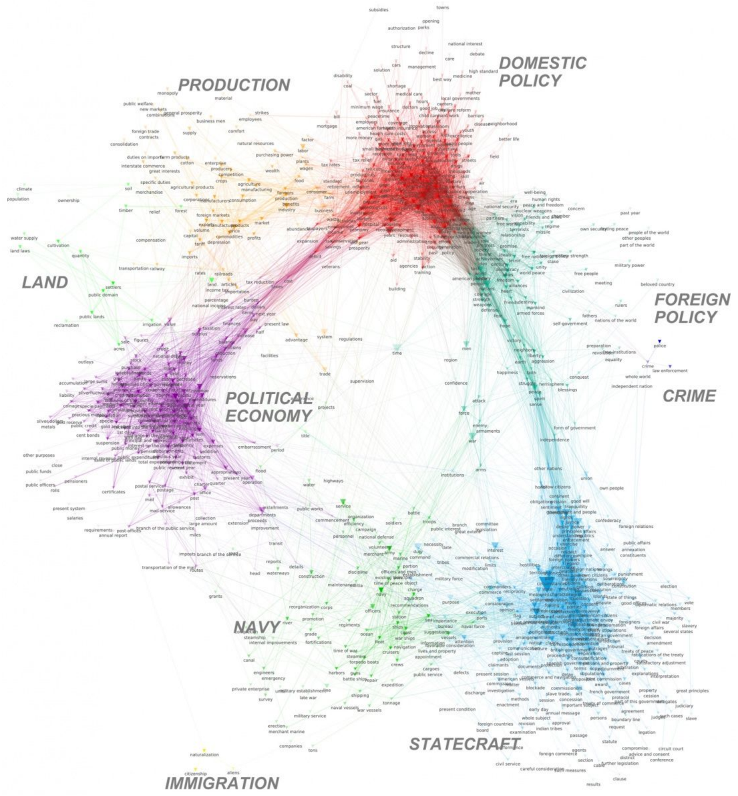


Processing of natural language data

Extract information from text sources. Natural language (NL) is structured, but has many exceptions and ambiguities. Rule based analysis of NL is not practical because of the many exceptions with classical methods.

The following tasks are typical:

- ➢ text/document **classification**
- ➢ document **clustering**
- ➢ entity **extraction** and relation modelling
- ➢ **sentiment** analysis
- ➢ document **summarization**
- ➢ **translation**



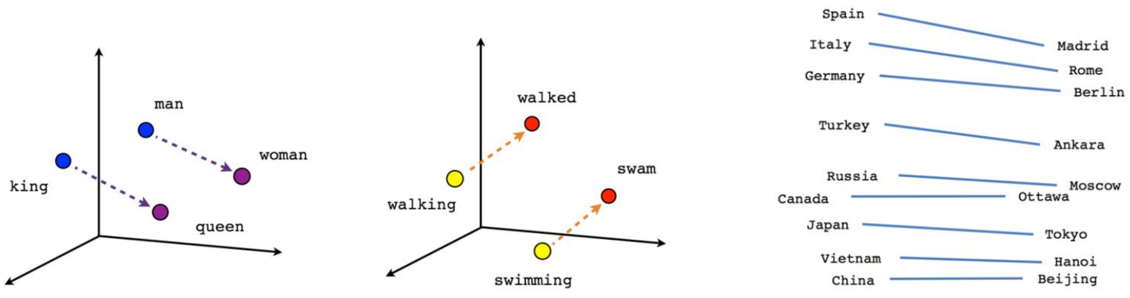
https://sciencenode.org/feature/text-mining-strikes-gold-in-political-discourse.php

Word embeddings

A numerical representation of words and texts is required. E.g. 400k words in English are represented as vectors with a dimension of 400k. For each word, exactly one dimension is set to non zero.

Idea

- ➢ Reduction of dimensionality by transformation into a **low dimensional space** (embedding).
- ➢ Typically this space dimension is between 50 and 1500.



Male-Female   Verb tense   Country-Capital

Dimensions

Word vectors				
dog	-0.4	0.37	0.02	-0.34
cat	-0.15	-0.02	-0.23	-0.23
lion	0.19	-0.4	0.35	-0.48
tiger	-0.08	0.31	0.56	0.07
elephant	-0.04	-0.09	0.11	-0.06
cheetah	0.27	-0.28	-0.2	-0.43
monkey	-0.02	-0.67	-0.21	-0.48
rabbit	-0.04	-0.3	-0.18	-0.47
mouse	0.09	-0.46	-0.35	-0.24
rat	0.21	-0.48	-0.56	-0.37

animal
domesticated
pet
fluffy

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/
https://medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf

word2vec

Creating a low dimensional representation of words from a **corpus** using 2 methods, CBOW and skip-gram. Words with similar meanings get a small distance in the new vector space. Uses 10 to 1500 dimensions. Uses a training method similar to the Autoencoder.

CBOW (continuous bag of words)

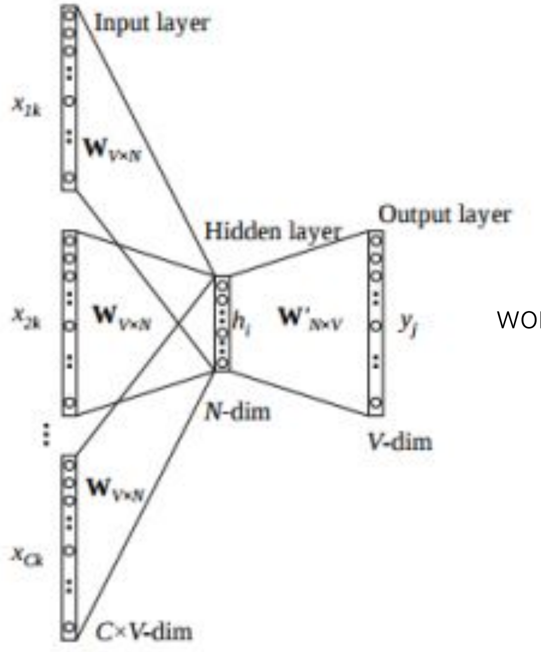
➢ Learns the prediction of the current word through C words in context (e.g. C=5)

Skip-gram

➢ Learns the prediction of the current context words for a word. (e.g. C=10).

CBOW model

context words



word

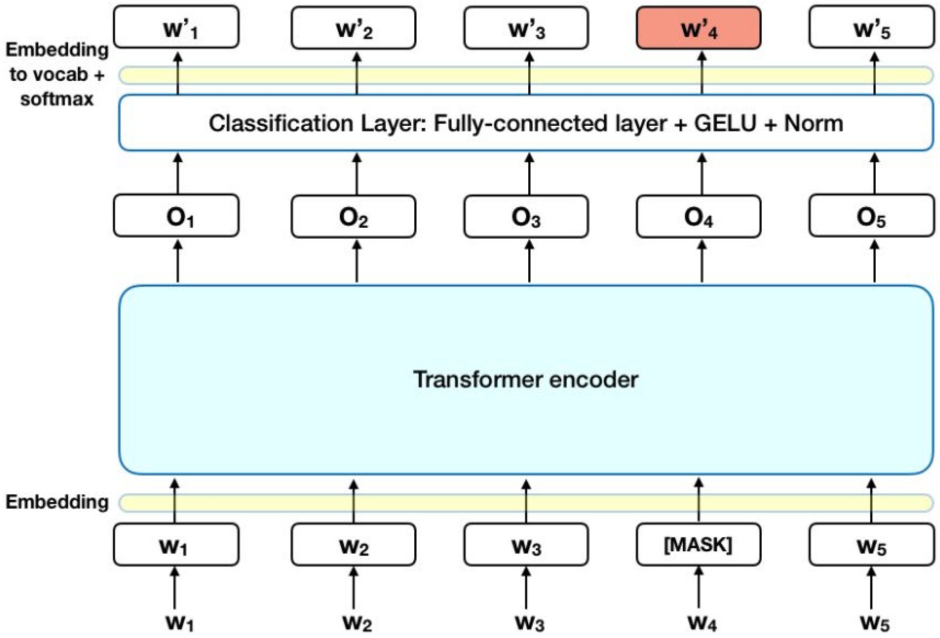
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

BERT model

BERT (bidirectional encoder representations from **transformers**)

BERT is a very recent advancement in embedding models which is non-directional. It learns the context of words independent from the direction.

- ➢ 15% of the words in each sample are masked. The model is trained to predict the masked words (**masked language model**).
- ➢ Additionally, the model is trained to predict if the second half of the input is a valid extension of the first half (**next sentence prediction**).



https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

BERT model background

Key information:

- ➢ BERT is able to **separate** different context **meanings** of the same word (e.g. *apple* as company and *apple* as fruit)
- ➢ BERT does not work on word level, but uses parts of words (e.g. **30k tokens**).
- ➢ BERT model size is huge (e.g. **345 million** parameters in BERT large).
- ➢ training a model is very expensive
- ➢ BERT is trained in two phases
  - ○ language pretraining
  - ○ task training (fine tuning)
- ➢ fine-tuned tasks can be
  - ○ Q&A
  - ○ translation
  - ○ completion
  - ○ classification, ...

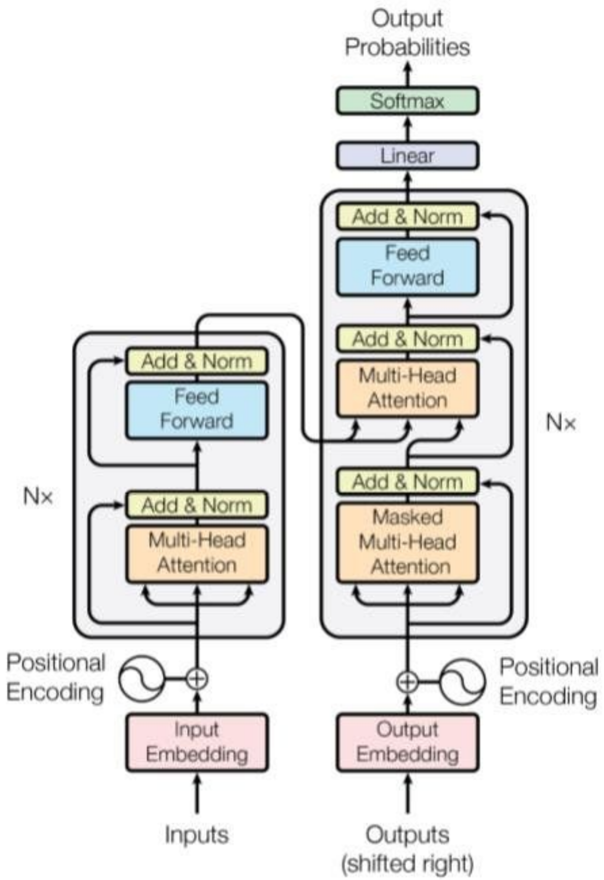
	Training Compute + Time	Usage Compute
$BERT_{BASE}$	4 Cloud TPUs, 4 days	1 GPU
$BERT_{LARGE}$	16 Cloud TPUs, 4 days	1 TPU

https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Transformer architecture

Ideas

- ➢ works with **tokens** instead of words
- ➢ uses **encoding** for tokens and position
- ➢ **attention mechanism** generates maps of important relations between individual words of a text in the input stream.
- ➢ attention maps create relations in the output stream
- ➢ transformers are often trained on autoregressive tasks (e.g. estimation of masked tokens)



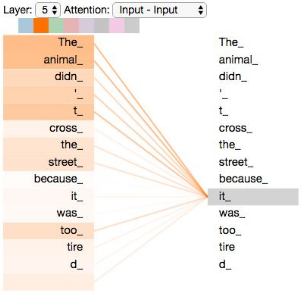
source: https://www.researchgate.net/figure/The-transformer-model-from-Attention-is-all-you-need-Viswani-et-al_fig1_339390384

Attention and scaled dot product attention

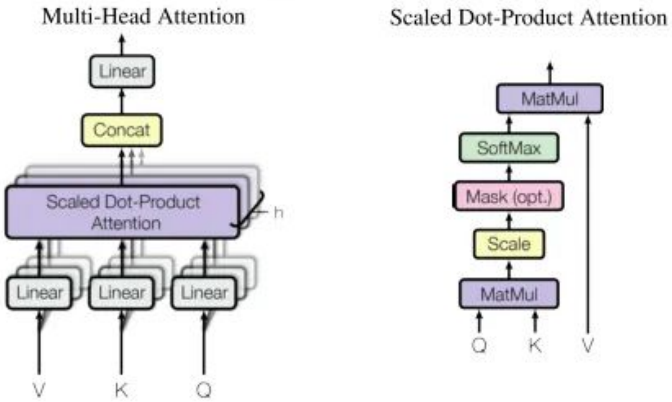
Key ideas:

- ➢ attention gives the decoder access to all the encoder's hidden states
- ➢ attention mechanism is used as a way for the model to **focus on relevant** information based on what it is currently processing
- ➢ attention controls the relevance of the encoder hidden states (**value V**) in processing the decoder state (**query Q**) and were calculated based on the encoder hidden states (**key K**)
- ➢ scaled dot product attention is the implementation of attention in the transformer architecture
- ➢ V,K,Q are basically the **linear transformations of the embedded inputs** in the encoder
- ➢ in the decoder, K,Q are from the encoder outputs and V comes from the embedded former outputs

self attention



dot product attention in transformers



source: https://arxiv.org/pdf/1706.03762.pdf
https://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/
https://youtu.be/mMa2PmYJlCo

GPT-3 model

GPT-3 is a **transformer** model with **175 billion** parameters

- ➢ attention allows it to filter and combine essential information
- ➢ can **complete text** and generate new text
- ➢ trained on **6 million** articles
- ➢ created by OpenAI
- ➢ recently bought from Microsoft

- ➢ shows first steps of deeper text comprehension. However, does not have deeper **symbol grounding**

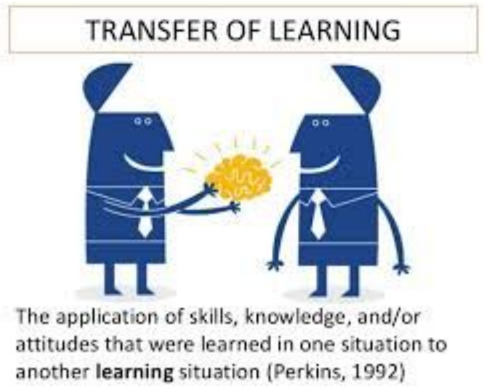
```
Q: What is your favorite animal?
A: My favorite animal is a dog.

Q: Why?
A: Because dogs are loyal and friendly.

Q: What are two reasons that a dog might be in a bad mood?
A: Two reasons that a dog might be in a bad mood are if it is hungry or if it is hot.

Q: How many eyes does a giraffe have?
A: A giraffe has two eyes.

Q: How many legs does a frog have?
A: A frog has four legs.

Q: Are there any animals with three legs?
A: No, there are no animals with three legs.

Q: Why don't animals have three legs?
A: Animals don't have three legs because they would fall over.
```

```
Q: How many eyes does a giraffe have?
A: A giraffe has two eyes.

Q: How many eyes does my foot have?
A: Your foot has two eyes.

Q: How many eyes does a spider have?
A: A spider has eight eyes.

Q: How many eyes does the sun have?
A: The sun has one eye.

Q: How many eyes does a blade of grass have?
A: A blade of grass has one eye.
```

sources: https://towardsdatascience.com/gpt-3-demos-use-cases-implications-77f86e540dc1
https://lacker.io/ai/2020/07/06/giving-gpt-3-a-turing-test.html

Pretrained models

The effort for training a word2vec model is high. It pays off to use ready-trained models as a basis and to modify them further.



Examples

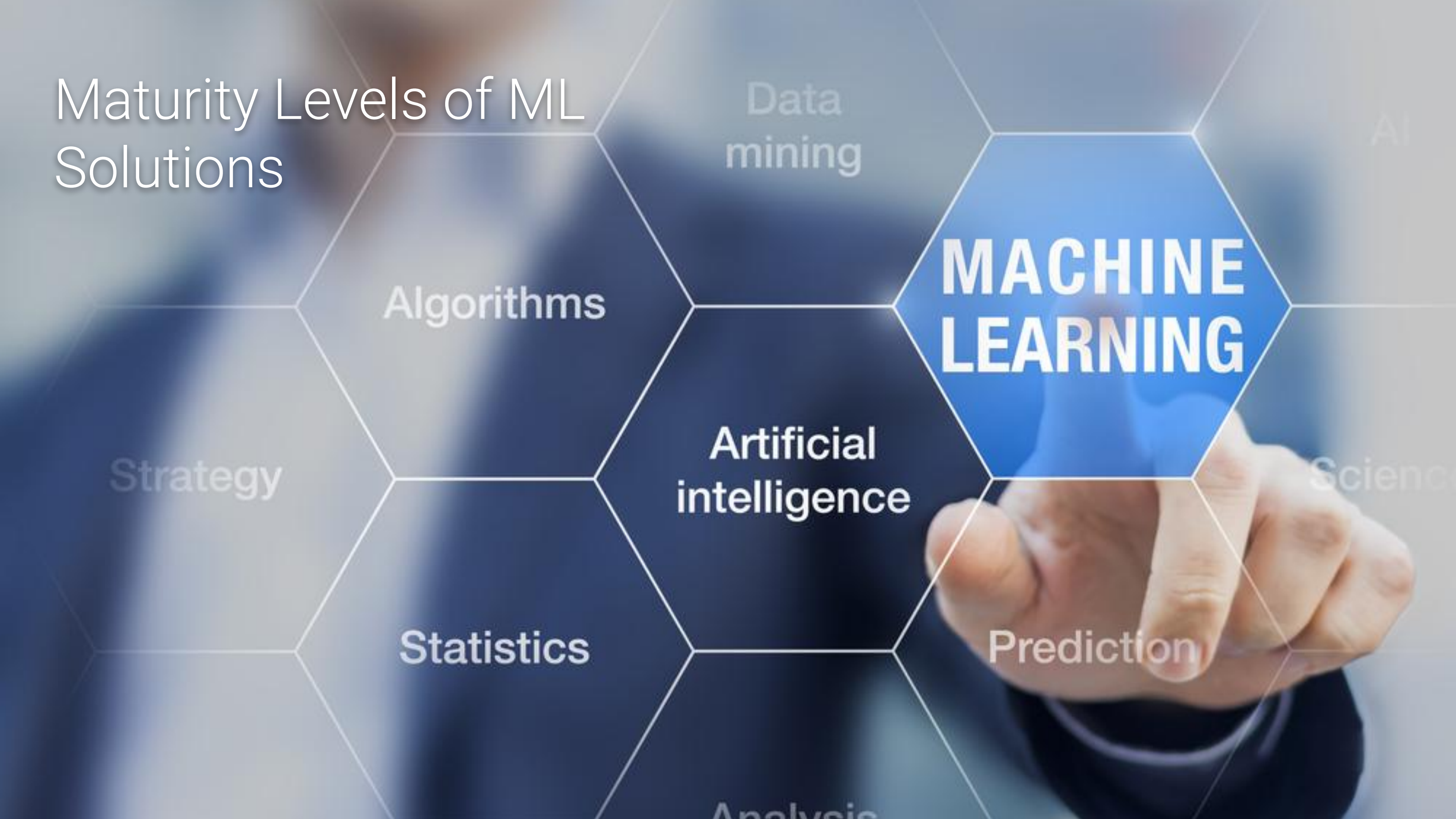
- **word2vec**, from text bodies with 100 billion words, 300 dimensions
- **GloVe**, Stanford University, several models from Wikipedia, Web and twitter.
- **BERT** models
- **GPT-3**, sorry, can't be downloaded, it is no longer open source

https://modelzoo.co/framework/keras

Artificial Intelligence

3 Life cycle of ML projects

Maturity Levels of ML Solutions



Data to AI Maturity

1 Manual Data Drudgery

Manual reports

Spreadsheets & PowerPoints communicate status

Disagreements on how data was processed

2 Death by Dashboards

Shadow data teams

Only privileged employees can create reports

Big spend on reporting, dashboarding or BI systems

Employees flooded with irrelevant data

Multiple, inconsistent sources of truth

3 Data Tells A Story

Glance-able answers start to simplify employee processes

Multi-source data merging

Consistent view of info up & down the organization

IT & business leadership coordinate work

Measurable results emerge

4 Emerging Intelligence

Consistent measurable results

Proactive information supports employees

Experience tuned for each customer and employee

Smart systems know what to look for

Data crosses silos

5 Transformed Organization

AI/ML is real

New ways of working

Employees focused on high value work, all low value work automated

Recommendations are right for the employee

New business models emerge

LEXTECH

source: https://towardsdatascience.com/how-to-measure-your-organizations-data-maturity-2352cbaf1896

Success factors for ML projects

Team mindset

- ➢ focus on **business impact** vs science
- ➢ start with simple **baseline** models
- ➢ implement complete **life-cycle** asap
- ➢ implement gradual improvement

Company mindset

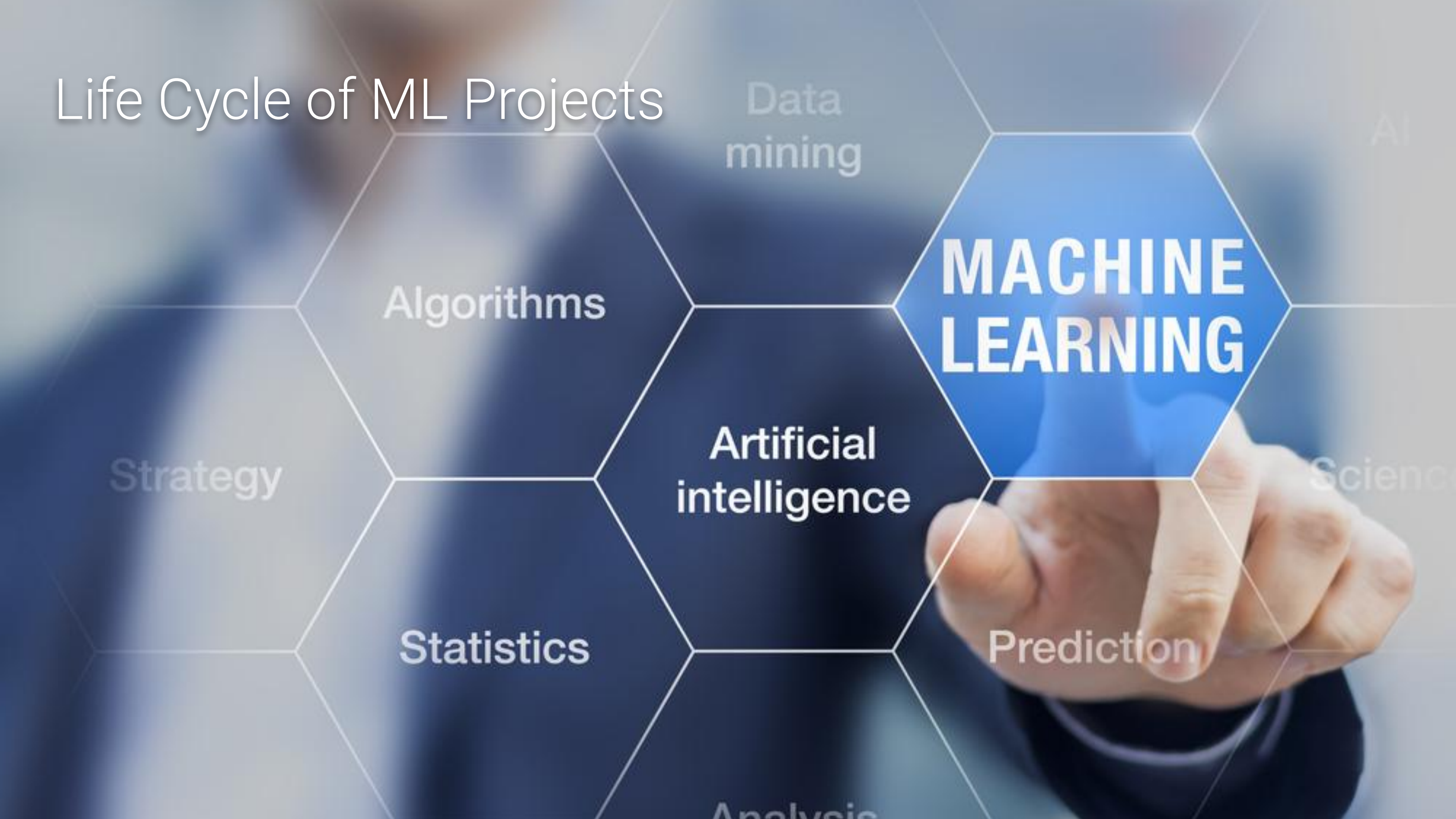
- ➢ set **realistic goals** for ML projects
- ➢ product management must understand ML
- ➢ manage customer expectations
- ➢ avoid isolation of ML teams

Fail-fast principle

- ➢ provide suitable ML infrastructure
- ➢ **test and monitor** extensively
- ➢ error tolerant project setup
- ➢ **continuous integration** for ML

source: https://towardsdatascience.com/the-secret-of-delivering-machine-learning-to-production-1f6681f5e30c
https://martinfowler.com/articles/cd4ml.html

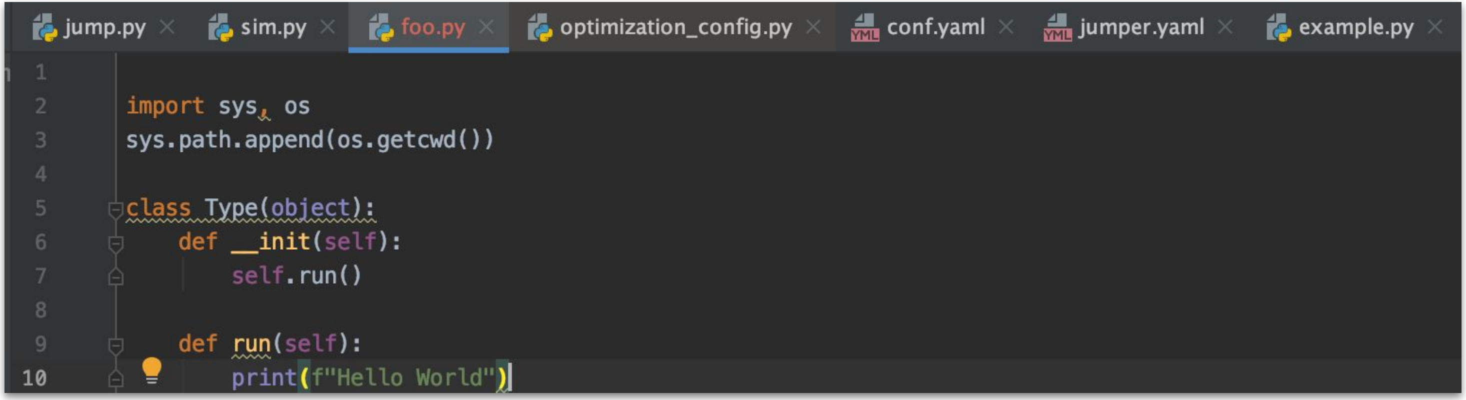
Life Cycle of ML Projects



Why are ML projects different to software projects?

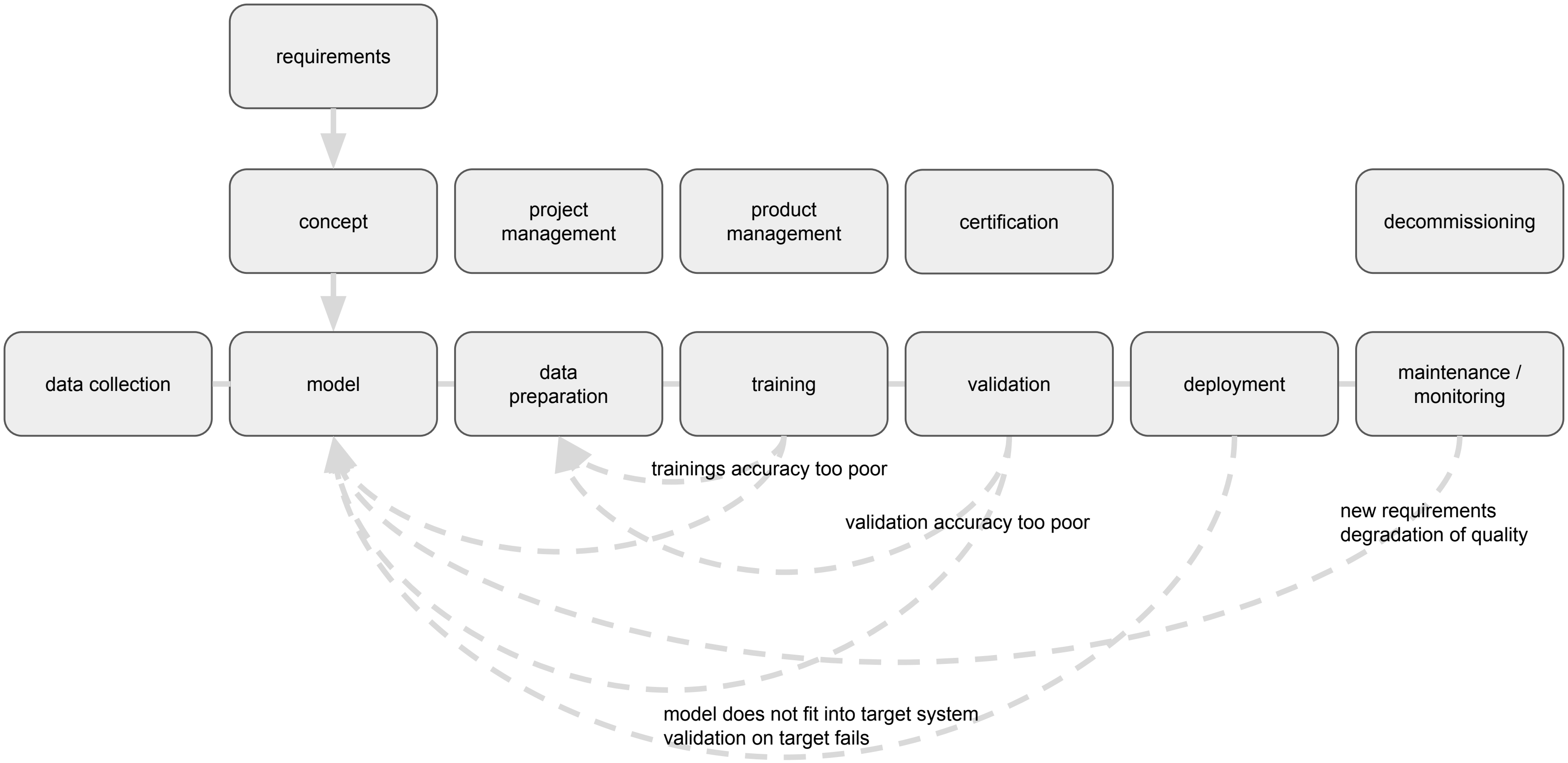
Development of software has become a well understood and managed capability of organizations.

Machine learning has not yet reached this status. Many ML projects have to deal with the uncertainty if the expected results can be delivered.



https://towardsdatascience.com/machine-learning-engineer-versus-software-engineer-fb59f8cba9dd

More realistic life cycle of ML projects



Machine learning problem framing

Description of **problem to solve**
- ➢ from the viewpoint of the user
- ➢ include cost of problem
- ➢ how might you solve your problem **without ML**?

Description of the **solution** using digitalization and machine learning
- ➢ what is the **ideal outcome** of the use of the model?
- ➢ how would the problem be reduced by using your approach

Description of the **machine learning solution**
- ➢ define success and failure **metrics**
- ➢ measurable quality metrics in context of model AND problem
- ➢ what type of **output** would you like the ML model to produce? e.g. classification of images, clustering of sensor data, …
- ➢ identify your **data sources** and **labels** (include metadata)
- ➢ identify your **data transformations**

Description of the operational aspects
- ➢ how will the output be **integrated** in a product
- ➢ resources
- ➢ data collection and processing

https://developers.google.com/machine-learning/problem-framing/framing
https://medium.com/thelaunchpad/a-step-by-step-guide-to-machine-learning-problem-framing-6fc17126b981

Quality parameters and test strategy

- The required quality of the machine learning solution must be defined very precisely in **writing**.
- It is necessary to define the **calculation** of the quality parameter **mathematically** OR as **computer code** and to agree upon it in writing with the customer.
- It is difficult to estimate the quality of **unknown new data samples** in the real production environment. So preparations shall be taken to handle situations with insufficient quality in a cooperative way.
- A good starting point is to define the **distribution** of the **test data** together with the customer.
- **Liability** if quality cannot be met.

https://tryolabs.com/blog/2013/03/25/why-accuracy-alone-bad-measure-classification-tasks-and-what-we-can-do-about-it/
https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Ground truth collection and creation

We have learned that the creation and quality assurance of ground truth data can be very expensive.

Clarify at least the following aspects:

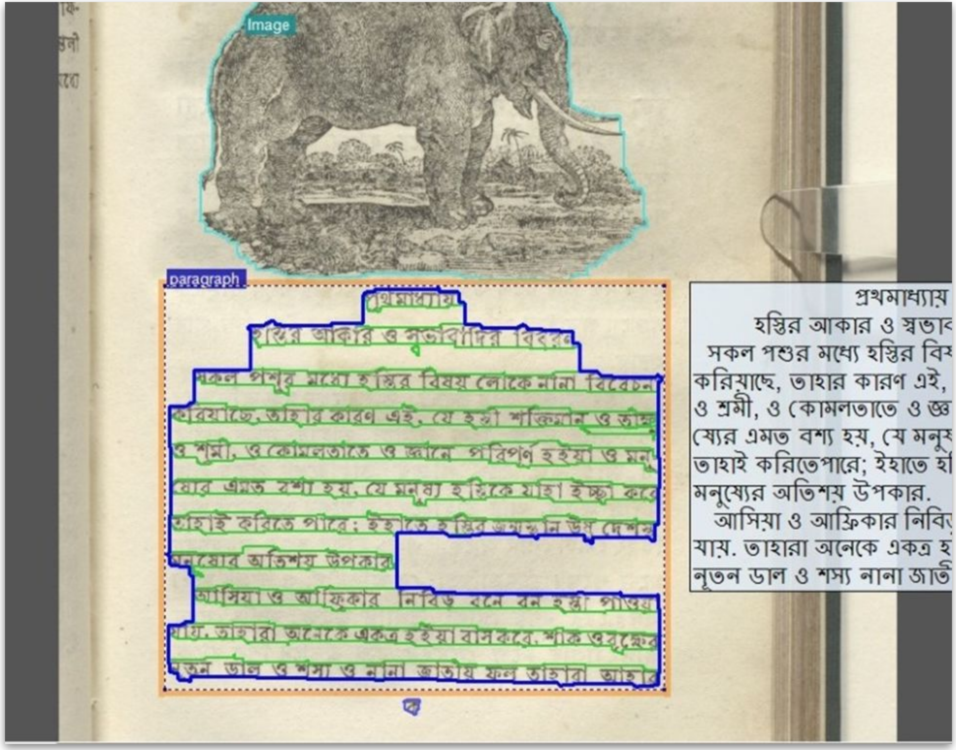
- ➢ **who** is responsible (delivery and cost) for ground truth generation
- ➢ if the initial ground truth data is not sufficient, how are the additional cost **divided**
- ➢ how can the contractor **access** the data, e.g. for measurement in the customer plant
- ➢ **ownership** of the resulting ground truth and rights of use

Ground truth generation

Supervised learning usually requires a larger number of samples with correct labels. The creation of this data set manually is usually not economically feasible.

solutions

- ➢ public and **open source datasets** as a basis
- ➢ **tool** support
- ➢ semi-automatic **generation of the GT**
- ➢ **augmentation** of existing datasets
- ➢ **synthesis** of ground truth data
- ➢ cheaper sources of **manual labor**



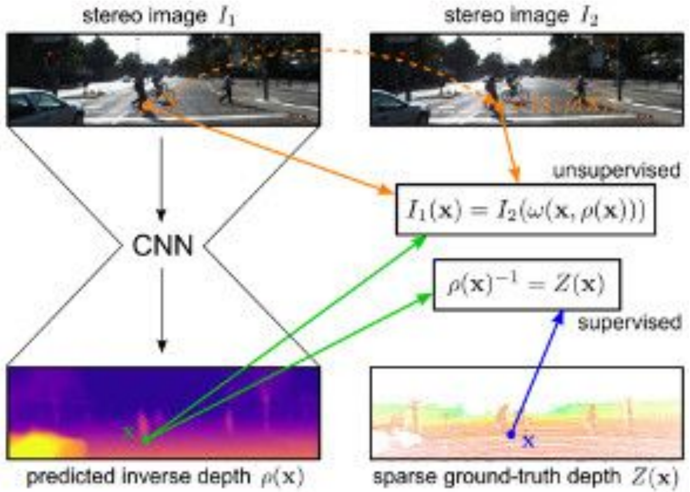
source: https://pro.europeana.eu/page/issue-13-ocr

Semi-automatic generation of ground truth

Filter the most relevant events from your unlabeled data. Estimation of the label by simpler model or older generation of your model. Preparation for import into ground truthing Tool.

possible filters and transformations

- ➢ **heuristics** and rules
- ➢ **anomaly** detection
- ➢ **clustering**
- ➢ previous generations of your **model**
- ➢ **measure** the ground truth



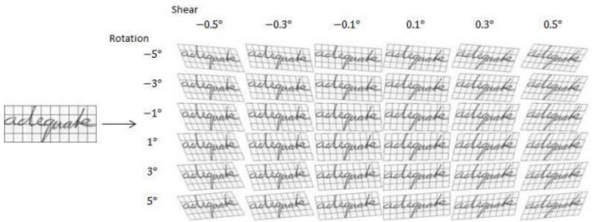
https://www.semanticscholar.org/paper/Semi-automatic-Ground-Truth-Generation-for-Chart-I-Yang-Huang/378e25cabd3c7c21b486f02208b21d6feff5cba8

Augmentation of training data

Extension of the ground truth dataset by generation of **artificially modified** versions of original samples. The applied change must correspond to transformations that also occur in the real environment.

typical transformations are

- ➢ **distortions**
- ➢ **superpositions**
- ➢ **noise**
- ➢ **partitioning**
- ➢ **errors**



source: https://m2dsupsdlclass.github.io/lectures-labs/slides/03_conv_nets/index.html#1

Synthesis of ground truth data

Artificial generation of ground truth data from physical models or basic data components.

examples

- ➢ **superimposed** audio tracks
- ➢ **simulated** water leaks
- ➢ **stickers** on traffic signs

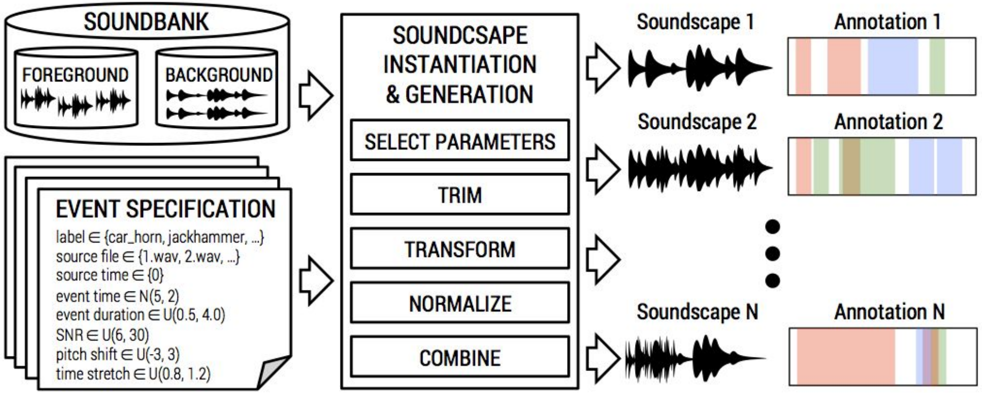


Figure 1: Block diagram of the Scaper synthesis pipeline.

source: https://github.com/unrealcv/synthetic-computer-vision
http://www.justinsalamon.com/news/new-tools-data-for-soundscape-synthesis-and-online-audio-annotation

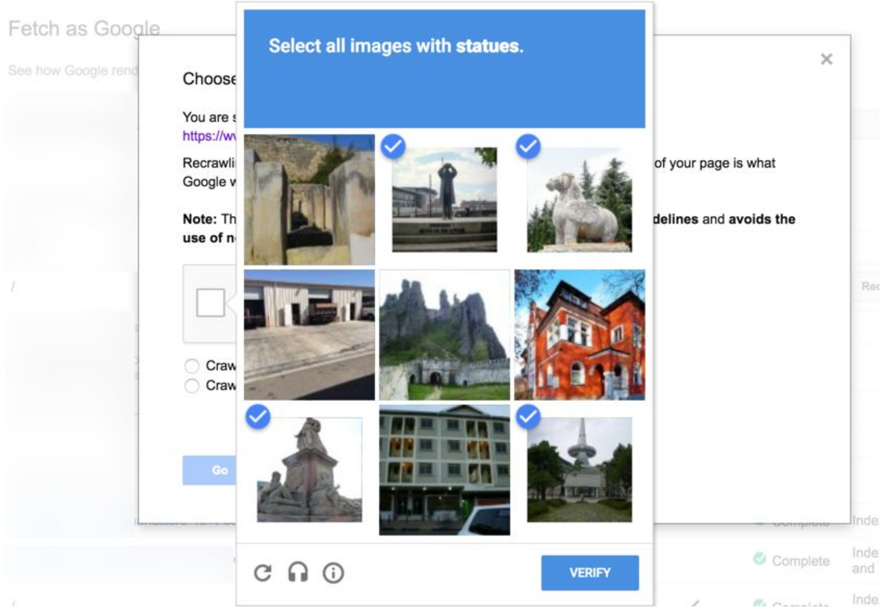
Manually creating ground truth

If ground truth has to be labelled manually, there are several alternatives.

- ➢ **lower cost employees**
- ➢ **low cost markets**
- ➢ Amazon Sagemaker Ground Truth
- ➢ fiverr and similar **platforms**
- ➢ **crowdsourcing**

However, it is important that the resulting quality is carefully monitored.

If you don't care for this aspect, then your expensive data scientists will do this work [sic]



source: https://medium.com/@thenextcorner/you-are-helping-google-ai-image-recognition-b24d89372b7e

Validation and Testing

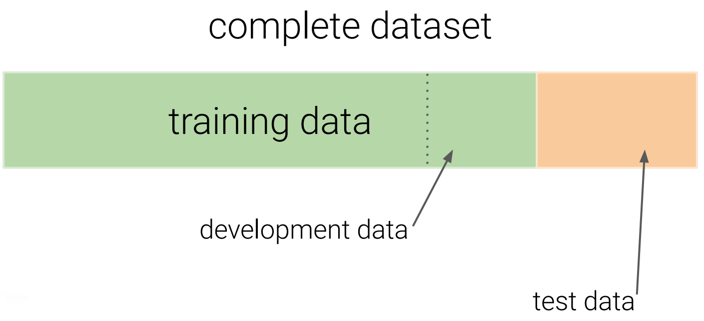
Debugging and testing

First, check the performance of the model during training. In the first step only the **accuracy** of the model using **training data** is measured.

Next the accuracy against **development data** can be measured. This delivers an estimation of the production level quality.

However, during model debugging using development data, the model starts to create a **bias** towards the development data.

In the final step a model is **tested** using the test data. Test data should be very similar to data expected from the production level use.



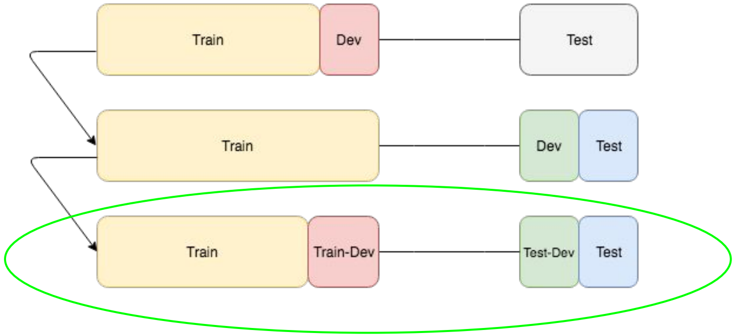
Thou shalt not use the test data for training.

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Training data split

Split up your valuable data for development

- **Training data** are for training
- **Dev data** are not used for training but support the debugging of the model and the detection of overfitting
- **Test data** is used for estimation of model performance in the production environment. Test data are never used for training. Test data should have the **same distribution** as real-world data.



https://kevinzakka.github.io/2016/09/26/applying-deep-learning/

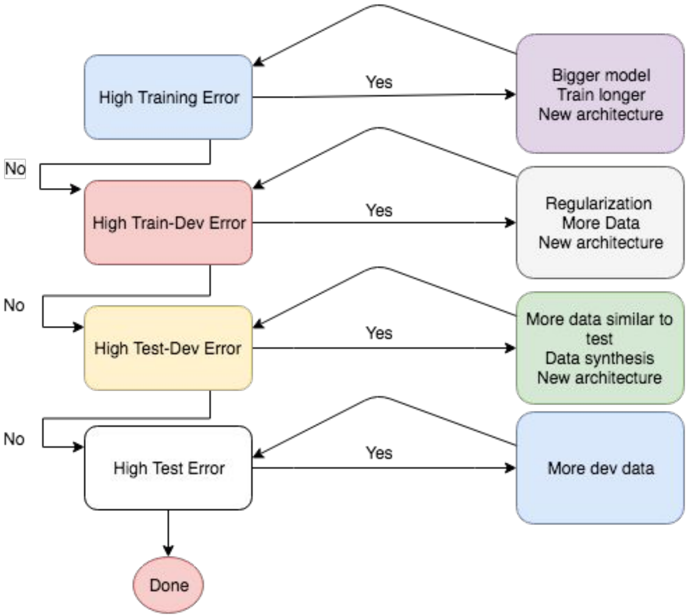
Control your training

Debugging workflow

- ➢ follow diagram

Possible actions

- ➢ train more
- ➢ bigger model
- ➢ different architecture
- ➢ regularization
- ➢ more data
- ➢ more data
- ➢ more data
- ➢ **more data** (no joke)



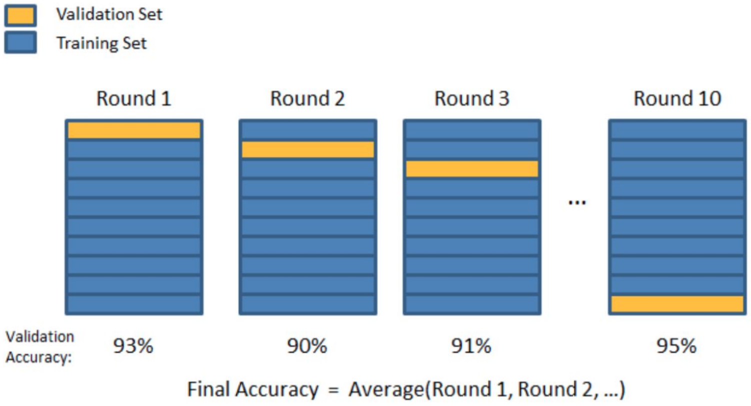
https://kevinzakka.github.io/2016/09/26/applying-deep-learning/

K-Fold cross-validation

How can you estimate how the system will behave in productive use?

Idea

Division of the training data record into K parts. In several rounds, one of the K parts is excluded from the training and only used for testing.
The final quality parameter (e.g. Accuracy) is then the mean value of all K rounds.



> Cross-validation requires an execution infrastructure called the **Machine Learning Pipeline**.

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

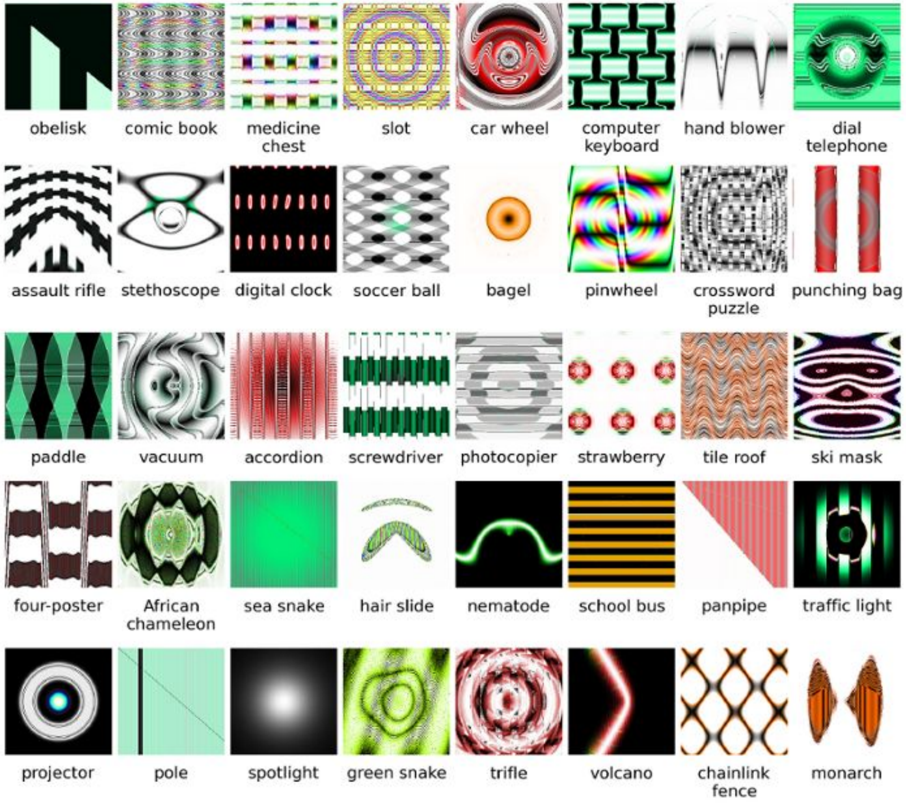
Warning

Unknown unknowns

➢ After training and validation, we know what the ML model has learned, but we don't know **how it handles unknown data**.

Approaches to reduce these unknowns are

➢ Real life data
➢ Augmentation
➢ Targeted hacking
➢ Perturbation testing



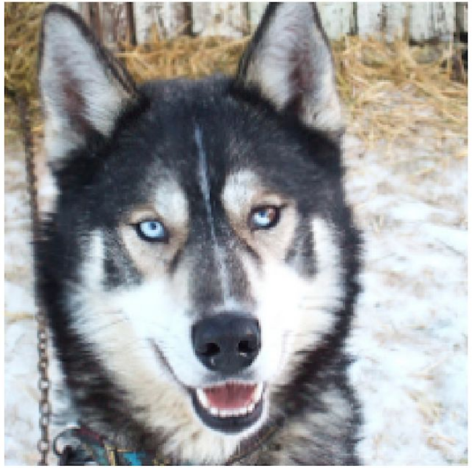
http://www.evolvingai.org/fooling

Machine learning generates statistics not causality

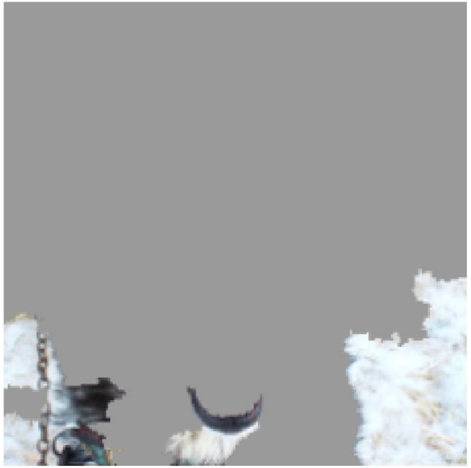
ML enforces **simple models.**
Errors in data may lead to low quality models.

E.g. Husky case from LIME paper:

- ➢ training data contained wolves in snowy background and huskies in gras background.
- ➢ Network learned to look at the background only



(a) Husky classified as wolf



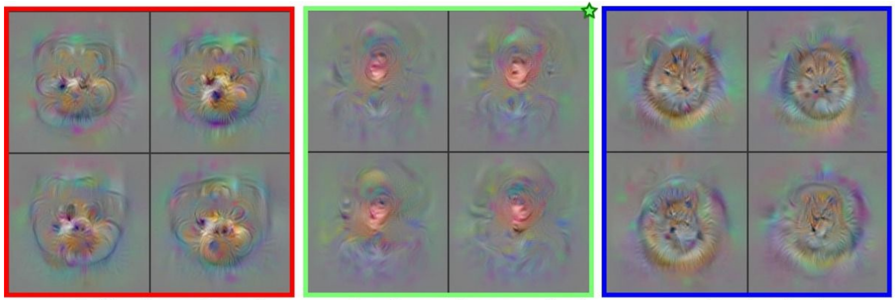
(b) Explanation

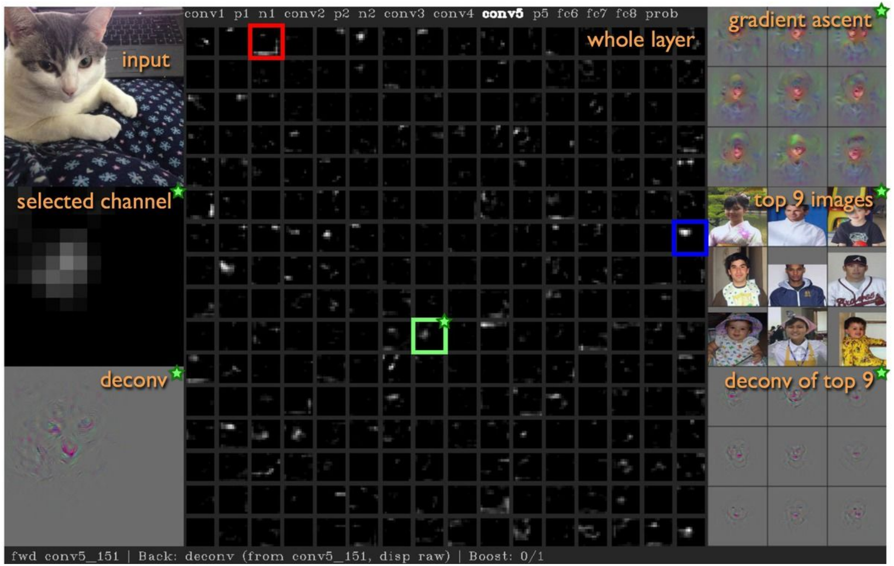
source: LIME paper https://arxiv.org/pdf/1602.04938.pdf

Deep visualization toolbox

Try to understand what a network has learned

- ➢ to understand the limits of abilities. Safety and robustness against unknown data.
- ➢ hacking. Understand which neurons trigger classification result the most
- ➢ leads to explainability and transparency
- ➢ possible with every type of neural network



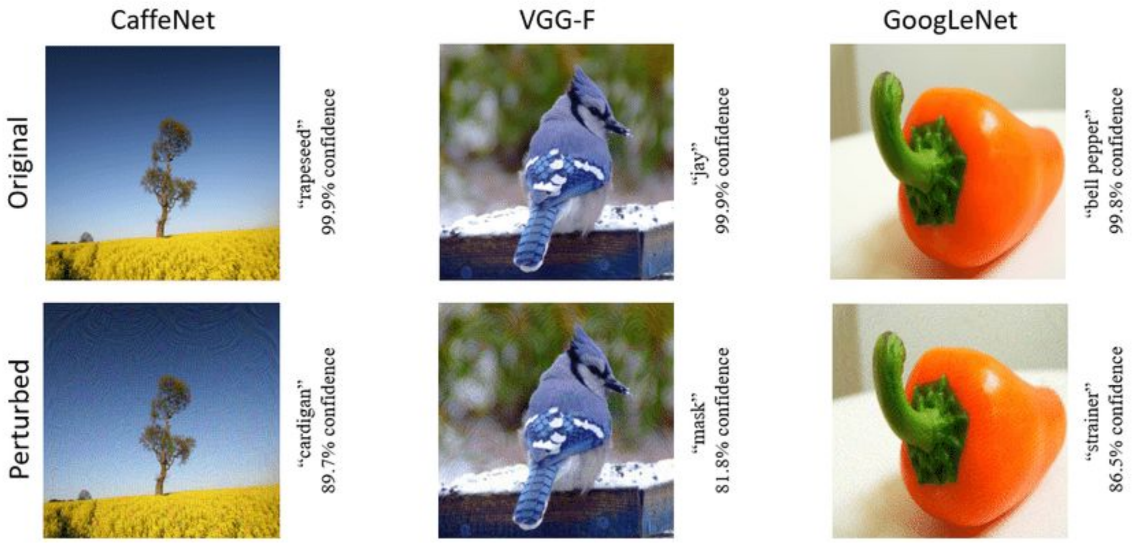


http://yosinski.com/deepvis and https://youtu.be/AgkfIQ4IGaM

Perturbation testing

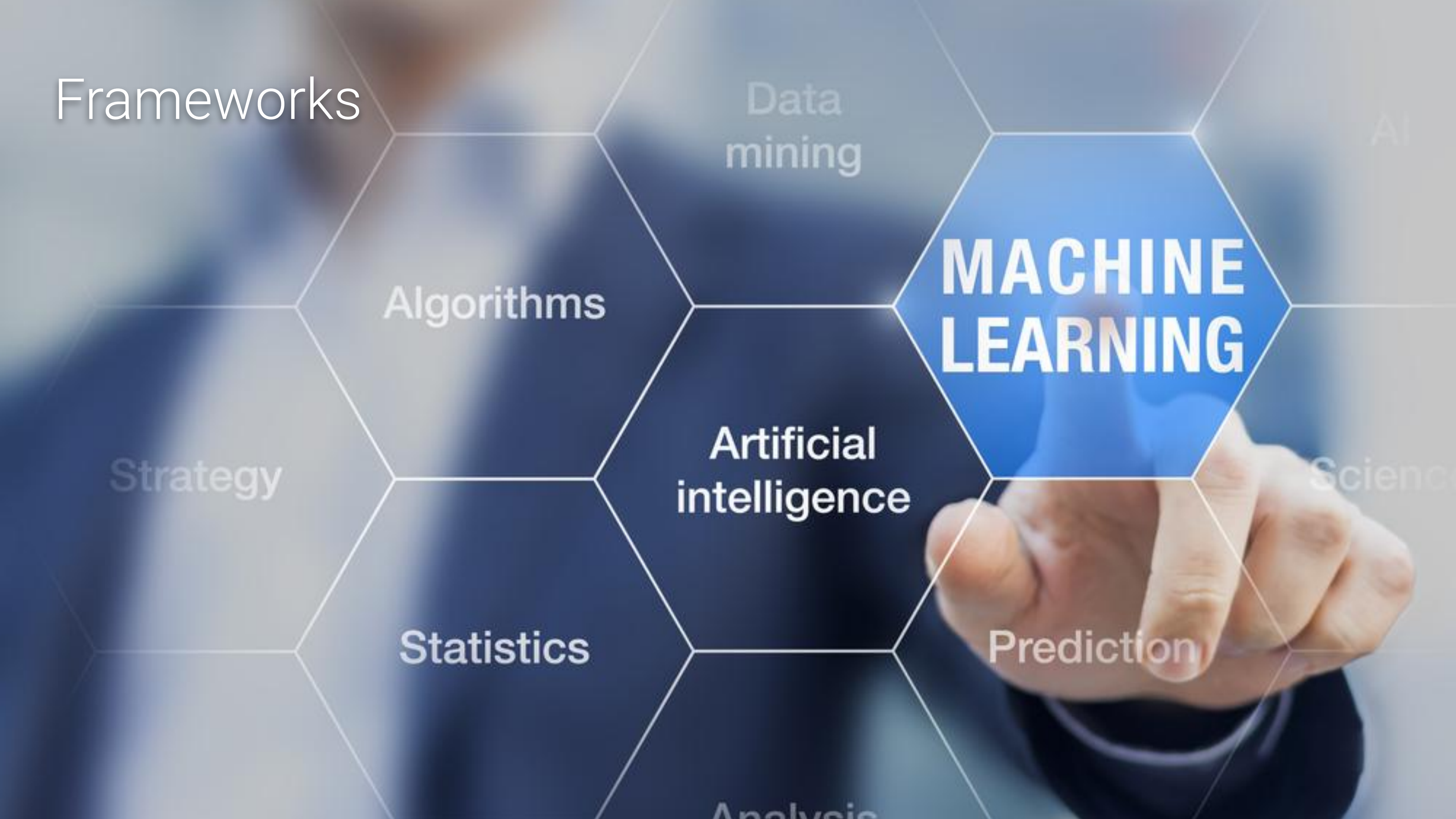
Intuition

- ➢ analyze how robust the predictions of a network are by collecting a set of predictions and add distortions to the input data
- ➢ rerun predictions and look for degradation of the prediction quality

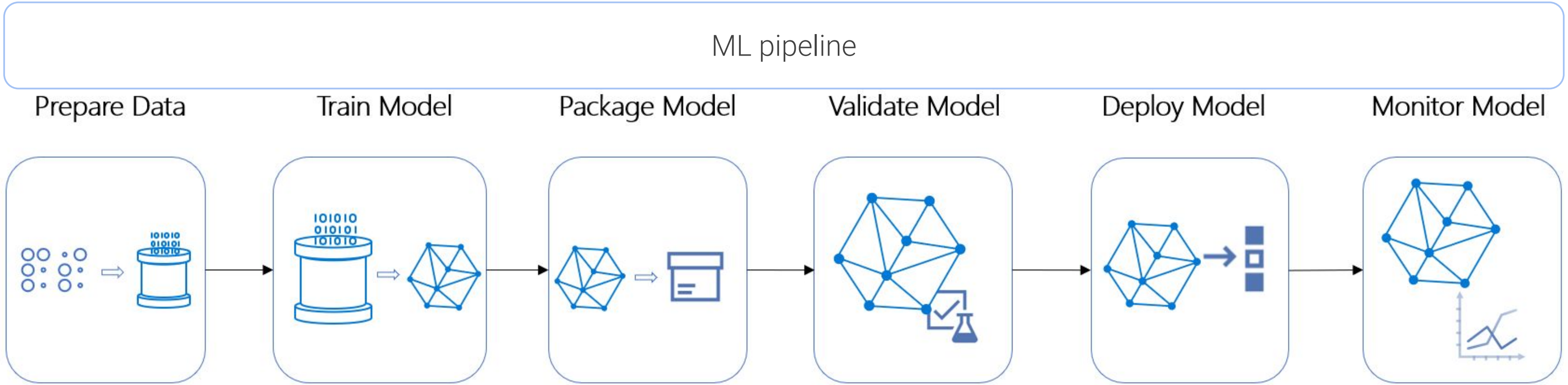


https://www.researchgate.net/figure/Example-of-attacks-on-deep-learning-models-with-universal-adversarial-perturbations_fig7_322221397
extended ideas: https://medium.com/intuitionmachine/deep-learning-perturbations-is-all-you-need-d630b6980587

Frameworks


Data
mining
AI
MACHINE
LEARNING
Algorithms
Artificial
intelligence
Strategy
Statistics
Prediction

Machine learning pipeline



Frameworks shall support the construction of a **machine learning pipeline**

https://docs.microsoft.com/en-us/azure/machine-learning/concept-ml-pipelines

ML frameworks (2020)

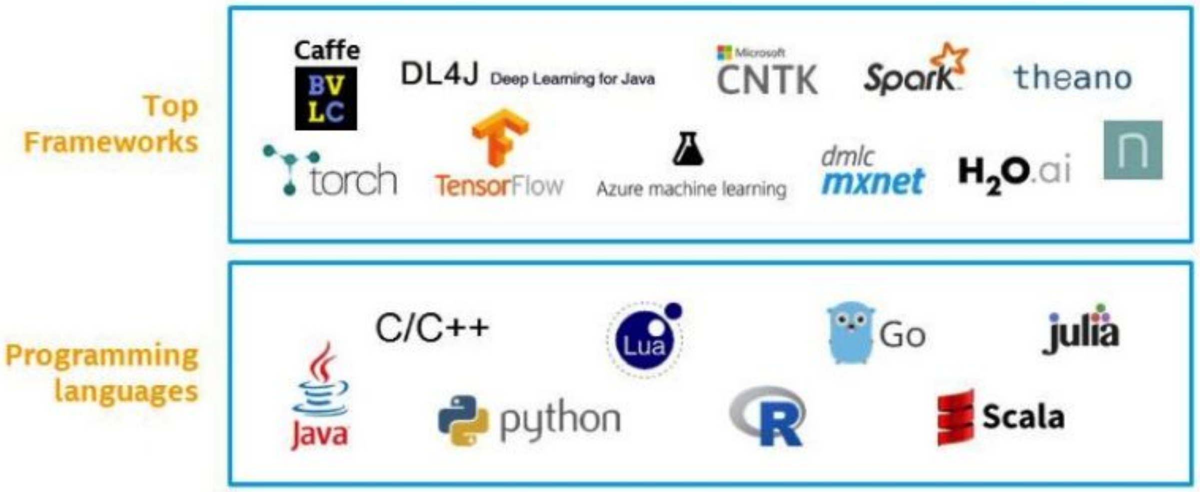
Sci-kit learn

tensorflow/Keras

pytorch

Apache Spark

Apache MXNet



http://bigdata-madesimple.com/machine-learning-becomes-mainstream-how-to-increase-your-competitive-advantage/

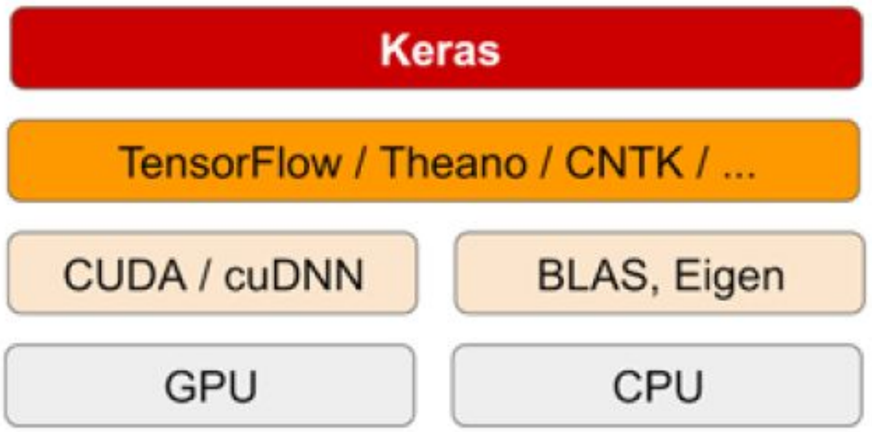
Keras on tensorflow

tensorflow

- ➢ open source library written in C++
- ➢ generates a computation graph
- ➢ graph is optimized and executed on CPU/GPU/TPU
- ➢ binding for python available

Keras

- ➢ high level neural network definition
- ➢ support for different computational backends (theano, tensorflow)



https://medium.com/@mjbhobe/why-i-love-keras-a65675b0e971
https://www.tensorflow.org/guide/keras/functional

Container technologies (Docker)

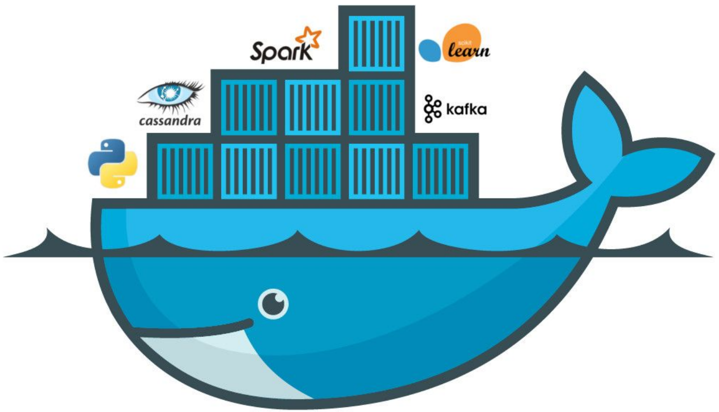
A docker container defines a **light-weight virtual machine** which contains all software components for an application. A docker container can be executed on many platforms.

Complete parts of the ML pipeline can be implemented in docker containers.

Containers are **deployed** on multiple servers for scaling.

Containers can be **plumbed** together to form applications

Special tools are used to **orchestrate** multiple docker containers of a pipeline.



https://towardsdatascience.com/machine-learning-models-as-micro-services-in-docker-a798e1f068a5
https://towardsdatascience.com/using-docker-to-set-up-a-deep-learning-environment-on-aws-6af37a78c551

Frameworks for embedded systems

tensorflow lite

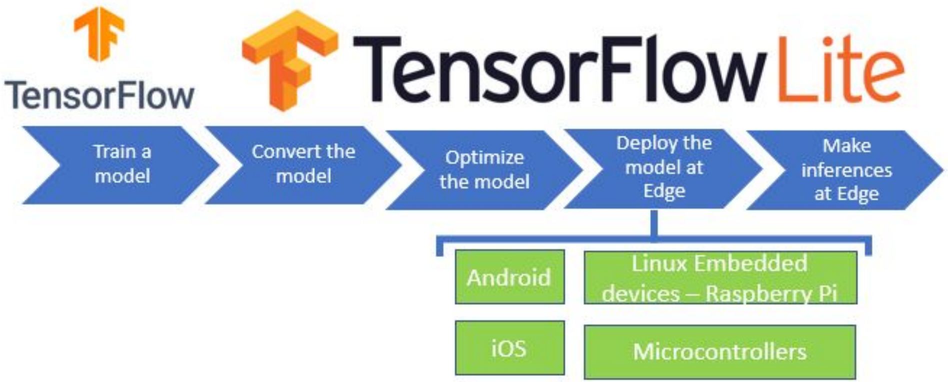
- ➢ tensorflow models can be down converted to run on embedded hardware

hardware oriented frameworks

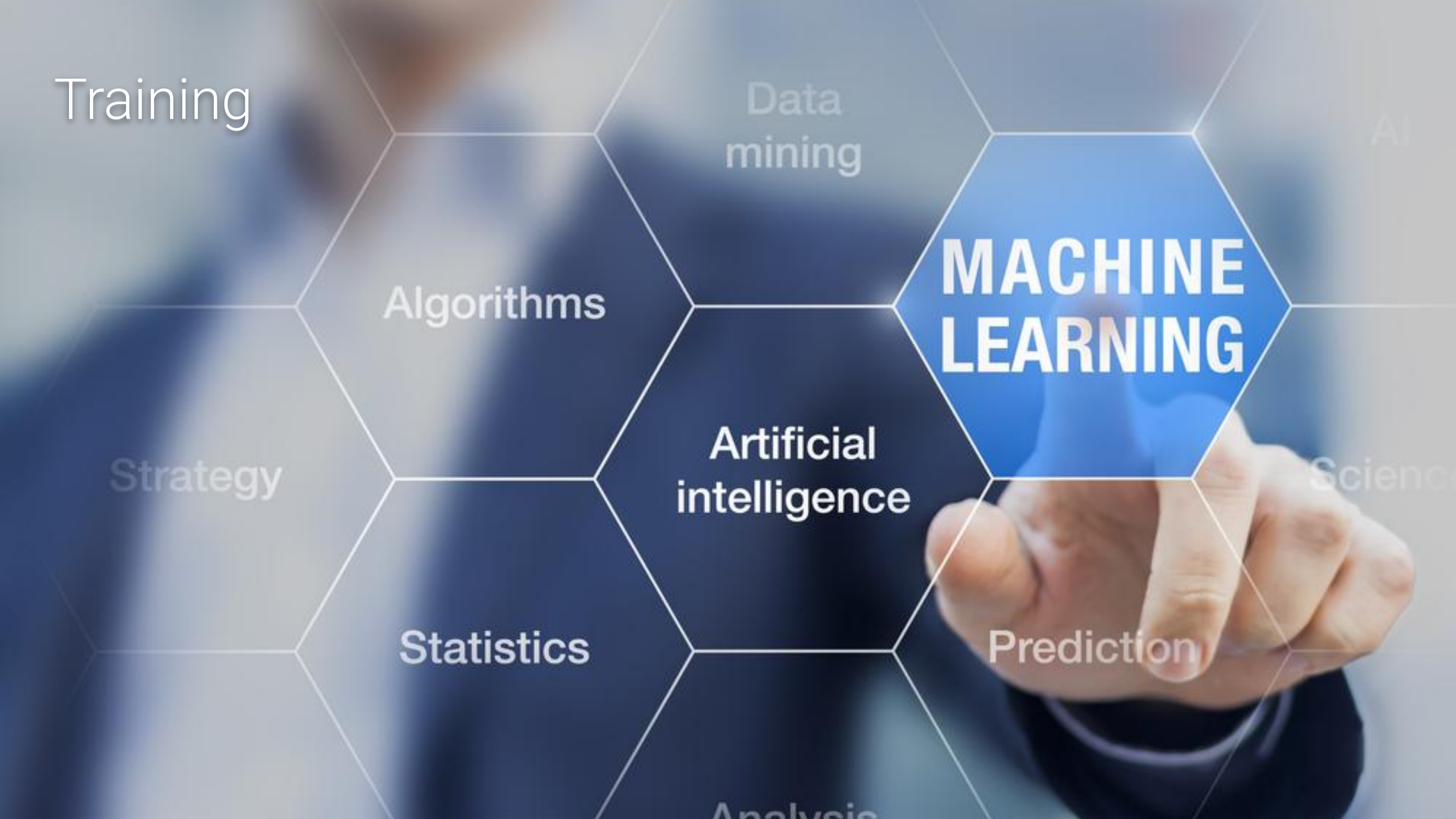
- ➢ Intel movidius framework
- ➢ Beckoff machine learning framework
- ➢ Matlab framework
- ➢ WindRiver,...

considerata

- ➢ look for **high compatibility** with higher level machine learning pipeline
- ➢ compatibility often breaks at very subtle levels (e.g. LSTM not supported)



Training

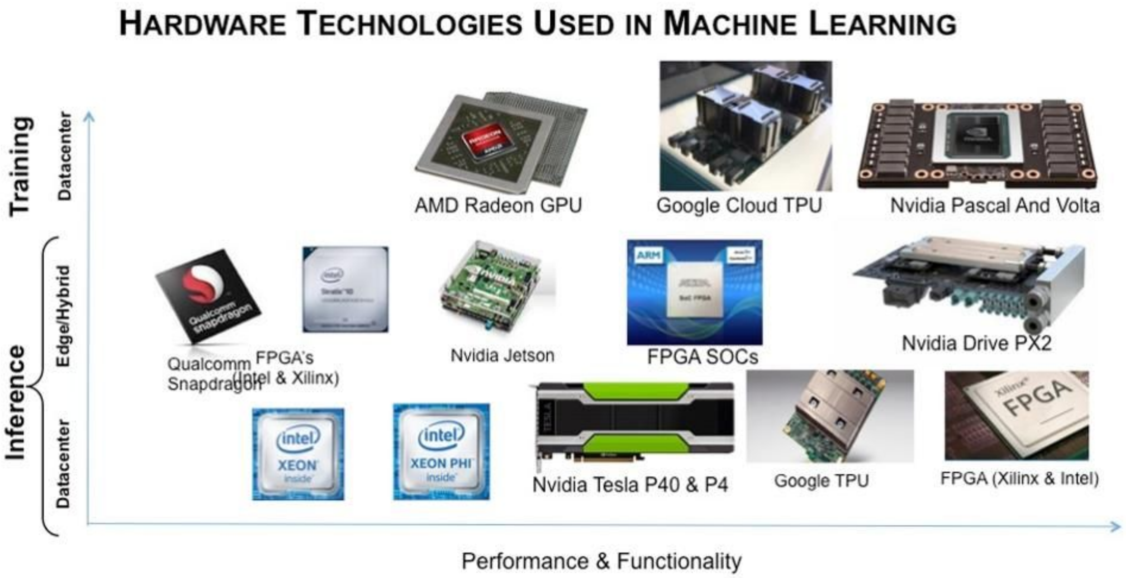

Data mining
Algorithms
MACHINE LEARNING
Strategy
Artificial intelligence
Statistics
Prediction

Training of models

Training models with large data sets requires a lot of **computing power**.

Solutions

- ➢ **cloud servers**
- ➢ **special hardware**
- ➢ **pre-trained models**
- ➢ **transfer learning**



http://www.moorinsightsstrategy.com/will-asic-chips-become-the-next-big-thing-in-ai/

Cloud services and hardware

In many cases it makes sense to only **rent** servers and hardware and not to buy them. The development of new technologies is very fast and therefore an investment in own hardware would be invalidated very quickly.

Cloud services

- ➢ pro: billing only according to time used
- ➢ con: transfer of training **data outside your company** (IP protection)
- ➢ con: **bandwidth** demand for transfer of large models and training data

Own hardware

- ➢ pro: IP protection and local transfer
- ➢ con: not easy to scale
- ➢ con: requires **maintenance**
- ➢ con: rapid aging

VS

http://www.moorinsightsstrategy.com/category/ai-and-machine-learning/

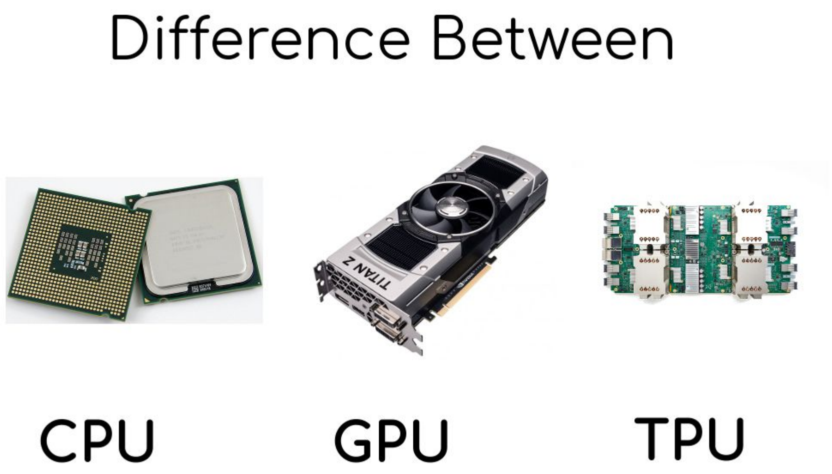
GPU and TPU types

NVIDIA GPUs
- ➢ graphics processing units
- ➢ e.g. RTX 8000: 48GB memory

Google TPUs
- ➢ custom processing units for tensorflow
- ➢ high energy efficiency
- ➢ e.g. V3 with 32GB memory

AMD GPUs
- ➢ cheap, but almost no support for frameworks

FPGAs
- ➢ very powerful, but requires highly skilled experts for setting up



FPGAs?

https://www.geekboots.com/story/cpu-vs-gpu-vs-tpu
https://lambdalabs.com/blog/choosing-a-gpu-for-deep-learning/

Transfer learning

Reuse of models that are already trained. Often only parts of the original network are reused and new parts are added.

Examples

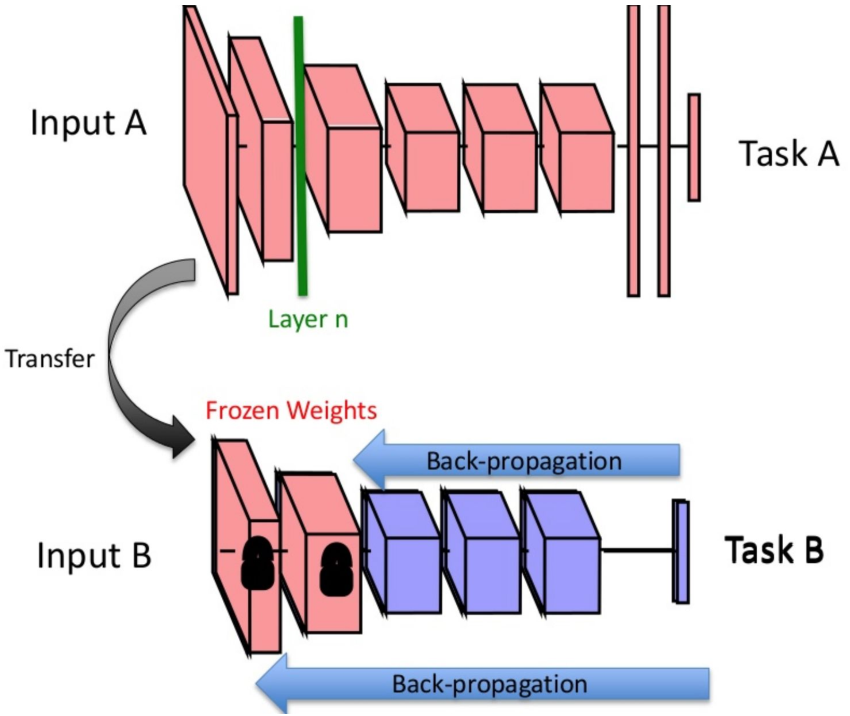
- ➢ old **CNN kernels** with newly trained DNN part
- ➢ word embedding models (BERT, ...)

http://ruder.io/transfer-learning/

Transfer learning and fine tuning

Idea

- ➢ A network is fully trained on data for task A.
- ➢ The trained network is split into static part (red) and a retrainable part (blue).
- ➢ The retrained part is trained with new data for task B. The weights of the static part are not changed.
- ➢ Framework needs to support such operations.



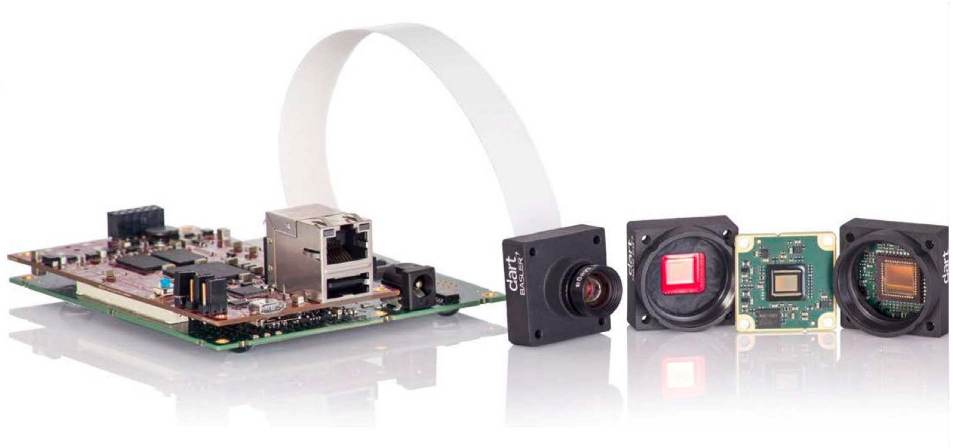
https://www.slideshare.net/fernandoconstantino/transfer-learning-breve-introduccin-a-modelos-preentrenados

Target systems

Different target systems for a model have very different computing resources (CPU, memory, energy) available.

Examples

- ➢ GPU Cloud Cluster
- ➢ cloud server
- ➢ standalone PC
- ➢ robots
- ➢ mini PC (e.g. raspberry pi)
- ➢ mobile phone
- ➢ embedded devices and edge devices
- ➢ smart sensors
- ➢ wearable device
- ➢ implants



The resources of the **target system** for deployment must be considered from the beginning.

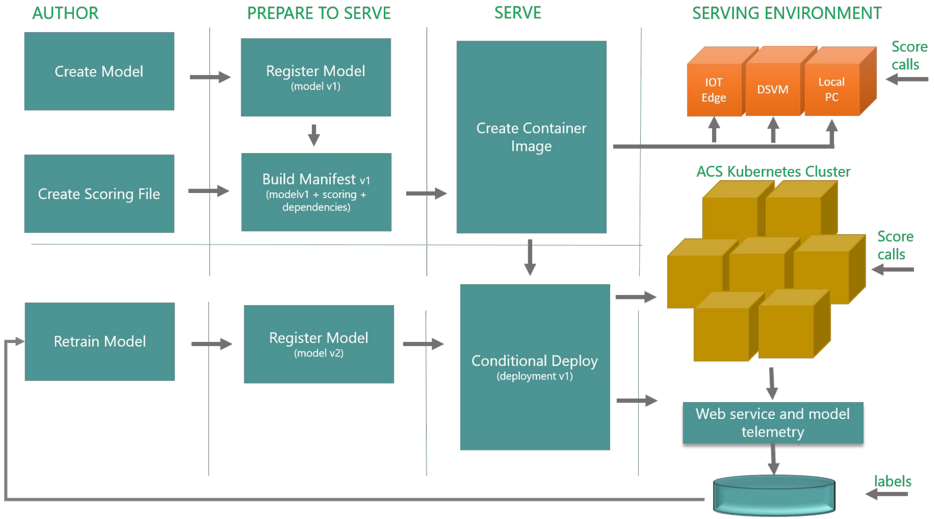
Model management

Management of many variants of one or more models throughout the entire lifecycle of an ML project

- ➢ versioning
- ➢ consistent reproduction
- ➢ tracking of performance
- ➢ permissions
- ➢ serving
- ➢ release
- ➢ retraining

storage

- ➢ git (until you have large binary models)
- ➢ database plus blob storage



https://docs.microsoft.com/en-us/azure/machine-learning/preview/model-management-overview

Monitoring of deployed models

The delivered quality of a model can vary over time due to several reasons

- **training data** may **not** be **representative** compared to customer data
- **change in the data** sent to the model by customers (distribution change)
- **software inconsistencies** (drivers, framework versions)
- data **preparation inconsistencies** (different transformations of features in model and on server)

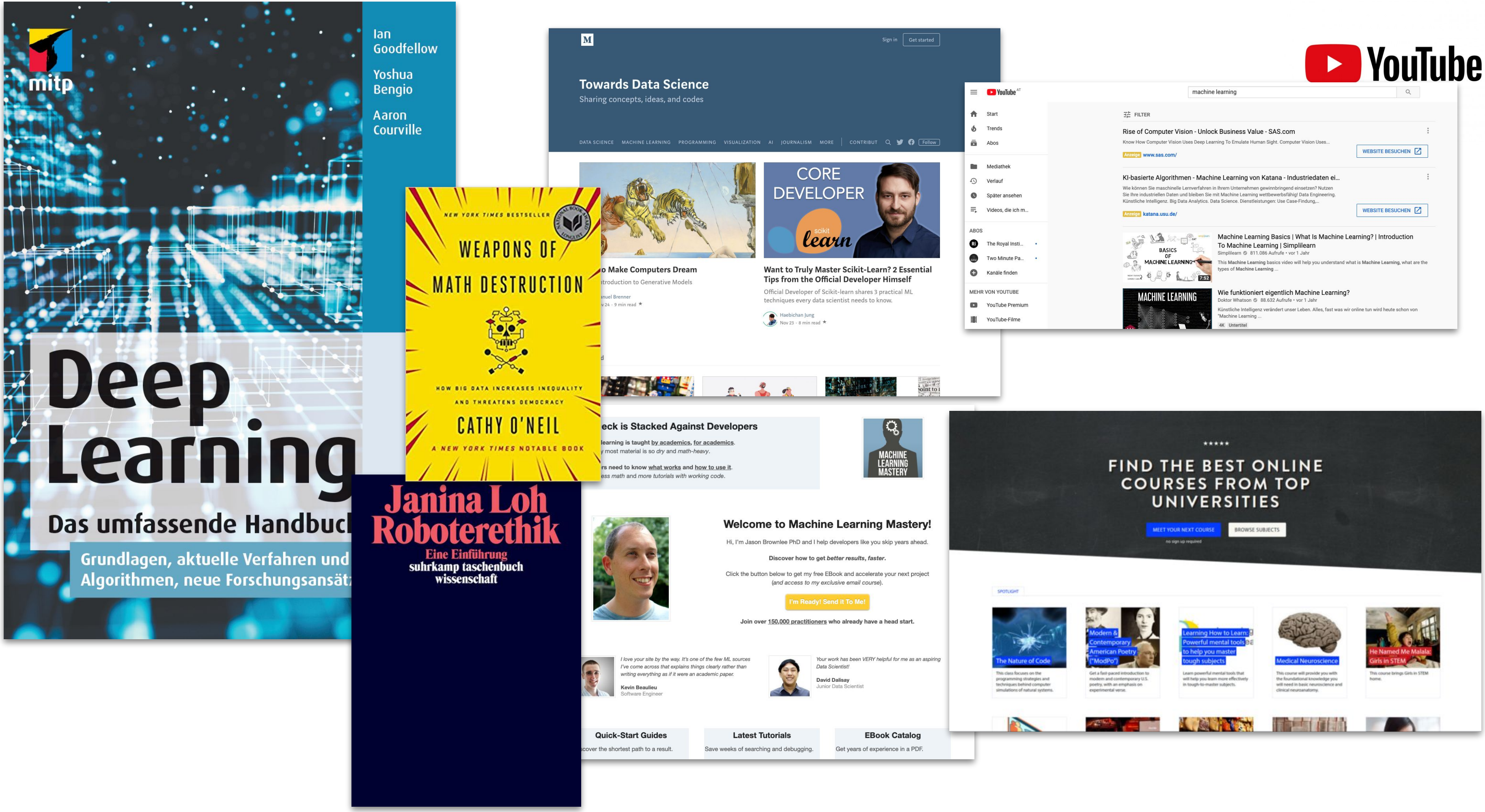
In order to identify such variations fast, it is required to monitor the resulting quality of a model permanently.

The following steps are to be considered:

- **data health** monitoring
  - distribution of the input data
  - min/max value changes
  - data outages
- prediction monitoring
  - **distribution of the results** (predictions)
  - **mean confidence** of the results
- system monitoring
  - load (CPU/memory/GPU)
  - response time
  - requests
- end-to-end monitoring
  - a good idea is also to use an external agent to send requests to the server with known data and to compare the prediction results with the expected results

https://christophergs.com/machine%20learning/2020/03/14/how-to-monitor-machine-learning-models/

How to Stay Informed and Keep on Learning


mitp
Ian Goodfellow
Yoshua Bengio
Aaron Courville
Deep Learning
Das umfassende Handbuc
Grundlagen, aktuelle Verfahren und
Algorithmen, neue Forschungsansät
NEW YORK TIMES BESTSELLER
WEAPONS OF MATH DESTRUCTION
HOW BIG DATA INCREASES INEQUALITY AND THREATENS DEMOCRACY
CATHY O'NEIL
A NEW YORK TIMES NOTABLE BOOK
Janina Loh
Roboterethik
Eine Einführung
suhrkamp taschenbuch wissenschaft
Towards Data Science
Sharing concepts, ideas, and codes
DATA SCIENCE MACHINE LEARNING PROGRAMMING VISUALIZATION AI JOURNALISM MORE CONTRIBUT
CORE DEVELOPER
o Make Computers Dream
Want to Truly Master Scikit-Learn? 2 Essential Tips from the Official Developer Himself
Official Developer of Scikit-learn shares 3 practical ML techniques every data scientist needs to know.
YouTube
machine learning
Rise of Computer Vision - Unlock Business Value - SAS.com
KI-basierte Algorithmen - Machine Learning von Katana - Industriedaten ei...
Machine Learning Basics | What Is Machine Learning? | Introduction To Machine Learning | Simplilearn
Wie funktioniert eigentlich Machine Learning?
eck is Stacked Against Developers
MACHINE LEARNING MASTERY
Welcome to Machine Learning Mastery!
Hi, I'm Jason Brownlee PhD and I help developers like you skip years ahead.
Discover how to get better results, faster.
Click the button below to get my free EBook and accelerate your next project (and access to my exclusive email course).
I'm Ready! Send it To Me!
Join over 150,000 practitioners who already have a head start.
Quick-Start Guides
Latest Tutorials
Save weeks of searching and debugging.
EBook Catalog
Get years of experience in a PDF.
FIND THE BEST ONLINE COURSES FROM TOP UNIVERSITIES
MEET YOUR NEXT COURSE
BROWSE SUBJECTS
The Nature of Code
Modern & Contemporary American Poetry ("ModPo")
Learning How to Learn: Powerful mental tools to help you master tough subjects
Medical Neuroscience
He Named Me Malala: Girls in STEM

thank you

I hope you will find some cool machine learning projects

Please send questions and comments to:
dietmar.millinger@aiaustria.com