Neural Networks 2 - Introduction

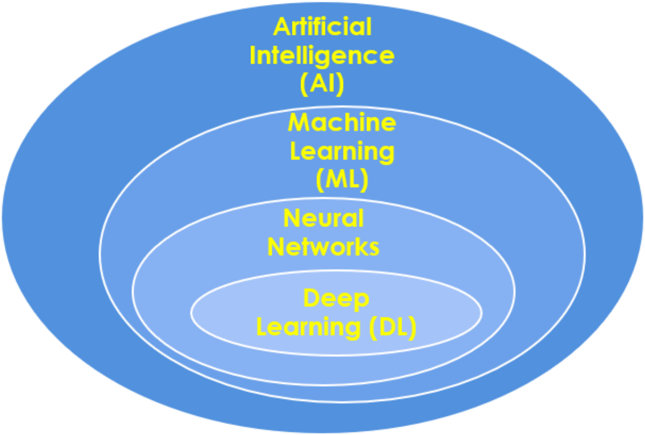
18NES2 - Lecture 1, Winter semester 2025/26

Zuzana Petříčková

September 23, 2025

Neural Networks 2 - Introduction

Introduction to Artificial Intelligence



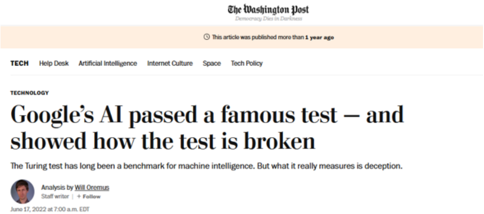
- What is AI (Artificial Intelligence)?

Introduction to Artificial Intelligence

- Turing Test (Alan Turing, 1950)
    - A computer/program passes the test if a human, during a five-minute conversation, cannot recognize that they are not talking to another human.
- First chatbot: Eliza (1966)

Introduction to Artificial Intelligence

- Turing Test (Alan Turing, 1950)
  - A computer/program passes the test if a human, during a five-minute conversation, cannot recognize that they are not talking to another human.
- The classical definition is no longer valid:
  - The first chatbot to surpass the Turing Test was Google BERT, 2022.


The Washington Post

This article was published more than 1 year ago

TECH Help Desk Artificial Intelligence Internet Culture Space Tech Policy

TECHNOLOGY

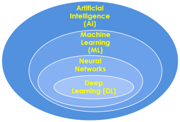
Google's AI passed a famous test — and showed how the test is broken

The Turing test has long been a benchmark for machine intelligence. But what it really measures is deception.

Analysis by Will Oremus
Staff writer | + Follow
June 17, 2022 at 7:00 a.m. EDT

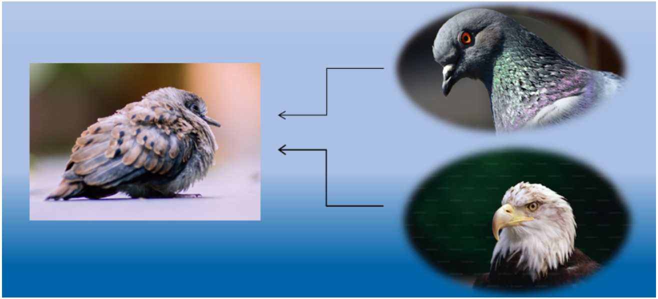
Introduction to Artificial Intelligence

- **Classical definition:** The ability of machines/computer programs to replicate human capabilities that we consider **intelligent**:
  - The ability to reason and solve problems, and plan
  - The ability to adapt to new environments and learn
  - Creativity, innovation, and decision-making
- **Modern definition:** A scientific field focused on developing **advanced systems** to solve **complex problems**
  - Image recognition, language translation, playing chess, medical diagnostics, autonomous vehicles, . . .
  - **AI is integrated into everyday life:**
    - Content personalization on social media, personalized advertisements, spam detection, search engines, chatbots, and more



Machine Learning

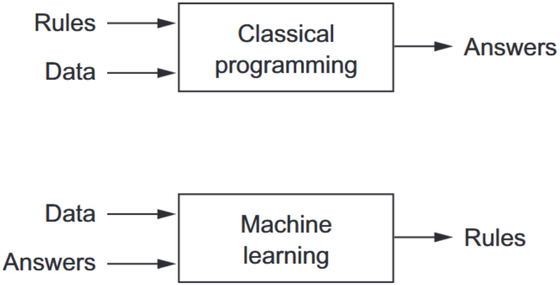
- Models and techniques that allow a computer system to **learn** from data or past experiences.
- Often inspired by biological principles, the model "builds itself."



Machine Learning

Principle

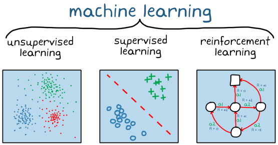
- Based on biological principles, the model "builds itself."
- The computational model learns from data (training set) or past experiences.



F. Chollet: Deep Learning with Python, Fig. 1.2

Machine Learning - Three Fundamental Types

- **Supervised Learning**
  - The model learns from labeled data (e.g., image classification)
  - Training set in the form of [*input, expected output*]
- **Unsupervised Learning (Self-Organization, Self-Supervised Learning)**
  - The model identifies patterns in unlabeled data (e.g., clustering images)
  - Training set in the form of [*input*]
- **Reinforcement Learning**
  - The model learns an optimal strategy based on past experiences, often using rewards and penalties (e.g., robotic soccer)



https://www.mathworks.com/discovery/reinforcement-learning.html

Machine Learning - Three Fundamental Types

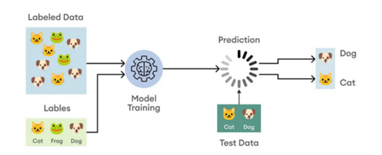
Supervised Learning

- **Training set format:** [input, expected output]
- **Learning objective:** The model should approximate an unknown function as accurately as possible $\rightarrow$ predict the correct output for any given input.
- **Generalization:** The model should produce accurate outputs even for data not included in the training set.
- **Typical applications:**
  - Medical diagnosis, fraud detection in banking
  - Time series forecasting
  - Image classification and segmentation
  - Natural language processing, speech recognition

Machine Learning - Three Fundamental Types

Supervised Learning - Task Types

- **Classification:** Predicting a class/category



- **Regression:** Predicting a numerical value (price, temperature, handwriting slant, etc.)

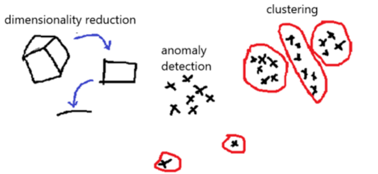
- **Structured Data Learning** (e.g., sentences in natural language)

Machine Learning - Three Fundamental Types

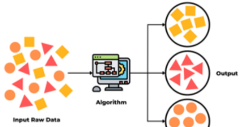
Unsupervised Learning (Self-Organization)

- **Training set format:** [input]
- **Learning objective:** Identify inner structure or patterns in data
- **Applications: Dimensionality reduction** (data compression, visualization), **Anomaly detection** (e.g., in banking transactions), **Clustering** (e.g., grouping customers based on behavior, plagiarism detection), e-commerce (recommendation systems)

Types of tasks:



https://towardsdatascience.com/unsupervised-learning-algorithms-cheat-sheet-d391a39de44a



https://eastgate-software.com/what-is-unsupervised-learning/#ftoc-heading-7

Machine Learning - Three Fundamental Types

Unsupervised Learning (Self-Organization)

- Methods: Clustering, association rules, autoencoders, generative models

Reinforcement Learning

- The program learns an optimal strategy based on past experiences.
- Often involves a system of rewards and penalties.
- Methods: Q-learning, Deep Q-Networks, etc.
- Applications: Gaming industry, robotics, resource management, machine translation, ...

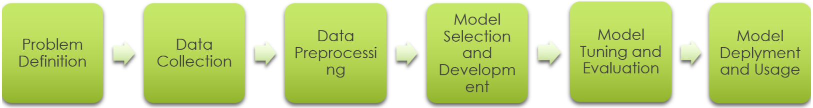
Typical Workflow for Solving a Machine Learning Task

- **Problem Definition**
  - What is the task? What kind of data do we have? What type of problem are we solving? What is the exact goal?
  - What machine learning models are suitable for this task?
  - Conduct a literature review of existing solutions, limitations, and challenges.
- **Data Collection**
  - Gather relevant data and try to understand it thoroughly.
  - Define a success metric (various evaluation metrics such as accuracy, precision, recall, etc.).

Typical Workflow for Solving a Machine Learning Task

- **Data Preprocessing**
  - Transform data into a format suitable for machine learning models.
  - Techniques include vectorization, normalization, handling missing values, and data augmentation.
- **Model Selection and Development**
  - Choose an evaluation protocol (e.g., k-fold cross-validation).
  - Build a simple baseline model for reference.
  - Determine the model type — depends on the problem domain.
  - Select a specific model within the chosen type (tuning hyperparameters, defining architecture, using pre-existing templates).

Typical Workflow for Solving a Machine Learning Task



- **Model Tuning and Evaluation**
  - Experiment with different architectures.
  - Apply optimization techniques such as dropout, regularization, and hyperparameter tuning.
  - Evaluate the model—preferably using unseen (test) data.
- **Deployment and Maintenance**
  - Deploy the model across different platforms and devices.
  - Monitor model performance over time and update as necessary.

Machine Learning

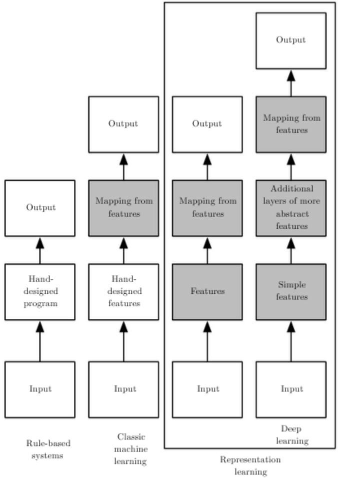
Examples of Classical Machine Learning Models

- Stored data models
- Linear or non-linear functions (e.g., linear regression, logistic regression)
- Decision trees, rule-based systems
- Bayesian networks, fuzzy systems, evolutionary algorithms
- Artificial neural networks
  - Inspired by the structure and function of the human brain:
    - Information processing (speed, parallelism)
    - Information storage mechanisms
    - Redundancy and control mechanisms

Modern Machine Learning ... Deep Learning

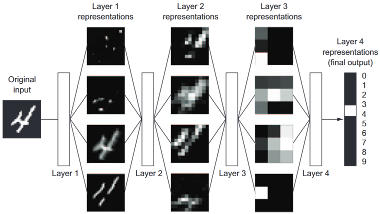
- Models based on deep neural networks

Machine Learning — Deep Learning



I. Goodfellow, Y. Bengio, and A. Courville: Deep Learning, 2016, Figure 1.5

- Utilizes artificial neural networks with multiple layers (so-called deep networks).
- The model automatically extracts features from data, reducing the need for extensive preprocessing.



F. Chollet: Deep Learning with Python, Fig. 1.6

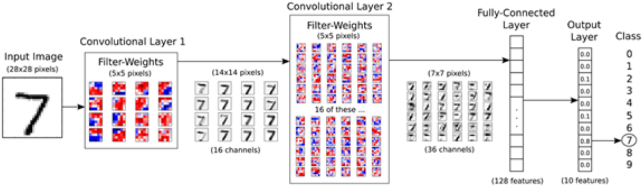
Machine Learning — Deep Learning

Advantages:

- Capable of processing vast amounts of data and identifying complex patterns.
- Flexible across various tasks (classification, recognition, content generation, etc.).

Disadvantages:

- Requires large amounts of data and significant computational power for training and inference.
- Functions as a "black box" — limited interpretability of learned representations.



About the Course Neural Networks 2

- This course is build on the previous one

What we covered in the course Neural Networks 1, 18NES1, Sommer semester 2024/25

- We covered basic "shallow" neural network models: the perceptron and single-layer models, Kohonen maps
- We also introduced basic deep models (MLP, convolutional neural networks)
- We tried to look inside neural network models and understand how they work
- We also touched on theory (to a limited extent)
- We got familiar with Python libraries for deep learning (mainly Keras)

What we will cover in the course Neural Networks 2

Course Neural Networks 2, 18NES2/18YNES2, Winter Semester 2025/26

- We will briefly revisit some models already covered in Neural Networks 1 (MLP, convolutional neural networks)
- We will introduce new ones (modern recurrent neural networks, advanced CNN variants, autoencoders, generative models)
- Most likely, we will not reach more complex models (GANs, transformers, reinforcement learning, etc.)
- The emphasis will be on a practical, engineering-oriented perspective
- Theory will be covered only at a very basic level

What we will cover in the course Neural Networks 2

Course Neural Networks 2, 18NES2/18YNES2, Winter Semester 2025/26

- We will explore real-world applications through Python examples
- We will address different types of tasks and how to solve them with deep neural networks
- We will experiment and try things out :)

Would you like to learn how to use or understand even more modern and complex neural network models?

- You can learn more about deep neural networks in follow-up courses:
  - **Machine Learning 2**
  - **Applications of Optimization Methods, and others**

Main Deep Learning Frameworks in Python

- **TensorFlow** – Open-source library by Google.
  - Powerful framework for AI applications (mobile, server).
  - Supports both static and dynamic computation graphs.
- **PyTorch** – Open-source library by Meta (Facebook).
  - Flexible and intuitive, ideal for research and academia.
  - Dynamic computation graphs, easy debugging.
- **Keras** – High-level universal API.
  - Beginner-friendly and easy to understand.
  - Great for fast prototyping. Runs on top of TensorFlow, JAX, or PyTorch.
- **PyTorch Lightning** – High-level wrapper for PyTorch.
  - Reduces boilerplate code in training routines.
  - Supports multi-GPU training, scaling, and reproducibility.
- **JAX** – Optimized for speed and experimental research.
- Previously popular **Theano** – now deprecated.

TensorFlow vs PyTorch – Comparison

TensorFlow – robust and production-ready, but more rigid:

- Part of a broader ecosystem (TensorBoard, TF Lite, etc.).
- Very efficient (C++/Python hybrid), supports distributed training, native TPU support.
- Optimized for deployment, mobile support (TF Lite), model compilation.
- Less developer-friendly: more code, harder to define custom models.
- Difficult debugging of complex models (C++ backend).

PyTorch – newer, rapidly evolving, research-focused:

- Pythonic, concise, and easier to use; gaining feature parity.
- Slightly less performant (pure Python), but highly flexible.
- Custom models and layers are very easy to implement and debug.

Other Useful Libraries

Data manipulation and numerical computing:

- **Scikit-learn (sklearn)** – classic ML algorithms; tools for data processing and model evaluation.
- **NumPy** – efficient numerical computing with arrays and tensors.
- **Pandas** – powerful data manipulation library for structured data (categorical, missing values).

Visualization:

- **Matplotlib** – general-purpose plotting (static, animated, interactive).
- **Plotly** – interactive visualizations.
- **Seaborn** – statistical data visualization (correlations, distributions, etc.).
- **TensorBoard** – visualization of the training progress, especially for TensorFlow.

Practical Examples

useful_python_libraries.ipynb

- Commented examples of the usage of useful Python libraries (NumPy, Pandas, Matplotlib...)

NN_libraries.ipynb

- Commented examples comparing major deep learning frameworks (Keras, TensorFlow, PyTorch, Lightning) on a simple binary classification task.
- Demonstration of automatic symbolic tensor differentiation in TensorFlow and PyTorch.
- Frameworks and GPU support in practice.

NN_libraries_installation.ipynb

- Brief installation guide for running the examples locally on your own machine.