Machine Learning and Artificial Intelligence

CSAMA 2026

Davide Risso

Table of contents

- Measuring Accuracy
- Autoencoders
- Foundation Models
- Trustworthy AI

Introduction

We can think of the data as being generated by “nature” with an unkown mechanism, which we can think of as a “black box”.

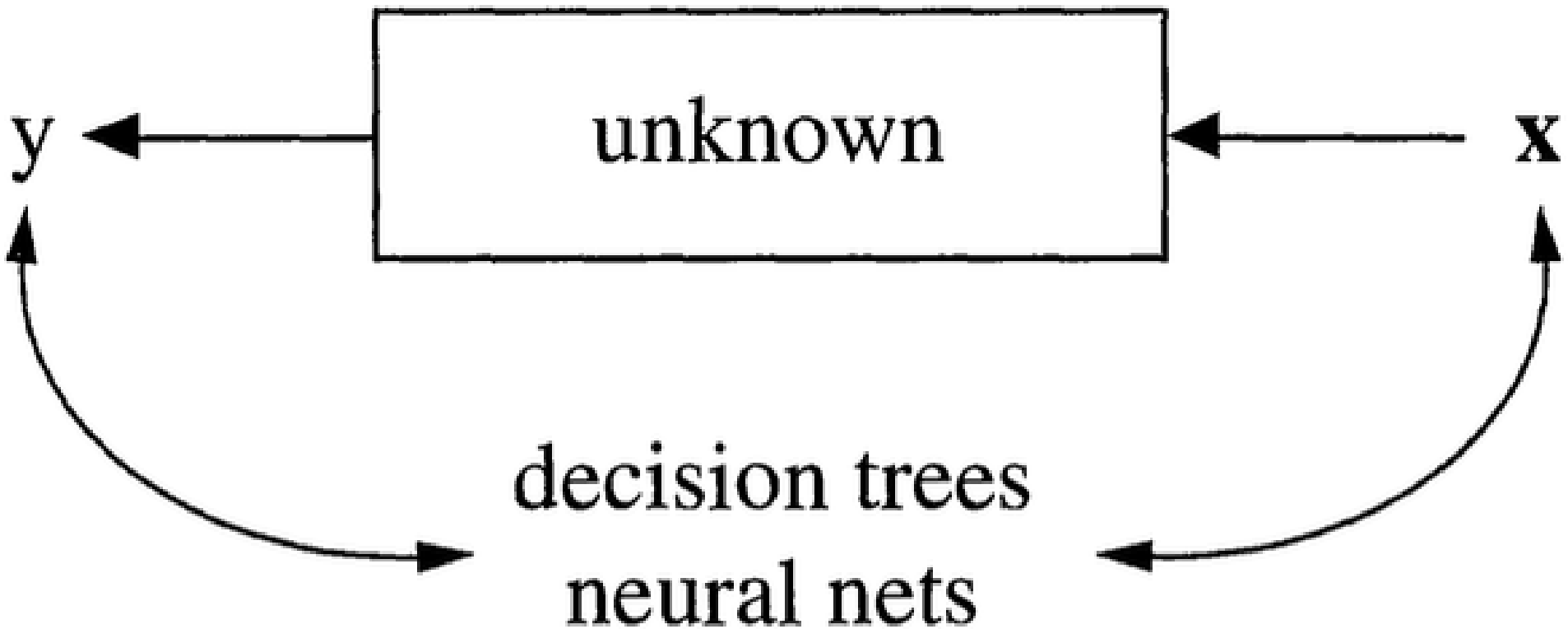
Leo Breiman

Machine Learning

In ML, we consider the data generating mechanism too complex to be described by a simple stochastic model.

Instead, we look for a function $f(x)$, seen only as an algorithm to *predict the response from the input variables*, without any assumption on the data generating process.

The Algorithmic Modeling Culture



Machine Learning

Machine Learning

We evaluate the model by looking at *how well it can predict new, unseen observations*.

We do not care much about the interpretation of the parameters of the model, but just about how well it works in predicting the response.

So we need a way to measure the *accuracy of the predictions*.

Measuring Accuracy

Example

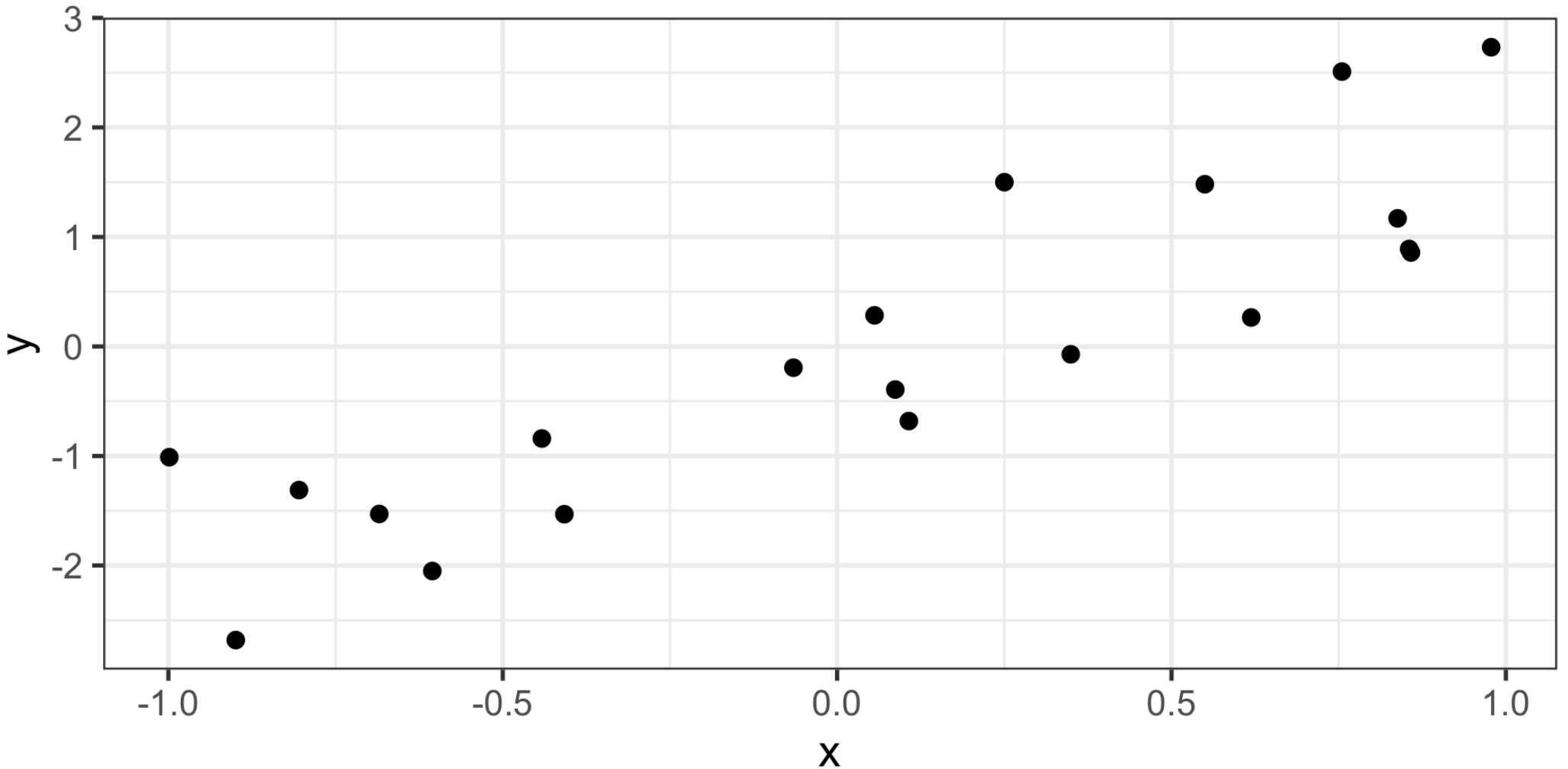
Imagine that we want to predict the value of the response y based on a predictor x.

We do not know the data generating distribution, but we collect some data on which to train a model.

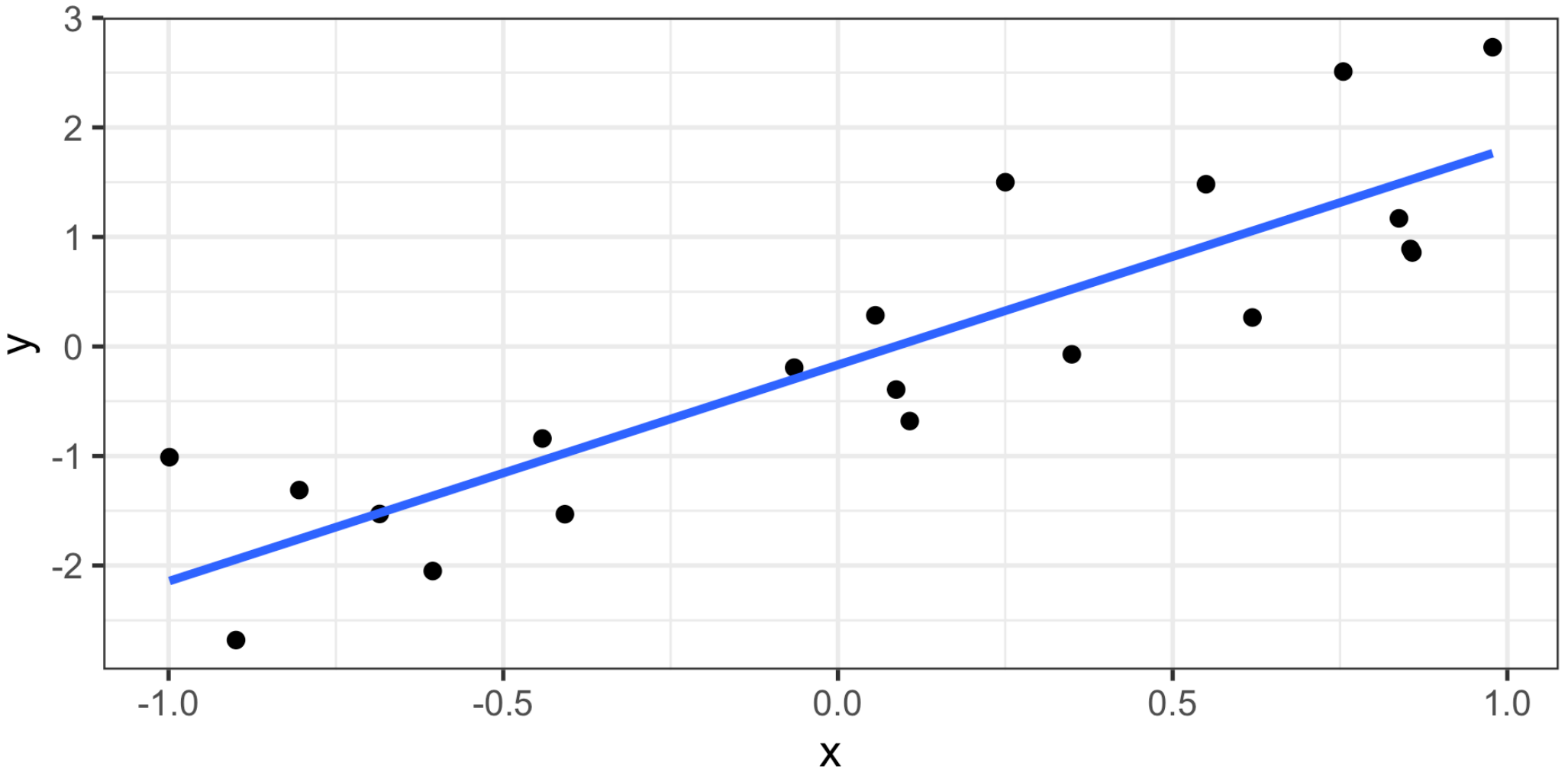
The data look like the next slide.

Which model do you prefer? Why?

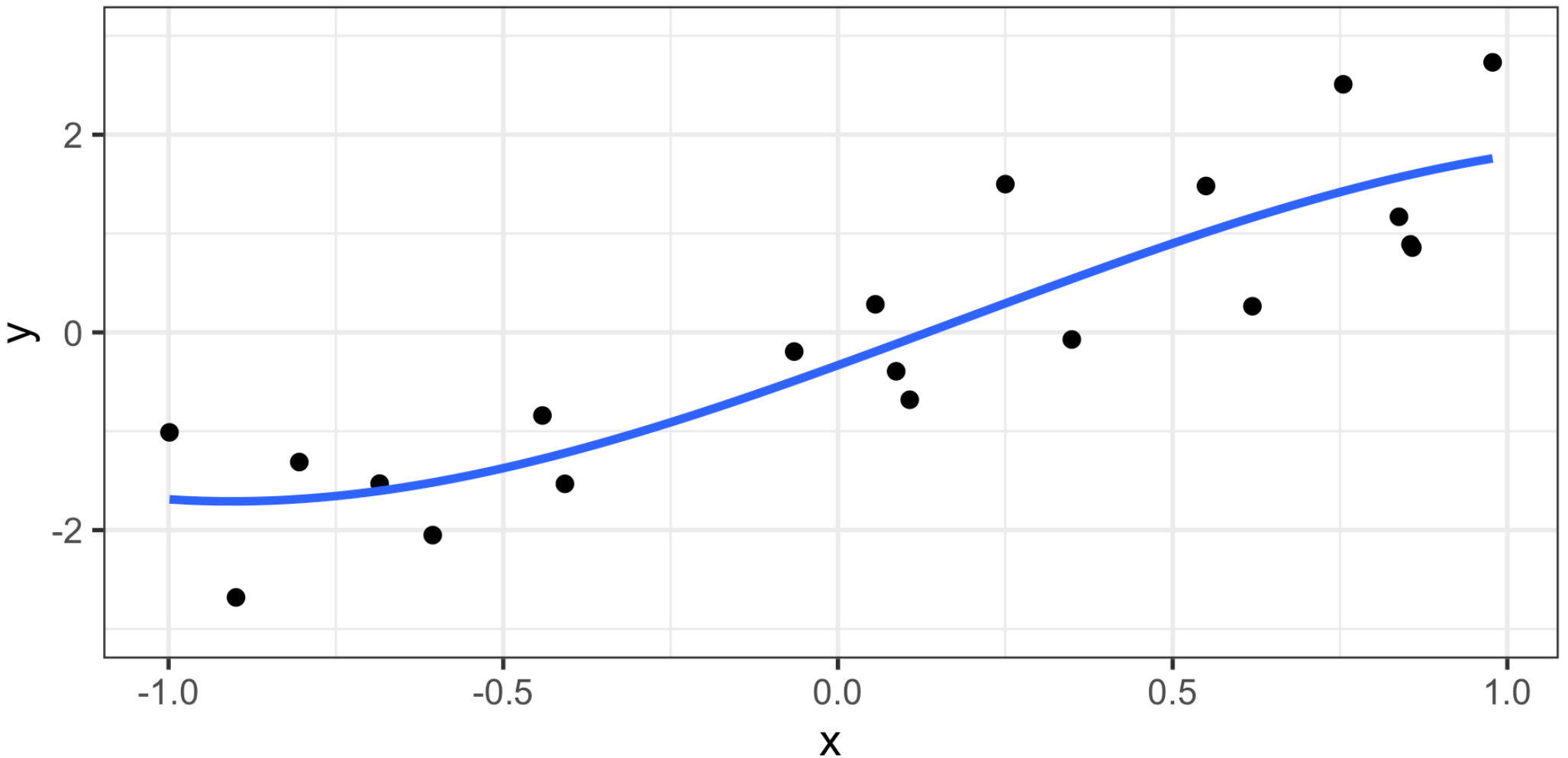
Example



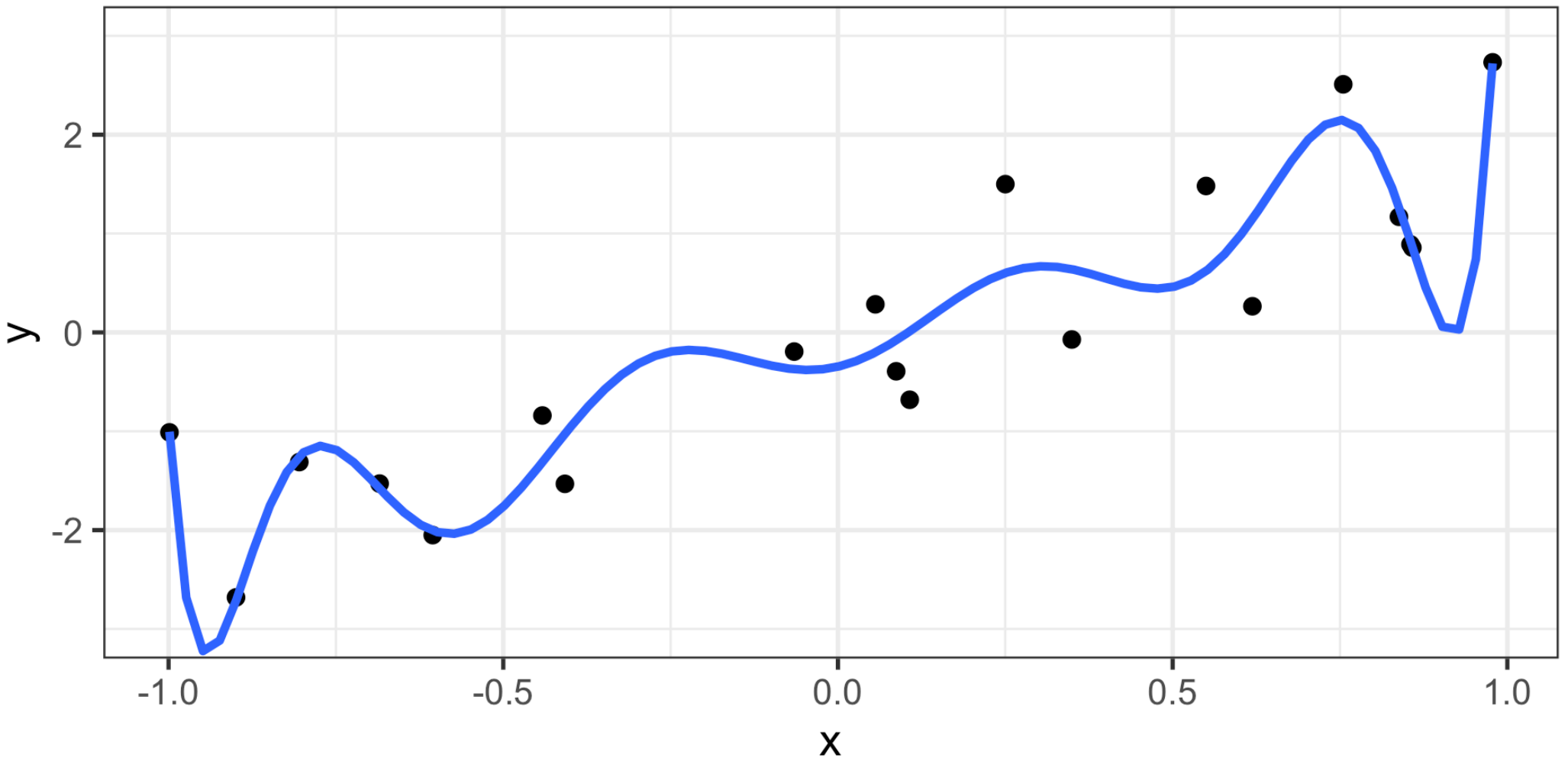
Linear model



Polynomial (degree=3)



Polynomial (degree=10)



The right balance

- The linear model is “too far” from the training data.
- The 10-degree polynomial follows “too closely” the training data.
- Intuitively, we need to find the right balance to capture the signal, without being fooled by the random fluctuations of the training data.

Training Mean Squared Error

In practice, we need some way of computing the model accuracy from the data. One common *loss function* is the Mean Squared Error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2.$$

Intuitevely, the more $\hat{f}$ accurately predicts y, the smaller the MSE will be.

However, if we compute the MSE on the *training data* we may end up *overfitting* the data.

Think of a model that interpolates the training data: it will have a training $\mathrm{MSE} = 0$, but will likely follow random noise in the training data too.

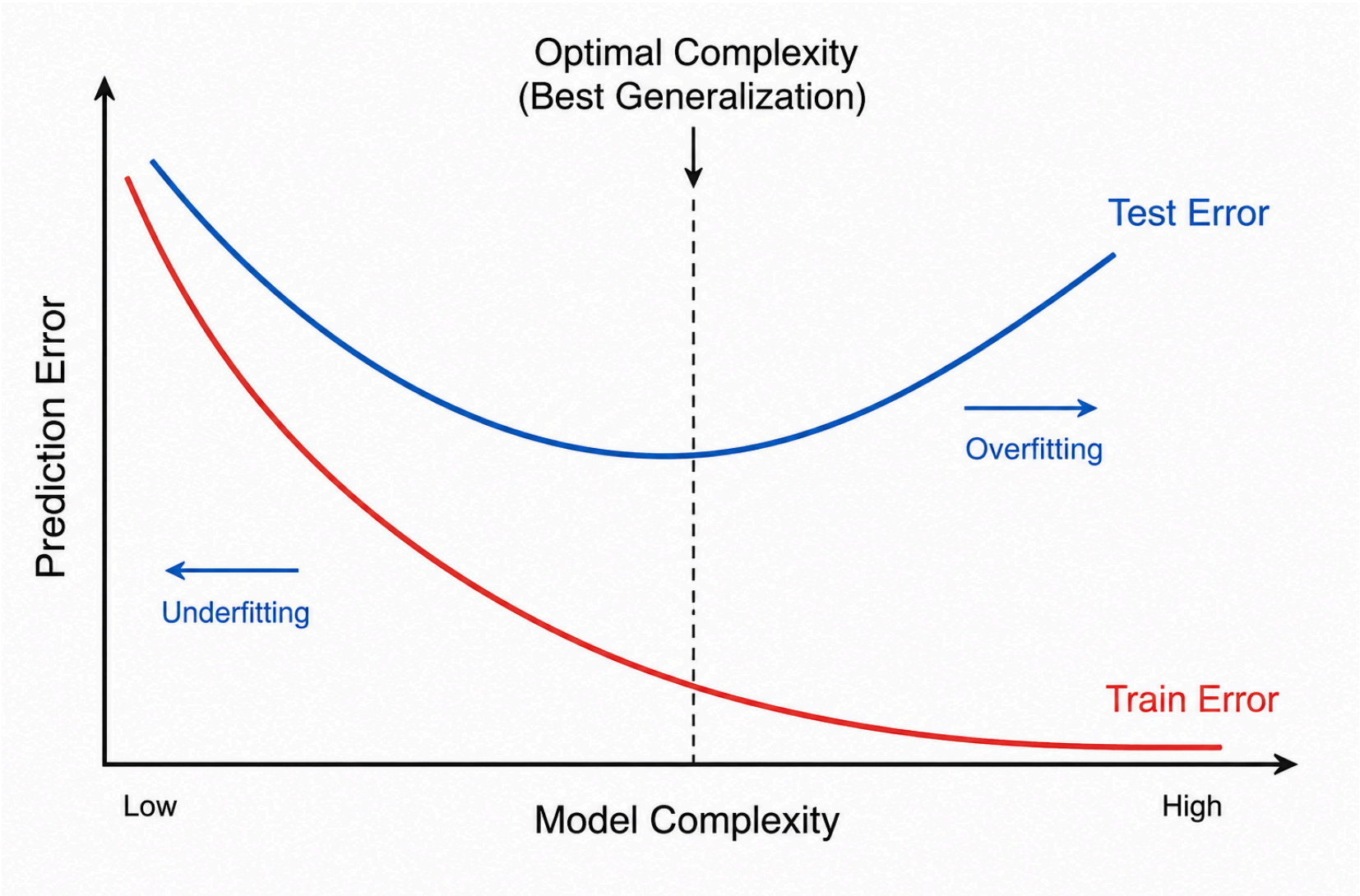
Test MSE

We are interested in the *accuracy of the predictions in new, unseen data*.

Hence, we can set aside a set of *test data* and compute the MSE on those.

You can think of the test MSE as a *better estimate* of the true population MSE.

Training and test error



Generated by ChatGPT (GPT 5.5 *Thinking* - May 2026)

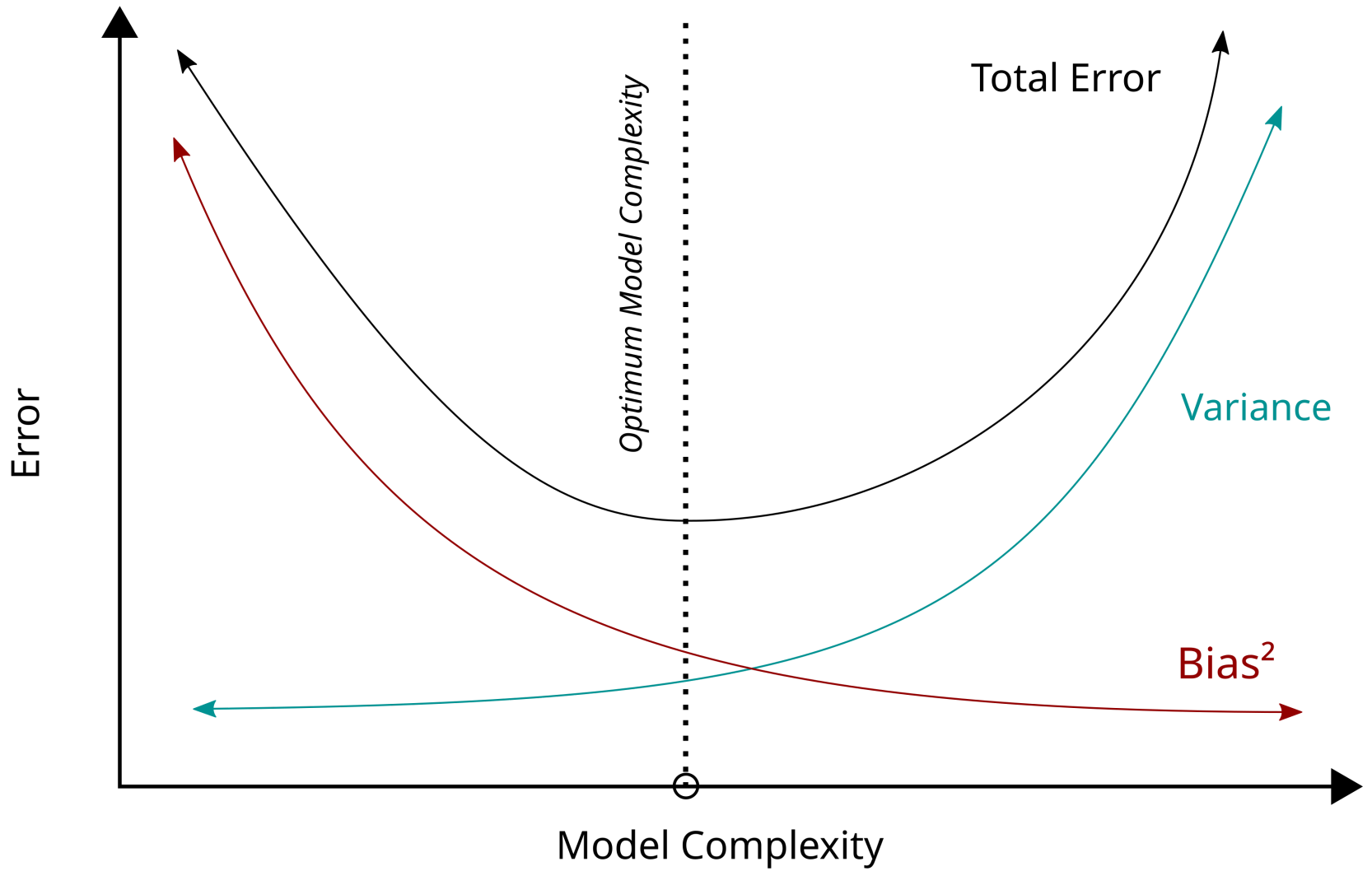
The Bias-Variance tradeoff

We can show that the reducible part of the MSE can be decomposed in two competing terms:

$$E[(y - \hat{y})^2] = \mathbf{Var}(\hat{f}(X)) + [\mathbf{Bias}(\hat{f}(X))]^2 + \mathbf{Var}(\varepsilon).$$

- The **variance** is the amount by which $\hat{f}$ would change if we estimated it using a different training data set.
- The **bias** is the error introduced by approximating Nature's generating mechanism by our prediction model.

The Bias-Variance tradeoff



Wikipedia - CC0

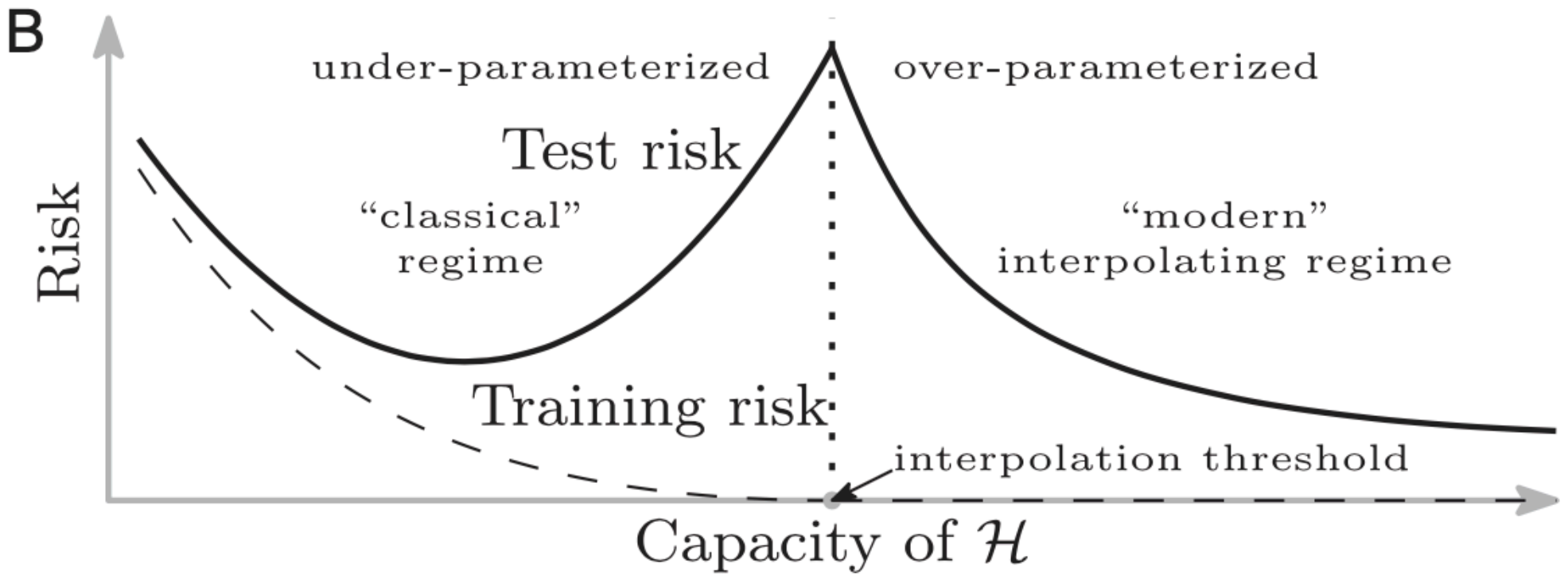
Double descent

In modern machine learning (think of LLMs), practitioners train huge models with billions of parameters that that have *very low or even zero training risk*.

While we would expect overfitting on the training data, these models often give very accurate predictions on new data.

So much so that practical guides for deep learning recommend to choose models that achieve “effortless zero-loss training” (i.e., interpolation) of the training data.

Double descent



Belkin et al. (2019)

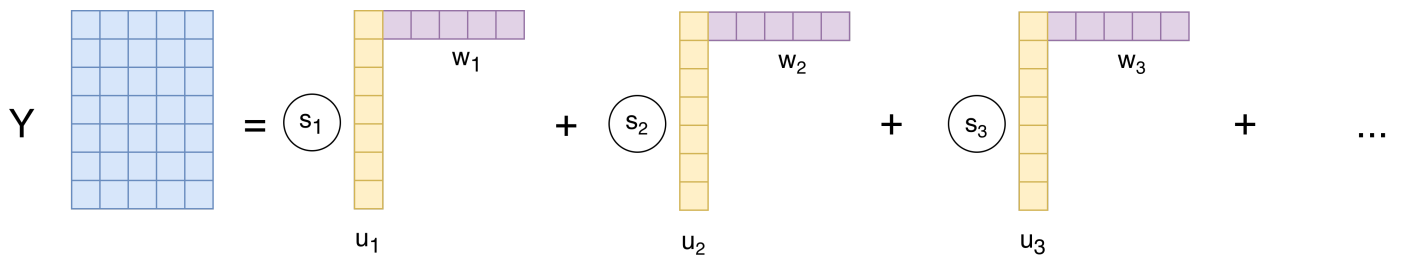
Autoencoders

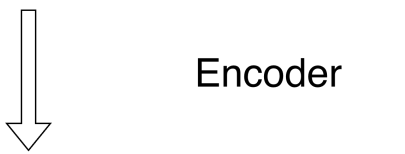
PCA as a linear autoencoder

We have seen PCA as a dimensionality reduction technique.

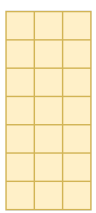
We can view it as a method that learns a low-dimensional representation of the data together with a reconstruction map.

PCA as a linear autoencoder

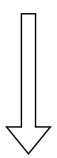




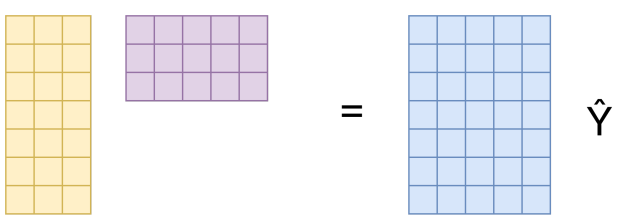
Encoder



Low-dimensional representation



Decoder



PCA as a linear autoencoder

1. An *encoder* maps each observation $y_i \in \mathbb{R}^m$ to a low-dimensional representation $u_i \in \mathbb{R}^d$.
2. A decoder maps u_i back to a reconstruction $\hat{y}_i \in \mathbb{R}^m$.

The overall reconstruction takes the form

$$\hat{Y} = f_d(f_e(Y))$$

Autoencoders

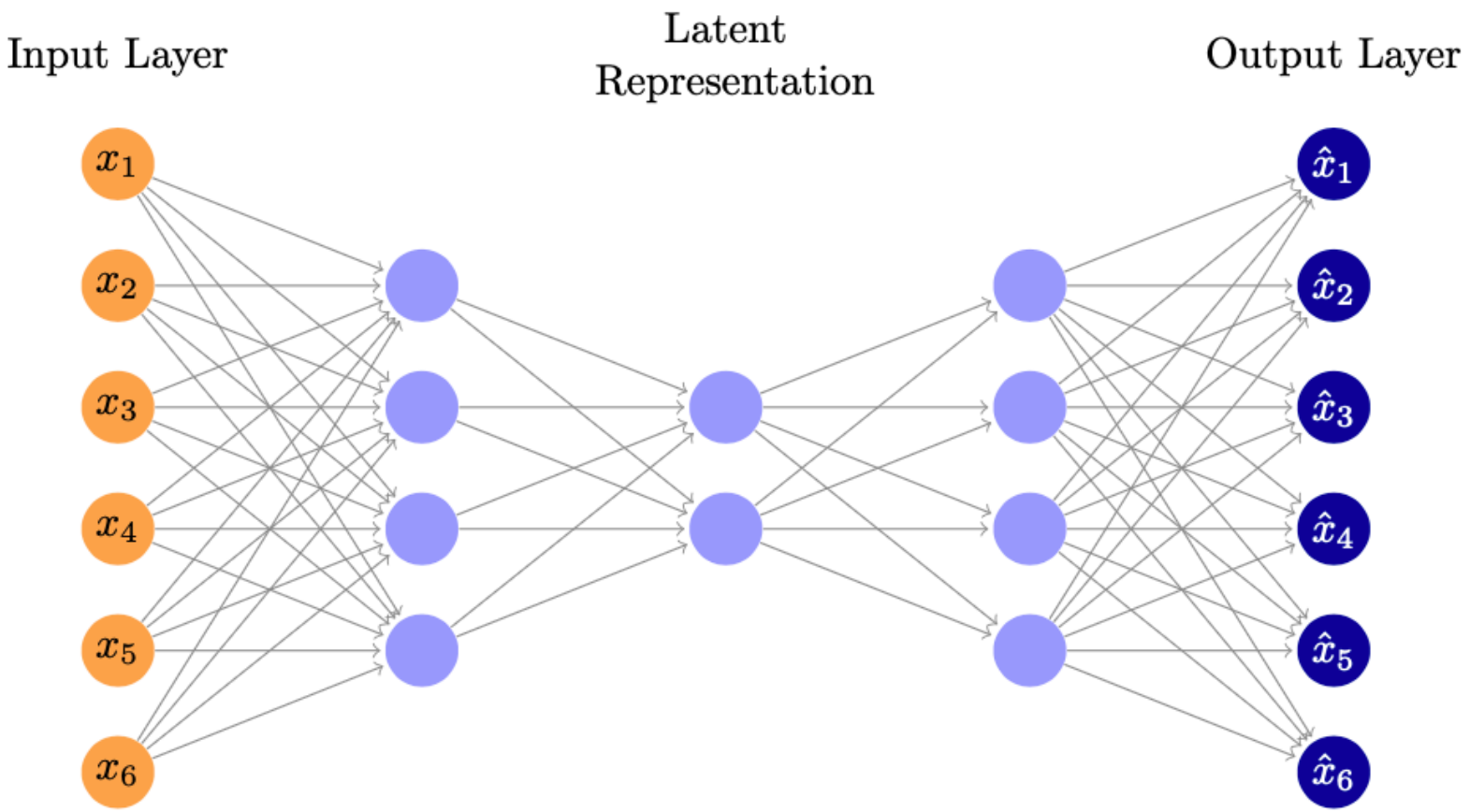
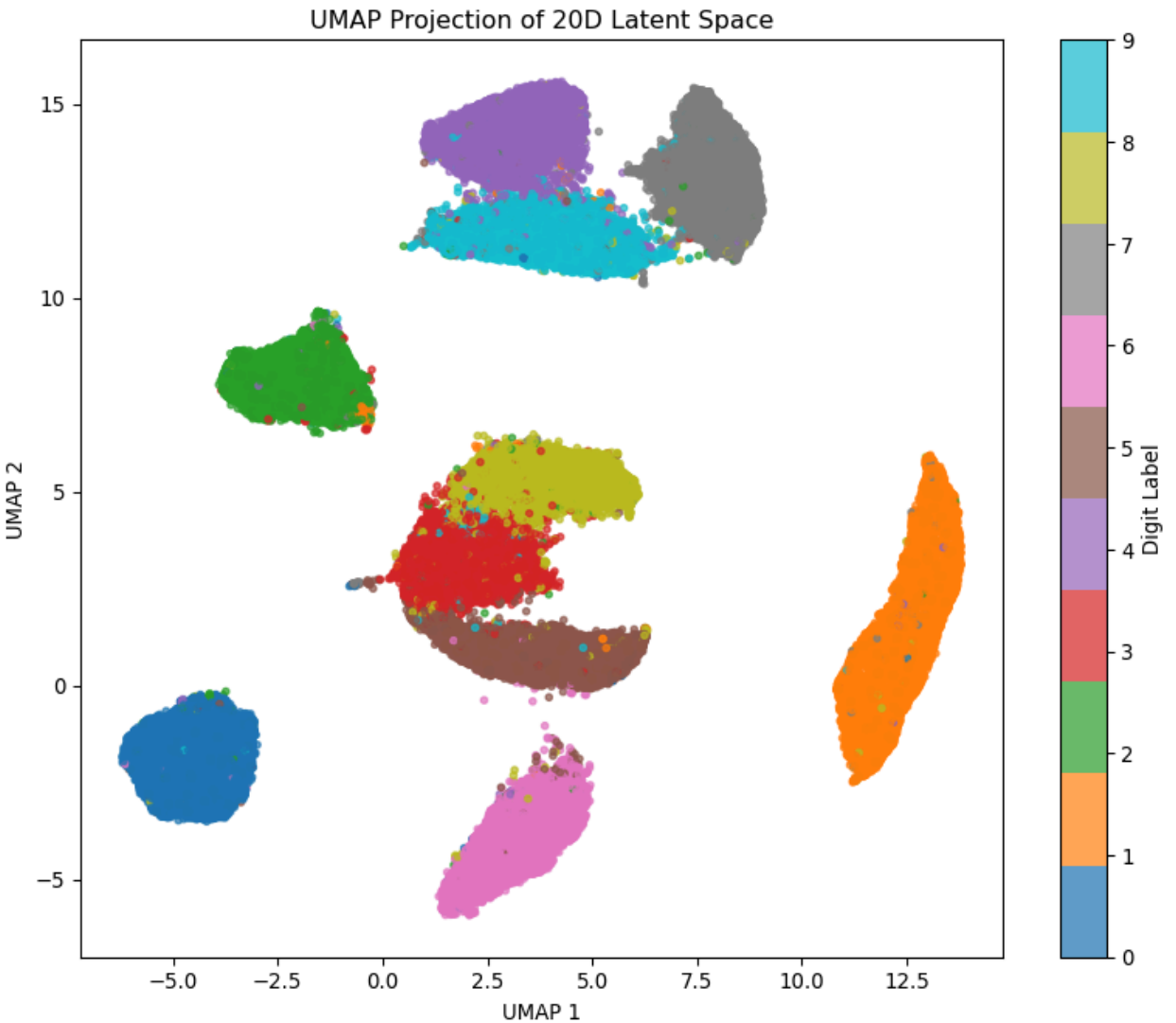


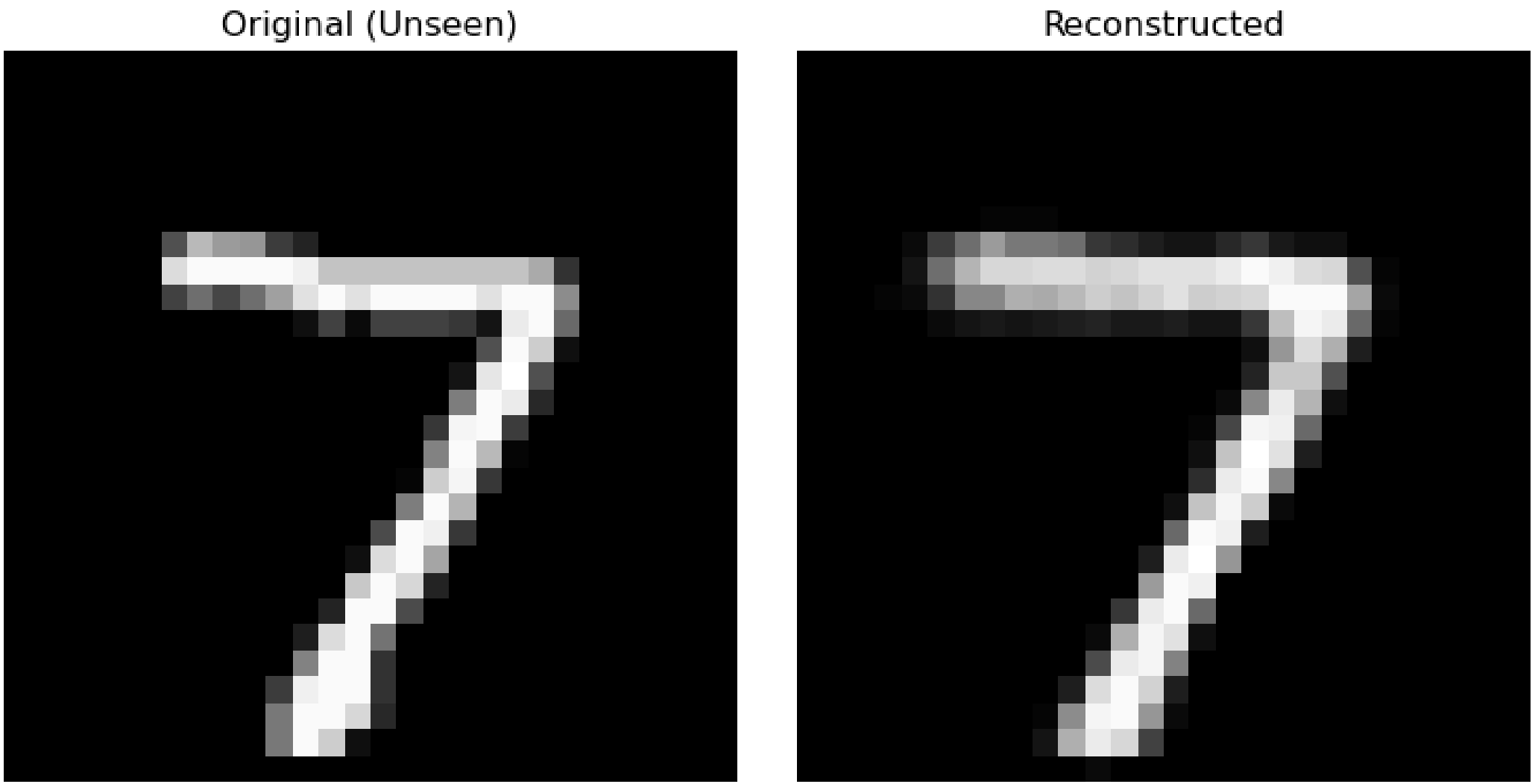
Figure 16.13: Illustration of an autoencoder with 3 hidden layers.

Example: MNIST digits



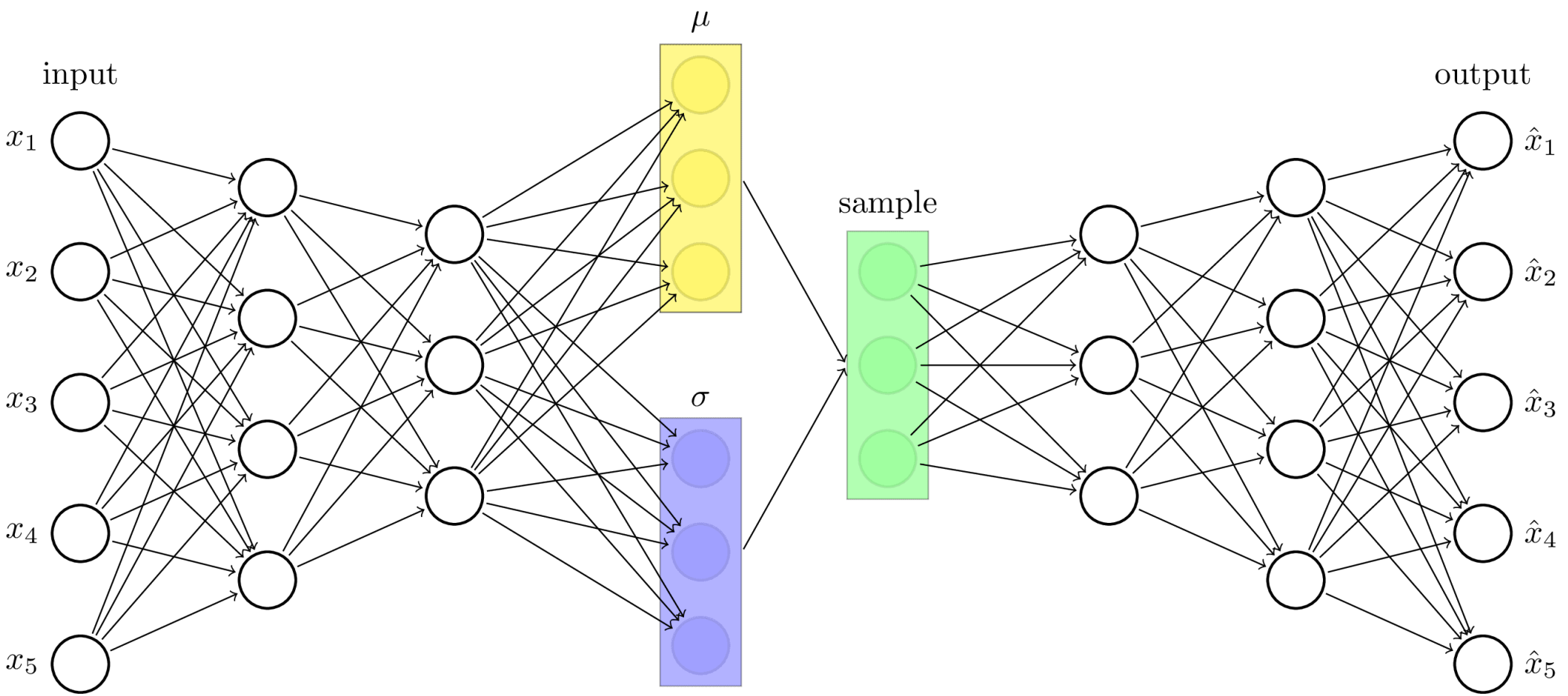
Autoencoder embedding

Example: MNIST digits



Autoencoder image reconstruction

Variational Autoencoders



Differences between AE and VAE

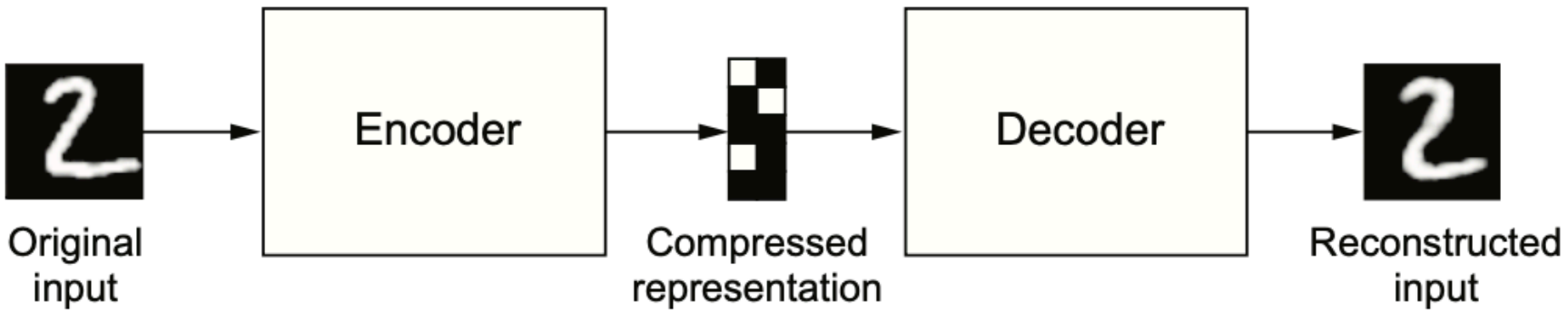


Figure 12.16 An autoencoder mapping an input *x* to a compressed representation and then decoding it back as *x'*

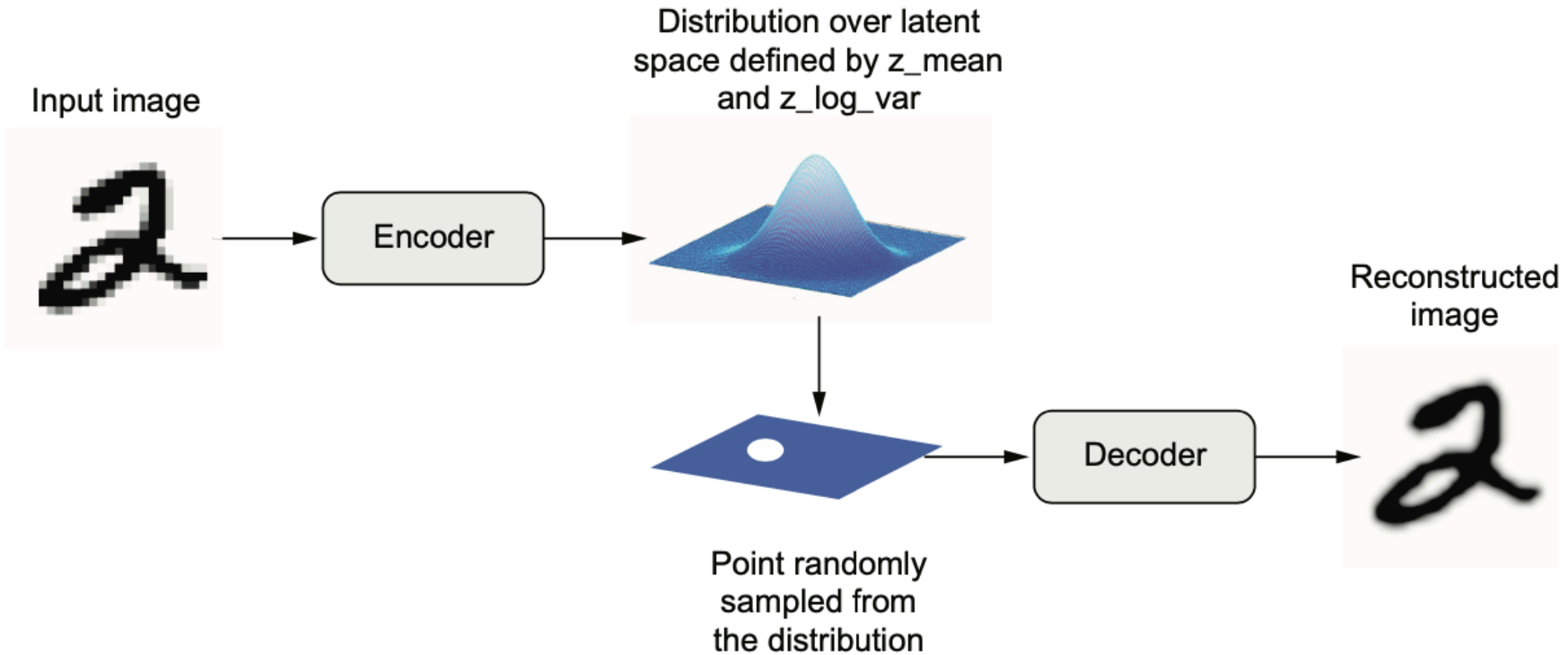


Figure 12.17 A VAE maps an image to two vectors, `z_mean` and `z_log_sigma`, which define a probability distribution over the latent space, used to sample a latent point to decode.

VAE as a generative model

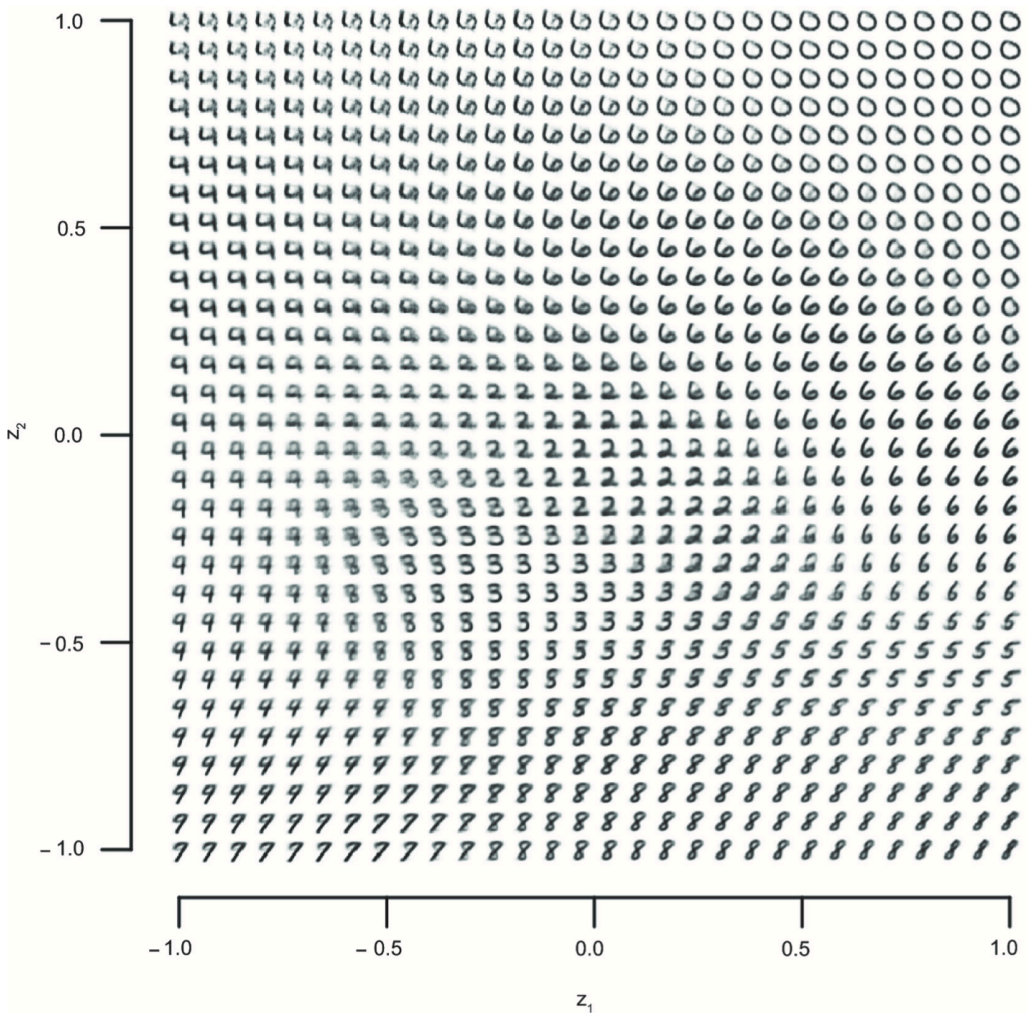


Figure 12.18 Grid of digits decoded from the latent space

Deep Learning with R

VAE as a generative model

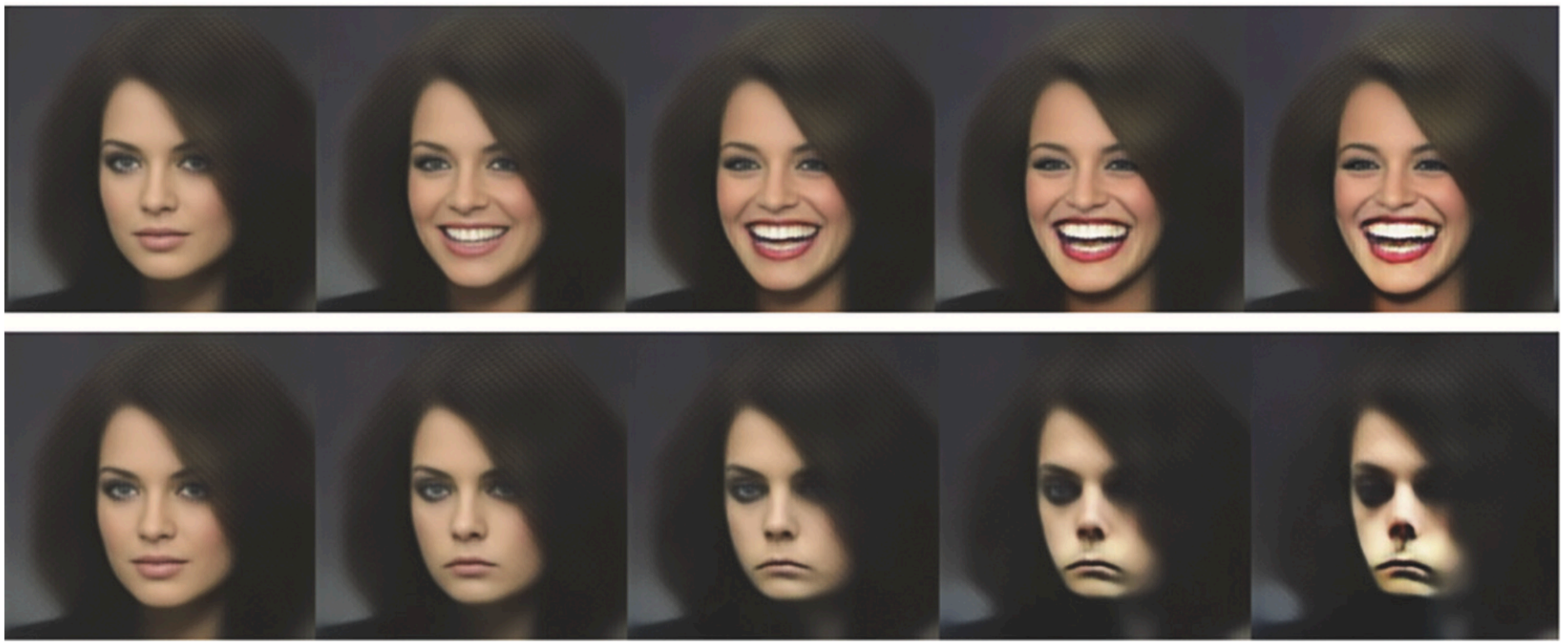
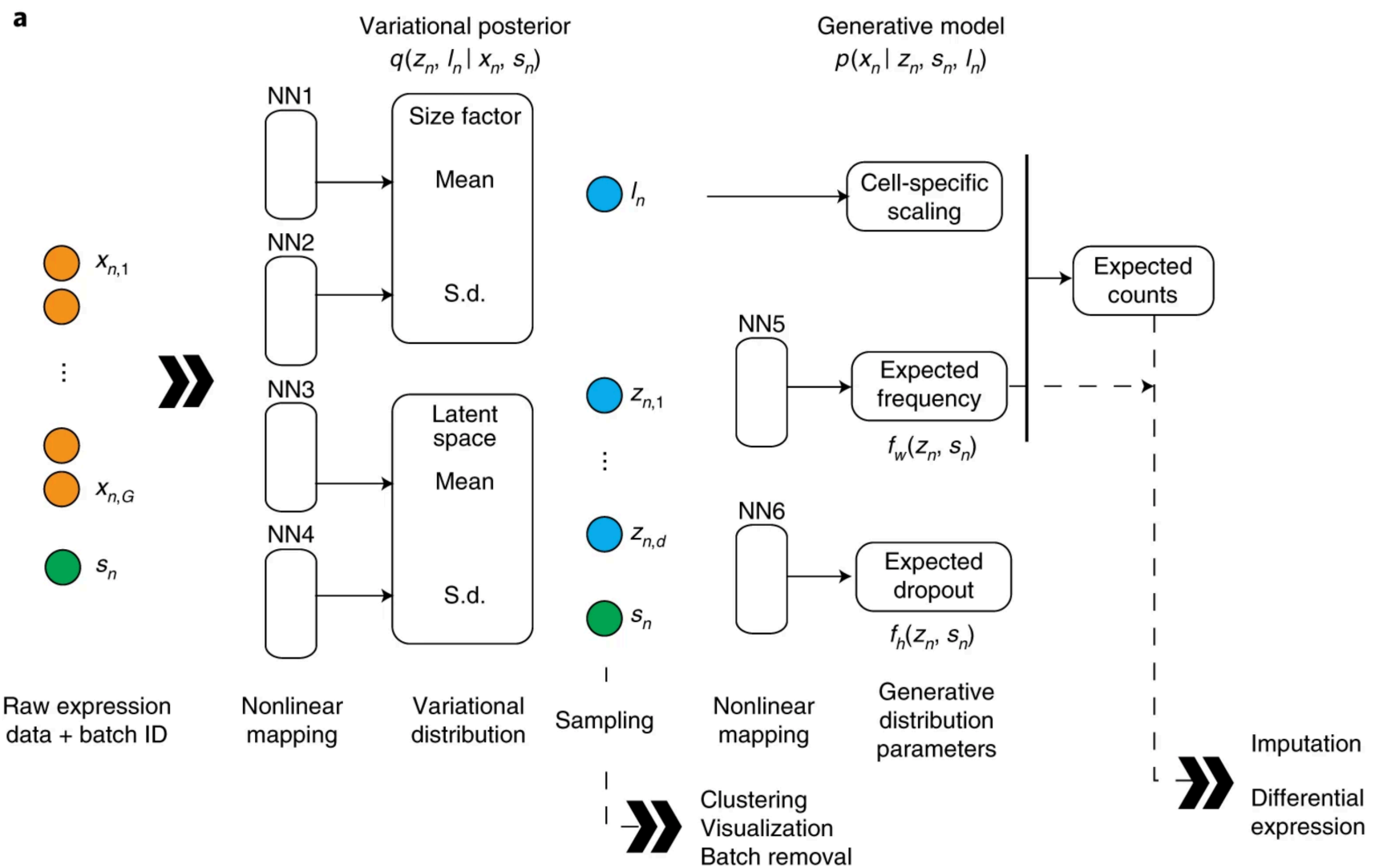


Figure 12.15 The smile vector

Deep Learning with R

VAE in single-cell omics



Lopez et al. (2018). scvi-tools.org

Foundation Models

Foundation models

A Foundation Model (FM) is "**any model** that is **trained on broad data** (generally using self-supervision at scale) that can be **adapted** (e.g., fine-tuned) **to a wide range of downstream tasks**."

(Stanford Institute for Human-Centered Artificial Intelligence)

FMs are typically trained on vast datasets and have a large complexity (billions of parameters), requiring multiple GPUs and long and costly training.

Large Language Models

The concept of FMs is that they provide pre-trained “whole-purpose” AI models that can be *fine tuned* for specific tasks.

Large Language Models (LLMs) are the most well-known example, but FMs are getting traction in other domains, including images (next lecture!) and omics data (with mixed success).

Artificial General Intelligence?

Autocomplete in overdrive

Temporary Chat

Memory is disabled for this chat, and it won't appear in your history.

OpenAI doesn't use OpenAI workspace data to train its models. For safety, we may keep a copy of this chat for up to 30 days.

Transformers

In 2017, Vaswani et al.'s seminal paper Attention is all you need showed that a sequence of *attention layers* could be used to build powerful language models.

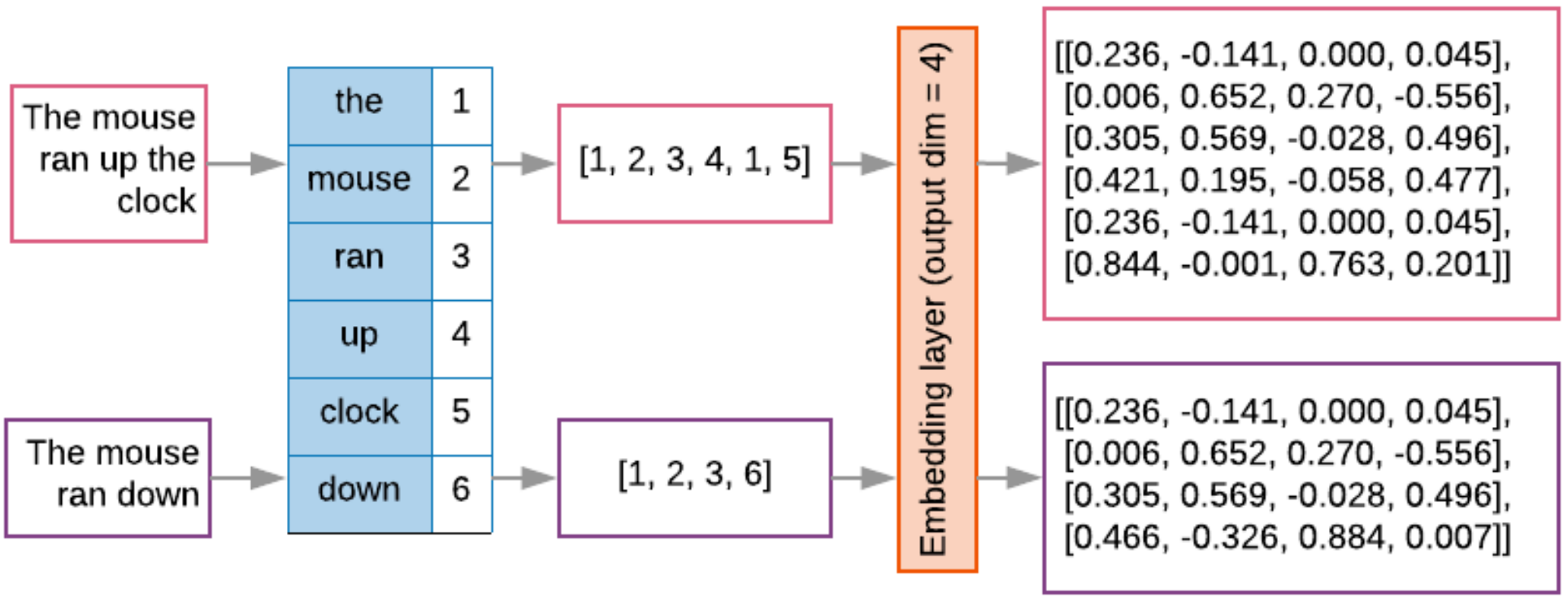
Their proposed **Transformer** architecture has revoluzionized deep learning for text and beyond.

Embeddings

Because deep learning works with tensors, we first need to turn words (or better *tokens*) into numerical vectors.

This process is far from straightforward and involves standardization, *tokenization*, and indexing.

Embedding



https://developers.google.com/machine-learning/guides/text-classification

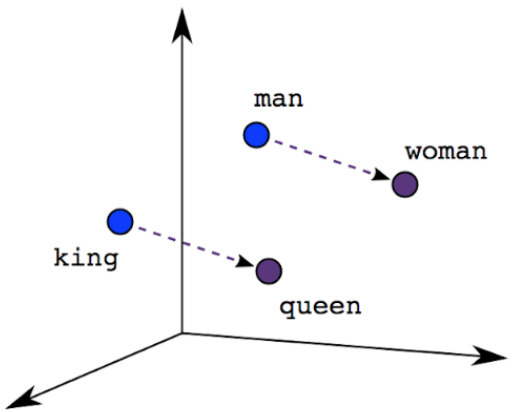
Embedding

An embedding is a dense vector space, learned from a large corpus of training data.

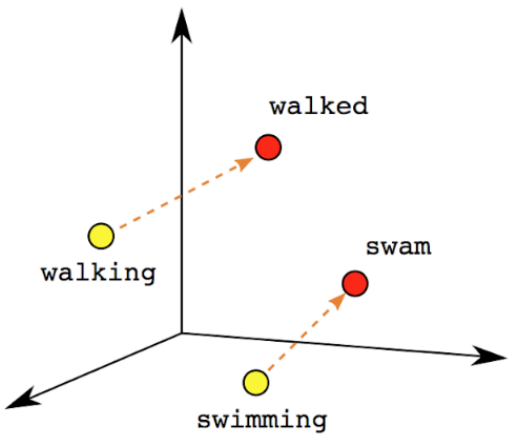
The *geometric relationship* between two word vectors in the embedding reflects the *semantic relationship* between the two words.

As such, we can define spatial relationships between tokens.

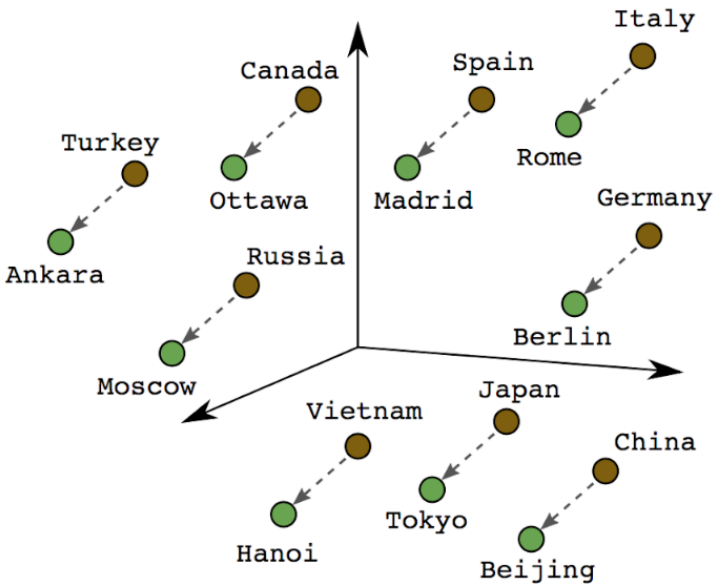
Embedding



Male-Female



Verb Tense



Country-Capital

https://developers.google.com/machine-learning/guides/text-classification

Self-attention

The simple, yet revolutionary idea in the Transformer architecture is to use a *self-attention* mechanism to provide the word embedding with context.

In an embedding, each word has a fixed position. However, in natural language the meaning of each word depends on the context.

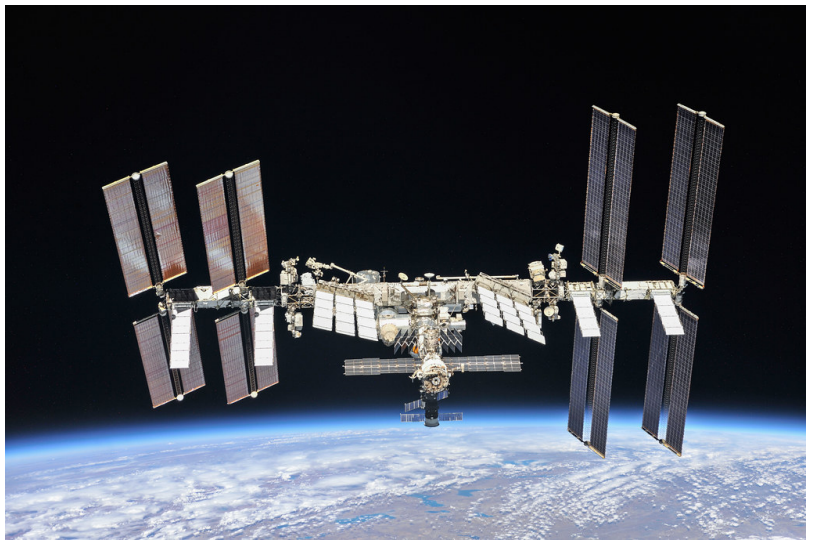
Self-attention

The role of self-attention is to modify the representation of a word based on its relations with other words in a sequence to yield **contex-aware token representations**.

Example: “The train left the station on time.”

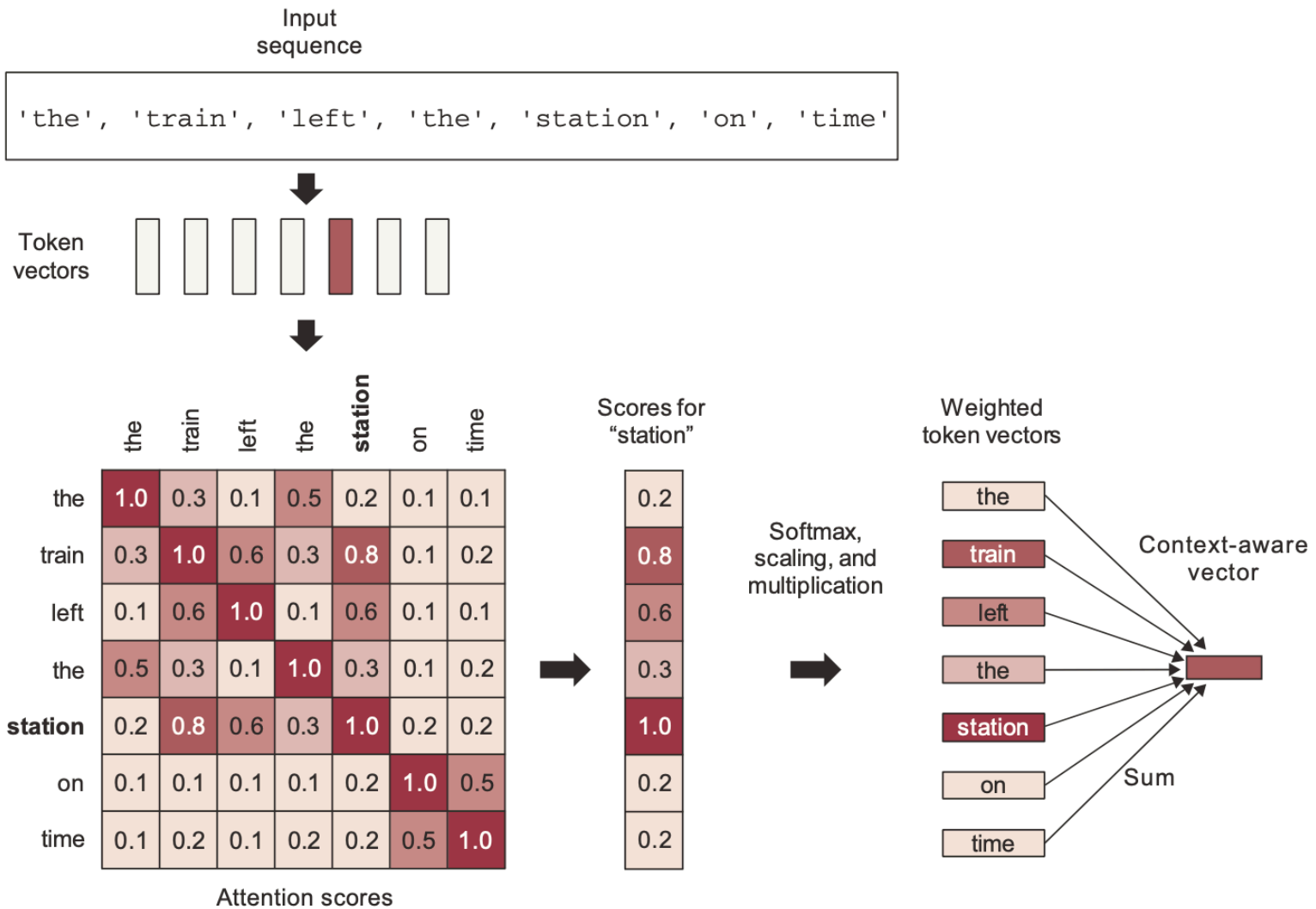
What kind of station are we talking about?

“Station”




CITY OF ORLANDO FIRE STATION ONE
DISTRICT 1
ENGINE 1
ENGINE 101
HAZ MAT 1
RESCUE 1

Self-attention



Deep Learning with R

Self-attention

1. Compute attention scores between the words (e.g. by using the dot product of the corresponding vectors).
2. Compute the sum of all the weights for each word to create a new, “context-aware” vector.

When we want to generate text, we have to *mask* the next words to train the model to predict them.

The encoder-decoder Transformer

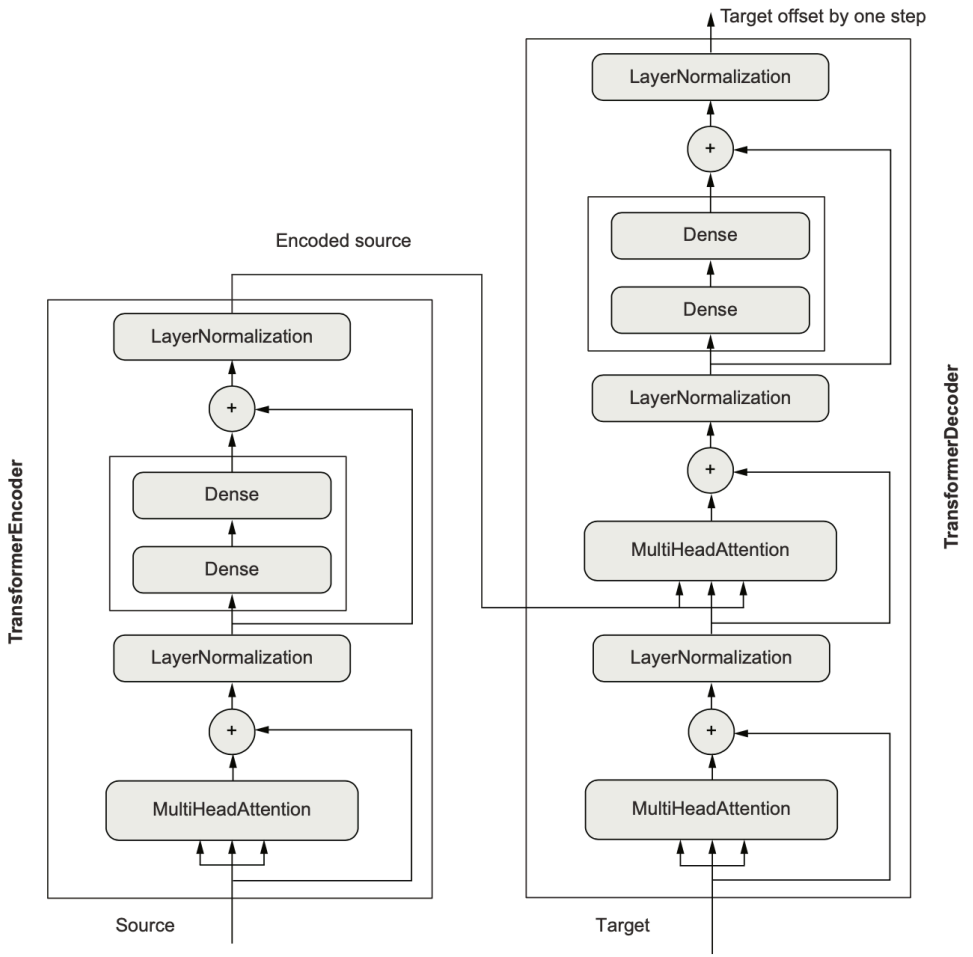


Figure 11.14 The `TransformerDecoder` is similar to the `TransformerEncoder`, except it features an additional attention block where the keys and values are the source sequence encoded by the `TransformerEncoder`. Together, the encoder and the decoder form an end-to-end Transformer.

Deep Learning with R

Text generation

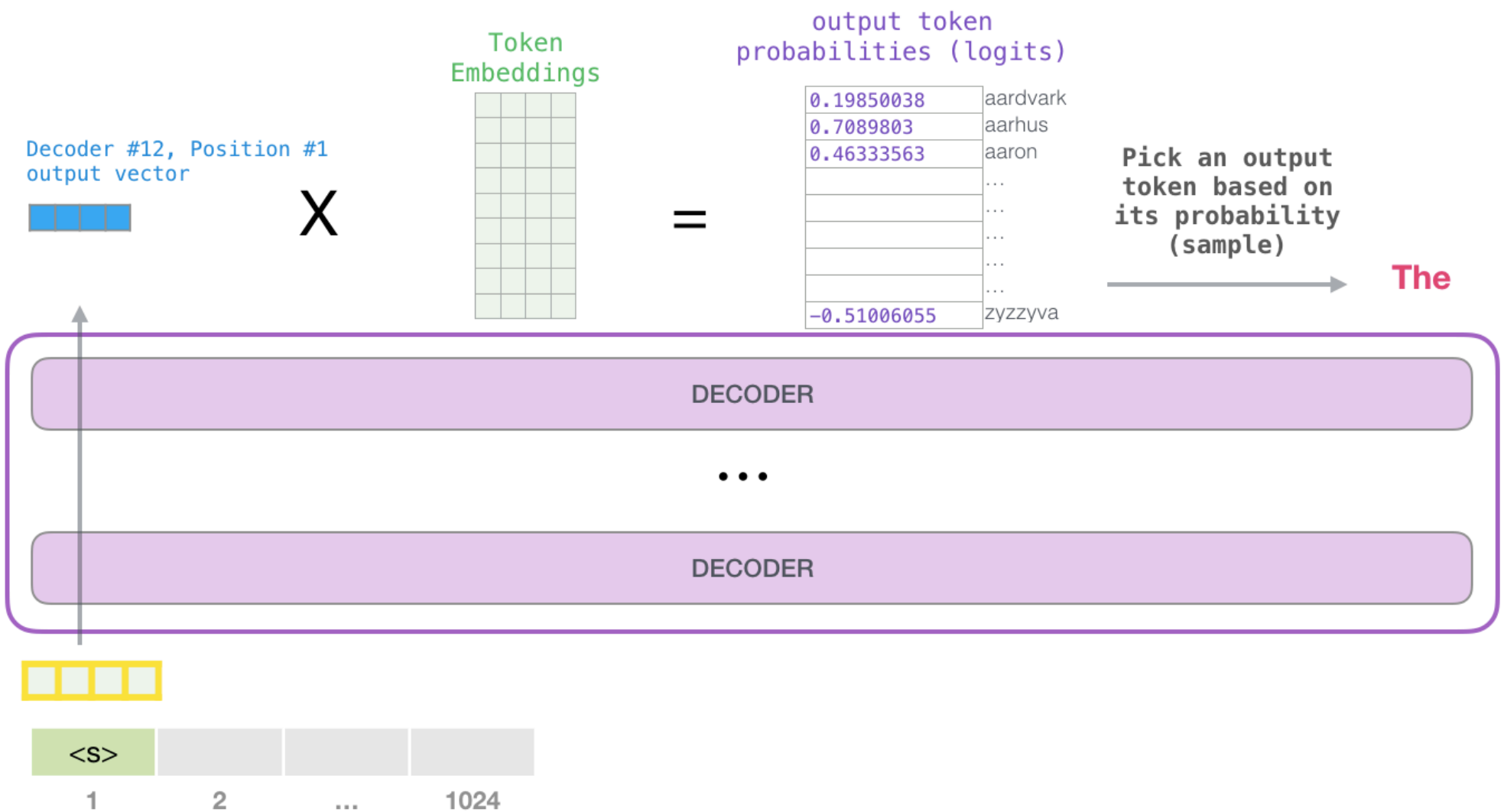
The output of the model is a **vector of probabilities** for the next token prediction.

Always choosing the highest probability would result in dry and repetitive text generation.

Hence, we use a tuning parameter called **temperature** (cf. entropy) to control the level of stochasticity that we want in the output.

This leads to better, more creative text generation.

Text generation



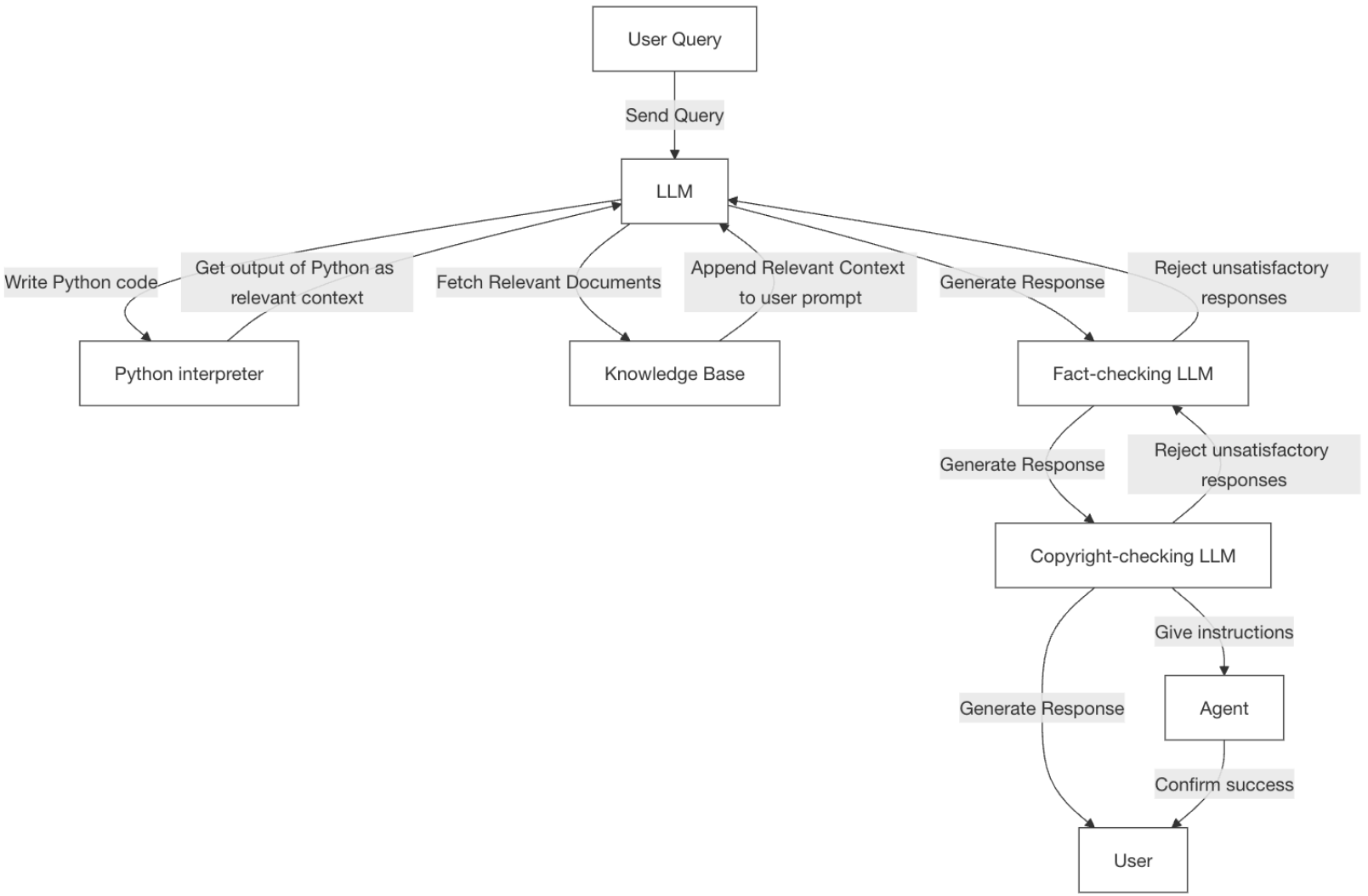
https://jalammar.github.io/illustrated-gpt2/

GPT

- **Multiple encoders and decoders**: LLMs use multiple encoder and decoder blocks: this is the *large* in LLMs.
- **Multi-head attention**: LLMs employ the self-attention mechanism independently multiple times to learn different features and then concatenate them together.
- **Positional embedding**: because the position of the words in the sentence matter, LLMs add a positional embedding too.

GPT-3 uses a stack of 96 decoder blocks, each with 1.8 billion parameters!

ChatGPT is not an LLM!



https://somerandomnerd.net/blog/just-an-llm

Trustworthy AI

Modern day oracles

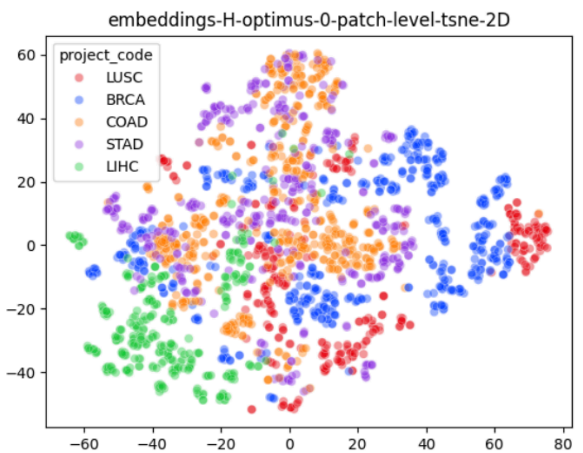
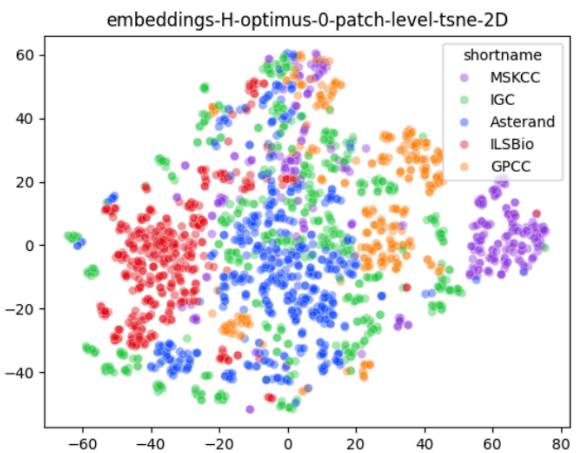
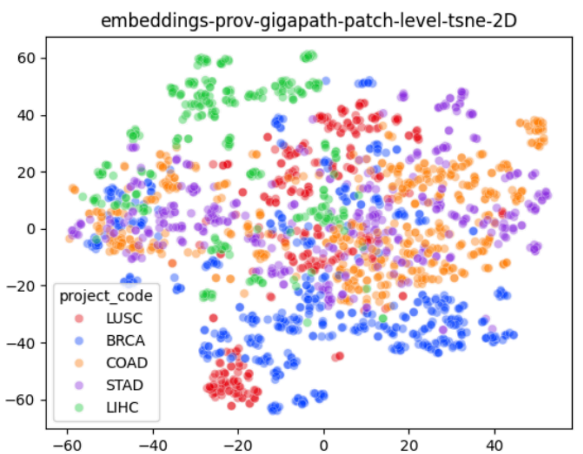
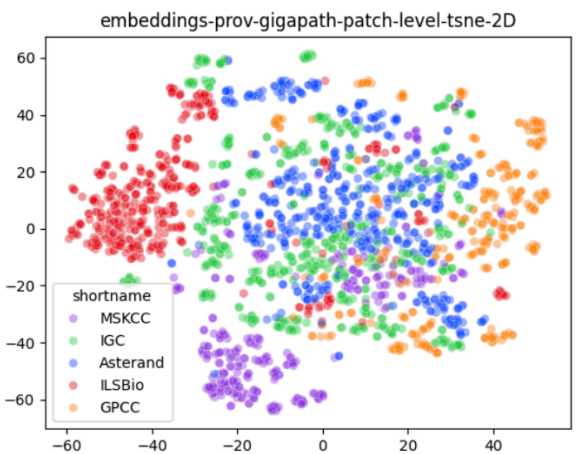
https://thebullshitmachines.com/

Algorithmic bias

generate an image of a surgeon and a nurse in an operation room

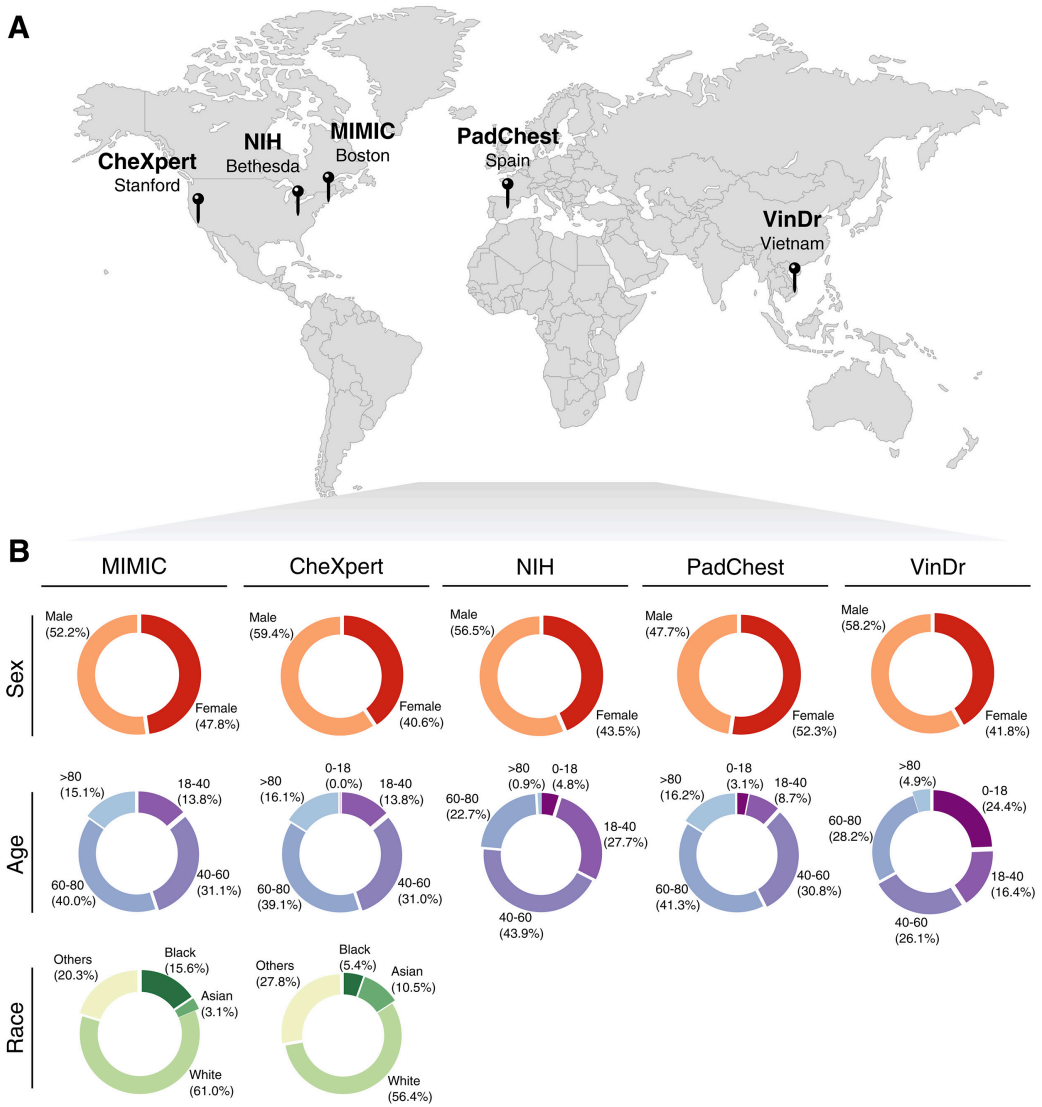
ChatGPT (GPT 5.5 *Thinking* - May 2026)

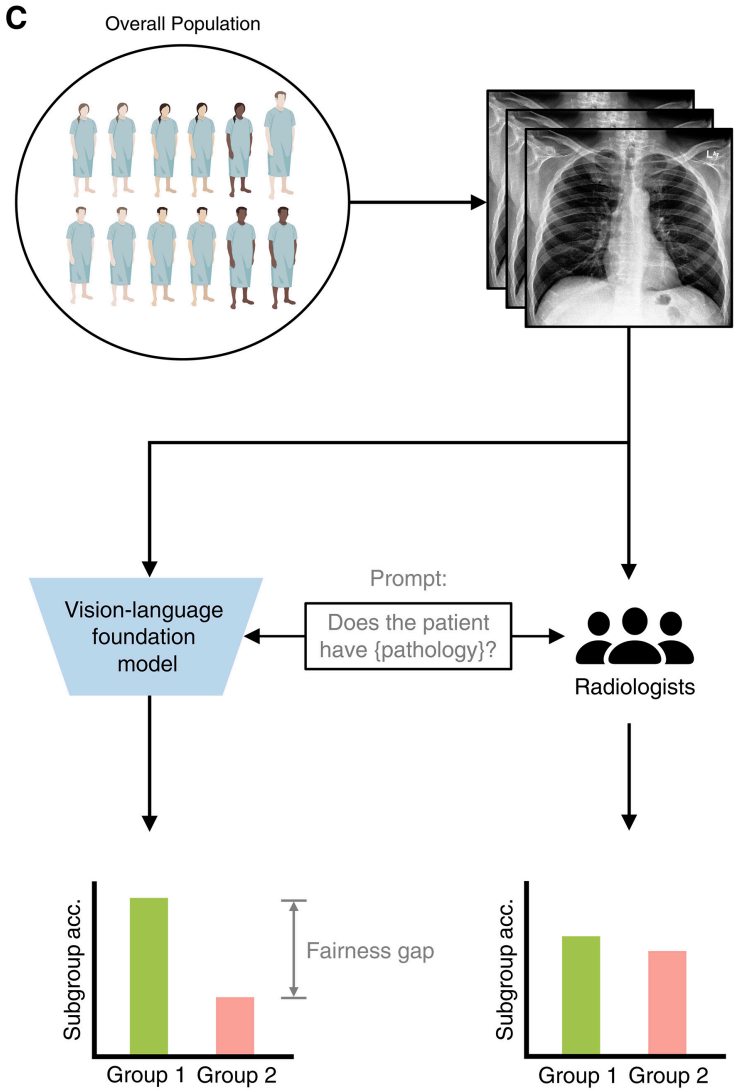
Algorithmic bias

H-optimus-0		
prov-gigapath		

de Jong et al. (2025)

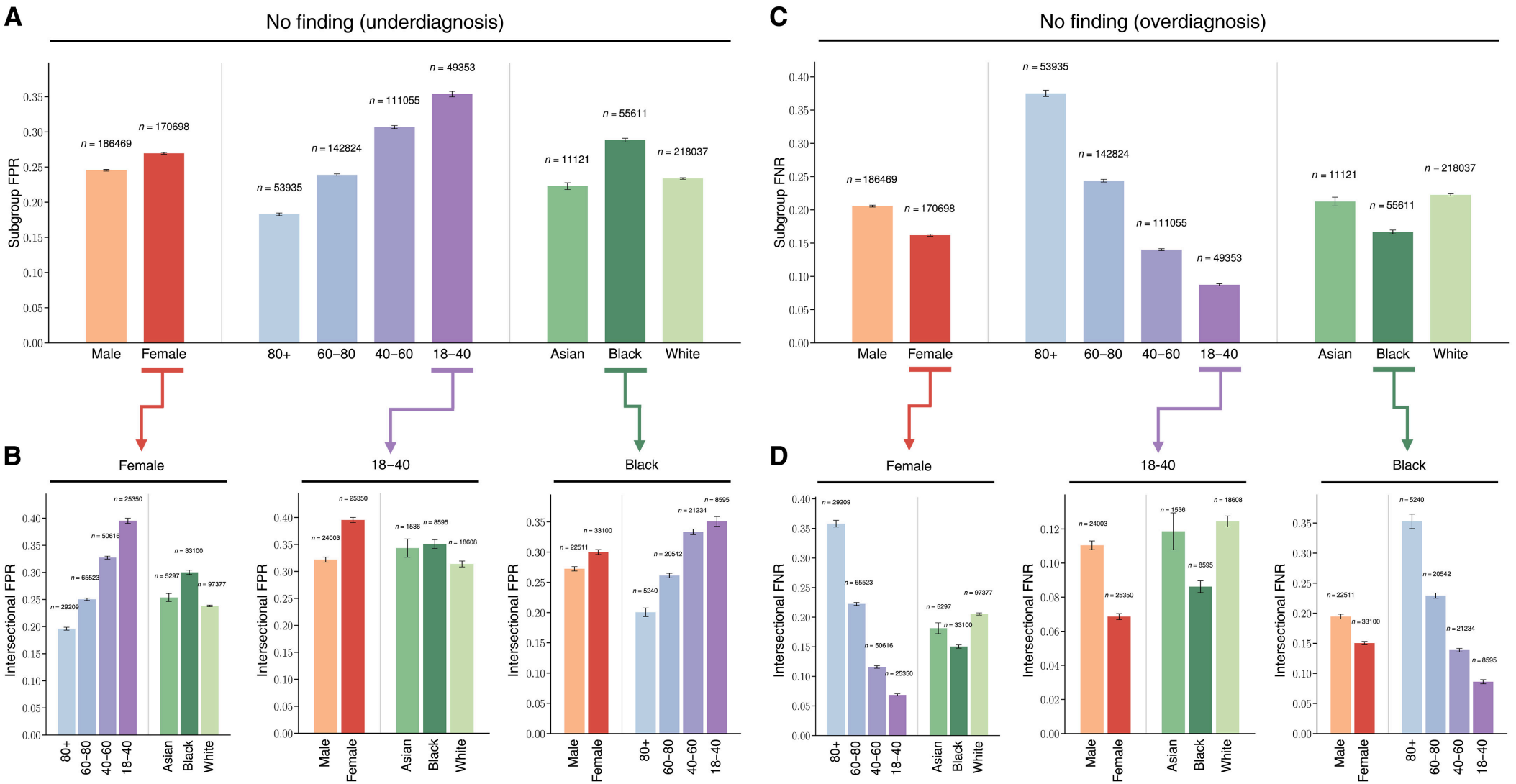
Why does it matter?





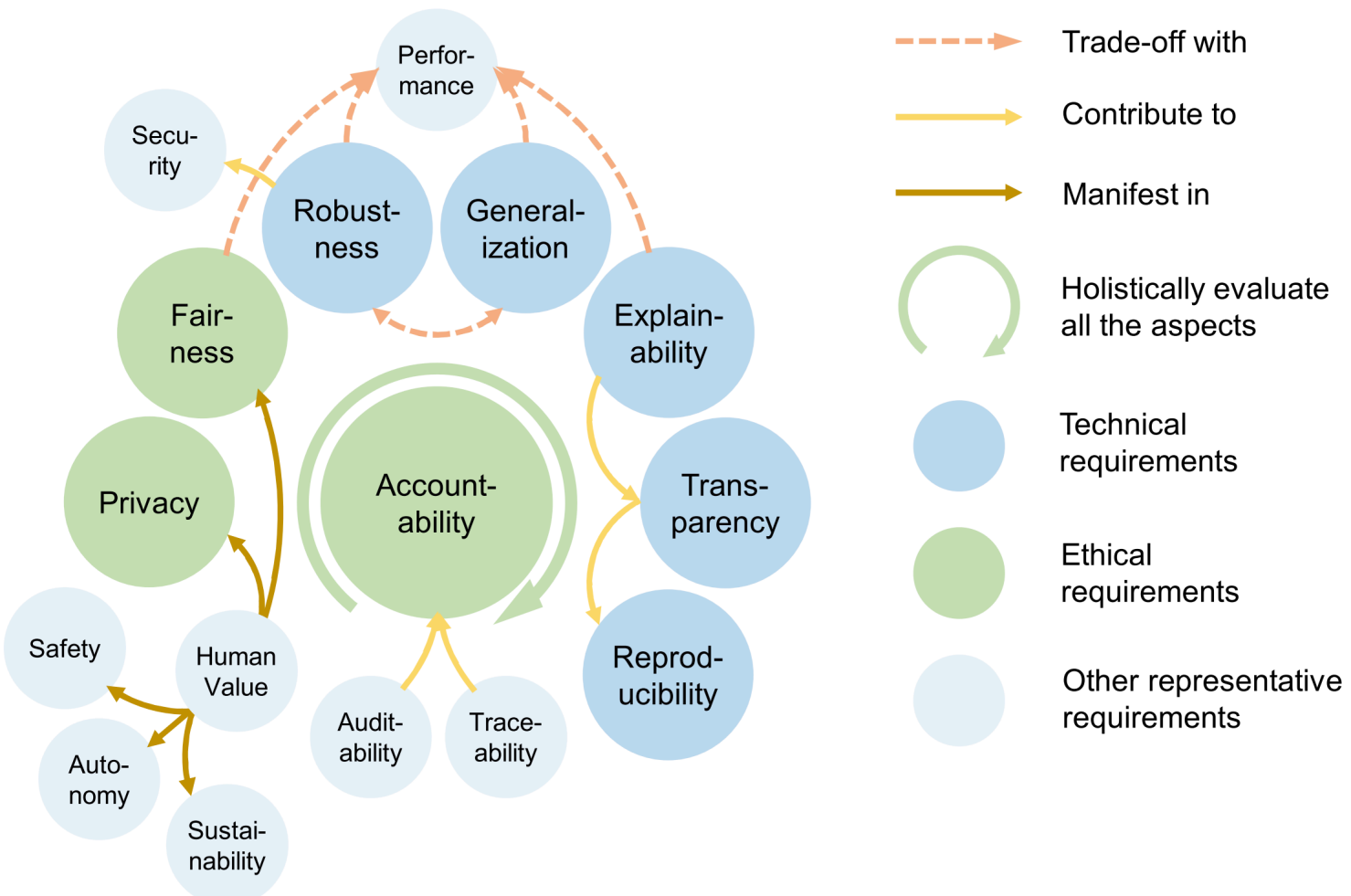
Yang et al. (2025)

Why does it matter?



Yang et al. (2025)

Trustworthy AI



Li et al. (2023)

Trustworthy AI

- **Robustness**: ability to deal with execution errors, erroneous inputs, or unseen data.
- **Generalization**: capability to distill knowledge from limited training data to make accurate predictions regarding unseen data.
- **Explainability**: understand how an AI model makes decision.
- **Transparency**: disclose information used by the AI system.
- **Reproducibility**: replication of results in controlled experiment, versioning.

Trustworthy AI

- **Fairness**: underprivileged groups might experience systematic disadvantage, need to mitigate bias.
- **Privacy protection**: protecting against unauthorized use of the data that can directly or indirectly identify a person.
- **Accountability**: who is accountable for the decisions made with (by?) the AI?
- **Intellectual property**: safeguarding creativity and the human contribution to the training data.

Thank you for your attention!

xkcd.com/329/