

---

---

**Information technology — Artificial  
intelligence (AI) — Overview of  
computational approaches for AI  
systems**





**COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
CP 401 • Ch. de Blandonnet 8  
CH-1214 Vernier, Geneva  
Phone: +41 22 749 01 11  
Email: [copyright@iso.org](mailto:copyright@iso.org)  
Website: [www.iso.org](http://www.iso.org)

Published in Switzerland

# Contents

Page

<b>Foreword</b> .....	<b>v</b>
<b>Introduction</b> .....	<b>vi</b>
<b>1 Scope</b> .....	<b>1</b>
<b>2 Normative references</b> .....	<b>1</b>
<b>3 Terms and definitions</b> .....	<b>1</b>
<b>4 Abbreviated terms</b> .....	<b>2</b>
<b>5 General</b> .....	<b>3</b>
<b>6 Main characteristics of AI systems</b> .....	<b>5</b>
6.1 General.....	5
6.2 Typical characteristics of AI systems.....	6
6.2.1 Adaptable.....	6
6.2.2 Constructive.....	6
6.2.3 Coordinated.....	6
6.2.4 Dynamic.....	6
6.2.5 Explainable.....	6
6.2.6 Discriminative or generative.....	6
6.2.7 Introspective.....	6
6.2.8 Trained or trainable.....	7
6.2.9 Accommodating various data.....	7
6.3 Computational characteristics of AI systems.....	7
6.3.1 Data-based or knowledge-based.....	7
6.3.2 Infrastructure-based.....	7
6.3.3 Algorithm-dependent.....	8
6.3.4 Multi-step or end-to-end learning-based.....	9
<b>7 Types of AI computational approaches</b> .....	<b>9</b>
7.1 General.....	9
7.2 Knowledge-driven approaches.....	10
7.3 Data-driven approaches.....	10
<b>8 Selected algorithms and approaches used in AI systems</b> .....	<b>11</b>
8.1 General.....	11
8.2 Knowledge engineering and representation.....	11
8.2.1 General.....	11
8.2.2 Ontology.....	12
8.2.3 Knowledge graph.....	12
8.2.4 Semantic web.....	14
8.3 Logic and reasoning.....	14
8.3.1 General.....	14
8.3.2 Inductive reasoning.....	15
8.3.3 Deductive inference.....	15
8.3.4 Hypothetical reasoning.....	16
8.3.5 Bayesian inference.....	17
8.4 Machine learning.....	18
8.4.1 General.....	18
8.4.2 Decision tree.....	18
8.4.3 Random forest.....	19
8.4.4 Linear regression.....	20
8.4.5 Logistic regression.....	21
8.4.6 K-nearest neighbour.....	21
8.4.7 Naïve Bayes.....	22
8.4.8 Feedforward neural network.....	22
8.4.9 Recurrent neural network.....	23

8.4.10	Long short-term memory network.....	24
8.4.11	Convolutional neural network .....	25
8.4.12	Generative adversarial network.....	26
8.4.13	Transfer learning .....	27
8.4.14	Bidirectional encoder representations from transformers.....	27
8.4.15	XLNet .....	28
8.5	Metaheuristics .....	29
8.5.1	General .....	29
8.5.2	Genetic algorithms .....	29
<b>Bibliography.....</b>		<b>31</b>

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives-or-www.iec.ch/members\\_experts/refdocs](http://www.iso.org/directives-or-www.iec.ch/members_experts/refdocs)).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see [www.iso.org/patents](http://www.iso.org/patents)) or the IEC list of patent declarations received (see <https://patents.iec.ch>).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see [www.iso.org/iso/foreword.html](http://www.iso.org/iso/foreword.html). In the IEC, see [www.iec.ch/understanding-standards](http://www.iec.ch/understanding-standards).

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 42, *Artificial intelligence*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](http://www.iso.org/members.html) and [www.iec.ch/national-committees](http://www.iec.ch/national-committees).

## Introduction

Artificial intelligence (AI)-related products, systems and solutions have become more common in recent years thanks to rapid software and hardware improvements that boost computational performance, data storage capabilities and network bandwidth. The intent of this document is to look at computational methods and approaches within AI systems. Based on ISO/IEC 22989<sup>1)</sup>, ISO/IEC 23053<sup>2)</sup> and ISO/IEC TR 24030, this document provides a description of the characteristics of an AI system and its computational approaches. The illustration of computational approaches in AI systems includes both machine learning and non-machine learning methods. To reflect state-of-the-art methods used in AI, this document is structured as follows:

- [Clause 5](#) provides an overall description of computational approaches in AI systems;
- [Clause 6](#) discusses the main characteristics of AI systems;
- [Clause 7](#) provides a general taxonomy of computational approaches, including knowledge-driven and data-driven approaches;
- [Clause 8](#) discusses selected algorithms used in AI systems, including basic theories and techniques, main characteristics and typical applications.

By giving an overview of different technologies used by AI systems, this document is intended to help users understand computational characteristics and approaches used in AI.

---

1) Under preparation. Stage at the time of publication: ISO/IEC DIS 22989:2021.

2) Under preparation. Stage at the time of publication: ISO/IEC DIS 23053:2021.

# Information technology — Artificial intelligence (AI) — Overview of computational approaches for AI systems

## 1 Scope

This document provides an overview of the state of the art of computational approaches for AI systems, by describing: a) main computational characteristics of AI systems; b) main algorithms and approaches used in AI systems, referencing use cases contained in ISO/IEC TR 24030.

## 2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 22989, *Information technology — Artificial intelligence — Artificial intelligence concepts and terminology*

ISO/IEC 23053, *Framework for artificial intelligence (AI) systems using machine learning (ML)*

## 3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 22989 and ISO/IEC 23053 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at <https://www.iso.org/obp>

— IEC Electropedia: available at <https://www.electropedia.org/>

### 3.1

#### **heuristic search**

search, based on experience and judgment, used to obtain acceptable results without guarantee of success

[SOURCE: ISO/IEC 2382:2015, 2123854, modified — Notes to entry removed.]

### 3.2

#### **fuzzy logic**

#### **fuzzy-set logic**

nonclassical logic in which facts, inference rules and quantifiers are given certainty factors

[SOURCE: ISO/IEC 2382:2015, 2123795, modified — Notes to entry removed.]

### 3.3

#### **generator**

neural network that produces samples usually to be classified by a discriminator

Note 1 to entry: Generators primarily appear in the context of generative adversarial networks.

**3.4**  
**discriminator**

neural network that classifies samples usually produced by a generator

Note 1 to entry: Discriminators primarily appear in the context of generative adversarial networks.

**3.5**  
**generative adversarial network**  
**GAN**

neural network architecture comprised of one or more generators and one or more discriminators that compete to improve model performance

**3.6**  
**platform**

combination of an operating system and hardware that makes up the operating environment in which a program runs

[SOURCE: ISO/IEC/IEEE 26513:2017, 3.30]

**3.7**  
**perceptron**

neural network consisting of one artificial neuron, with a binary or continuous output value that is determined by applying a monotonic function to a linear combination of the input values and with error-correction learning

Note 1 to entry: The perceptron forms two decision regions separated by a hyperplane.

Note 2 to entry: For binary input values, the perceptron cannot implement the non-equivalence operation (EXCLUSIVE OR, XOR).

[SOURCE: ISO/IEC 2382:2015, 2120656, modified — term revised, “or continuous” added to definition and Notes 3 and 4 to entry removed.]

## **4 Abbreviated terms**

AI	artificial intelligence
ASIC	application-specific integrated circuit
BERT	bidirectional encoder representations from transformers
BPTT	back propagation through time
CNN	convolutional neural network
CPU	central processing unit
DAG	directed acyclic graph
DNN	deep neural network
ERM	empirical risk minimization
FFNN	feedforward neural network
FPGA	field programmable gate array
GDM	gradient descent method
GPU	graphics processing unit

GPT	generative pre-training
IoT	internet of things
KG	knowledge graph
KNN	k-nearest neighbour
LSTM	long short-term memory
MFCC	Mel-frequency cepstrum coefficient
MLM	masked language model
NER	named entity recognition
NLP	natural language processing
NSP	next sentence prediction
OWL	web ontology language
QA	question answering
RDF	resource description framework
RNN	recurrent neural network
RTRL	real-time recurrent learning
SPARQL	SPARQL protocol and RDF query language
SQL	structured query language
SRM	structure risk minimization
SVM	support vector machine
URI	uniform resource identifier
XML	extensible markup language

## 5 General

Advances in computational approaches are an important driving force in the maturation of AI to become capable of processing various tasks. Initial AI methods were primarily rules-based and knowledge-driven. More recently, data-driven methods such as neural networks have gained prominence. AI computational approaches continue to evolve in industry and academia and are an important consideration in AI systems.

Computational approaches for AI systems are often categorized based on various criteria. One such categorization is by the purpose of the AI system. This purpose-based categorization is adapted from studies of AI<sup>[4]</sup> and includes an exemplary categorization of common types.

- a) Search methods. These approaches can be further divided into various types of search: classical, advanced search algorithms, adversarial search and constraint satisfaction.
  - 1) Classical search algorithms solve problems by a search over some state space and can be divided into uninformed searches and heuristic searches, which apply a rule of thumb to guide and speed up the search.

- 2) Advanced search algorithms include those that search in a local subspace, those that are nondeterministic, those that search with partial observation of the search space and online versions of search algorithms.
  - 3) Adversarial search algorithms search in the presence of an opponent and are generally used in games. These include notable algorithms such as alpha-beta pruning and also include stochastic and partially observable variations.
  - 4) Constraint satisfaction problems are solved when each variable in the problem has a value that satisfies all the constraints.
- b) Logics, planning and knowledge. These approaches can be further divided into three cases: logics, planning and state space search, and knowledge representation.
- 1) Logics, such as propositional logic and first-order logic, are used in classical AI to represent knowledge. Problem solution in such computational systems involves inference over the logic using algorithms such as resolution.
  - 2) Planning in classical AI systems involves search over some state space as well as algorithmic extensions to deal with planning in the real world. Methods to deal with the complexity of real-world planning involve time and resource constraints, hierarchical planning where problems are solved at abstract levels first before fine-grain details, multi-agent systems that handle uncertainties and dealing with other agents in the system.
  - 3) Knowledge representation is a kind of data structure for describing knowledge using predicate logic, "if-then" generation and knowledge frame representation.
- c) Uncertain knowledge and reasoning. Approaches in this area deal with potentially missing, uncertain or incomplete knowledge. They generally use either probability or fuzzy logic to represent concepts. Probabilistic computational systems reason using Bayes rule, Bayesian networks or (in time-dependent situations) hidden Markov models or Kalman filters. Another set of computational approaches is used for decision-making, including those based on utility theory and decision networks.
- d) Learning. Computational approaches in this area deal with the problem of making the computer learn similarly to a human. Approaches can be grouped into learning from examples, knowledge-based learning, probabilistic learning, reinforcement learning, deep learning approaches, GANs and other learning approaches.
- 1) Learning from examples involves supervised learning approaches that learn a machine learning model from labelled data. It includes methods such as decision trees, linear and logistic regression approaches, artificial neural networks, non-parametric approaches (e.g. the KNN), SVMs and ensemble learning methods (e.g. bagging, boosting and variants of random forest).
  - 2) Knowledge-based learning approaches include logic-based approaches, explanation-based learning and inductive logic programming.
  - 3) Probabilistic learning involves computational approaches such as Bayesian methods and expectation-maximization methods.
  - 4) Reinforcement learning involves computational systems that receive feedback, make decisions and take actions in environments to maximize the overall reward. Notable algorithms include temporal difference-learning and Q-learning.
  - 5) Deep learning neural approaches involve modern computational approaches with many hidden layers, including deep feedforward networks, regularisation, modern optimization methods, CNNs and sequence learning methods such as LSTM networks.

- 6) GANs involve two competing networks, a generator and discriminator. The generator produces samples and the discriminator classifies each sample as real or fake. After this iterative process, trained generators can be used in applications such as creating artificial images.
  - 7) Other learning approaches include unsupervised learning, which involves identifying the natural structure of data sets; semi-supervised learning, which deals with partially labelled data sets; online learning algorithms, which continue to learn as they receive data; networks and relational learning, ranking and preference learning, representation learning, transfer learning and active learning.
- e) Inference. These approaches embody the application of an AI system in estimating parameters or aspects of (or classifying new or unobserved data based on) learned, acquired or defined parameters. Bayesian inference is the act of taking statistical inference from a Bayesian point of view. Approximate inferences, such as variational inference, solves the inference problem by taking the best approximation of the statistics. Monte Carlo algorithms generate samples from a known distribution that is difficult to normalize, then infer statistics from generated samples. Causal inference involves inferring the causal connections of the observed data.
  - f) Dimensionality reduction. These computational approaches involve reducing the number of dimensions of data by either dimensionality reduction (feature extraction) algorithms, which identify a new smaller number of attributes to represent data, or feature selection, which chooses a subset of the most appropriate attributes.
  - g) Communicating, perceiving and acting. Computation approaches in these areas are associated with the fields of NLP (including tasks such as language modelling, text classification, information retrieval, information extraction, parsing, machine translation and speech recognition), computer vision (including image processing and object recognition) and robotics.

These categories and subcategories are not mutually exclusive. For instance, deep learning approaches [d)5)] can be either supervised [d)1)] or unsupervised [d)7)], reinforcement learning [d)4)] can be achieved through deep learning [d)5)], and approaches for machine translation or object recognition [g)] can be learning approaches [d)].

ISO/IEC 22989 specifies concepts and terminologies relevant to AI computational approaches. ISO/IEC 23053 provides a framework for AI systems using machine learning, encompassing machine learning algorithms, optimization algorithms and machine learning methods. ISO/IEC TR 24030 collects and analyses AI use cases.

## 6 Main characteristics of AI systems

### 6.1 General

Not all AI systems are based on machine learning or neural networks. To demonstrate the breadth of AI systems, some frequently encountered characteristics of AI systems are described in 6.2 and 6.3. These characteristics are broadly conceptual and not tied to a specific methodology or architecture. In the aggregate these characteristics differentiate AI systems from non-AI systems.

Some characteristics of AI systems are common and apply widely to different use cases. Others are specific to a small number of use cases within a specific industry. This clause contains a list of characteristics of AI systems which is not exhaustive but contains attributes intrinsic to many AI systems. While the list is not limited to a specific base technology (such as AI systems built with neural networks), it does not encompass every type of dynamic AI system.

## 6.2 Typical characteristics of AI systems

### 6.2.1 Adaptable

Some AI systems adapt to different changes in itself and the environment in which it is deployed. Such adaptation depends on many factors, including data in the system's domain, architecture or other technical decisions made at its implementation.

AI systems often operate on server-side cloud computing environments with access to high-performance computing and other resources. With the growth of IoT systems capable of general-purpose computing on GPUs and multi-core CPUs, or AI processing on application-specific processors and accelerators, AI system adaptability now extends to IoT implementation considerations, such as near-real-time data processing, optimization for low latency and power-efficient performance.

### 6.2.2 Constructive

Some AI systems construct or generate a static or dynamic output based on specified input criteria. This applies to methods including unsupervised learning and generative learning.

### 6.2.3 Coordinated

Some AI systems coordinate between agents. Agents can also be AI systems in their own right, but do not need to be. Many simultaneous constraints can govern agent behaviour, including static or dynamic ways. Coordination can be exhibited either explicitly through direct negotiation among the systems or implicitly through reaction to changes in the environment.

### 6.2.4 Dynamic

Some AI systems exhibit dynamic decision-making based on external data sources. These data sources can come from other software platforms, from physical environments or from other sources.

### 6.2.5 Explainable

Some AI systems provide a mechanism to explain what precipitated a decision or output. This output can take many forms and can be explicit or implicit with respect to AI system design.

An explainable AI system can contribute to or complement trustworthiness, accuracy and efficiency. Explainability can also contribute to comparison and optimization of machine learning model performance by generating insights into factors that degrade performance. Explainability can be an important counter to deceptive behaviour in AI systems.

### 6.2.6 Discriminative or generative

Some AI systems are discriminative, designed primarily to distinguish between possible outputs such as by excluding prior probabilities. Alternatively, some AI systems are generative, designed primarily to represent relevant aspects of data, such as by including prior probabilities.

### 6.2.7 Introspective

Some AI systems self-monitor to adapt to their environment or to provide insight into their functionality, such as in an audit situation. This self-monitoring can be adaptable, situation-dependent or static, and can take different forms depending on the system architecture.

To support introspective AI systems, performance monitoring functionality collects and reports performance metrics regarding CPU, GPU or application-specific processor compute resources, memory and other system resource usage. This information can be used to configure AI system resources, such as memory allocation, kernel configuration and load balancing across a multi-processor or hybrid

hardware system, and enable an AI system to handle parallelization and acceleration for machine learning model training or inference.

### 6.2.8 Trained or trainable

Some AI systems are trained on a data set before deployment or trained dynamically (through adaptation) as the system is used. Systems with these characteristics have numerous possible system architectures (e.g. neural networks, hidden Markov models).

### 6.2.9 Accommodating various data

Some AI systems deal with large amounts of heterogeneous data that are structured or unstructured, static or streaming. AI systems can draw insights from varied data sets to help humans make better and more accurate decisions.

## 6.3 Computational characteristics of AI systems

### 6.3.1 Data-based or knowledge-based

A characteristic of data-based AI computational approaches is that the computational model is trained on one or more data sources to acquire knowledge.

Considerations for data used in AI systems include acquisition, storage and access.

- a) Data acquisition. An AI application's use case and task typically dictate the type of data to be acquired for training. Typical AI system tasks reflected in ISO/IEC 22989 and ISO/IEC 23053 include classification, categorization, (conceptual) clustering, regression, prediction, optimization, NLP (text or speech), perception and system control or behaviour guidance. Depending on the application and task, AI system developers can collect training data through intelligent hardware (e.g. smart bracelet, smart watch, smart phone), IoT sensors (e.g. gravity sensor, temperature sensor, humidity sensor), cameras, microphones or other sensors.
- b) Data storage. Collected data are stored in the format and structure consistent with the AI system application and task. Storage approaches and constraints can differ during training and evaluation. In addition, distributed and shared storage can be important data storage considerations.
- c) Data access. Rapid access and retrieval of large amounts of data is often necessary in AI systems. Load-balancing techniques are often used to address challenges in data concurrency and network overloading.

In addition to perception-based tasks and applications, cognitive intelligence has become an important aspect of AI systems in which cognitive computing is integrated with industrial knowledge. Using techniques such as NLP and KG, AI systems can reveal implicit knowledge and give insights into relations, logic or patterns that are not easily found by human observers.

**EXAMPLE** Using KGs, accumulated business process data can be converted into organizational experience and knowledge. This can be used in turn to reduce communication costs among different departments.

### 6.3.2 Infrastructure-based

AI systems can face simultaneous challenges in computing platform design optimization, computing efficiency in complex heterogeneous environments, highly parallel and scaled computing frameworks, and the computing performance of AI applications. One possible solution to such challenges is to use powerful infrastructures to provide computing capability.

Such infrastructures can include sensor, server, network, processor, storage and other elements. Silicon-based processors are often used for both training and reasoning in learning approaches. When handling large amounts of training data or complex DNN structures, the training processes typically need to execute large-scale calculations on multi-processor systems or processor or accelerator clusters.

Compared to training, inference is less computationally intensive, but can still require significant matrix operations. Training and inference have traditionally been implemented on cloud-based servers, though use cases that demand real-time processing can implement inference on edge devices.

Depending on the technical architectures, silicon-based processors for AI encompass general-purpose processors (e.g. CPU, GPU and FPGA), semi-customized processors based on FPGA, fully customized ASIC processors and brain-like computing processors. Vision processing units, deep-learning processing units, neural network processing units and other application-specific processors are also suitable for different AI scenarios and functions.

Sensors with microprocessors to collect, process and transmit information can be used to create a full awareness of the external environment. Large-scale sensor deployment and application can support data acquisition in AI applications. Further, specialized sensor requirements are needed for smart home, smart medical and smart security applications. The development of intelligent sensors for AI applications relies on important factors such as high precision, high reliability, miniaturization and integration, and high sensitivity.

### 6.3.3 Algorithm-dependent

ISO/IEC 22989 defines machine learning as a process of optimizing model parameters through computational techniques, such that the model's behaviour reflects the data or experience. Machine learning methods find patterns from observed data or samples and use these patterns to make predictions on input data without being explicitly programmed. Machine learning methods vary in part based on differences in learning approaches and computational frameworks.

A machine learning method includes three basic elements: loss functions, learning criteria and optimization algorithm. Differences across machine learning methods can be viewed as functions of these elements. For example, linear classification methods, such as perceptron, logistic regression and SVM, differ in terms of learning criteria and optimization algorithms.

- a) From a loss function perspective, machine learning methods can be classified as linear or nonlinear. A robust method needs a small expected risk or error, which uses loss function to quantify the difference between predicted data and real data. Common loss functions include 0-1 loss function, quadratic loss function, cross-entropy loss function, hinge loss function, mean absolute error loss function, Huber loss function, log-cosh loss function and quantile loss function.

Moreover, other broad categories of loss functions include ranking, distribution-based, classification and regression.

- b) Learning criteria of supervised learning includes ERM and SRM. An ERM reduces the average loss on training data set. An SRM avoids overfitting problems by introducing parameter regularization based on ERM to limit the model capability. Unsupervised learning has a variety of learning criteria. For example, maximum likelihood estimate is often used in density estimation, reconstruction error minimization is often used in unsupervised feature learning.
- c) The task of optimization is to find the optimum machine learning model. It consists of parameter optimization and hyper-parameter optimization. Common optimization algorithms include gradient descent method, early stopping, batch gradient descent, stochastic gradient descent, mini-batch gradient descent, gradient descent methods utilizing the coefficient of momentum, root mean square propagation and adaptive moment optimization.

In the face of massive data processing and complex knowledge reasoning, a large computing task is typically divided into smaller computing tasks. Such distributed computational frameworks are based on cloud computing, edge computing and big data technologies. The deep learning framework is the basic underlying computational framework for deep learning, which generally includes neural network architectures and a stable deep learning interface to support the distributed learning. Some frameworks can be transferred to run on multiple platforms such as cloud computing platforms and mobile devices.

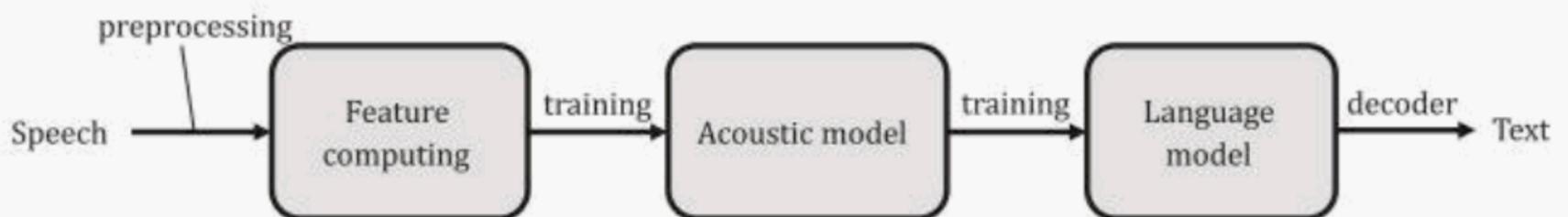
### 6.3.4 Multi-step or end-to-end learning-based

As opposed to multi-step learning, where a problem is divided into multiple stages to be solved step by step, end-to-end learning seeks to solve problems such that results are directly obtained from input data.

Machine learning processes often consist of several independent modules. For example, a typical NLP application includes segmentation, part of speech tagging, syntactic analysis, semantic analysis and other independent steps. Each step is an individual task, results from each step impact the next step, potentially affecting the entire training process.

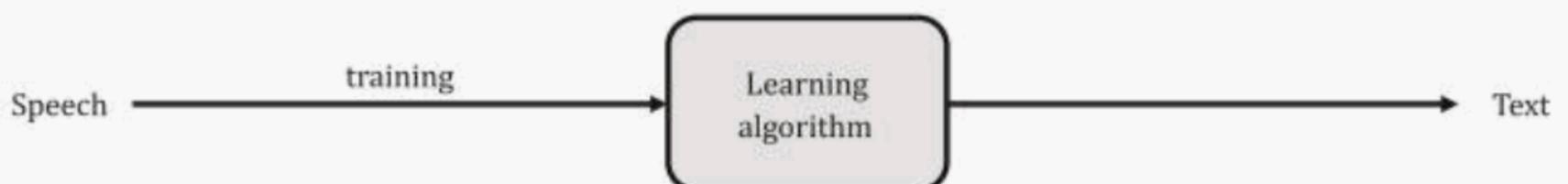
In end-to-end learning, as can be done with deep learning, a prediction result is obtained from the input to the output. Typically, errors are transferred through back propagation in each layer of the network. The representation of each layer is adjusted according to such errors until the network is convergent or achieves the desired performance. In such end-to-end processes, data labelling before each independent learning task is no longer needed.

Taking speech recognition as an example, in multi-step speech recognition as shown in [Figure 1](#), speech is converted into speech feature vectors (e.g. the MFCC features), groups of vectors are then classified into various phonemes using machine learning, the original texts of speech with maximum probability are finally restored through phonemes. In this process, feature vectors produced by the feature computing and phonemes are processed by the acoustic model. The acoustic model and language model are trained separately.



**Figure 1 — Multi-step learning-based speech recognition**

For end-to-end learning-based speech recognition as shown in [Figure 2](#), the entire process from feature extraction to phoneme expression can be directly completed by a DNN. Given enough labelled training data, including pairs of voice data and text data, at the beginning of recognition process, the end-to-end learning-based speech recognition can show good performance.



**Figure 2 — End-to-end learning-based speech recognition**

## 7 Types of AI computational approaches

### 7.1 General

Computational approaches in AI can be decomposed into knowledge-driven and data-driven.

Knowledge-driven AI computational approaches primarily consist of a series of rule-based methods. Taking expert system as an example, learning, reasoning and decision-making for a use case are realized through a set of conceptualized objects and “if-then” logic rules. A large knowledge base storing extensive knowledge from domain experts is used to support the expert system.

By contrast, data-driven AI computational approaches use large amounts of data as the fundamental resources processed by algorithms to simulate human thinking and decision-making processes. Typical data-driven computational approaches include machine learning, which can be decomposed as linear or logistic regression, probabilistic graphical model, decision tree, neural networks and other approaches.

Figure 3 shows this decomposition of AI computational approaches.

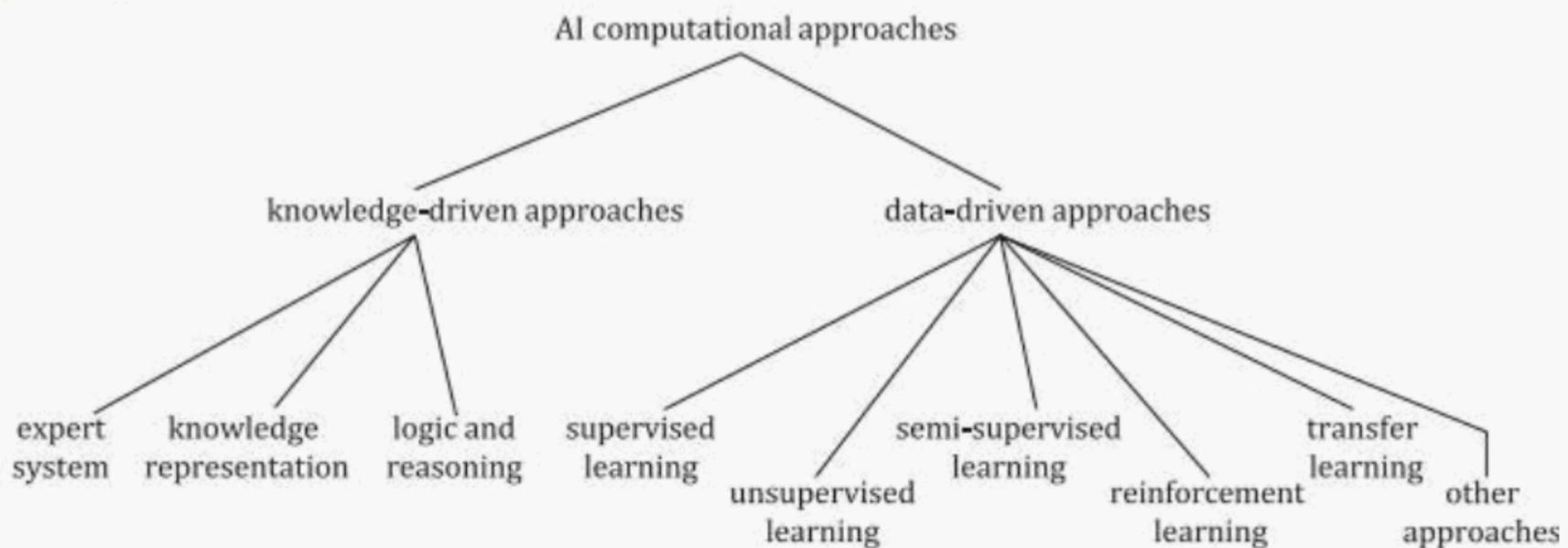


Figure 3 — AI computational approaches

## 7.2 Knowledge-driven approaches

Knowledge-driven approaches mimic the functions of human intelligence from symbols and logic rules. The human cognitive process is regarded as a symbolic operation process. This kind of approach has two basic assumptions:

- a) information is represented as symbols;
- b) symbols are manipulated by explicit rules (such as logical operations).

## 7.3 Data-driven approaches

Data-driven approaches rely on data in their AI computational model. Various machine learning approaches are associated with different types of data, as described in ISO/IEC 23053. These approaches include: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, hybrid learning, statistical inference, multi-task learning, active learning, transfer learning, ensemble learning and online learning.

Typical learning approaches are described in this clause.

- Supervised learning. Supervised learning approaches establish machine learning models using labelled training data. Machine learning models then predict the classification or mapping of input data. As training data labels become richer and more accurate, machine learning model predictions and analytics derived from these predictions are more useful and reliable. Typical machine learning algorithms for supervised learning include regression and classification. Supervised learning is used in NLP, information retrieval, text mining, handwriting recognition, spam detection and other fields.
- Unsupervised learning. Unsupervised learning approaches map inputs to outputs using unlabelled training data. Unsupervised learning does not need labelled training data, which can reduce data storage and calculation demands, improve algorithmic speed and avoid classification errors caused by mislabelled training data. Typical unsupervised machine learning algorithms<sup>[5]</sup> include single-class density estimation, single dimensionality reduction for featuring and clustering. Unsupervised learning is used in economic prediction, anomaly detection, data mining, image processing, pattern recognition and other fields.

- Semi-supervised learning. Semi-supervised learning approaches use both labelled data and unlabelled data. By training on a small amount of labelled data and a large amount of unlabelled data, such algorithms can maximize the usefulness of limited amounts of labelled training data. Typical semi-supervised learning algorithms include self-training and co-training.
- Reinforcement learning. Reinforcement learning approaches involve an agent interacting with its environment to achieve a predefined goal. These approaches learn mappings from an environment to a behaviour to maximize the functional value of a reinforcement signal. In cases where the external environment provides little information, reinforcement learning algorithms rely on their own experience to learn. Reinforcement learning has been successfully applied to industrial control, chess gaming, robot control, automated driving and other fields.
- Transfer learning. Transfer learning approaches use stored, abstracted knowledge gained from data in an application domain and apply it to the tasks in different application domains. Transfer learning is often used when adequate data are not available in a given application domain to reliably train models. Transfer learning is suitable for applications with a limited number of variables, such as text classification, sensor network for location tasks and image classification.

Machine learning starts from observations or training data, attempting to find patterns beyond those obtained through principal analysis to realize accurate predictions. Machine learning algorithms include logistic regression, hidden Markov, SVM, KNN, Adaboost, Bayesian network, decision tree and many others. Results from machine learning approaches are typically explainable with respect to the behaviour of machine learning algorithms and models.

Machine learning provides a framework for training with limited data availability and can be used for regression analysis, pattern classification, probability density estimation and other tasks. Statistics is an important theoretical basis of machine learning, used widely in fields such as NLP, speech recognition, image recognition, information retrieval and biological informatics.

Neural networks are networks of neuron layers connected by weighted links with adjustable weights. Neural networks take input data and produce an output, often a prediction. Neural networks with several hidden layers are referred to as deep learning. Deep learning can be seen as an approach to combine both feature representation and learning. Deep learning is often less explainable than conventional machine learning. Typical deep learning approaches include deep belief network, CNN, restricted Boltzmann machine and RNN. CNNs are often used for spatial distribution data, while RNNs are often used for temporal distribution data based on their use of memory and feedback from previous layers.

## **8 Selected algorithms and approaches used in AI systems**

### **8.1 General**

ISO/IEC TR 24030 provides a template through which detailed information on AI use cases is collected. The template includes descriptions of use case features including tasks, methods, platforms, topology, terms and concepts used. Deep learning, machine learning and neural networks are among the most frequently mentioned computational approaches in AI use cases collected in ISO/IEC TR 24030.

In this document, a state of the art of selected algorithms and approaches used in AI system is described in terms of theories and techniques, main characteristics and typical applications.

### **8.2 Knowledge engineering and representation**

#### **8.2.1 General**

Knowledge engineering and representation is a very broad field in AI concerned with expressing knowledge in forms which are machine processable and using machine processing to perform knowledge-related tasks, most particularly reasoning in all its forms.

There are many philosophical and practical decisions involved in how knowledge is represented for machine processing in order to be both accurate and useful for whatever purposes people have in mind, thus design and engineering are essential elements of the process. All knowledge representation and reasoning-based projects commit to some kind of knowledge model, whether explicitly realized or not. The choice of knowledge model is a key factor in whether or not project objectives can be met. This field is still being developed, especially with respect to the engineering implications of the choices.

### 8.2.2 Ontology

#### 8.2.2.1 Theories and techniques

“Ontology” is a term which originates from philosophy, where it refers to what people assert to exist. In a sense it is relevant to the knowledge structures known as ontologies, as philosophically speaking, it delineates a model of the world which implicitly is a statement of what exists. However, an “ontology” as a knowledge structure is essentially a knowledge model of whatever domain people wish to declare knowledge about. In the context of information technology, “ontology” generally means an explicit knowledge structure which has a formal logical model and supports reasoning. The most prevalent kind of ontology as knowledge structure is that implemented by the W3C’s semantic web technology stack based on RDF<sup>[6]</sup> and OWL<sup>[7]</sup>.

#### 8.2.2.2 Main characteristics

As a foundational method of cognitive AI, ontology focuses on the conceptual classification of entities and the relationship among different concepts. It provides formal language, common definitions, a logical relationship and a stable conceptual model for knowledge used in AI. It functions as a knowledge description schema which enables AI systems to acquire, reuse and share knowledge.

An ontology model consists of specifically defined concepts that relate to compositional elements such as structure, entity, term, attribute, function and axiom. Formal descriptive languages such as XML, RDF(s) and OWL are used in ontology engineering.

Ontology is a kind of abstract, cognitive model for the real world. Within an ontology, objects’ definitions, attributes and inter-relationships are specifically illustrated. This enables underlying general or domain knowledge to be easily acquired and processed by computers and machines.

#### 8.2.2.3 Typical applications

Ontologies are widely used in knowledge engineering, such as in KGs, information searches and QA tasks.

### 8.2.3 Knowledge graph

#### 8.2.3.1 Theories and techniques

A KG can be seen as a knowledge structure composed of nodes and links. The word “graph” refers to the mathematical concept of a graph. The nodes in the graph represent things (entities) and the links represent relationships between the things. It is considered to be a knowledge structure because a combination such as entity1 - > relationship R - > entity2 is taken to be an assertion that entity1 is in relationship R with entity2.

**EXAMPLE** Anne - > motherOf - > David asserts that Anne is the mother of David. This relationship is directional, in that Anne is the mother of David but David is not the mother of Anne.

KGs are a distinct contrast to tabular data structure. They are a more flexible, extensible approach which supports more advanced forms of reasoning. KGs are not yet widely adopted as they are slightly more challenging to manage than the traditional approaches and are not yet widely taught in tertiary institutions.

### 8.2.3.2 Main characteristics

In general, a KG is simply a kind of semantic network<sup>[8]</sup>. It is a graph-based data structure consisting of nodes and edges. The whole graph expresses various entities, concepts and semantic relationships.

Compared with conventional knowledge representation methods (e.g. ontology or semantic networks), a KG is characterized by:

- full coverage of entities and concepts;
- diverse semantic relations;
- easy-to-use structure;
- generally high quality of knowledge representation.

KG has become the most important knowledge engineering approach in AI systems. The process of building a KG can enable machines with cognition capabilities.

The underlying computational process of KG generally includes knowledge extraction, knowledge representation, knowledge storage, knowledge modelling, knowledge fusion and knowledge computation.

- a) Knowledge extraction: knowledge is extracted from structured, semi-structured and unstructured data, especially from text data. According to different objects, knowledge extraction includes entity extraction (e.g. NER), relation or property extraction and event extraction.
- b) Knowledge representation: knowledge representation is a kind of data structure for describing knowledge using predicate logic, “if-then” generation and framework representation.
- c) Knowledge storage: the objects of knowledge storage include basic attribute knowledge, relation knowledge, event knowledge, temporal knowledge and resource knowledge. Common storage methods are table-based and graph-based. Specifically, graph-based knowledge storage includes property graphs, RDF and hypergraph methods.
- d) Knowledge modelling: the goal of knowledge modelling is to build a data model for a KG, which is essential for the entire KG construction. There are top-down and bottom-up methods for modelling. The top-down approach defines the data schema or ontology then gradually refines it to form a well-structured hierarchical classification. The bottom-up approach summarizes and organizes existing entities to form foundational concepts and then gradually abstracts them to form higher-level concepts.
- e) Knowledge fusion: knowledge fusion usually extracts knowledge from data level and concept level. The data-level fusion focuses on entity linking and entity resolution, while the concept-level fusion focuses on ontology alignment and cross-language fusion.
- f) Knowledge computation: knowledge computation aims to obtain implicit knowledge based on the information provided by a KG. Examples include:
  - 1) using ontology or rule-based reasoning approaches to abstract concepts and entities;
  - 2) using entity linking approaches to predict implicit relationships between entities;
  - 3) using social computing approaches to structure entities into a KG and provide knowledge-related paths.

### 8.2.3.3 Typical applications

KG is widely used in knowledge retrieval, intelligent recommendation and QA tasks.

## 8.2.4 Semantic web

### 8.2.4.1 Theories and techniques

The semantic web is a worldwide web of knowledge. It is a knowledge graph which utilizes the semantic web technology stack based on RDF. All entities and relationships in the semantic web have URIs, so they can be located anywhere on the web (though knowledge that organizations generate using semantic web technologies is not necessarily made public).

As the semantic web technologies are graph-based, the basic unit of knowledge is an RDF triple: entity - > relationship - > entity. URIs and triples combine to form a graph of URI nodes and links: RDF, RDF-schema and OWL.

The semantic web technology stack is a W3C standard and is an open standard. The stack includes SPARQL<sup>[9]</sup>, a query language analogous to SQL<sup>[10]</sup> and shapes constraint language<sup>[11]</sup>, a language based on the idea of shapes used to ensure that the knowledge graph conforms to the desired enforced constraints. Like the web, the semantic web is decentralised, scalable, extensible and flexible. Unlike the web, it is designed to be processed directly by machines rather than consumed by humans.

### 8.2.4.2 Main characteristics

The semantic web is based on formal semantics known as “description logics” which is a deductive form of logic. Assertions can have logical characteristics and relationships between entities can be transitive or reflexive, which supports reasoning (e.g. Sarah is taller than Anne, Anne is taller than David, therefore Sarah is taller than David). There are various reasoners available to perform reasoning functions.

Key ideas in RDF are classes, subclasses and instances. RDF has two types of properties (relationships): those that relate classes to other classes and those that relate classes to literals.

**EXAMPLE** Parent is a subclass of the class human. Anne is an instance of parent. Inference from this can be that Anne is human. This is a form of syllogistic reasoning and is supported through class inheritance.

Examples of syntaxes for the semantic web include:

- XML-based syntaxes whose usability varies (.ttl or Turtle is generally preferred by practitioners);
- formal logic-based syntaxes that resemble logical formulae;
- syntaxes that are partly in natural language (Manchester syntax) for improved human readability;
- syntaxes that are fully readable as natural language (fully defined but not yet implemented by tools such as Sydney syntax), which help domain experts perform knowledge checking without having to understand technical syntax.

### 8.2.4.3 Typical applications

Typical uses for semantic web include development of improved search capabilities and incremental modelling in healthcare applications.

## 8.3 Logic and reasoning

### 8.3.1 General

Logic and reasoning are to do with using the existing knowledge to generate more knowledge and ensuring it has been done correctly and reliably. Formally it belongs to the area of epistemology in philosophical enquiry, although informally there are many common-sense methods. It is fundamentally important for mathematics, science and any area of life where veracity matters. AI offers the prospect of using machines to automate the process, thus smoothing the generation of knowledge and checking

veracity. Leveraging machine capabilities for logic and reasoning is currently still under development and needs more attention.

### 8.3.2 Inductive reasoning

Inductive reasoning concerns the notion of generalizing from examples. It is commonly used in many everyday scenarios. It is not used in pure mathematics, which is the domain of deduction alone. Generally speaking, it starts with examples which are observations of some kind and uses them to create a theory. Given enough supporting evidence, some theories become generally accepted rules.

For example, scientists have studied many life forms, and all of them have been found to have DNA. Scientists probably generalize from this the theory that “all life forms have DNA”. While it accurately describes all life forms studied to date, people do not know whether it is true for all life forms at all times and places. It always remains a logical possibility that people will one day study a life form and find that it does not have DNA.

This is called a “black swan” event. Prior to the European colonisation of Australia, European naturalists had only ever encountered white swans, thus they believed that all swans are white. However, in western Australia, there are black swans, and when this was discovered by the naturalists of European origin, their theory was proved false and they were forced to revise it.

There are cases in which an inductive theory becomes a fact, but only when it concerns something that can be empirically proven: they are bounded in time and space rather than being abstract generalisations. For instance, based on many kinds of observations, it is theorised that the earth is round. It took some time but eventually it was conclusively proven, through empirical methods, and has been accepted as a fact for some hundreds of years (to everyone who understands basic science). The heliocentric orbit of the earth is another example.

There are also inductive theories which are accepted as scientific laws. These are abstract generalizations that will not be empirically proven for all times and places but are believed to be true at all times and places. An example is the conservation of mass. People believe this to be a fundamental law of physics that is unbreakable, although technically it is still based on inductive reasoning.

### 8.3.3 Deductive inference

Deductive reasoning is a form of reasoning which starts with a set of propositions or assertions known as “premises” or “axioms” and uses only reasoning methods which guarantee that if the premises are true, any conclusions made are also true. It is also necessary that the terms have clear meaning, e.g. the referents are clear. A proposition is a statement which asserts something; it can be evaluated as either true or false.

Historically, deductive reasoning patterns have been known since the times of Aristotle. There are several recognized forms of deductive argument and these have been known since antiquity.

a) Propositional logic.

1) Modus ponens form (the law of detachment):

premise 1: if P then Q

premise 2: P

conclusion: Q (where P and Q are assertions).

2) Modus tollens form (the law of contrapositive), which reasons in the opposite direction to modus ponens:

premise 1: if P then Q

premise 2: not Q

conclusion: not P.

3) Transitivity:

premise 1: if P then Q

premise 2: if Q then R

conclusion: if P, then R.

The forms relate to propositional logic, which is the simplest form of logic, dealing with propositions. It uses logical connectives (and, or, not) to combine propositions into compound forms. There are truth tables which help people to evaluate compound forms as true or false based on whether the component propositions are true or false, e.g. if P is true and Q is false, then (P and Q) is false, but (P or Q) is true, not (P) is false and not (Q) is true.

There are more complicated forms of logic which also use deductive methods. These use the same logical connections (and, or, not) as propositional logic.

b) First order logic.

First order logic is also known as predicate logic or quantification logic. The key difference between first order logic and propositional logic is that first order logic uses variables and quantifiers. The quantifiers used are “all” and “exists”. Whereas in propositional logic one only makes assertions such as “Socrates is a person”, in first order logic one can say “there exists an x such that x is Socrates and x is a person”. This enables new deductive forms of reasoning. There are also higher forms of logic, which add expressivity but lose reasoning ability.

c) Other forms of logic which support deductive methods.

- 1) Modal logic: modal logics use modifiers to qualify statements. The traditional form uses the notions of possibility and necessity as modals, but there are also modals that are temporal (relate to time), deontic (relate to obligation and permission), epistemic (knowledge) and doxastic (belief). The semantic use frames containing “possible worlds” where every iteration of truth value over all the variables are enumerated. Possible worlds are characterized by their values and binary accessibility relations between them.

Binary relationships can themselves have logical properties such as being reflexive, symmetric and transitive (note these are used in the semantic web). The system of logic is characterized by the properties of its relations.

- 2) Spatial logic: if A is in front of B and B is in front of C, then A is in front of C. “In front of” is taken to be a transitive relationship. An observer viewpoint is implied as the concept of “front” is relative.
- 3) Temporal logic: if A happens after B and B happens after C, then A happens after C. Here A, B and C are all events and “happens after” is assumed to be a transitive relationship (e.g. assuming Newtonian physics frame of reference or a single observer in a relativity scenario).
- 4) Mathematics: everything in mathematics is supposed to be deductively true. Several well-known proof patterns are by contradiction.

### 8.3.4 Hypothetical reasoning

Hypothetical reasoning is widely used in science and mathematics but can be applied in any aspect of life. It is used in forensics, legal arguments and everyday reasoning.

As its starting point, it takes an assertion which is potentially true or untrue, but one does not know which. This is known as the “hypothesis” and in science and mathematics it is usually a scientific or mathematical theory which people want to test.

The hypothesis makes certain predictions, that is, things which logically follow from it. Ideally these are things which can be empirically tested, but in many cases, they are not and one then says that the theory is “unfalsifiable”, i.e. there are no tests that can be made which show it to be false.

The form of reasoning used is modus tollens<sup>[12]</sup>, which is:

hypothesis: P

premise: if P then Q.

The conclusion now rests on whether Q turns out to be true. If it is false, then by modus tollens, P is false and the hypothesis P have been falsified. This is a deductive conclusion: it is guaranteed that P is not true. If Q turns out to be true, it does not prove that P is true, it is merely consistent with P being true. One then needs to come up with another premise and another condition to test.

**EXAMPLE** Hypothesis: it has been raining this morning. Premise: if it has been raining this morning, the path will be wet. Empirical observation: is the path wet? The empirical result can be yes or no (this can be drawn as decision tree branch).

- Empirical result 1: no; conclusion: it has not been raining this morning (if the premise is correct and observation is valid, this is deductively true).
- Empirical result 2: yes; conclusion: none.

The path being wet is logically consistent with it having rained this morning, but it doesn't prove it. Perhaps the path is wet because it was hosed, because there was a water fight, or perhaps an elephant was bathing itself (while the last option sounds implausibly unlikely in most scenarios, it is not impossible. But, an “open world assumption” is always needed, until people know the facts).

In everyday life, people often take the truth of Q as proof of the truth of P, which is a logical fallacy known as “affirming the consequent”<sup>3</sup>.

In mathematics, the pattern of taking a hypothesis, assuming it is true and reasoning to a false conclusion is called a “proof by contradiction”. It shows that the hypothesis is not true by showing that the hypothesis and conclusion are not logically consistent with each other.

### 8.3.5 Bayesian inference

Bayesian inference is a form of statistical inference. It is noted that statistical inference is an entirely separate endeavour to mathematical inference. Pure mathematics concerns only deductive inference: if a mathematical proof is valid then the conclusions always logically follow from the premises<sup>4</sup>.

The subject matter of statistics is probability, thus statistical inference primarily concerns the notion of likelihood and the probability that a certain future event will or won't happen. The computation makes use of the idea of randomness and distribution models. For instance, if someone tosses a coin, there is a probability of 50 % that it will land heads up and 50 % that it will land tails up. However, the result on any given toss is subject to randomness. Over 20 tosses it can be highly improbable to land heads up 20 times and far more likely to land heads up around 10 times, thus a bell-shaped probability distribution forms, peaking at 10.

The likelihood that something will happen (or has happened) is often affected by whether a certain condition applies. For instance, if the grass is wet then it is likely it has rained that day. However, it

---

3) Diagnostic reasoning: doctors often take a cluster of symptoms to indicate the presence of a specific condition and make a diagnosis saying that “symptoms X, Y and Z are consistent with condition C”. Technically, this is “affirming the consequent”: the diagnostician is assuming there is no other possible explanation for the symptoms. See 8.3.5.

4) Judging whether a mathematical proof is valid is an interesting exercise as it relies on one's mathematical reasoning abilities, but generally speaking, if a proof is valid then all suitably qualified mathematicians can agree that it is so. In some fields, such as number theory, some hypotheses have testable predictions about how numbers behave, but in most fields of pure mathematics there is nothing that can be empirically tested as a cross-check to the proof. See 8.3.4.

only increases the probability, it doesn't prove it. There are obviously other explanations, such as water has been sprinkled on the grass. Another way to look at it is in terms of belief revision in the light of new evidence. For instance, take the belief that a professor is in the office. If someone sees the professor's jacket is not on the back of the chair as it was earlier in the day, it can be necessary to revise the assessment of the probability that the professor is in the office. This is the key idea behind Bayesian inference, which is based on Bayes' theorem, that the posterior probability is proportional to the prior probability multiplied by the likelihood.

## 8.4 Machine learning

### 8.4.1 General

Machine learning approaches have drawn the attention of industry and academia. Machine learning algorithms and approaches are mentioned in most ISO/IEC TR 24030 use cases. Learning methods such as decision tree, random forest, linear or logistic regression, KNN, Naïve Bayes and neural networks-related approaches are introduced in this clause.

### 8.4.2 Decision tree

#### 8.4.2.1 Theories and techniques

Decision trees are supervised machine learning algorithms that recursively partition data space based on attribute values. The partitioning is described as a DAG where nodes within the graph are associated with attribute tests and leaf nodes correspond to either a class value or a regression value. [Figure 4](#) shows an example of classification of the famous iris data set. Rules can be read from the decision tree. For example, "If petal width  $\leq 0,8$  then class = setosa" or "If petal width  $> 0,8$  and petal length  $\leq 4,75$  then class = versicolor".

Internal nodes can have two or more children. Nodes associated with numeric attributes are often split into two children with a binary test, as shown in [Figure 4](#). Nodes associated with categorical attributes either use a binary test (i.e. attribute is equal to or not equal to a specific value) or more commonly there is a child for each possible value the attribute takes.

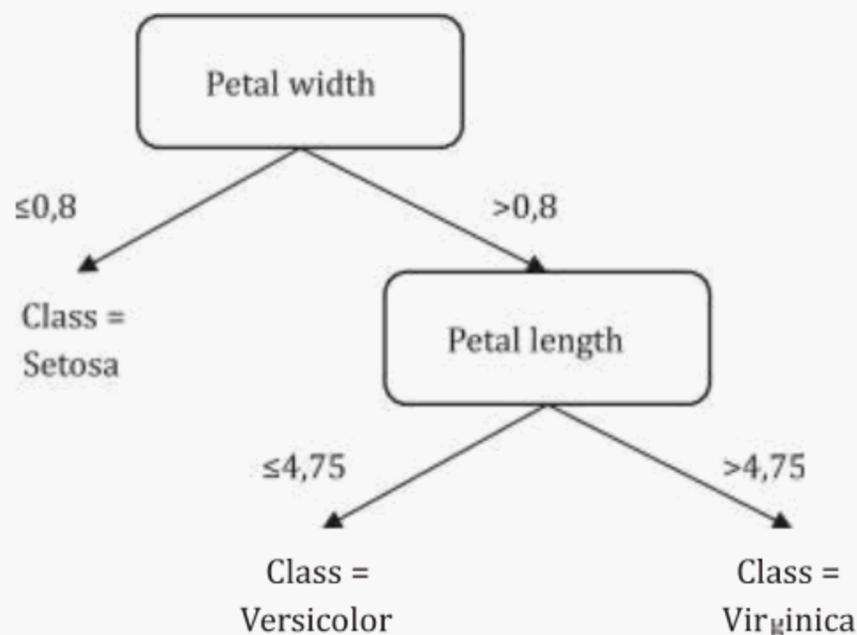


Figure 4 — Example of a decision tree

#### 8.4.2.2 Main characteristics

Decision tree induction algorithms build the decision tree for a specific training data set. This involves using a statistical test to choose the best attribute to split the data at each step. The test partitions the data from the parent node into partitions based on the chosen attribute. Desired tests are those that reduce the mixture of classes in the data associated to the parent node to data sets containing mostly

the same class (i.e. less disordered or purer) associated with the child nodes. Common statistical tests are information gain or the Gini index. The former leads to the ID3<sup>[13]</sup>, C4.5<sup>[14]</sup> or C5.0<sup>[15]</sup> family of algorithms and the latter leads to the classification and regression tree<sup>[16]</sup> algorithm.

Decision tree induction continues until some termination criteria is reached. Common criteria include stopping when all data at a node has the same class value or when a node has fewer than the specified data points associated with it. Maximally building a tree until all data at the nodes is the same class generally leads to overfitting of the training set. Consequently, tree induction is either terminated before this occurs (known as early stopping) or the maximal tree is pruned after training completes.

Decision tree induction is a simple supervised machine learning algorithm, producing trees that are relatively easy for a human to understand. Because decision tree induction does not need significant data pre-processing, it typically runs rapidly. It can handle numeric and categorical data as well as missing values. However, due to its relative simplicity, it often does not produce results as accurate as other supervised methods.

### 8.4.2.3 Typical applications

Decision tree induction can generate clear tree structures for different feature-based prediction results. It is typically explainable to domain experts and can be used for tasks such as classification and prediction.

## 8.4.3 Random forest

### 8.4.3.1 Theories and techniques

A random forest<sup>[17]</sup> is an ensemble supervised machine learning method. It consists of a set of  $n$  decision trees, each built to solve the machine learning problem. The majority vote of the decision trees in the ensemble is used to predict or classify new data points.

The  $i^{\text{th}}$  decision tree  $t_i$  in the ensemble is constructed from a bootstrap sample  $b_i$  of the training set. A bootstrap sample is a sample taken with replacement from the training set. On average it contains about two thirds of the data points from the sampled data set. This means that the  $n$  bootstrap samples are slightly different from one another, leading to a diversity of trees in the ensemble, in turn contributing to robustness of the ensemble learner as a whole.

Random forests also introduce diversity into the ensemble of trees by limiting the choice of attributes to be tested while building the tree to be from  $m$  randomly drawn attributes from the set of available attributes. Usually  $m$  is much less than  $d$ , the total number of attributes. Frequently,  $m$  is set to  $\sqrt{d}$ . In summary, the main parameters controlling the random forest algorithm are  $n$  and  $m$ , together with the type of decision tree built.

### 8.4.3.2 Main characteristics

Random forests have two useful properties: it is possible to calculate a test error for the ensemble without needing a holdout set and it is possible to estimate the relative importance of the attributes in the data set to the machine learning problem. These properties arise from the use of the out of bag samples. The out of bag sample  $o_i$  for the  $i^{\text{th}}$  decision tree is the set of data points from the training set that are not selected in bootstrap sample  $b_i$ . Due to the fact that  $o_i$  is not used to train decision tree  $t_i$ , it is used to estimate its test error. The test error of the entire ensemble can be estimated by averaging over all trees and bootstrap samples.

Similarly, by permuting the values of a particular attribute  $a$ , and calculating the difference in test error for a tree before and after the permutation, it is possible to estimate the importance of the attribute. Large differences in estimated test error suggest attribute  $a$  is important for solving the problem. Conversely, a small or no difference in estimated test error suggests attribute  $a$  is unimportant. By averaging over multiple permutations and trees, the algorithm calculates a mean decrease in error, which is then used to rank the importance of attributes. Sometimes, mean decrease in Gini index is used instead of accuracy.

As an ensemble method, random forest has most advantages on interpretation (particularly with the use of variable importance scoring) and unbiased test error for the ensemble. However, it sometimes performs poorly on data sets with huge numbers of partly correlated attributes.

### 8.4.3.3 Typical applications

Random forests are a robust supervised machine learning algorithm. Interpretation of the resulting model is facilitated by the variable importance score and by inspection of the individual trees in the forest. Variable importance from a random forest is also commonly used as a feature selection method for pre-processing data.

## 8.4.4 Linear regression

### 8.4.4.1 Theories and techniques

Linear regression is a regression method that builds a function of independent variables to predict some target variables' values. Linear regression is the simplest type of regression analysis. In linear regression, the expected value of  $y$ , given a set of independent variables  $x = \{x_1, x_2, \dots, x_p\}$ , where  $p$  is the number of independent variables, is shown in [Formula \(1\)](#):

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_px_p \quad (1)$$

where  $b_0$ ,  $b_1$ ,  $b_2$  and  $b_p$  are the coefficients of the model. Once the coefficients  $b_p$  are estimated, then one is able to predict the value of  $y$  given new values of the independent variable  $x$ .

In linear regression, the aim is to find the linear hyperplane that best fits the data points. The most common method to find the parameters is ordinary least-squares regression, where the best fit is defined as the hyperplane that minimizes the squared orthogonal distance between data points and the hyperplane. Metrics such as R-square values and residual standard error are often used to assess the fit of the model, and F-statistics are used to assess the significance of the linear relation.

### 8.4.4.2 Main characteristics

Linear regression models operate with several assumptions. Firstly, the relationship between dependent and independent variables is assumed to be linear. Secondly, the errors (i.e. difference between actual and estimated  $y$  values) are independent. Thirdly, the variance of the errors is constant with respect to the response. Fourthly, the errors are assumed to be from a normal distribution. Lastly, the independent variable  $x$  is assumed to be measured without error.

The linear regression model can be easily extended to create polynomial and other nonlinear regression models. To extend to the polynomial regression model, the terms in the linear regression model are replaced with polynomial terms such as  $x^2$ . One can also extend linear regression to take into account interactions between independent variables by adding terms that are multiplications of the independent variables.

### 8.4.4.3 Typical applications

Linear regression is used when there is a linear relationship between independent and dependent variables, and one needs to find the value of the dependent variable. For example, given the house prices and the various features in an area, predict sales value of a house.

## 8.4.5 Logistic regression

### 8.4.5.1 Theories and techniques

Logistic regression is a method for classification where the relationship between the independent variable  $X$  and the dependent variable  $Y$  is modelled according to [Formula \(2\)](#):

$$\log \frac{p(X)}{1-p(X)} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p \quad (2)$$

where  $p(X) = P(Y = 1|X)$  and  $\beta$  are the coefficients.

where

$$p(X) = P(Y = 1|X);$$

$b_0, b_1, b_2$  and  $b_p$  are the coefficients.

The left-hand side of the formula is the log-odds (or logit) transformation of  $p(X)$  and logistic regression assumes it can be expressed as a linear function. The coefficients of the logistic regression are generally estimated using the maximum likelihood method. Maximum likelihood aims to estimate the parameters such that the predicted probability  $p(x)$  of each sample is as close as possible to the actual sample value.

### 8.4.5.2 Main characteristics

In logistic regression, the target variable is predicted to be between 0 and 1, inclusive. In other words, a probability of being the second class. A threshold is set for the probability to determine whether the new data sample belongs to class 1 or class 2.

### 8.4.5.3 Typical applications

Logistic regression is used for small data sets where there is a linear relationship between independent and dependent variables. Logistic regression is most suited to data sets where there are only two classes. While it is possible to use logistic regression for multiclass classification, other methods such as linear discriminant analysis are more naturally suitable.

## 8.4.6 K-nearest neighbour

### 8.4.6.1 Theories and techniques

A KNN is the simplest machine learning method that is used for both classification and regression. It is sometimes referred to as a “lazy” algorithm as it does not learn a function from the training data. A KNN is a non-parametric method as it does not assume fixed model structure.

KNN makes a prediction for a new data point by finding the “nearest neighbours” in the training data. The nearest neighbours are found through distance metrics. Common metrics include Euclidean, Manhattan and Mahalanobis. In classification, the algorithm finds the  $k$ -nearest neighbours and the class label is the most common label in these closest neighbours. In regression, the estimated value is often the average of the nearest neighbours.

### 8.4.6.2 Main characteristics

As KNN depends on distance measures, it is generally recommended that the values of continuous attributes are normalized to keep certain attributes from overwhelming the resulting distance measure.



Since KNN is a non-parametric method, it is especially suited for data sets that lack clear decision boundaries or that cannot be easily modelled. One weakness of KNN is that there is no satisfactory way of dealing with categorical attributes.

#### 8.4.6.3 Typical applications

KNN is a very easy model to understand and implement, and it often gives reasonable performances. Hence it is a good baseline method to try before more advanced techniques. As a KNN requires the whole data set to perform classification or regression, it can be very slow for prediction when data size is large or data dimensionality is high.

#### 8.4.7 Naïve Bayes

##### 8.4.7.1 Theories and techniques

Naïve Bayes is generalized into some representation or model. The Naïve Bayes model uses the Bayesian theory for learning and inference, as shown in [Formula \(3\)](#):

$$P(B|A) = P(A|B) P(B) / P(A) \quad (3)$$

Where  $P(B|A)$  is the probability of event  $B$  given event  $A$ ,  $P(A|B)$  the probability of event  $A$  given event  $B$ ,  $P(A)$  and  $P(B)$  are the probabilities of events  $A$  or  $B$ .

where

$P(B|A)$  is the probability of event  $B$  given event  $A$ ;

$P(A|B)$  is the probability of event  $A$  given event  $B$ ;

$P(A)$  and  $P(B)$  are the probabilities of events  $A$  or  $B$ .

In Bayesian classification  $B$  is the class variable and  $A$  is the set of attributes. In Naïve Bayesian classification, it is assumed that the attributes are independent from each other, as shown in [Formula \(4\)](#):

$$P(C|A) = P(A_1|C) P(A_2|C) \dots P(A_i|C) \quad (4)$$

The posterior probability of  $P(C|A)$  is then calculated based on the likelihood,  $P(A_i|C)$ , and the prior,  $P(C)$ .

##### 8.4.7.2 Main characteristics

To find probabilities for a discrete variable, the counts of each possible class or value of the attribute are tabulated. If the variable is continuous, then one needs to estimate the probability distribution function of the variable.

##### 8.4.7.3 Typical applications

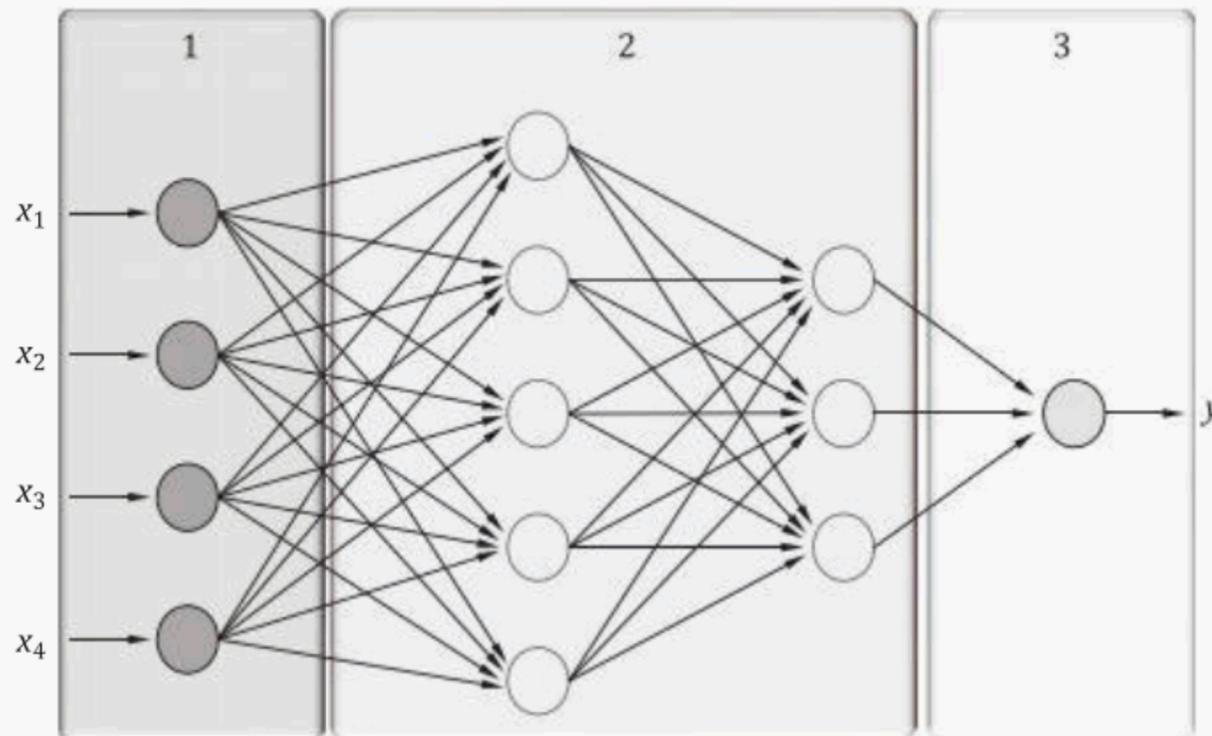
Naïve Bayes models are used for classification, especially when the data set is not too big. It can be used for multiclass classification. It does not work well if the assumption of conditional independence of attributes is not met. For very small data sets, if  $P(A_i|C)$  is zero, then the entire prediction fails.

#### 8.4.8 Feedforward neural network

##### 8.4.8.1 Theories and techniques

FFNNs have a topological structure where each neuron belongs to a layer. Each layer of neurons receives input signals from neurons in a previous layer and output signals to neurons in the next layer. The layers between input layer and output layer are hidden layers. The signal propagates from the input layer to

the output layer in one direction without feedback, which can be represented by a DAG. Figure 5 shows an example of an FFNN.



#### Key

- 1 input layer
- 2 hidden layer
- 3 output layer

**Figure 5 — Example of an FFNN structure**

#### 8.4.8.2 Main characteristics

FFNNs have a strong fitting ability and can approximate common continuous nonlinear functions, which can be used for complex feature transformation or approximation of a complex conditional distribution. In machine learning, the input features have a great influence on the classifier. Taking supervised learning as an example, good features can greatly improve the performance of classifier. Therefore, in order to achieve good classification effects, feature extraction is needed, where the original feature vector of the sample is converted to a more effective feature vector. As a multilayer FFNN can be regarded as a nonlinear composite function, it can also be applied as a feature transformation method, where its output can be used as the input of classifier for classification. When the structure of hidden layers and neurons is optimized, multilayer FFNNs can accurately approximate complex continuous functions. One of the weaknesses of FFNNs is that they are prone to overfitting, such that the model is not able to reliably generalize to new data.

#### 8.4.9 Recurrent neural network

##### 8.4.9.1 Theories and techniques

An RNN is a type of neural network that has short-term memory capabilities<sup>[18]</sup>. Neurons can receive information from both other neurons and themselves in the RNN, therefore forming a network structure with a loop. Figure 6 depicts an example of an RNN. Mathematically, an RNN can be regarded as a dynamic system, which uses a function to describe the change of all states in a given space with time. A fully connected RNN is similar to any nonlinear dynamical system<sup>[19]</sup>.

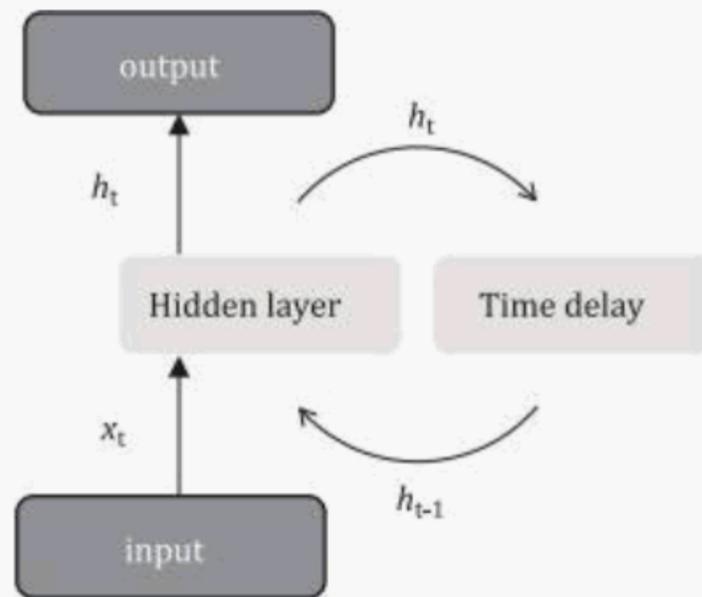


Figure 6 — Example of an RNN structure

#### 8.4.9.2 Main characteristics

RNNs can be applied to different types of machine learning tasks, including sequence-to-category tasks, synchronous sequence-to-sequence tasks and asynchronous sequence-to-sequence tasks.

- The sequence-to-category task is mainly used for classification of sequence data, where a machine learning model takes sequence data as input and outputs category data. For example, in text classification, the input data is the sequence of words, the output is the category of the text.
- The synchronous sequence-to-sequence task is mainly used for sequence labelling, where input and output are applied to each sample the length of input and output sequences are the same. For example, in part-of-speech tagging, each word needs to be labelled with its corresponding part-of-speech tag.
- In asynchronous sequence-to-sequence tasks, also known as encoder-decoder model, input and output sequences do not need to be strictly corresponding, and do not need to maintain the same length. For example, in machine translation, the input is the word sequence of the source language, the output is the word sequence of the target language.

A GDM is often used for learning parameters in RNN. Methods of the BPTT and the RTRL are used to calculate the gradient. The function of the BPTT is to pass the error information forward step by step according to the reverse order of time. Because the output dimension of general RNN is lower than the input dimension, a BPTT has the characteristics of less calculation but more spatial complexity. Because the RTRL method does not need gradient feedback, it is suitable for tasks requiring online learning.

A relatively long input sequence will cause the problem of gradient explosion and disappearance<sup>[20],[21]</sup>, also known as long-term dependence problem. Improvements to RNNs, such as gating mechanisms, have been developed to mitigate this problem.

#### 8.4.9.3 Typical applications

RNNs are widely used in speech recognition, language modelling and natural language generation.

#### 8.4.10 Long short-term memory network

##### 8.4.10.1 Theories and techniques

A LSTM network is a special RNN that can learn long-term dependence information<sup>[22]</sup>. Input gates, forget gates and output gates are used in a LSTM network to control the information flow. The input gates are selectively “remembered”. The forget gate selectively forgets the input from the previous node. The output gate determines which outputs will be treated as the current state.

### 8.4.10.2 Main characteristics

A LSTM network can learn long-distance dependencies because of the introduction of a gate mechanism which is the most important characteristic to control the circulation and loss of features. The LSTM network recall rate is high because the long-range dependence is strong and the amount of data that can be processed is large. A LSTM network can be used as a kind of “intermediate state” to describe the sequence and the results can also be applied as features for subsequent use.

The LSTM networks mitigate the problem of gradient explosion or gradient disappearance in simple RNNs. The LSTM networks can currently handle sequences of 100 orders of magnitude, but the vanishing gradient problem can still persist for sequences of 1 000 orders of magnitude or longer.

Since each LSTM network cell has four fully connected layers, if the time span of a LSTM network is large and the network is deep, this calculation can be large and time consuming.

The LSTM networks are also theoretically capable of fitting arbitrary functions in which assumptions and constraints around the problem are significantly relaxed.

### 8.4.10.3 Typical applications

Applications of the LSTM network include machine translation, language modelling<sup>[23]</sup> and speech recognition.

## 8.4.11 Convolutional neural network

### 8.4.11.1 Theories and techniques

A CNN is a kind of feedforward neural network. CNNs are multilayer perceptrons inspired by biological ways of thinking. CNNs include an input layer and a combination of convolution layers, activation layers, pooling layer, fully connected layer and normalization layers. An example of the CNN structure in the form of a LeNet-5 network<sup>[24]</sup> machine learning model is shown in Figure 7.

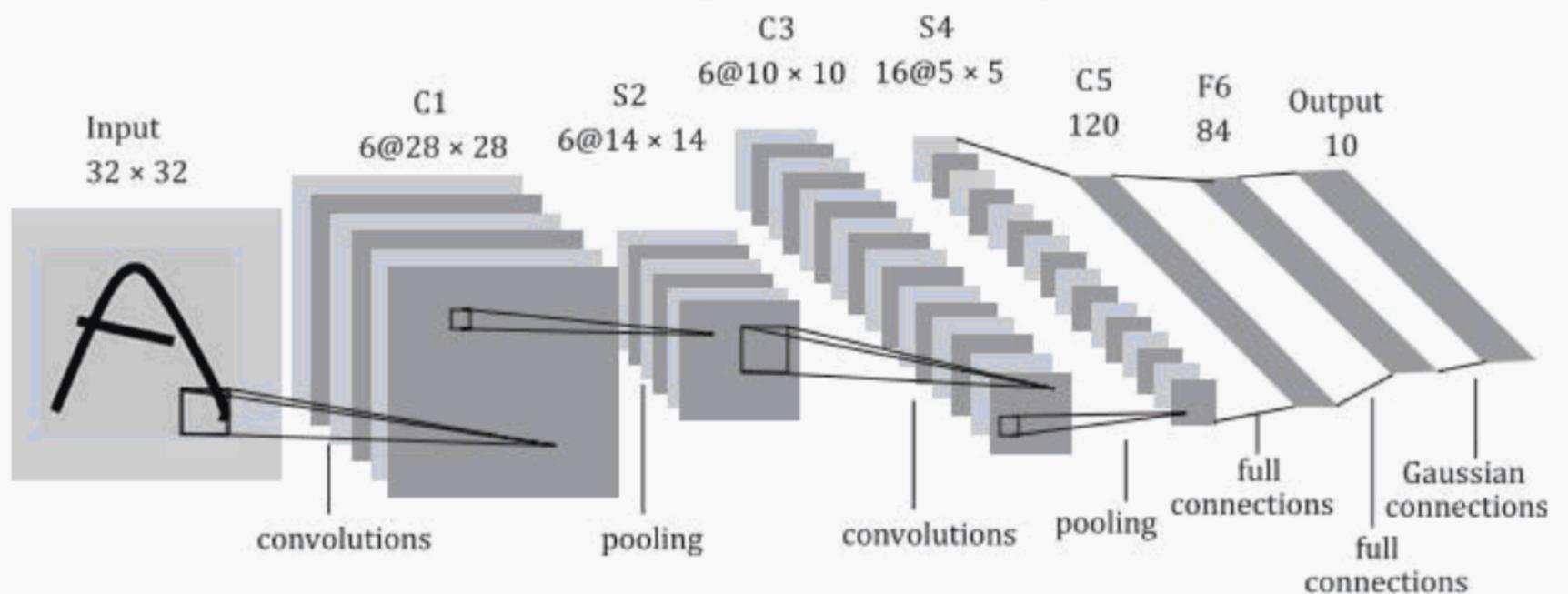


Figure 7 — Example of a CNN machine learning model structure: LeNet-5 network

### 8.4.11.2 Main characteristics

A CNN is essentially an input-output mapping. It can learn a large number of mapping relations between input and output without any precise mathematical expression between input and output. As long as the CNN is trained in a known mode, the network has the ability to map between input and output pairs<sup>[25]</sup>.

One of the most important characteristics of CNNs is that it has an inverted triangle shape, which limits excessive gradient loss in back propagation neural networks.

In using machine learning for image processing, a convolution kernel is slid on an image (or certain feature) to obtain a new set of features through its convolution operations. The convolution layer is used to extract local features, where convolution kernels function as feature extractors. CNNs have two important features:

- local connections, where each neuron in the convolutional layer is only connected to the neuron in a certain local window in the next layer, forming a local connection network;
- weight sharing, where filters for all neurons per layer are the same.

In this case, the network can learn in parallel, which is also one of the advantages of CNNs relative to the neurons interconnected networks.

Since CNNs share a convolution core, the demand for high-dimensional data processing is reduced. In order to optimize performance (e.g. to accurately predict image classes), significant manual effort can be necessary in feature selection, weight training and parameter adjustments. Large training data sets are often required, which in turn can necessitate the use of GPUs, multi-core CPUs or application-specific processors. CNNs are also not always explainable, such that it is unclear what aspects of input data are reflected in output mappings.

### 8.4.11.3 Typical applications

CNNs are used widely in image recognition and classification, e.g. in face recognition applications.

## 8.4.12 Generative adversarial network

### 8.4.12.1 Theories and techniques

GANs use adversarial training and generative models to produce samples that conform to real data distributions<sup>[26]</sup>. GANs train generator networks that produce samples intended to cause misclassifications on the part of the discriminator network. GANs also train discriminator networks that attempt to determine whether a sample is real data or is produced by the generator network. In this fashion, the two opposing networks are continuously trained. Convergence occurs when the discriminator network no longer accurately classifies samples as real or fake, such that the generator network is producing samples that conform to the real data distribution.

### 8.4.12.2 Main characteristics

GANs can represent latent dimensions and manifest hidden relationships among data. Compared with the single objective optimization task, the optimization objectives of the generator network and the discriminator network in GAN are opposing. Therefore, the GAN training process can be difficult and unstable and the ability of two networks needs to be balanced. If the discriminator network is too robust in initial adversarial stages, the generator network then cannot improve. Hyperparameter optimization is required in the training process such that in each iteration, the discriminator network is slightly more robust than the generator network.

### 8.4.12.3 Typical applications

GANs are usually used in image processing tasks such as text-to-image generation<sup>[27]</sup>, image-to-image translation<sup>[28]</sup> and image inpainting<sup>[29]</sup>.

### 8.4.13 Transfer learning

#### 8.4.13.1 Theories and techniques

Transfer learning methods store and abstract knowledge gained from training data for a given problem and apply this knowledge to a different problem. Knowledge in the form of a machine learning model is under some circumstances adaptable to a new task or domain. This is particularly useful when it is difficult or impossible to acquire and label training data for this new task or domain.

#### 8.4.13.2 Main characteristics

Transfer learning is based on identifying opportunities to apply existing knowledge to a new domain, transferring labelled data or knowledge structures (e.g. machine learning models) for use in this new domain, and tuning or optimizing the transferred knowledge against a new target area or task.

Transfer learning is designed to reduce dependence on large amounts of domain-specific labelled data. However, data are not always adaptable to new domains, and transferred knowledge does not always adequately reflect the characteristics of data in new domains.

Depending on what is being transferred, transfer learning approaches can be based on entity transfer, feature transfer or parameter sharing. Transfer learning can also be classified as inductive or transductive. Generally, in inductive transfer learning, the source and target domains are related to each other and the source domain relies on large numbers of training samples. In transductive transfer learning, transferring process happens using data from different domains.

#### 8.4.13.3 Typical applications

Typical transfer learning applications include image recognition and NLP<sup>[30]</sup>, where models trained to translate text between two languages are used to translate text into a third language.

### 8.4.14 Bidirectional encoder representations from transformers

#### 8.4.14.1 Theories and techniques

BERT is a kind of language model that uses unlabelled training data to obtain a rich semantic representation of the text. That representation, learned through a specific NLP task, can in turn be re-used to achieve other NLP tasks, often after fine-tuning or by applying downstream algorithms. BERT is a deeply bidirectional transformer<sup>[31]</sup> encoder. Its input includes a series of fusional vectors such as character, text and position information. Its output is a semantic vector.

A typical structure for a BERT model<sup>[32]</sup> is shown in Figure 8.  $E_1, E_2$  and  $E_N$  are embedding representation.  $T_1, T_2$  and  $T_N$  are final outputs.  $T_{rm}$  are intermediate representations of a given token.

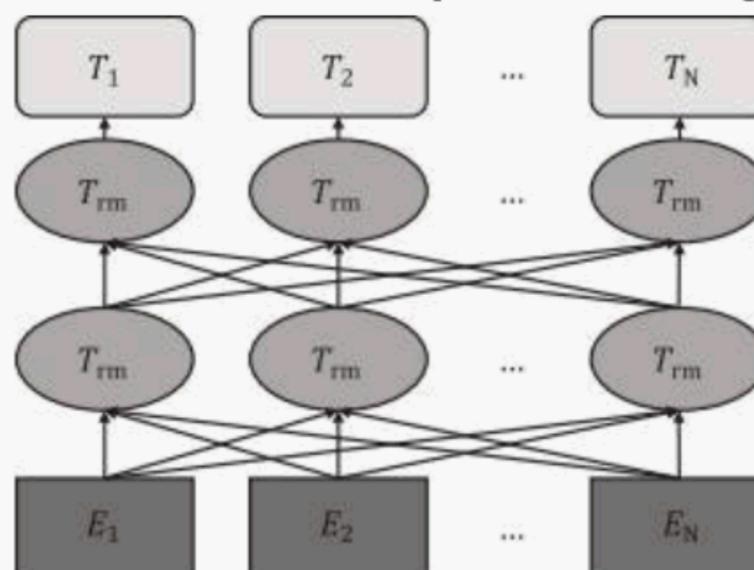


Figure 8 — Example of BERT model structure

#### 8.4.14.2 Main characteristics

BERT is designed to process a word using information in a bi-directional fashion. As opposed to other language model pre-training (e.g. embeddings from language models<sup>[33]</sup>, GPT<sup>[34]</sup>) methods, BERT does not predict the most likely word based on all preceding words. Instead, it randomly masks some words and uses all the uncovered words to make a prediction.

BERT alleviates the unidirectionality constraint by using MLM pre-training. Given a sentence, an MLM randomly erases one or several words and attempts to predict the erased words based on the remaining words. This makes the model more reliant on contextual information for predictions and gives the model a certain error correction capability. BERT also uses NSP to determine whether a second sentence closely follows a first sentence in the text.

By performing joint training with MLP and NLP tasks, the machine learning model will output the vector representation of each word which describes comprehensively and accurately the overall information of the input text (whether single sentence or sentence pair). Subsequent fine-tuning tasks provide better initial values of machine learning model parameters. However, any inconsistencies in the pretraining and generation processes will reduce effectiveness in natural language generation tasks.

BERT has pre-training and fine-tuning steps<sup>[35],[36]</sup>. The algorithm is trained on unlabelled data over different tasks in pre-training. The parameters of the algorithm are then fine-tuned using labelled data from specific tasks. Typical fine-tuning parameters include batch size, learning rate and number of epochs. BERT trains the deep bidirectional representation through jointly adjusting the bidirectional transformers in all layers. Therefore, it only needs an additional output layer to fine-tune the pre-trained machine learning model for various tasks and for a number of tasks there is no need to design a task-specific architecture. Excessive use of masks in training can affect the performance of the machine learning model in difficult-to-predict ways.

#### 8.4.14.3 Typical applications

BERT has contributed to the advancement of NLP tasks such as corpus of linguistic acceptability (CoLA), Microsoft research paraphrase corpus (MRPC), multi-genre natural language inference (MultiNLI), question natural language inference (QNLI), quora question pairs (QQP), recognition textual entailment (RTE), semantic textual similarity benchmark (STS-B) and Stanford sentiment treebank (SST-2).

### 8.4.15 XLNet

#### 8.4.15.1 Theories and techniques

Further improving on GPT and BERT, XLNet is an autoregressive permutation language model<sup>[37]</sup>. XLNet continuously predicts the next word from left to right, not in the natural order of the sentences but in the order of prediction. A dual flow self-attention mechanism is implemented. This “dual stream” is represented as two transformers for each layer in the model, a query stream and a context stream.

#### 8.4.15.2 Main characteristics

To solve the problem of bidirectional context, a permutation language model uses the position information of the target when making predictions. The content stream encodes all the content at the current moment, while the query stream refers to the previous history and the current position to be predicted. XLNet attempts to learn bidirectional contextual information by maximizing the log-likelihood of all possible factoring orders.

XLNet is autoregressive, which BERT is not. This property enables more consistency in how data appears to the model between the pre-training and fine-tuning steps. This property also avoids BERT’s assumption of token independence.

XLNet is often computationally intensive to implement.

### 8.4.15.3 Typical applications

Typical applications of XLNet include reading comprehension, document ranking, QA tasks, text classification and NLU.

## 8.5 Metaheuristics

### 8.5.1 General

Metaheuristics refers to a class of optimization algorithms that provide sufficiently good solutions for an optimization problem. They generally do not identify the optimum solution but can quickly find a reasonable solution by using a heuristic or rule to search the solution space. They usually do not make strong assumptions about the problem being solved, which means they can be used on a wide set of problems, particularly those that are stochastic or non-differentiable. However, because they do not make strong assumptions about the problem, they generally perform worse than custom algorithms designed for specific problems. Metaheuristics are often population-based, searching many points in the search space at the same time, which means they tend to avoid local optima. They are also often stochastic algorithms. Commonly used metaheuristic algorithms include those from evolutionary computation, typified by genetic algorithms, genetic programming or evolution strategies, simulated annealing and swarm intelligence algorithms, inspired by artificial life and typified by particle swarm optimization and ant colony optimization.

### 8.5.2 Genetic algorithms

#### 8.5.2.1 Theories and techniques

Genetic algorithms are one of the earliest and most famous forms of evolutionary computation. They operate on the basis of genetics and selection of the fittest. They were introduced and popularized around the 1990s<sup>[38],[39]</sup>.

There are various theories underpinning genetic algorithms. The most widespread is the building block hypothesis<sup>[39]</sup> that shows how genetic algorithms put together short, high-fitness building blocks of the solution to build into higher-performing solutions.

#### 8.5.2.2 Main characteristics

Solving an optimization problem using a genetic algorithm involves being able to represent potential solutions to the problem on a genome and being able to evaluate a genome using a fitness function, which scores the genome on how well it solves the problem. A genome commonly consists of a simple list of bits, doubles or other symbols, often representing the parameters in the optimization problem. Genomes can also be more complex representations such as trees, matrices or strings that need to be interpreted by a simple computer language.

The algorithm consists of four steps: initialisation, selection, genetic operators and termination. The algorithm breeds successive generations of genomes, with later generations better solving the problem.

In the initialisation step, a population of genomes is created. Generally, randomly generated genomes are used but it is possible to seed the population with weak solutions. It is, however, important that there is a diversity of initial genomes in the population.

After initialization, selection is performed. Here, genomes in the population are scored with the fitness function, and randomly selected for breeding, with the probability of selection proportional to the fitness. In other words, highly fit genomes, those that better solve the problem, are more likely to be selected for breeding. Common selection approaches include roulette wheel selection and tournament selection.

Once pairs of genomes are selected for breeding, genetic operators are applied. There are a range of genetic operators with the most common being crossover (or recombination) and mutation.

Crossover combines parts of the two parent genomes to produce children. Common crossover variants are single-point crossover or two-point crossover. In essence, one or two cut points are randomly set along the length of genome. The child genome comprises the first part of one parent genome before the cut point and the second part of the other parent genome after the cut point. For the two-point variant, the first and third parts of the genome come from one parent and the second part comes from the other parent. The intention of crossover is to share parts of good solutions to identify stronger children. Weaker children are removed in subsequent generations through selection.

The other main genetic operator is mutation, where symbols in the genome are randomly changed. The aim of mutation is to avoid premature convergence by reintroducing diversity into the population. Crossover is applied at a high probability, generally around 0,6 to 0,9, and mutation at a very low rate, typically 0,1 or less.

Successive generations of population are produced in this way until the algorithm is terminated using termination criteria, which can be as simple as running for a predefined number of generations or until the best member of the population satisfies some predefined criteria.

### **8.5.2.3 Typical applications**

Genetic algorithms are used on a wide variety of applications. They excel where problems are non-differentiable, where the fitness function cannot be written mathematically and where the problem domain is stochastic.

## Bibliography

- [1] Russell Stuart J., Norvig Peter *Artificial Intelligence: A Modern Approach*. Essex: Pearson
- [2] ISO/IEC 2382:2015, *Information technology — Vocabulary*
- [3] ISO/IEC/IEEE 26513:2017, *Systems and software engineering — Requirements for testers and reviewers of information for users*
- [4] ISO/IEC/TR 24030, *Information technology — Artificial intelligence (AI) — Use cases*
- [5] Jiao Jian Application and prospect of artificial intelligence in smart grid. *IOP conference series: Earth and Environmental Science*. IWRED 2020, 2020
- [6] RDF Schema 1.1 — W3C Recommendation 25 February 2014. Available from: <http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>
- [7] OWL 2 Web Ontology Language Document Overview (Second Edition) — W3C Recommendation 11 December 2012. Available from: <http://www.w3.org/TR/2012/REC-owl2-overview-20121211/>
- [8] Hu L., Jiang Y., Li Y. Optimization Design of Internet Fraud Case Based on Knowledge Graph and Case Teaching. *Proceedings of the 2019 7th International Conference on Information and Education Technology*. ICIET 2019, 2019
- [9] SPARQL 1.1 Query Language — W3C Recommendation 21 March 2013. Available from: <http://www.w3.org/TR/2013/REC-sparql11-query-20130321/>
- [10] ISO/IEC 9075 (all parts), *Information technology — Database languages — SQL*
- [11] Shapes Constraint Language (SHACL) — W3C Recommendation 20 July 2017. Available from: <https://www.w3.org/TR/2017/REC-shacl-20170720/>
- [12] Copi Irving M., Cohen Carl, McMahon Kenneth *Introduction to Logic 14th edition*. Essex: Pearson Education Limited
- [13] Quinlan J. R. Induction of Decision Tree. *Machine Learning*, 1986, **1**, 81-106
- [14] Quinlan J. R. *C4.5: Programs for Machine Learning*. San Francisco: Morgan Kaufmann Publishers Inc, 1993
- [15] Quinlan J.R. Data Mining Tools See5 and C5.0. *RuleQuest Research* [online]. St. Ives NSW, Australia, October 2020 [viewed 21 November 2020]. Available from: <https://www.rulequest.com/see5-info.html>
- [16] Breiman Leo, Friedman Jerome, Stone Charles J., Olshen R.A. *Classification and Regression Trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984
- [17] Breiman L. Random Forests. *Machine Learning*. 2001, **45** (1), 5–32
- [18] Li Bin, Zhuang Xiaoying Multiscale computation on feedforward neural network and recurrent neural network. *Frontiers of Structural and Civil Engineering*. 2021
- [19] Haykin Simon *Neural networks and learning machines, volume 3*. Upper Saddle River, NJ: Pearson, 2009
- [20] Bengio Yoshua, Simard Patrice, Frasconi Paolo Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*. 1994, **5**(2), 157–166
- [21] Hochreiter Sepp, Bengio Yoshua, Frasconi Paolo, Schmidhuber Jürgen *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. Wiley-IEEE Press, 2001, 237-243

- [22] Hochreiter Sepp, Schmidhuber Jürgen Long short-term memory. *Neural computation*. 1997, **9**(8), 1735–1780
- [23] Dai A., Le Q. Semi-supervised Sequence Learning. *arXiv preprint*. 2015. arXiv: 1511.01432
- [24] LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, 11
- [25] Hu Zhenlong, Zhao Qiang, Wang Jun The prediction model of worsted yarn quality based on CNN-GRNN neural network. *Neural Computing and Applications*. 2018
- [26] Goodfellow Ian, Pouget-Abadie Jean, Mirza Mehdi, Xu Bing David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*. 2014, 2672–2680
- [27] Xu T., Zhang P., Huang Q., Zhang H., Gan Z., Huang X. et al. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks. *arXiv preprint*. 2017. arXiv: 1711.10485v1
- [28] Zhu J.-Y., Park T., Isola P., Efros A.A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv preprint*. 2020. arXiv: 1703.10593v7
- [29] Liu G., Reda F.A., Shih K.J., Wang T.-C., Tao A., Catanzaro B. Image Inpainting for Irregular Holes Using Partial Convolutions. *arXiv preprint*. 2018. arXiv: 1804.07723v2
- [30] Jeremy H., Sebastian R. Universal Language Model Fine-tuning for Text Classification. *arXiv preprint*. 2018. arXiv: 1801.06146v5
- [31] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N. et al. Attention is all you need. *arXiv preprint*. 2017. arXiv:1706.03762v5
- [32] Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint*. 2019. arXiv: 1810.04805v2
- [33] Peters M.E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K. et al. Deep contextualized word representations. *arXiv preprint*. 2018. arXiv: 1802.05365v2
- [34] Radford A., Narasimhan K., Salimans T., Sutskever I., Improving language understanding by generative pre-training. OpenAI [online]. San Francisco, USA, 11 June 2018 [viewed 23 November 2020]. Available from: <https://openai.com/blog/language-unsupervised>
- [35] Song Z., Xie Y., Huang W., Wang H. Classification of Traditional Chinese Medicine Cases based on Character-level Bert and Deep Learning. *IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. 2019
- [36] Parcheta Zuzanna, Sanchis-Trilles Germán, Casacuberta Francisco, Rendahl Robin Combining Embeddings of Input Data for Text Classification. *Neural Processing Letters*. 2020
- [37] Yang Z., Dai Z., Yang Y., Carbonell J., Salakhutdinov R., Le Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint*. 2020. arXiv: 1906.08237v2
- [38] Goldberg David *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley Professional, 1989
- [39] Holland John *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1992



