Deep Learning 1

Raoul Grouls, 7 Mei 2024

Hello world

- Raoul Grouls, data alchemist, https://the-pttrn.nl/
- Research @ HAN, ‘internal representations of the world in AI models’
- Teaching @ HAN, HU

raoul.grouls@han.nl
https://github.com/raoulg


MY ENTHOUSIASM
THE TEACHING MATERIALS
LEARNING GOALS
THE EXTRAS
imgflip.com

What is intelligence?

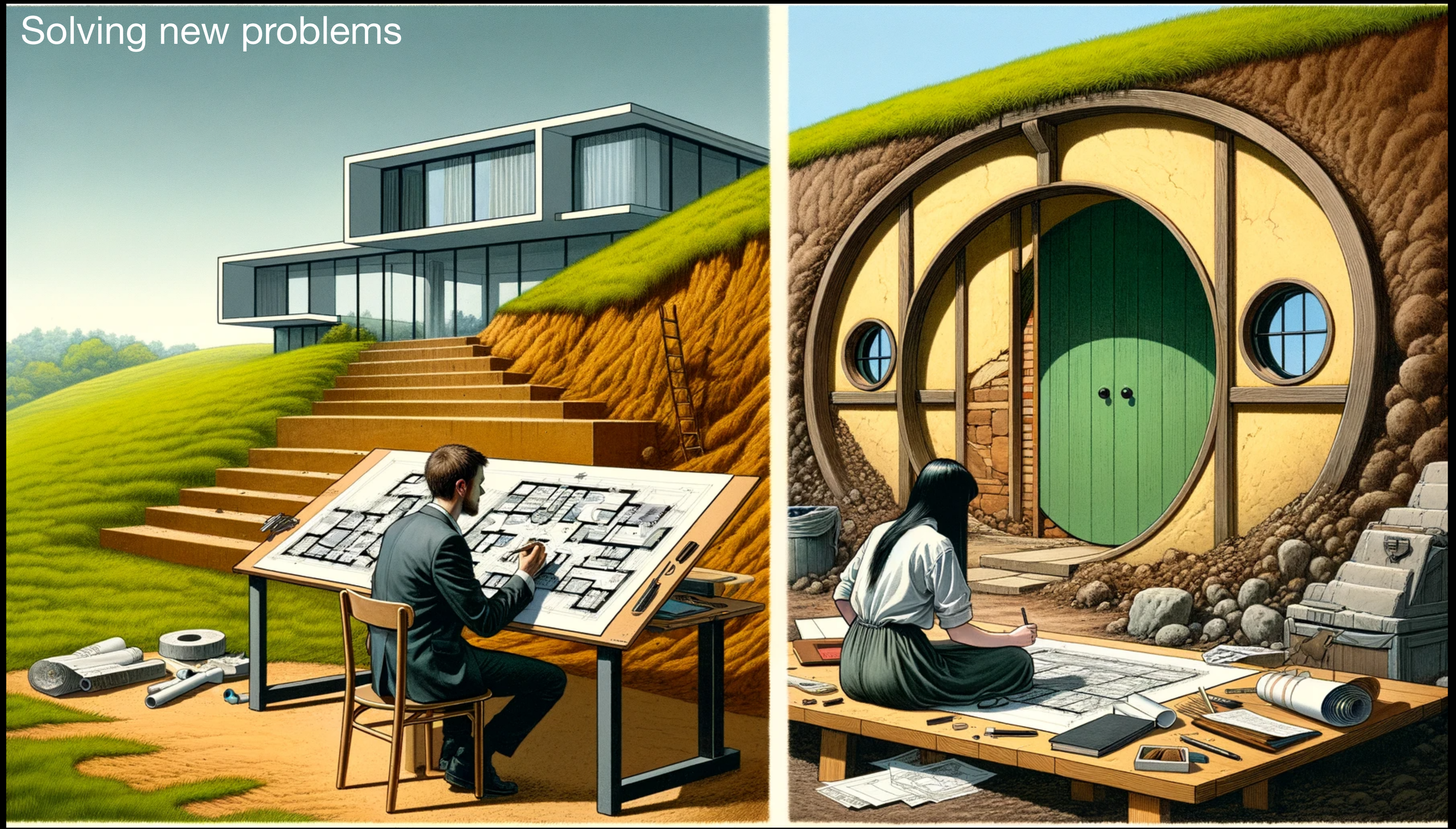
Exercise:
3 minute break out
Find a definition of intelligence

Abstraction

Memorization versus understanding

Memorization versus understanding

Solving new problems



Clarifying some terminology

This is an hierarchical list, where every item is a subset of the preceding item.

1. **Automation**: Autonomous agent executing fixed rules
2. **Artificial Intelligence**: Autonomous agent optimizing an objective (including Expert Systems, Fuzzy Logic, Symbolic AI, Constraint Satisfaction Problems, Cellular Automata, Genetic Algorithms and Swarm Intelligence)
3. **Machine Learning**: Learning from data. This includes both gradient-based and non-gradient-based methods. (linear regression, random forests, Support Vector Machines)
4. **Deep Learning**: Complex learning from data, gradient-based and with backpropagation (CNN, RNN, Transformers)

What is deep learning?

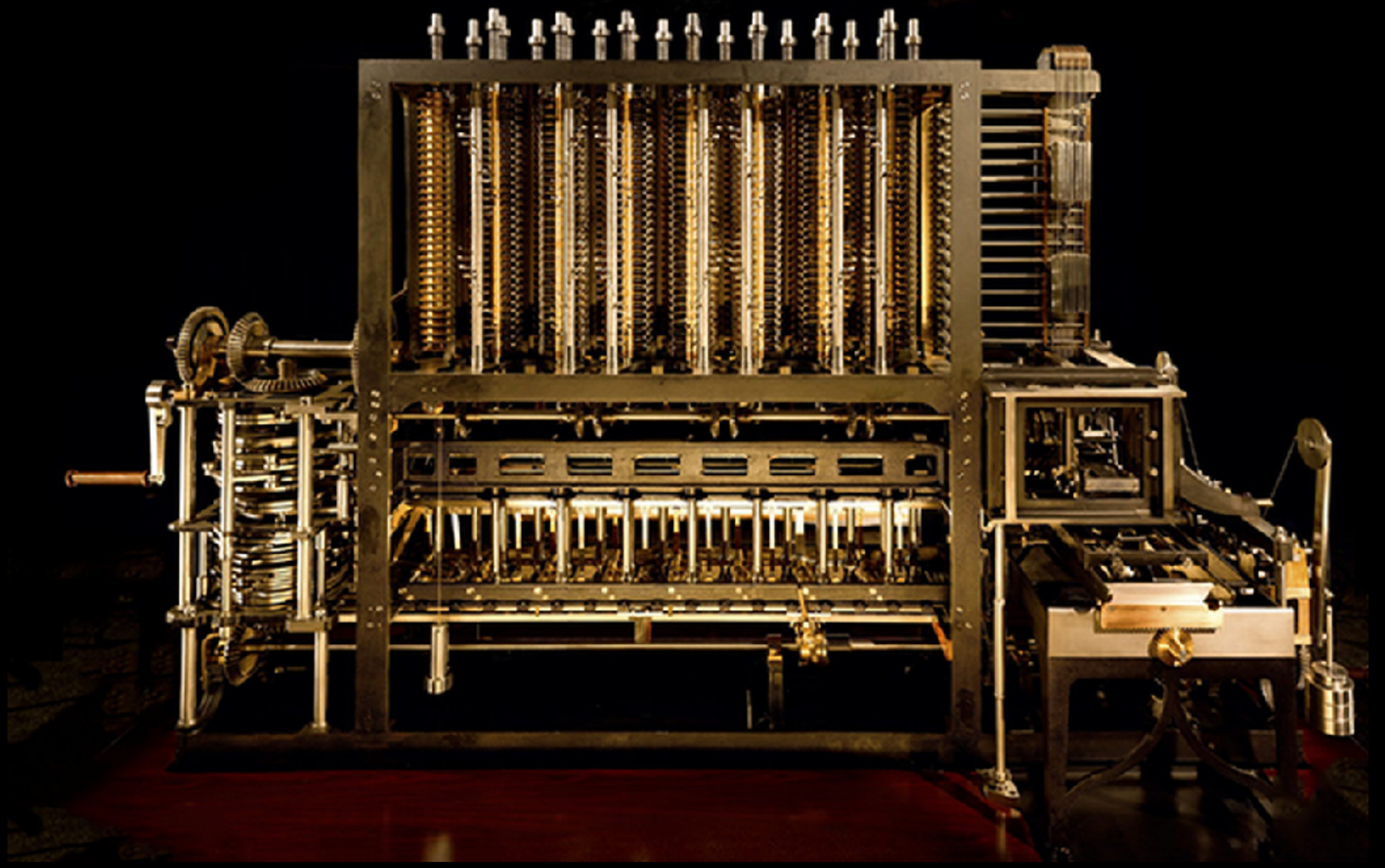
A machine learning technique that uses multiple layers of non-linear transformations

complex learning techniques using gradient-based optimization and backpropagation to process large amounts of data effectively

A technique to solve problems by providing data, and objective and often some sort of label, letting the computer find the solution using neural networks.

A very short history of AI

- 1842 Ada Lovelace & Charles Babbage work on the “analytical engine”
- 1950 Mathematics of Neural Networks
- 1970 Symbolic AI (expert systems)
- 1990-2010 statistical foundations
  - Swarm Intelligence
  - “Commodore 64” models
- 2012 Deep learning
- 2017 Transformers
- 2023 chatGPT4

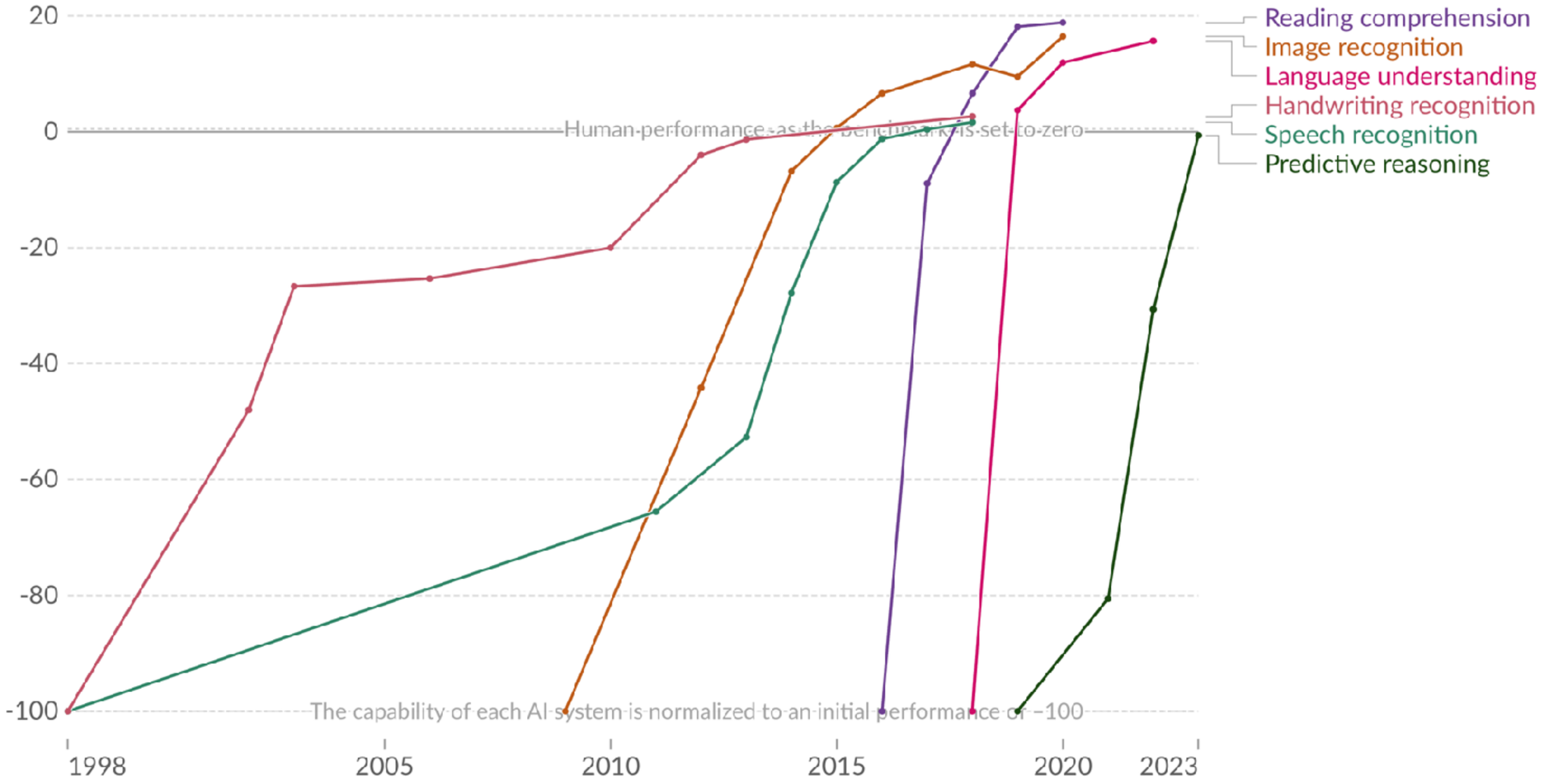


Wizard vs Prophet

Test scores of AI systems on various capabilities relative to human performance

Our World in Data

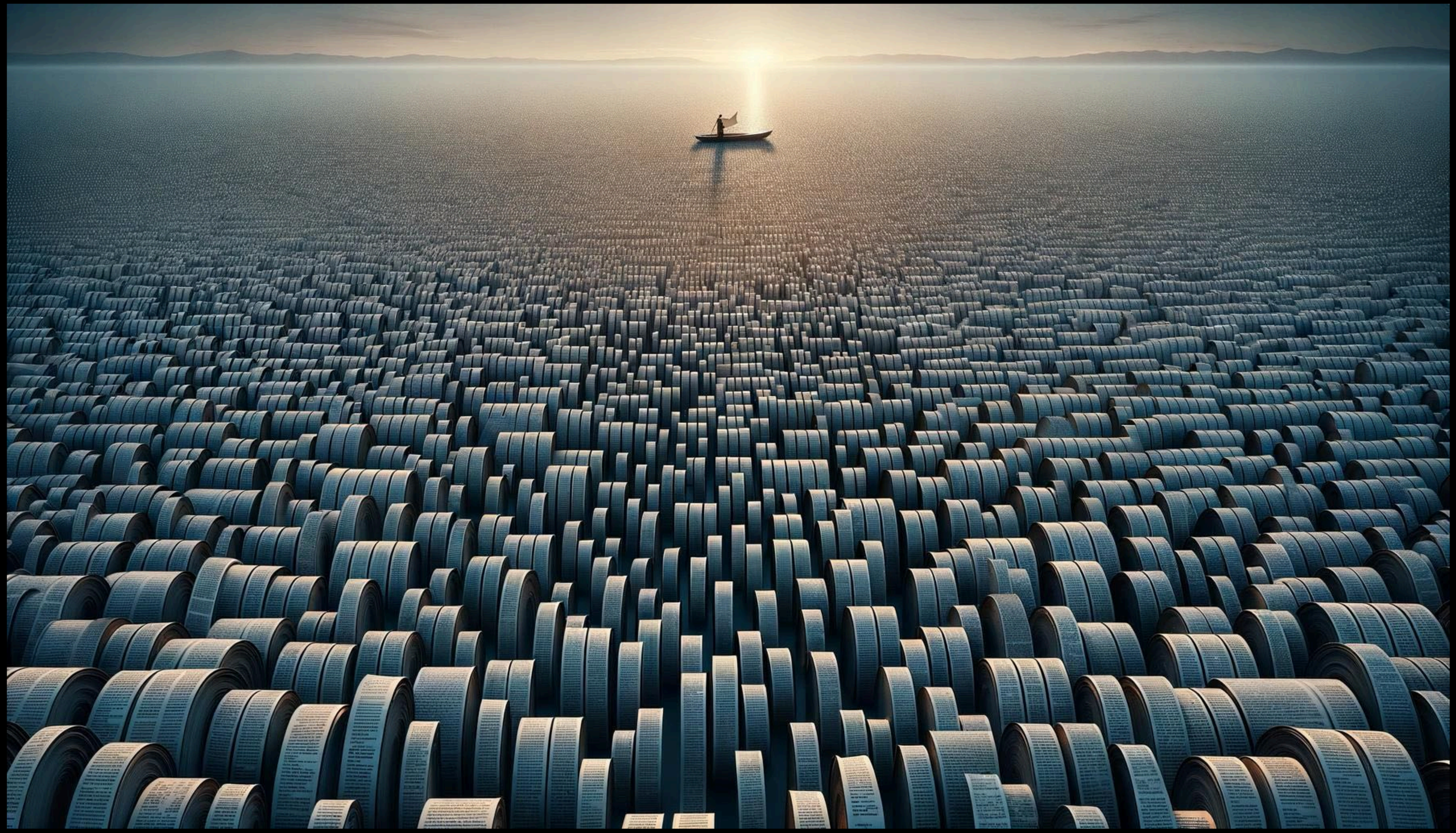
Within each domain, the initial performance of the AI is set to −100. Human performance is used as a baseline, set to zero. When the AI's performance crosses the zero line, it scored more points than humans.

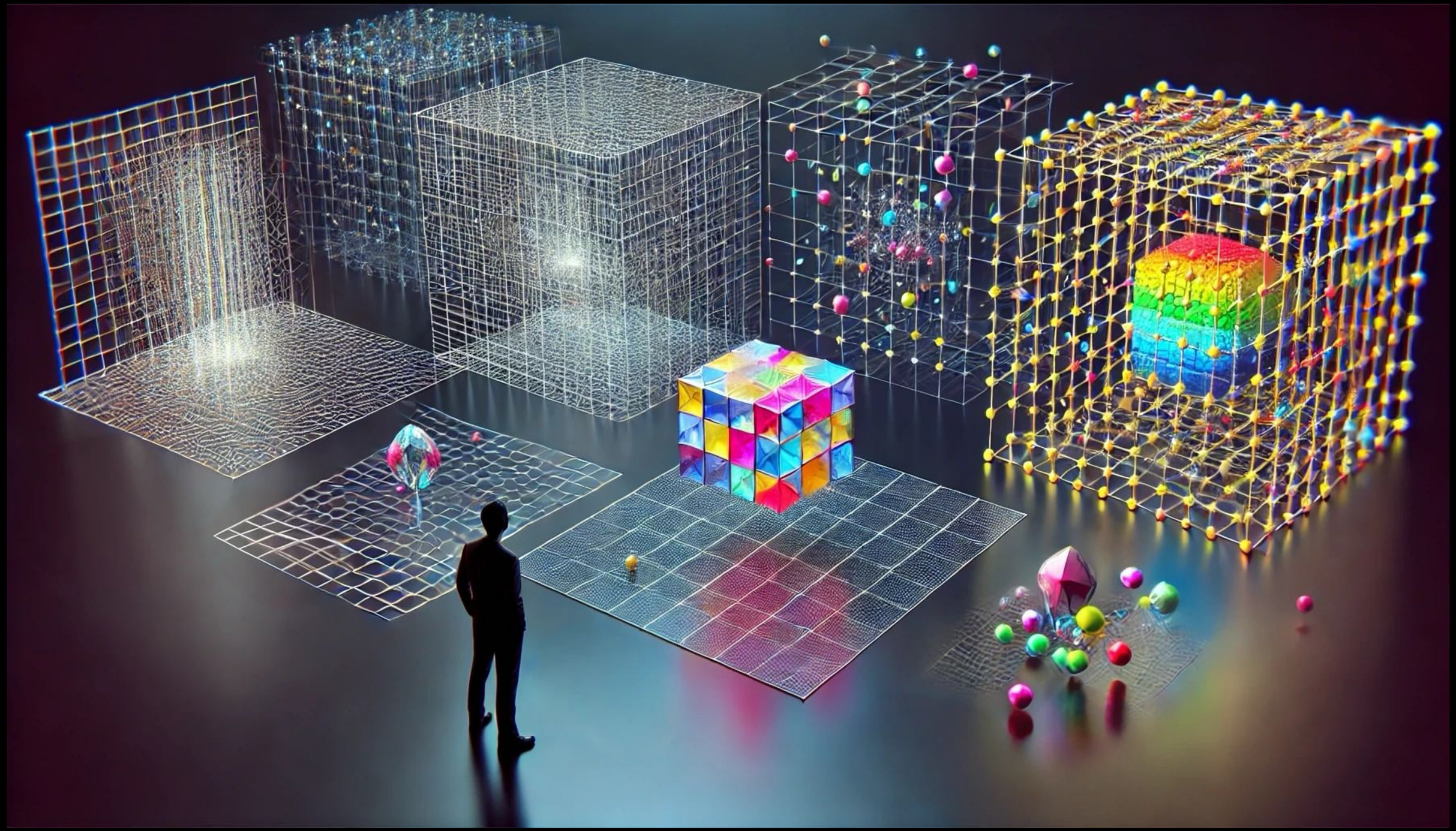


Data source: Kiela et al. (2023)

OurWorldInData.org/artificial-intelligence | CC BY

Note: For each capability, the first year always shows a baseline of −100, even if better performance was recorded later that year.

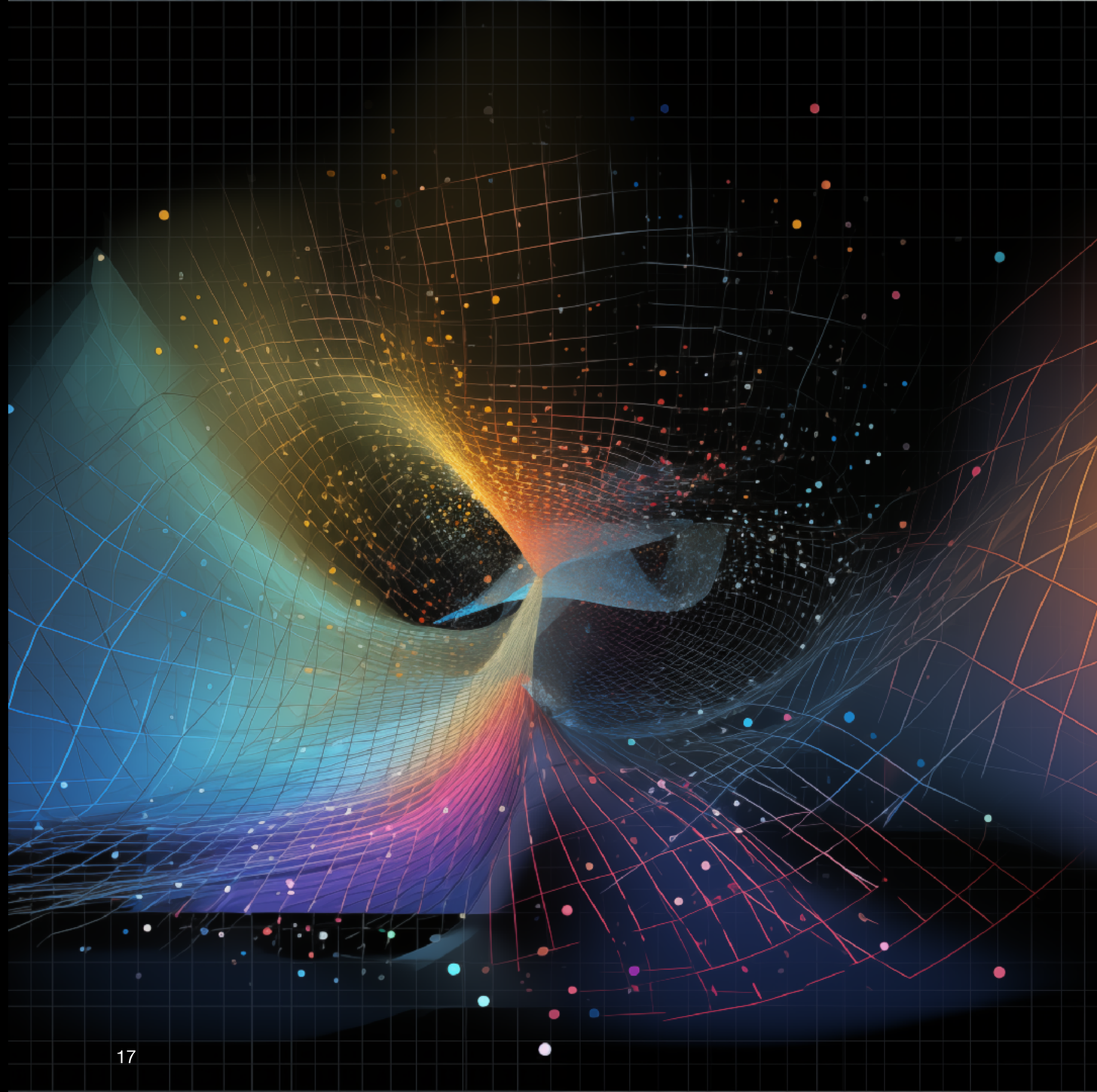


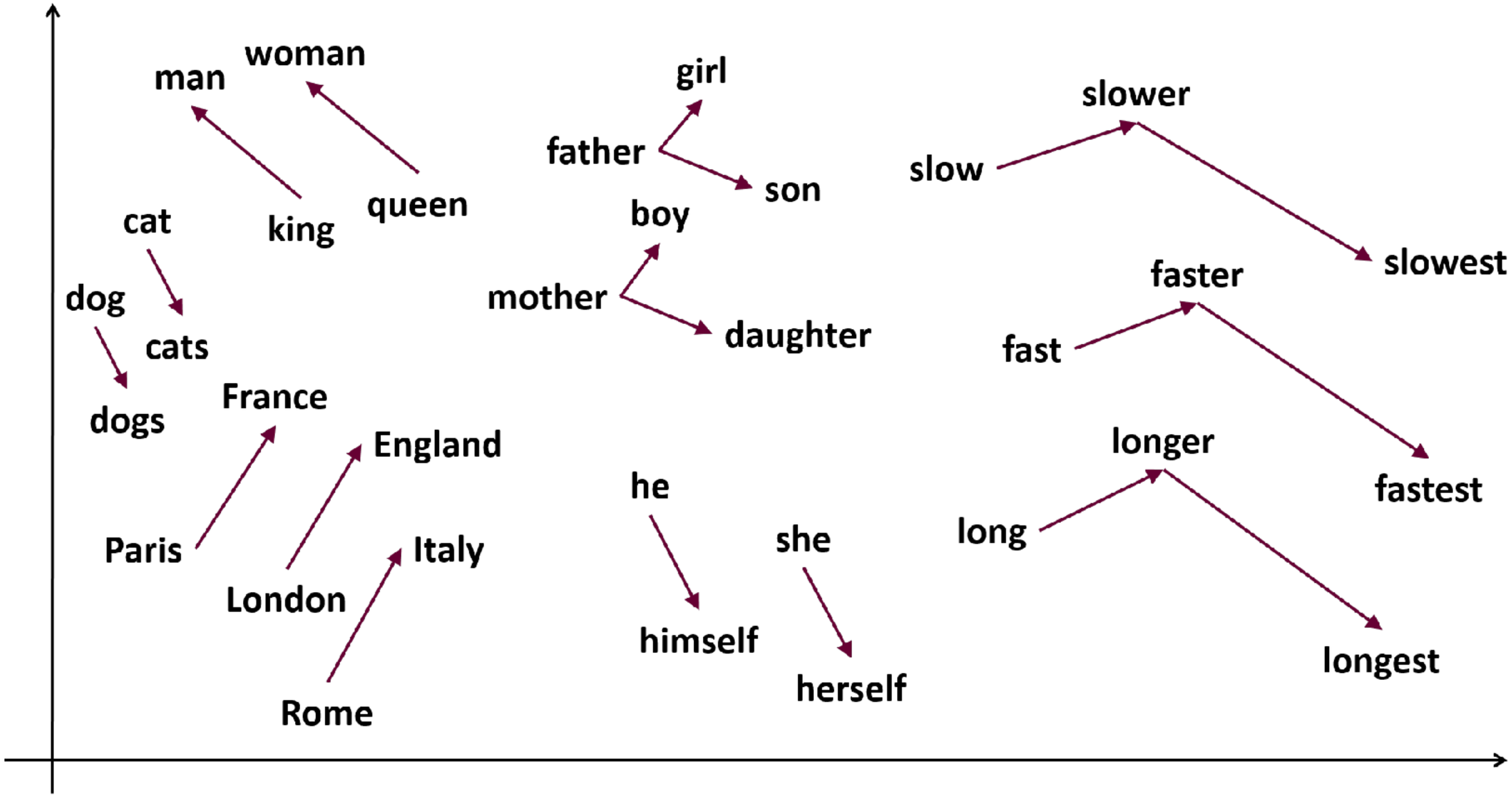


Atomen in het zichtbare heelal : 10^{80}

In een vectorruimte met 80 assen ($\mathbb{R}^{80}$), waarbij je op elke as 10 locaties gebruikt, heb je ruimte voor elk atoom in het heelal

LLMs gebruiken een semantische vectorruimte van 768 dimensies (512+256), soms zelfs 1536 : $\mathbb{R}^{768}$




man
woman
king
queen
cat
cats
dog
dogs
France
England
Paris
London
Italy
Rome
girl
father
son
boy
mother
daughter
he
himself
she
herself
slow
slower
slowest
fast
faster
fastest
long
longer
longest

Complexity 1990 - SVM

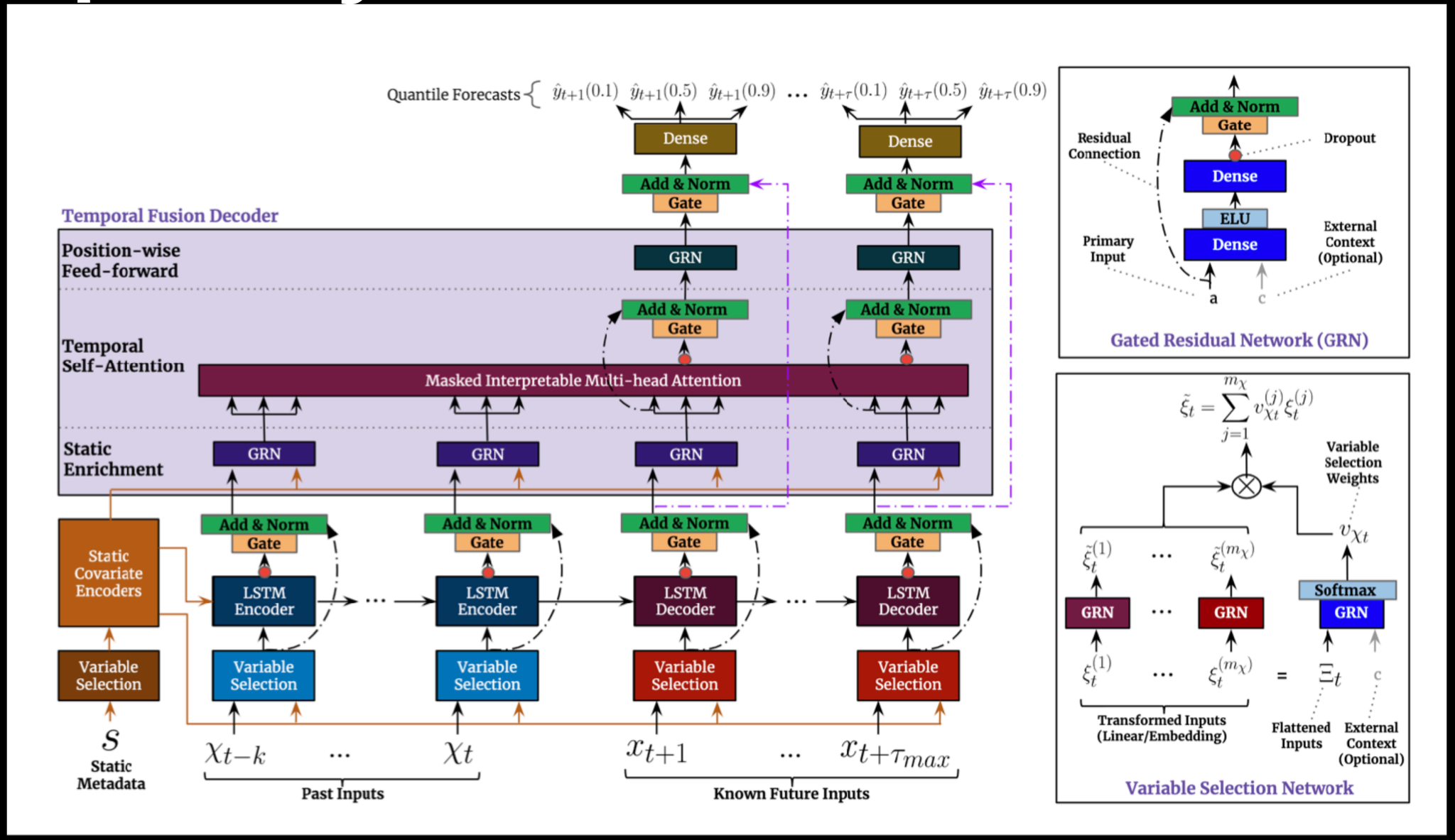
$$max_a \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j y_i y_j K(x_i, x_j)$$

$$K(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2)$$

$$J(w, b) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} max(0, 1 - t^i(w^T x^i + b))$$

m	number of observations
a	counter for errors
x	observations
y	labels
γ, C	hyperparameters
w, b	weights we need to learn

Complexity 2021 - TFT



Linear models

Even simpler models are called “retarded”

Linear models are one of the simplest models available. Their advantages are:

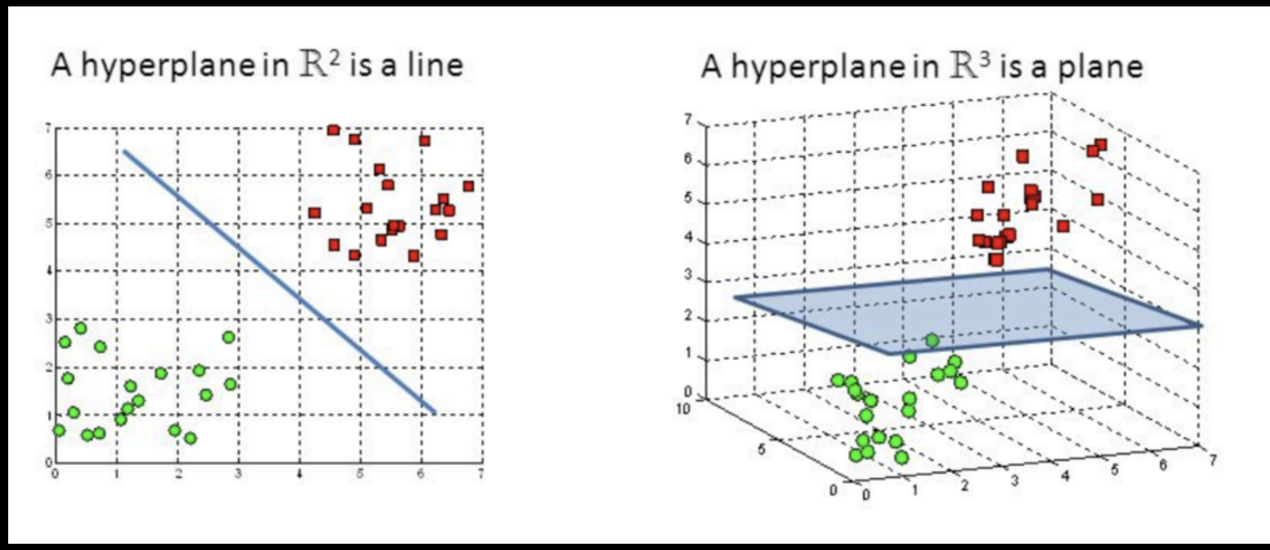
- The simplicity helps us to avoid overfitting
- They are fast and we don’t have to worry about getting stuck in a local minimum.
- They are ideal as a baseline model

Hyperplanes

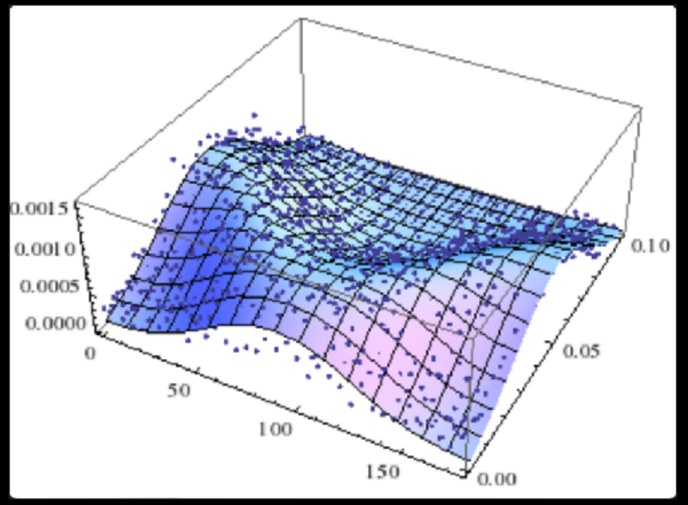
- For classification, our outcome are discrete classes (e.g. “yes” or “no”)
- For regression, our outcome is a real number (e.g. 1.4 or 26.834)

We can use a linear model for both cases. We call these models “hyperplanes”: they have one dimension less than the ambient space.

With classification, we want the data to be separated by the hyperplane. With regression, we want to points to get as close as possible to the hyperplane



Hyperplane for classification



Hyperplane for regression

Linear models

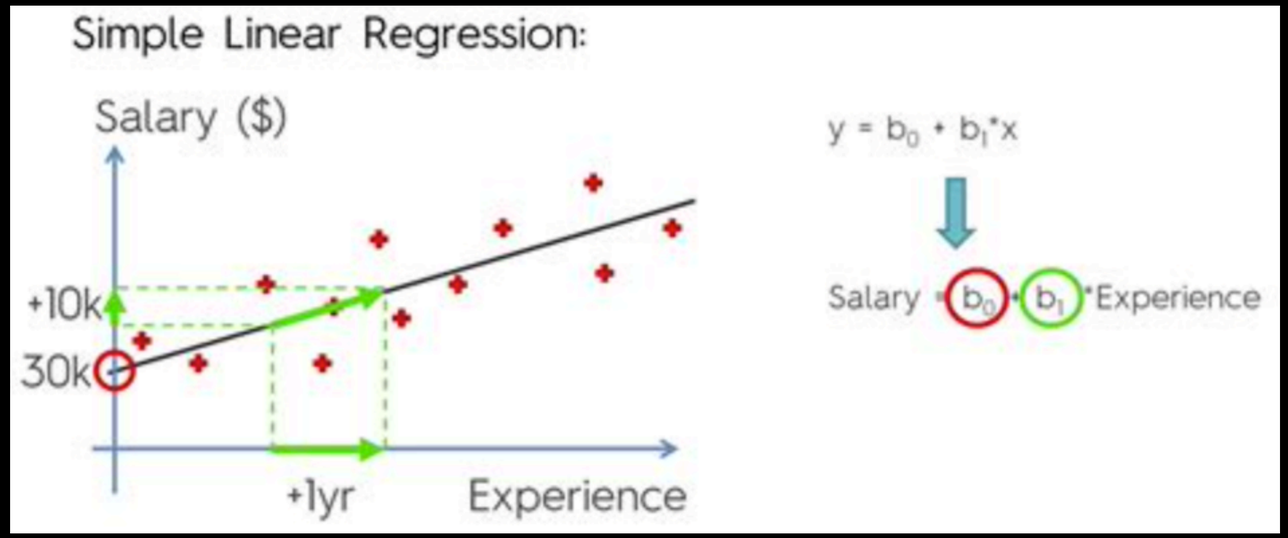
The basic mathematical shape of a linear model is:

$$Y = WX + b$$

- $Y = \{y_1, \ldots, y_m\}$ are labels, so we have m labels.
- Every label Y_i has corresponding features $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$, so we have n features.
- We can store the observations in an (m, n) matrix X
- We can store n weights for every feature in a matrix W
- b is an additional bias weight

The matrix notation is concise. We could write this out in full as:

$$y_i = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

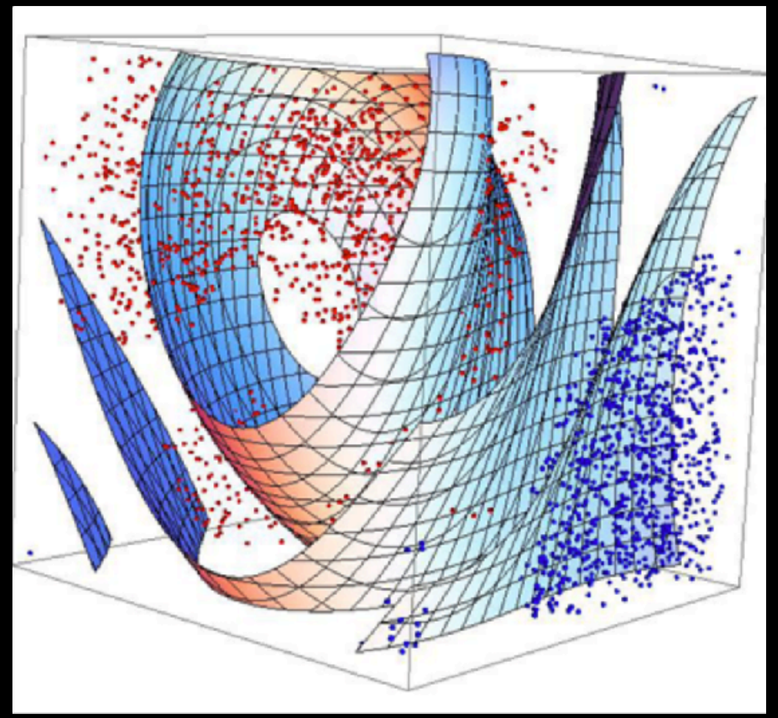


Non-linear models

A lot of data is non-linear. This means we would need a curved hyperplane.

One trick to do this is the kernel-trick, which is commonly used with Support Vector Machines, which we touched upon in the short history.

Deep learning has found another trick, we will look at the labs to see how this works.



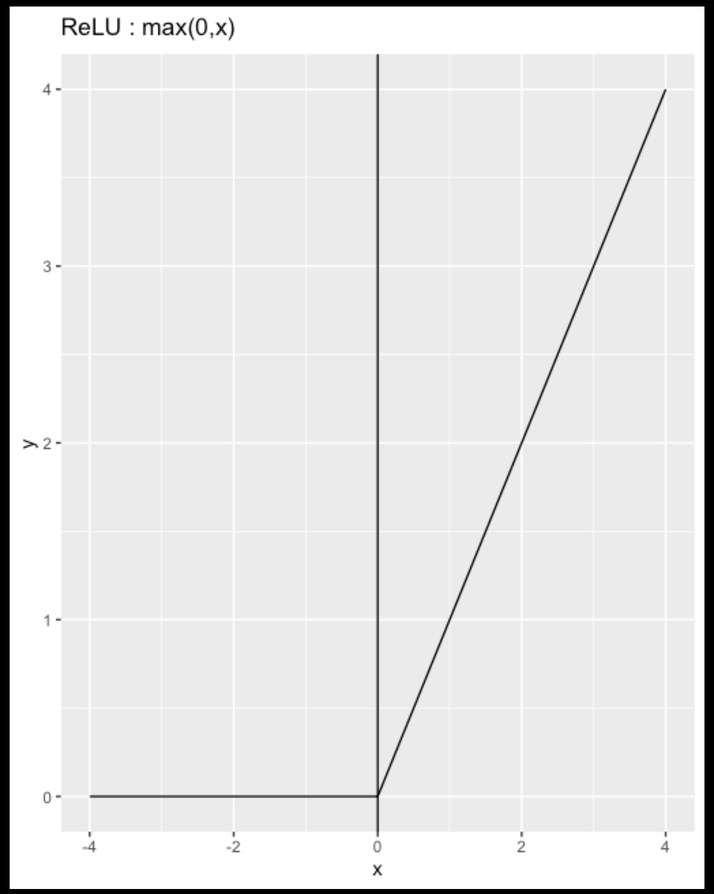
Machine learning

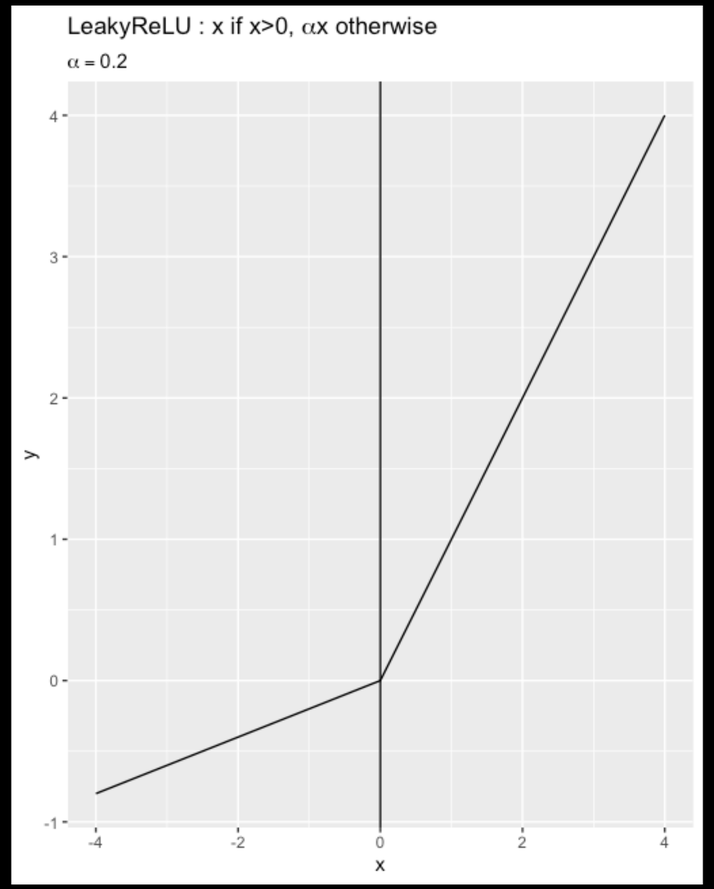
$$f\colon X \to y$$

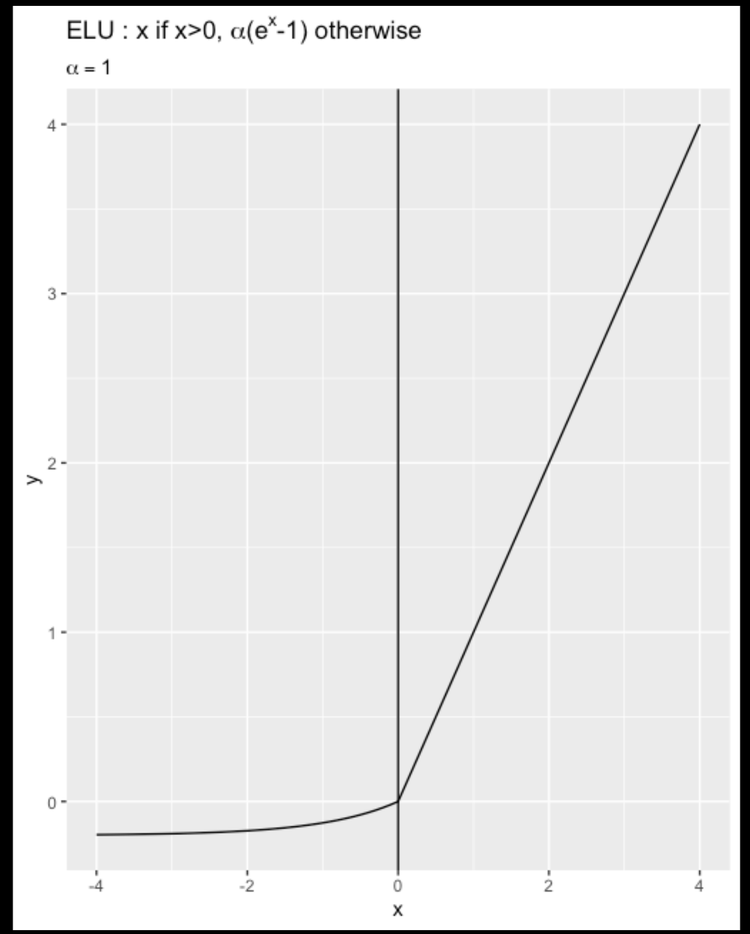
Deep Learning

Data	$X = \{\overrightarrow{x_1}, \ldots, \overrightarrow{x_n} \mid \vec{x} \in \mathbb{R}^d\}$
	$y = \{y_1, \ldots, y_n \mid y \in \{0,1\}\}$
Trainable Weights	W, b
(Non)linearity	$f(X) = WX + b \quad \sigma(X) = max(0,X)$
Predict	$\hat{y} = f_n \circ \sigma \circ f_{n-1} \circ \ \ldots \ \circ \sigma \circ f_1$
Loss	$Loss(y, \hat{y})$
Optimize	$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$

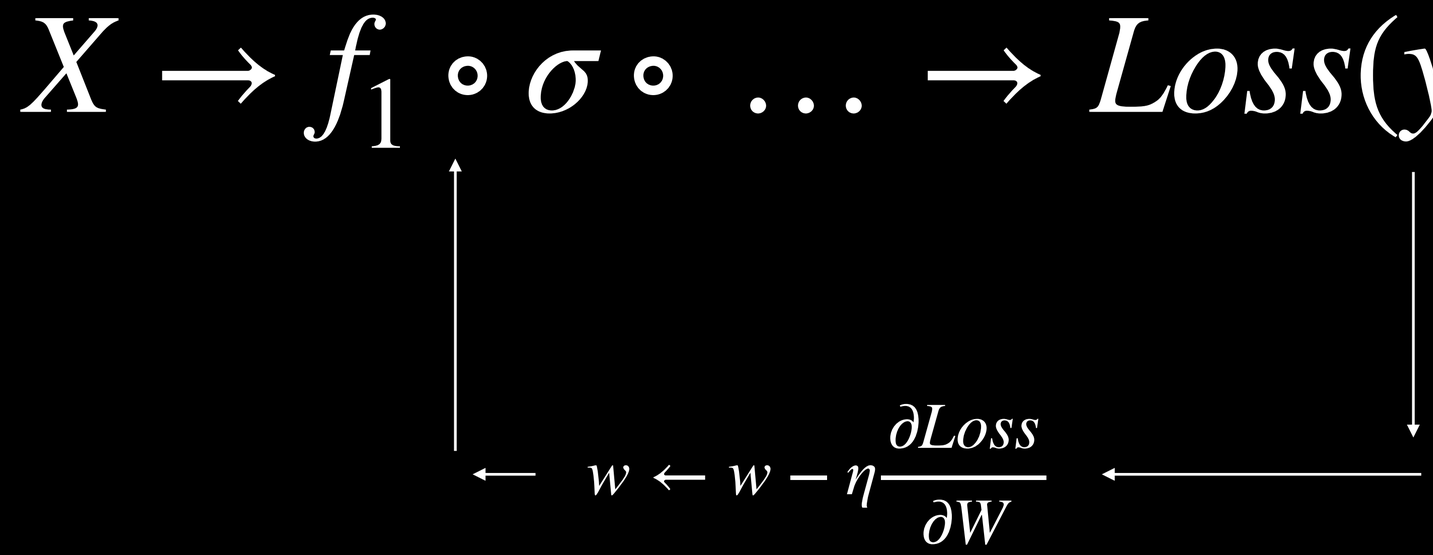
Activations





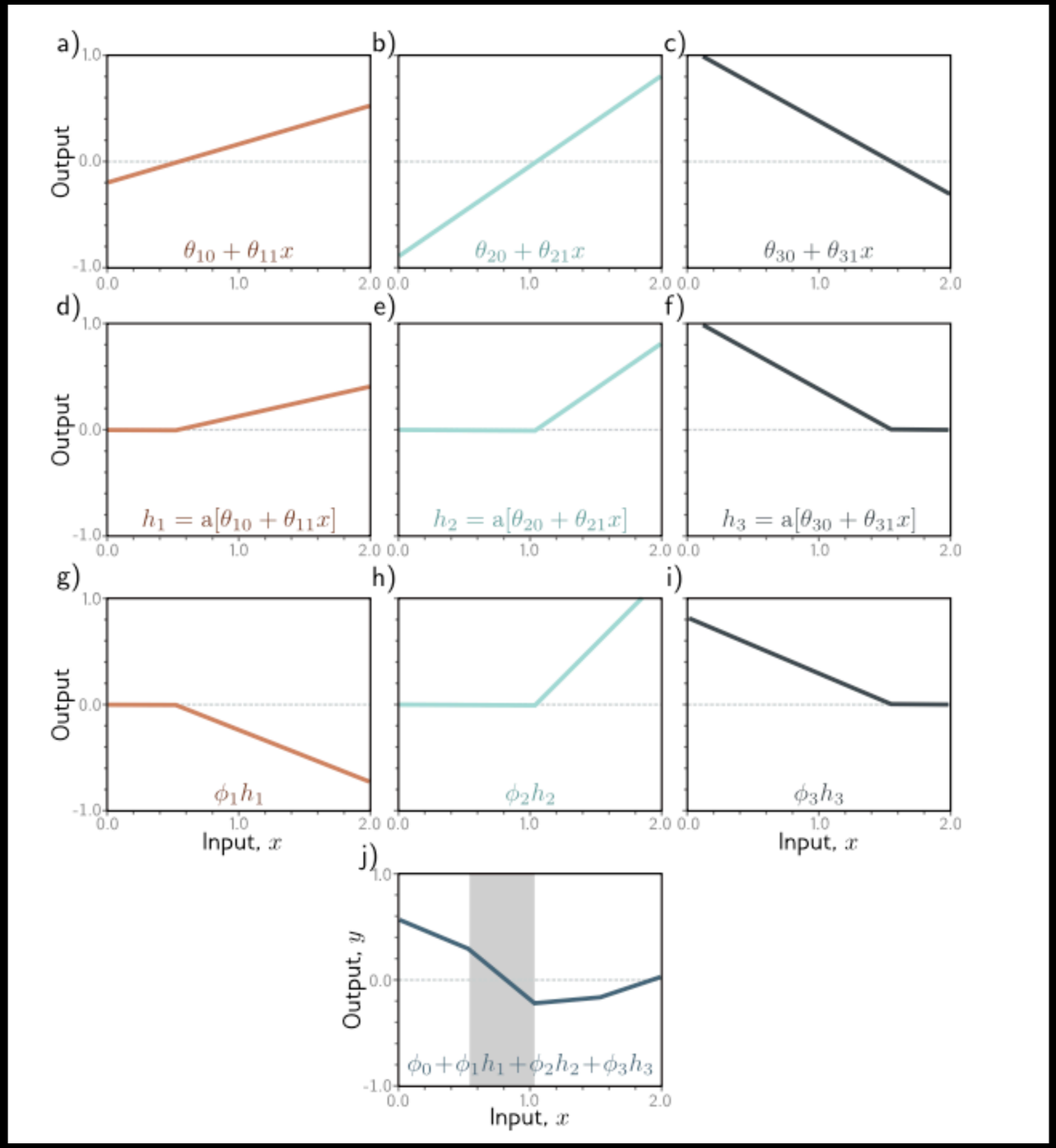


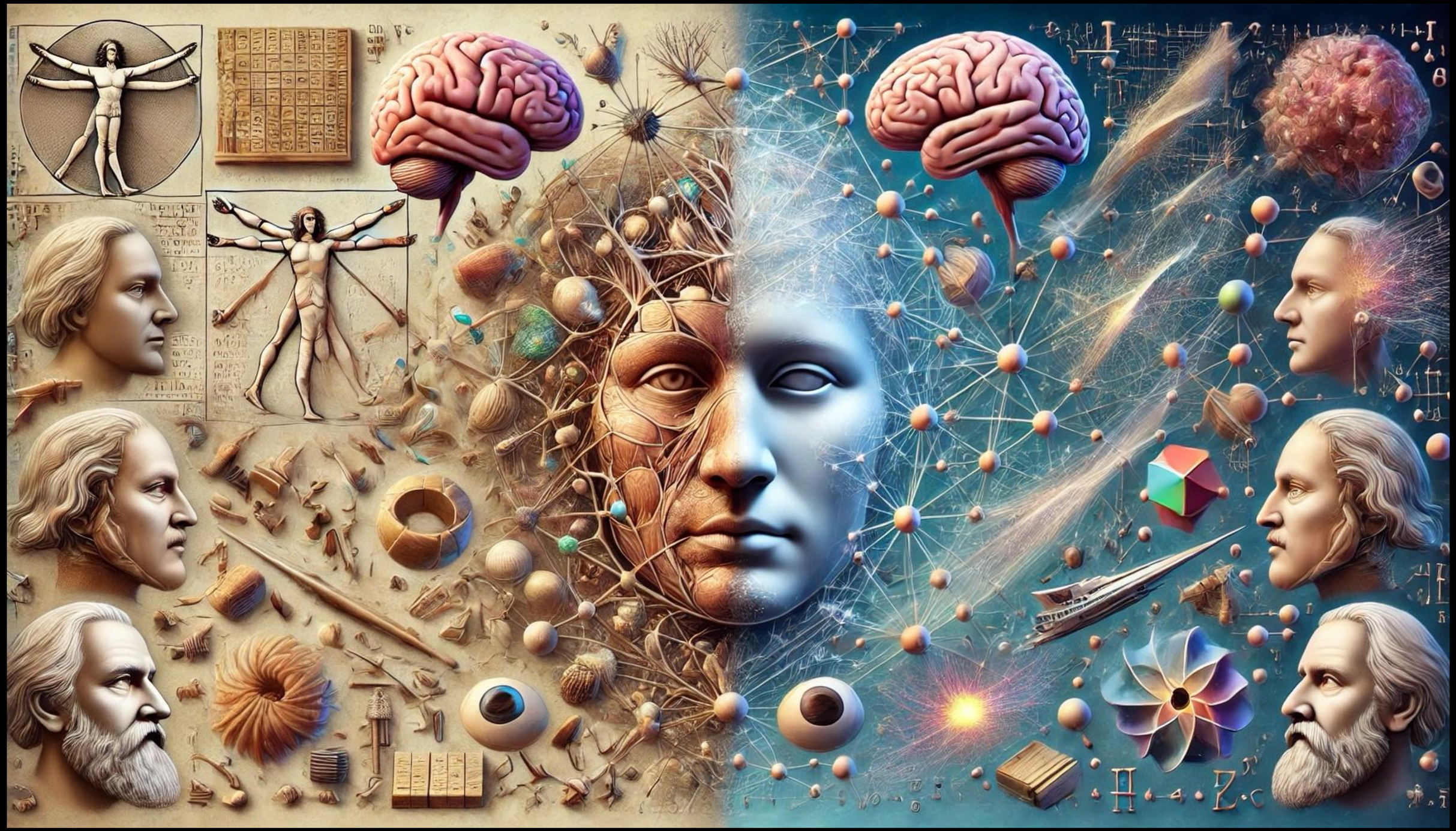
Training


$X \rightarrow f_1 \circ \sigma \circ \ \ldots \ \rightarrow Loss(y, \hat{y})$
$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$

The universal approximation theorem proves that :

*for **any** continuous function, there exists a shallow network that can approximate this function to **any** specified precision*





Code style

See references/codestyle

Topic	Testing a Concept	Proof of Concept	Product	Deployment
Prefer pathlib.Path over os.path	💡	🏅	🏅	🏅
Prefer loguru over print	💡	🏅	🏅	🏅
Use pydantic for all settings	💡	🏅	🏅	🏅
pyproject.toml for dependencies	💡	🏅	🏅	🏅
Use cookiecutters	💡	🏅	🏅	🏅
Git	💡	🏅	🏅	🏅
Use formatters and linting	💡	🏅	🏅	🏅
Use typehinting	🐌	💡	🏅	🏅
Makefiles	🐌	💡	🏅	🏅
Encapsulation	🐌	💡	🏅	🏅
Open-Closed Principle	🐌	💡	🏅	🏅
single responsibility	🐌	💡	🏅	🏅
Abstract classes (ABC, Protocol)	🐌	🐌	💡	🏅
Write tests (pytest)	🐌	🐌	💡	🏅