Machine Learning for Electron Microscopy

Juan Rojo
VU Amsterdam & Theory group, Nikhef
juanrojo.com

Electron Microscopy Characterisation of the Nanoscale
Applied Physics MSc program, TU Delft

24/06/2022

Lecture outline

Part I: A Crash Course on Machine Learning

- Basic concepts and terminology
- Supervised Learning and regression
- Artificial Neural Networks and their training
- Convolutional Neural Networks
- Reinforcement and Adversarial Learning

Based on ***Machine Learning for Physics and Astronomy***, UvA/VU BSc *natuur- en sterrenkunde:*
https://github.com/LHCfitNikhef/ML4PA

Part II: Machine Learning for Electron Microscopy

- Background subtraction in electron energy loss spectroscopy
- Automated identification of structural defects
- Improving data acquisition performance in STEM
- Predicting image features from partial inputs

Part I

A crash course on ML

Why this course?

Machine Learning rightly deserves to be part of the **toolbox of a modern scientist**

- Essential for building **modern models and algorithms** in various areas of physics and astronomy

- Very **fast developments** both in algorithms and in computing platforms have significantly extended the breadth of problems that can be tackled with ML

- Deep **physical connections** with many problems in physics and astronomy, *e.g.* quantum computation, condensed matter systems, conservation laws, ….

- Applied to problems even in very **formal fields,** *e.g.* string theory

- Large interested in the community, **societal implications** (AI hype)

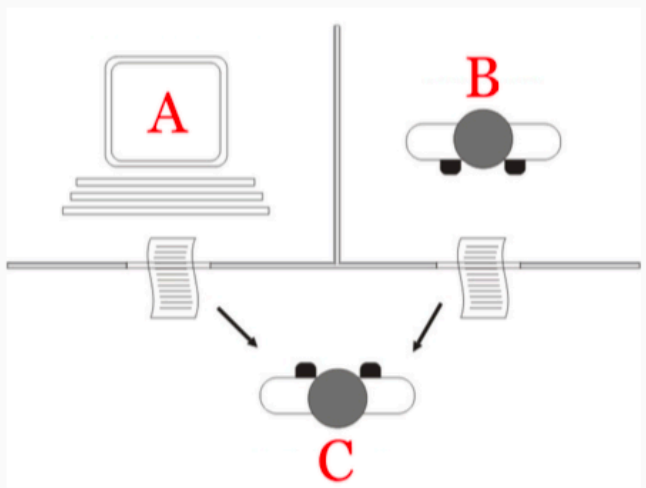
Furthermore, **expertise in ML/AI** is powerful asset for also for careers outside academia

Problems in AI

Most problems tackled with Artificial Intelligence fall in **two categories**

(1) abstract and formal: *easy for computers* but *difficult for humans*

knowledge-based approach



e.g. chess (DeepBlue)

(chess is deterministic game with finite number of options)

(2) intuitive, hard to formalize: *easy for humans* but *difficult for machines*

concept capture and generalisation

e.g. pattern recognition

Problems in AI

Most problems tackled with Artificial Intelligence fall in **two categories**

To excel in tasks which are intuitive to humans but difficult to machines, an A.I. system needs to **acquire its own knowledge**: the Machine is Learning

unlike in chess, where there is no new knowledge to acquire once rules are spelled out

Machine Learning algorithms allow computers the ability to **carry out a task without being explicitly programmed how to do it** by **learning from examples**

(2) intuitive, hard to formalize: *easy for humans* but *difficult for machines*

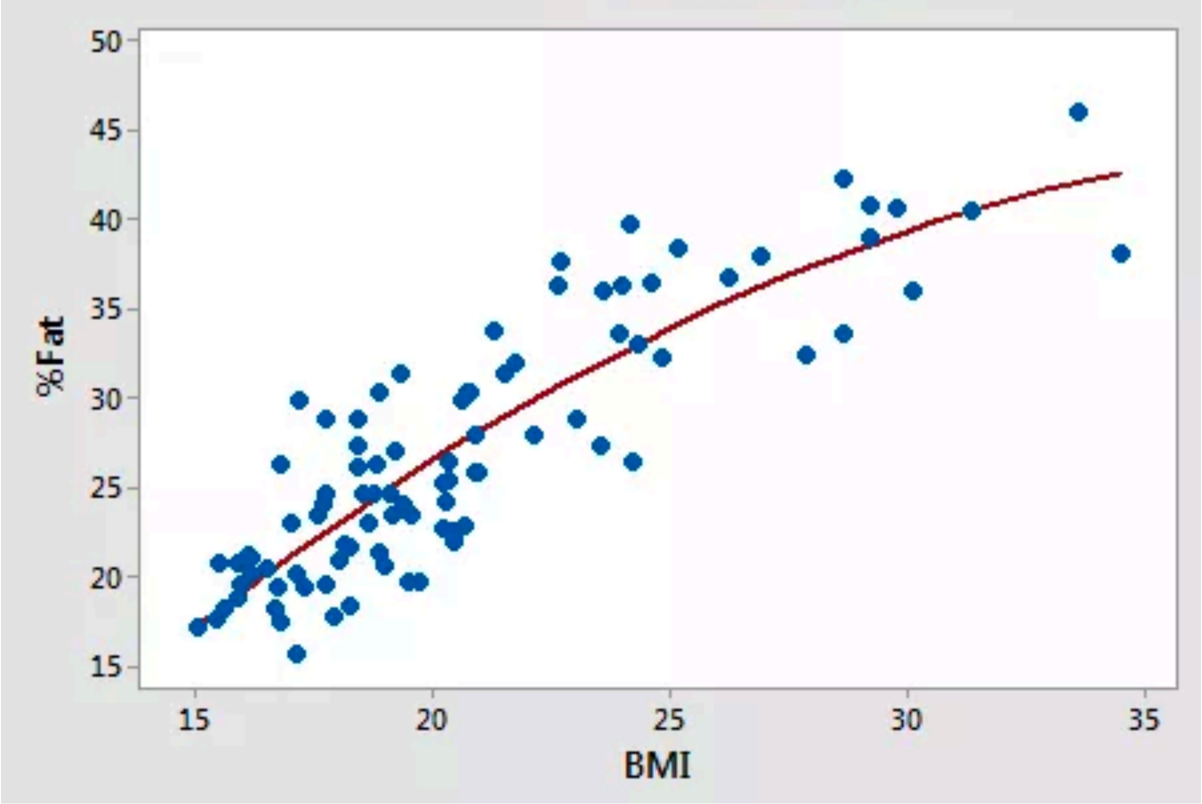
concept capture and generalisation

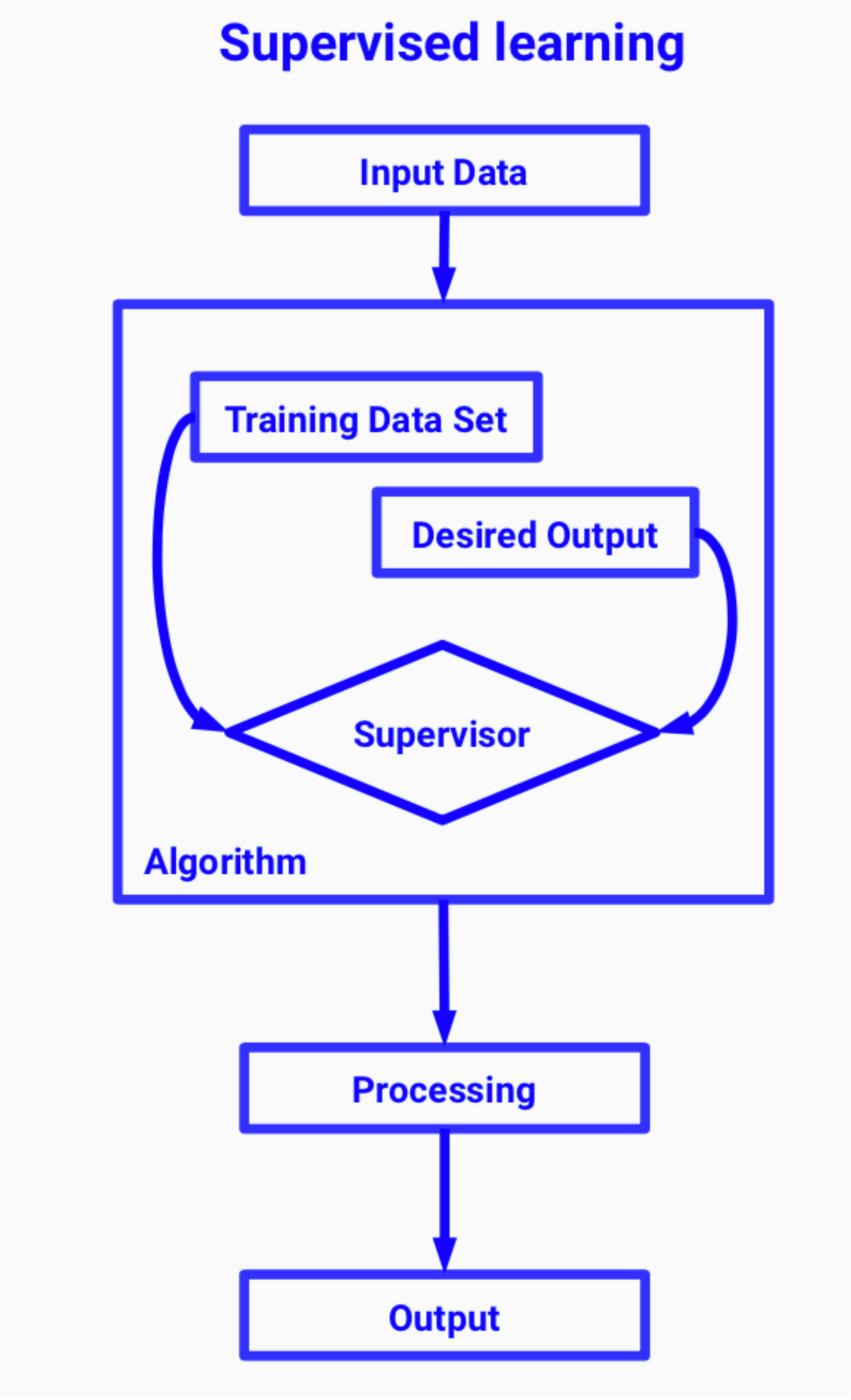
e.g. pattern recognition

Learning to learn

Machine Learning algorithms can be divided into several classes, including

- **Supervised Learning**: regression, classification, …



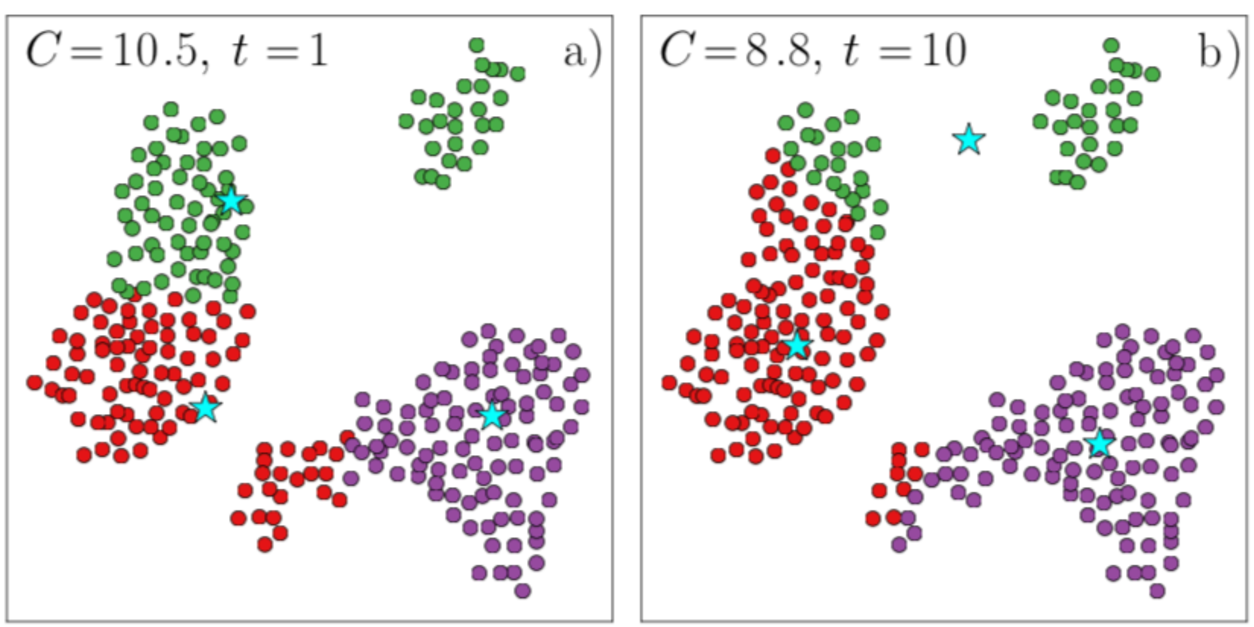


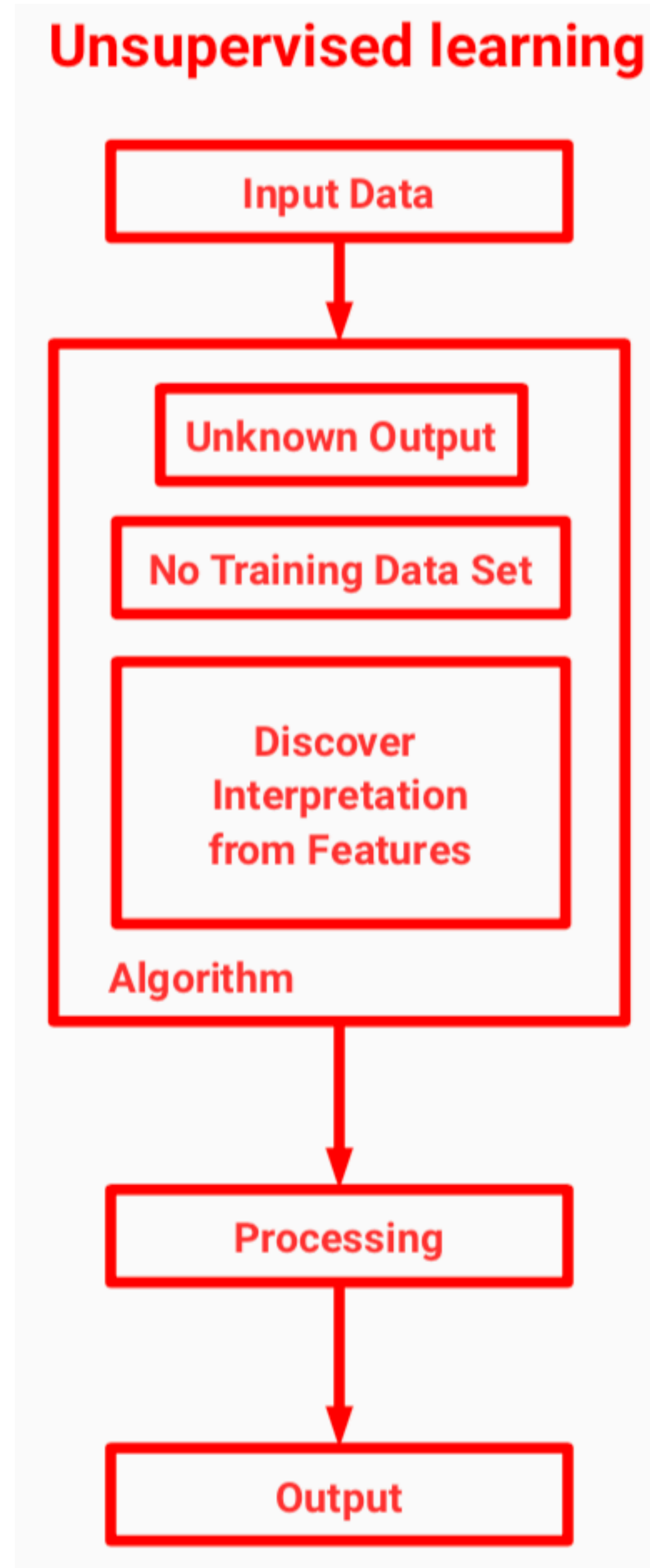
Learning to learn

Machine Learning algorithms can be divided into several classes, including

- **Supervised Learning**:
  regression, classification, …

- **Unsupervised Learning**:
  clustering, data dimensional reduction, ….





Learning to learn

Machine Learning algorithms can be divided into several classes, including

- **Reinforcement learning**: efficiently react to changing environment

Supervised Learning:

and Regression

Supervised learning

We denote as **supervised learning** the ML task of **learning a function** that maps a **vector of inputs** to a **vector of outputs** from a finite set of training example

note that some assumptions will be needed: a function is an **infinite-dimensional object** but learning takes place from a **finite number of examples**

main property of supervised learning: the **training samples are labeled**

$$\boldsymbol{x}_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,p}\right) \quad \rightarrow \quad y_i$$

data point (with p features) ***label***

the label can be **discrete** (signal/noise, cat/dog) or **continuous** (output of function)

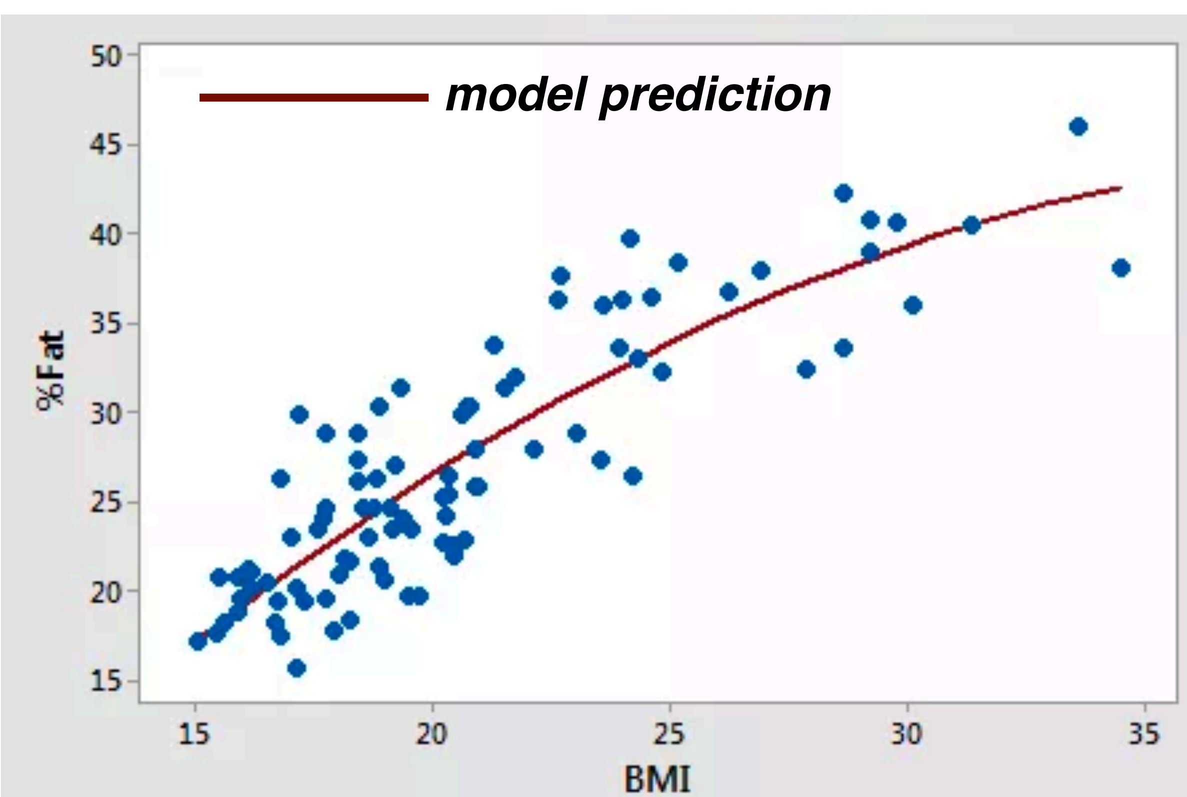
Supervised learning

We denote as **supervised learning** the ML task of **learning a function** that maps a **vector of inputs** to a **vector of outputs** from a finite set of training example

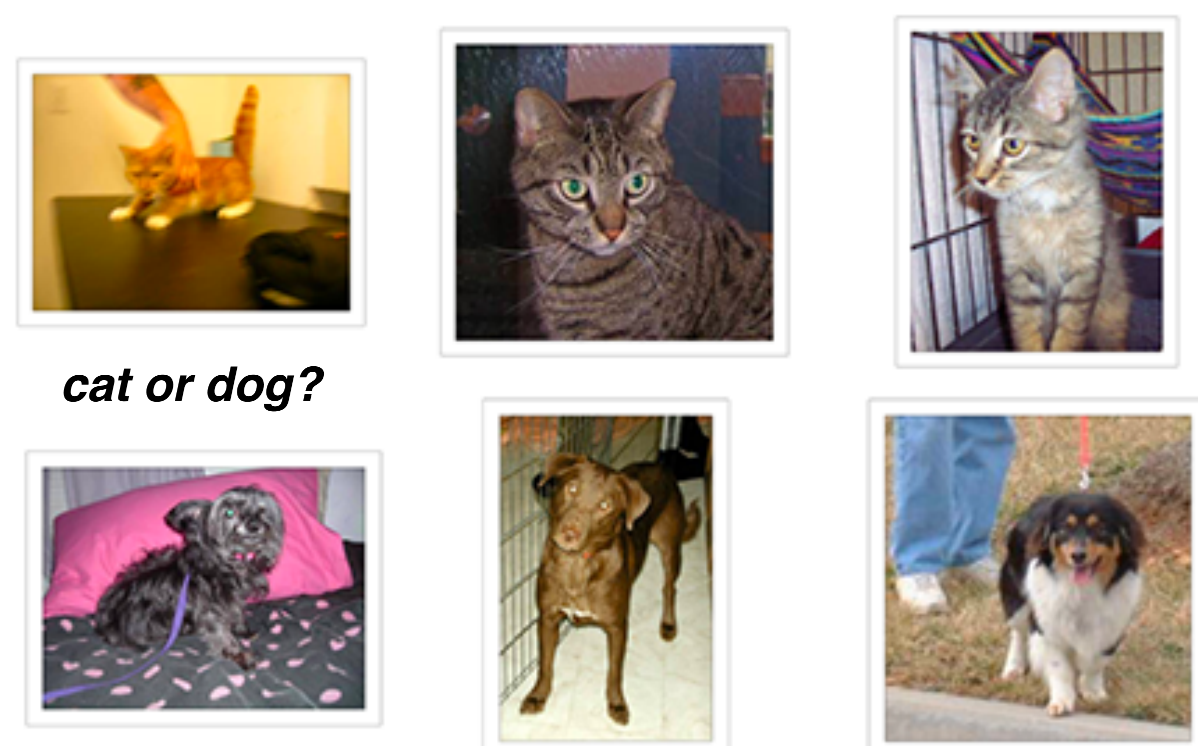
note that some assumptions will be needed: a function is an **infinite-dimensional object** but learning takes place from a **finite number of examples**

main property of supervised learning: the **training samples are labeled**

continuous outputs: regression



discrete outputs: classification



Setting up the problem

problems in **Supervised Machine Learning** are defined by the following ingredients:

(1) Input dataset: $\mathscr{D} = (\boldsymbol{X}, \boldsymbol{Y})$

array of ***independent*** *variables*

array of ***dependent*** *variables*

$$\boldsymbol{X} = \left(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\right) \qquad \boldsymbol{Y} = \left(y_1, y_2, \ldots, y_n\right)$$

$$\boldsymbol{x}_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,p}\right)$$

each independent variable contains p features

(2) Model: $f\left(\boldsymbol{X}, \boldsymbol{\theta}\right)$

mapping between dependent and independent variables

$$f \, : \, \boldsymbol{X} \to \boldsymbol{Y}$$

model parameters

$$\boldsymbol{\theta} = \left(\theta_1, \theta_2, \ldots, \theta_m\right)$$

the more complex the problem, the more flexible the model

Setting up the problem

problems in **Supervised Machine Learning** are defined by the following ingredients:

(1) Input dataset: $\mathscr{D} = (\boldsymbol{X}, \boldsymbol{Y})$

(2) Model: $f\left(\boldsymbol{X}, \boldsymbol{\theta}\right)$

(3) Cost function: $C\left(\boldsymbol{Y}; f(\boldsymbol{X}; \boldsymbol{\theta}\right)$

The cost function measures how well the model (for a specific choice of its parameters) is able to describe the input dataset

Fitting the model means determining the values of its parameters which **minimise the cost function**

$$\left.\frac{\partial C\left(\boldsymbol{Y}; f(\boldsymbol{X}; \boldsymbol{\theta}\right)}{\partial \theta_i}\right|_{\theta=\theta_{\rm opt}} = 0$$

Best-fit model

What is the best strategy to **determine the model parameters**?

Seems a silly question, surely those are simply **minimum of cost function?**

$$\hat{\theta} = \arg\ \min_{\theta}\left\{ \sum_{i=1}^{n} \left(y_i - f_{\alpha}(x_i;\boldsymbol{\theta}_{\alpha})\right)^2 \right\}$$

However this is in general **not the case**, because:

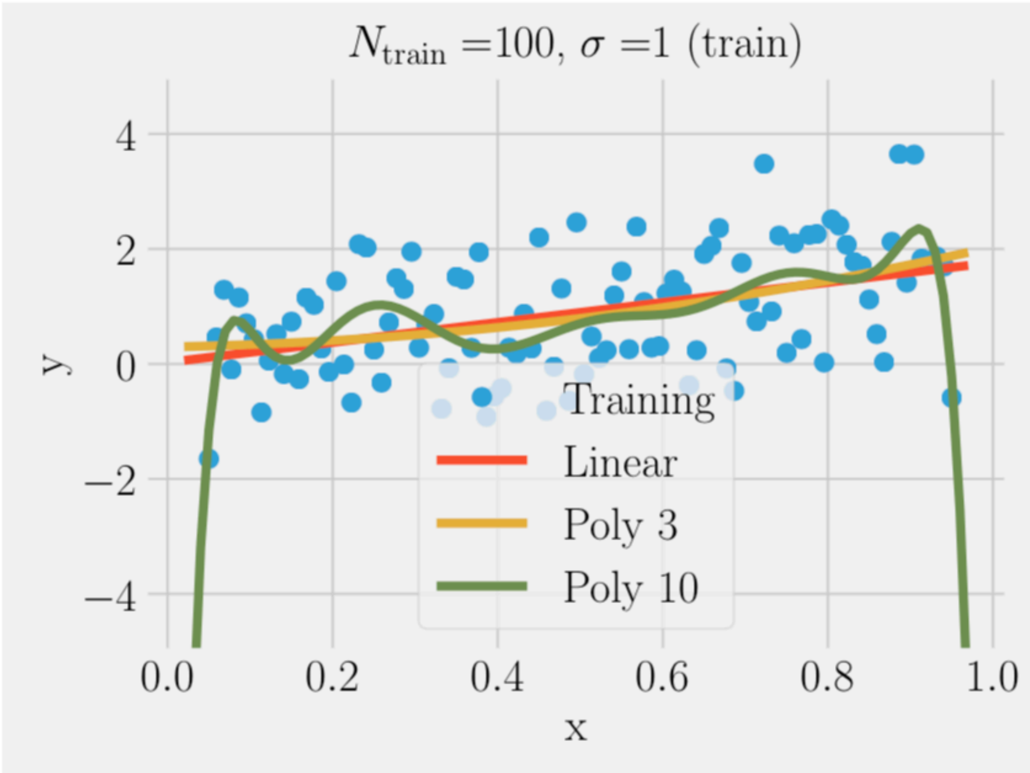
- Real world data is **noisy**: we want to **learn the underlying law**, not the statistical fluctuations

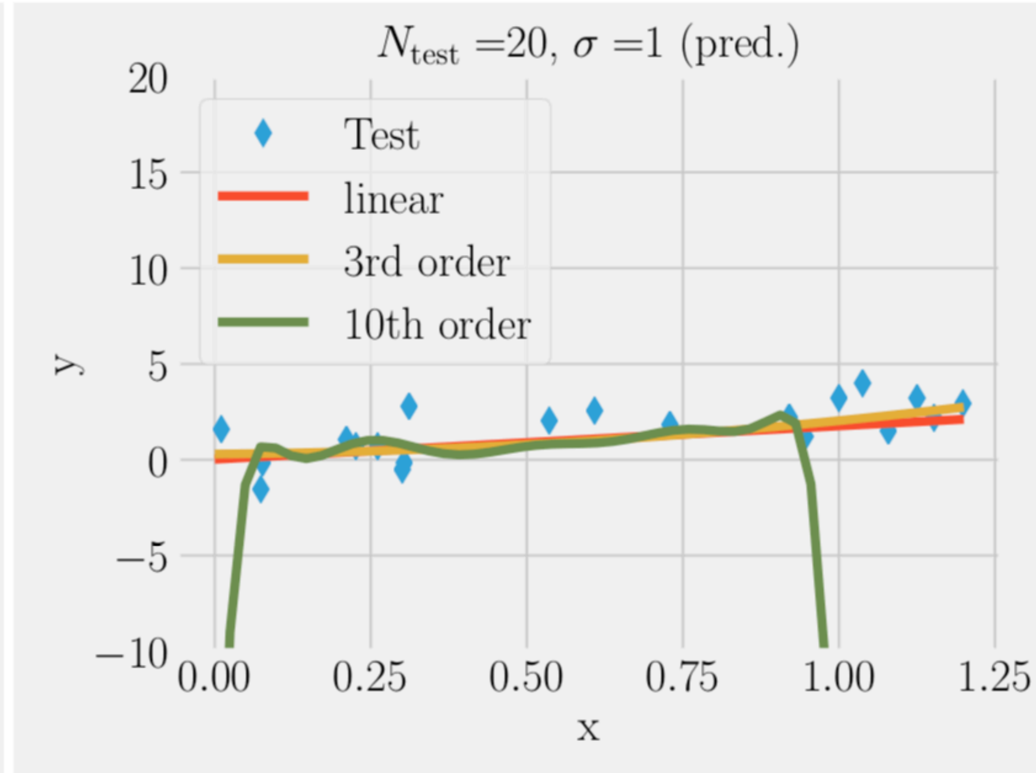
- More than fitting the data, our real goal is to create a model that **predicts future/different data:** we need figures of merit outside the training dataset!

- To ensure that our model describes the underlying law (and thus one can safely generalise) rather than the noise, a **regularisation procedure** needs to be used

Model fitting

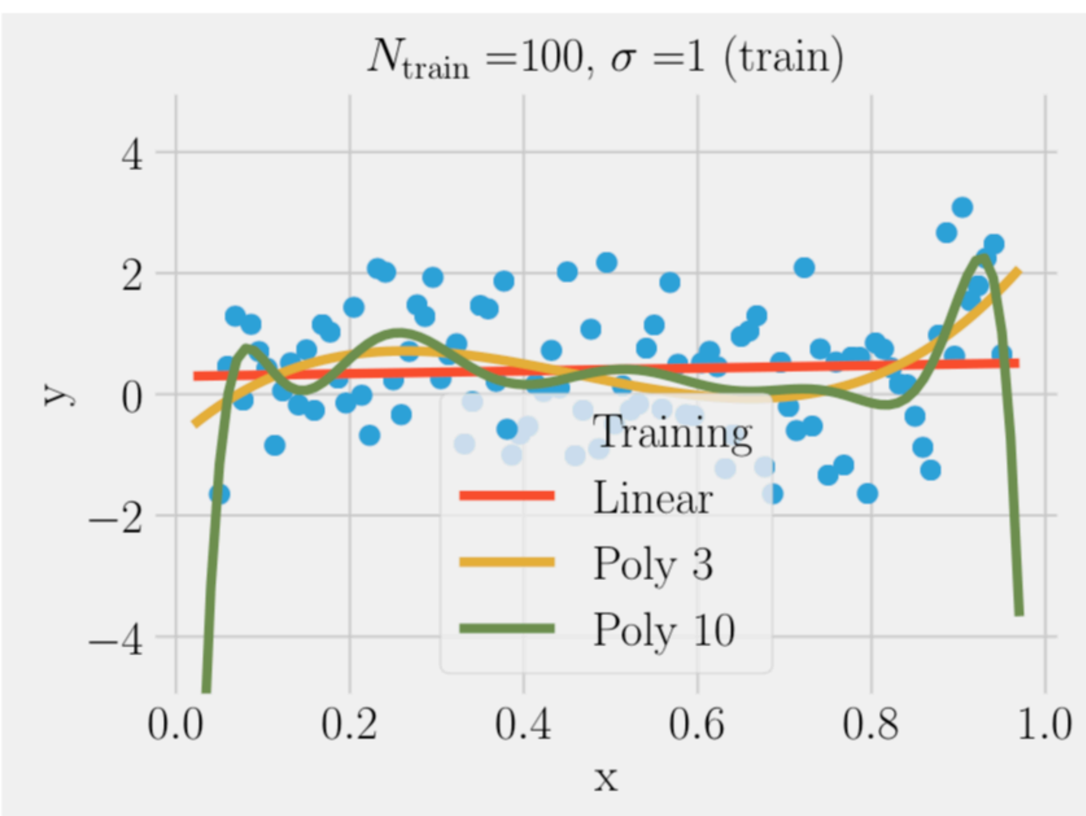
$$f(x) = 2x\,, \quad x \in [0,1]$$

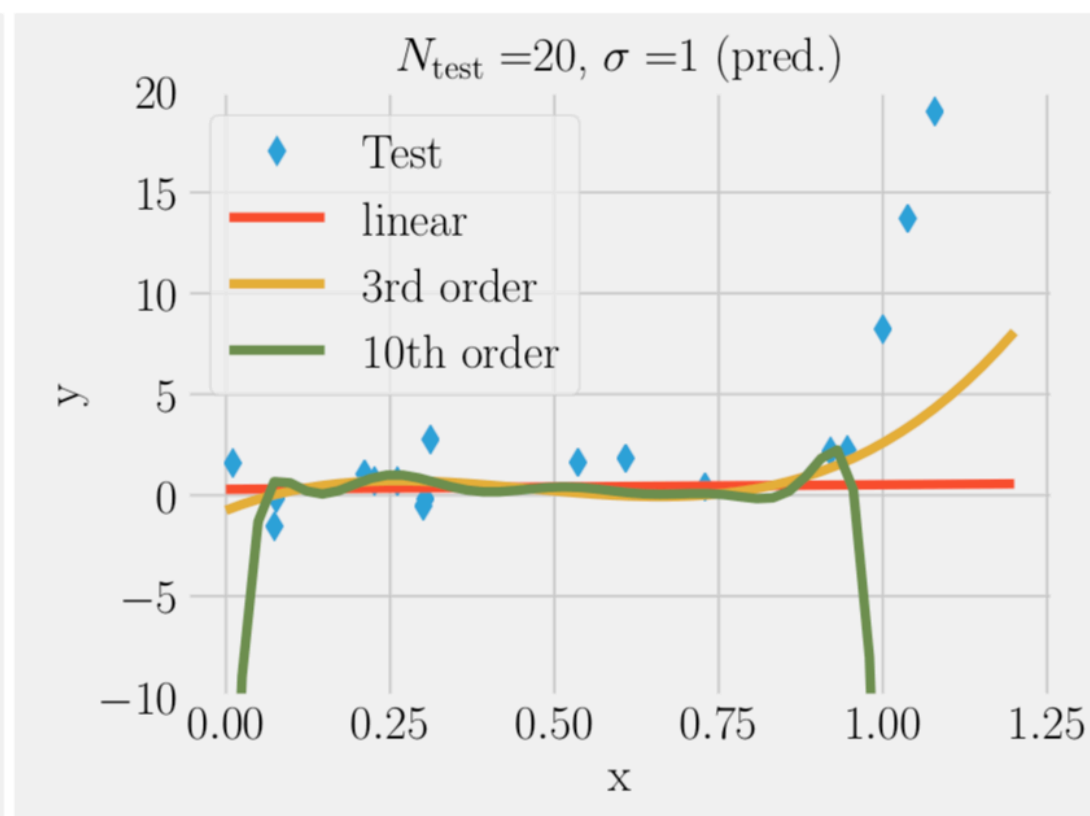




$$f(x) = 2x - 10x^5 + 15x^{10}\,, \quad x \in [0,1]$$

in the presence of noise, models with less complexity can exhibit improved predictive power





Cross-validation (regularisation)

In-sample (training) error $\longrightarrow$ $$E_{\mathrm{tr}} \equiv C\left(\boldsymbol{Y}_{\mathrm{tr}}, f(\boldsymbol{X}_{\mathrm{tr}}; \widehat{\boldsymbol{\theta}})\right)$$

Out-of-sample (validation) error $\longrightarrow$ $$E_{\mathrm{val}} \equiv C\left(\boldsymbol{Y}_{\mathrm{val}}, f(\boldsymbol{X}_{\mathrm{val}}; \widehat{\boldsymbol{\theta}})\right)$$

Splitting the data into mutually exclusive training and validation sets provides an unbiased estimate for the **predictive performance of the model**

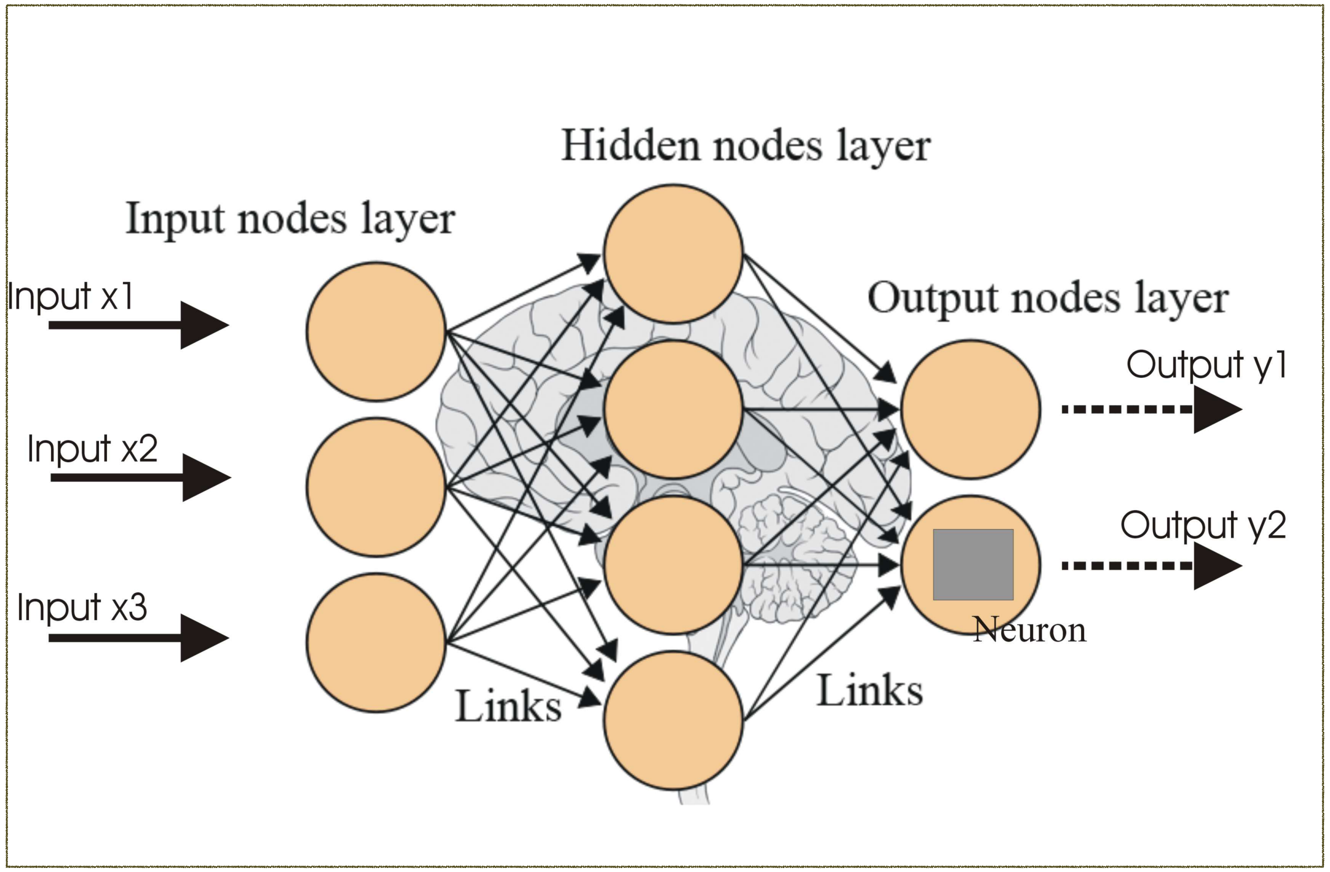
In ML problems one should select the model that **minimises** the out-of-sample error $\boldsymbol{E_{val}}$, since this is the model that generalises in the most efficient way

Fitting is not predicting: in general the model that describes better a given set of data will not be the one that generalises and predicts better related datasets

(Deep) Neural Networks

Artificial Neural Networks

Inspired by **biological brain models**, **Artificial Neural Networks** (ANNs) are mathematical algorithms designed to excel where domains as their evolution-driven counterparts outperforms traditional algorithms in tasks such as **pattern recognition, forecasting, classification**, ...



in ML context, ANN provide a flexible, powerful non-linear model for many problems

Neural Networks

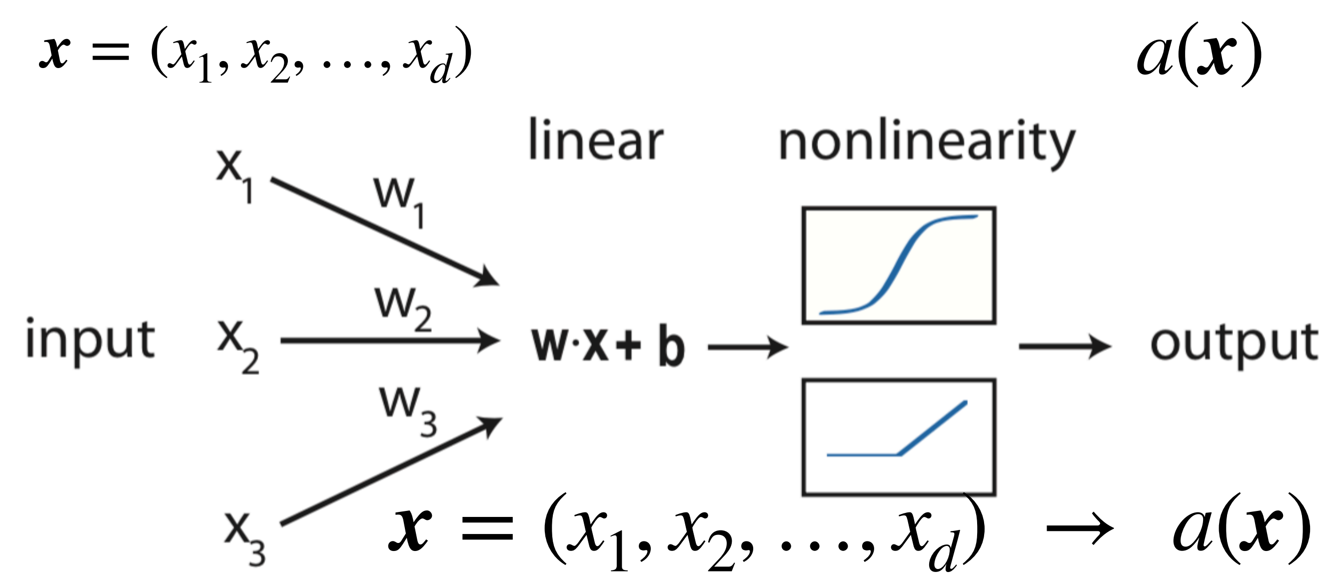
Neural Nets can be defined as **neural-inspired nonlinear models** for supervised learning

- The basic unit of a NN is the **neuron,** a transformation of a set of *d* input features into a scalar output

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \xrightarrow{\textit{mapping}} a(\boldsymbol{x})$$

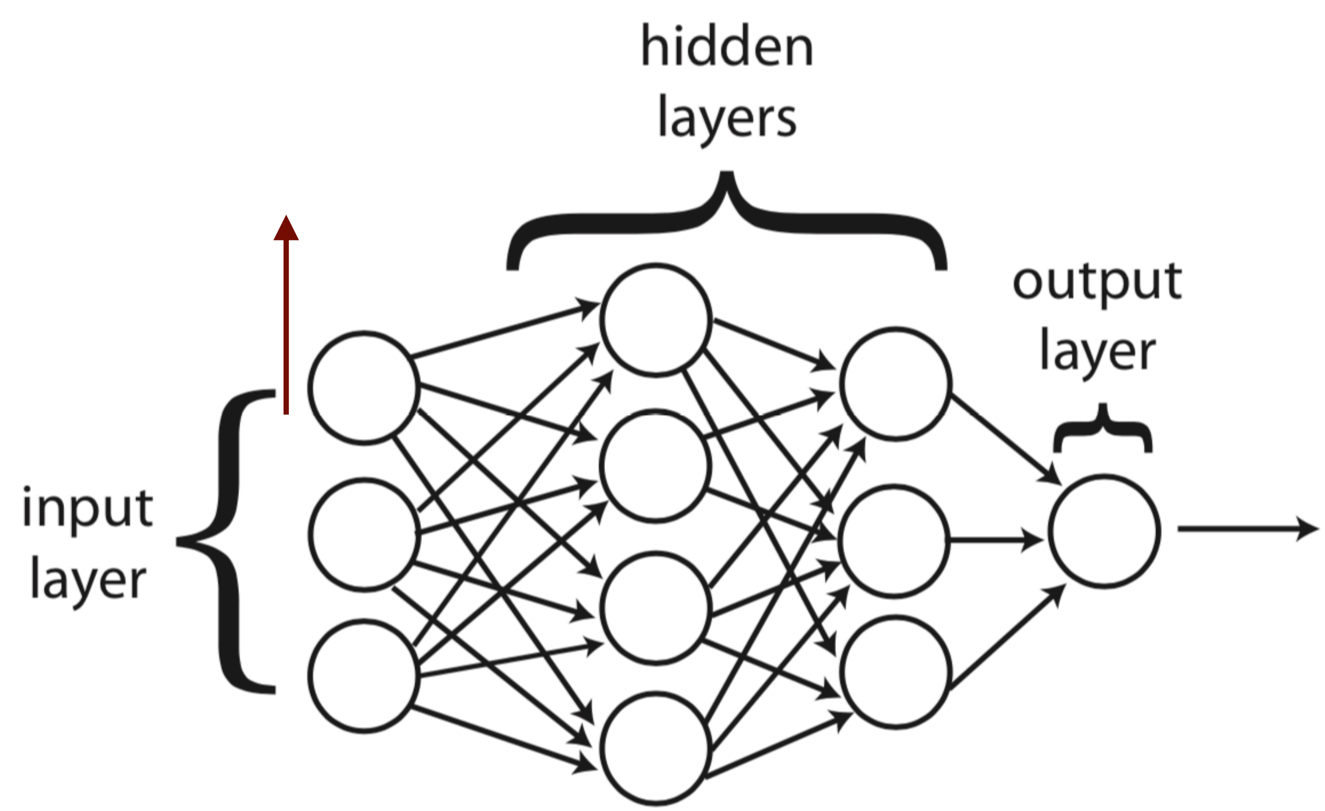
neuron inputs

neuron output (activation state)



- These neutrons are arranged in **layers**, which in turn are stacked on each other. The intermediate layers are called **hidden layers**

- Here we will focus on **feed-forward NNs,** where the output of the neurons of the previous layer becomes the input of the neurons in the subsequent layer



Neural Networks

Neural Nets can be defined as **neural-inspired nonlinear models** for supervised learning

the neural network output has two components

a **linear operation**, weighting the various inputs

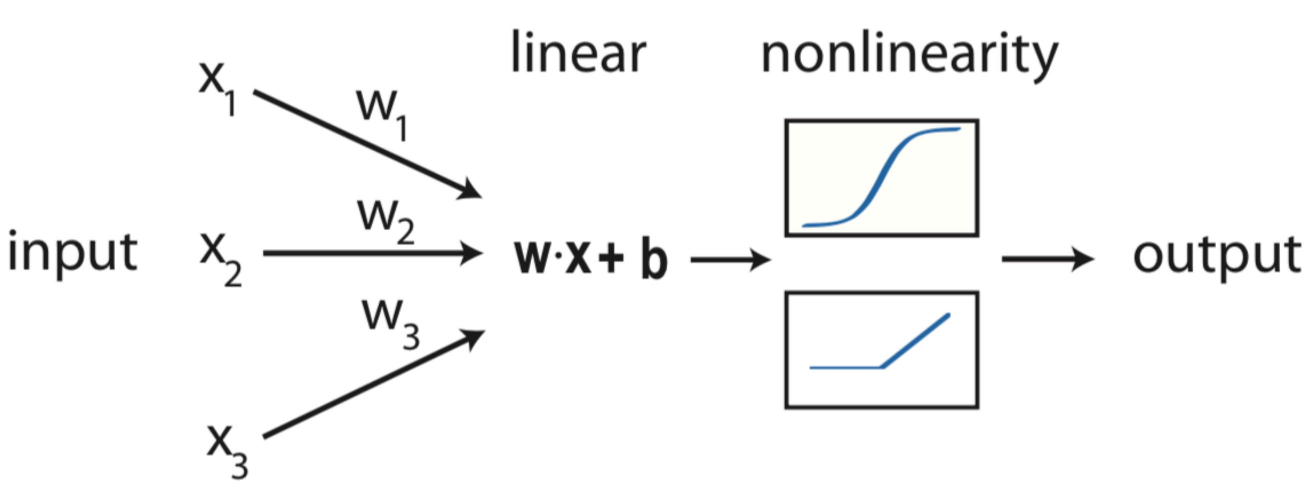
$$z^{(i)} = \boldsymbol{x}^T \cdot \boldsymbol{\theta}^{(i)} + \boldsymbol{\theta}_0^{(i)}$$

neuron inputs — *i-th neuron parameters* — *i-th neuron bias*

a **non-linear transformation**

$$a^{(i)}(\boldsymbol{x}) = \sigma_i(z^{(i)})$$

i-th neuron activation state — *i-th neuron activation function (non-linear)*



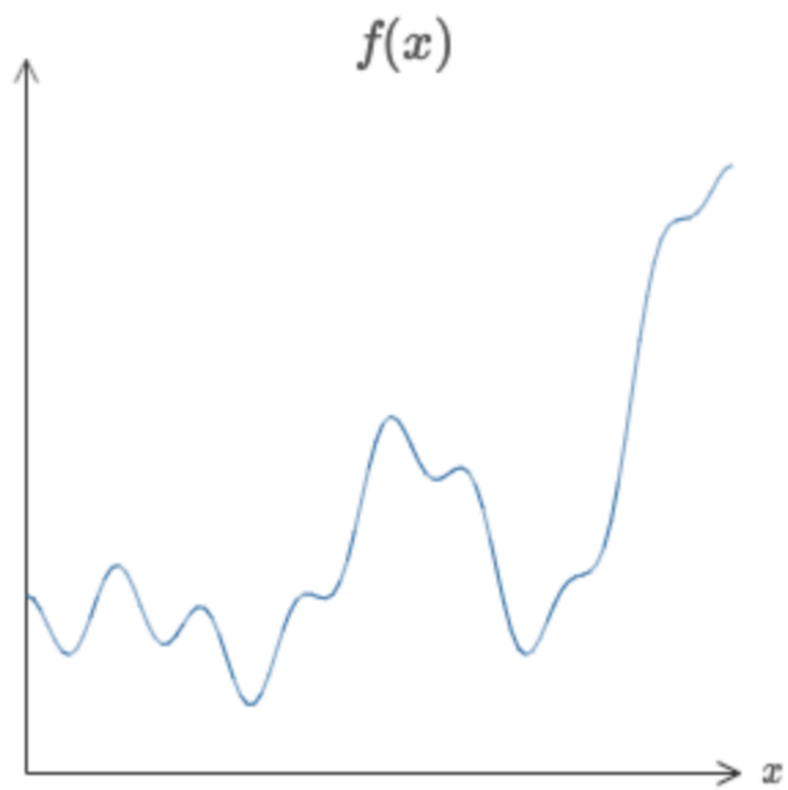
the parameters of NNs are often called ``weights'' and ``thresholds''

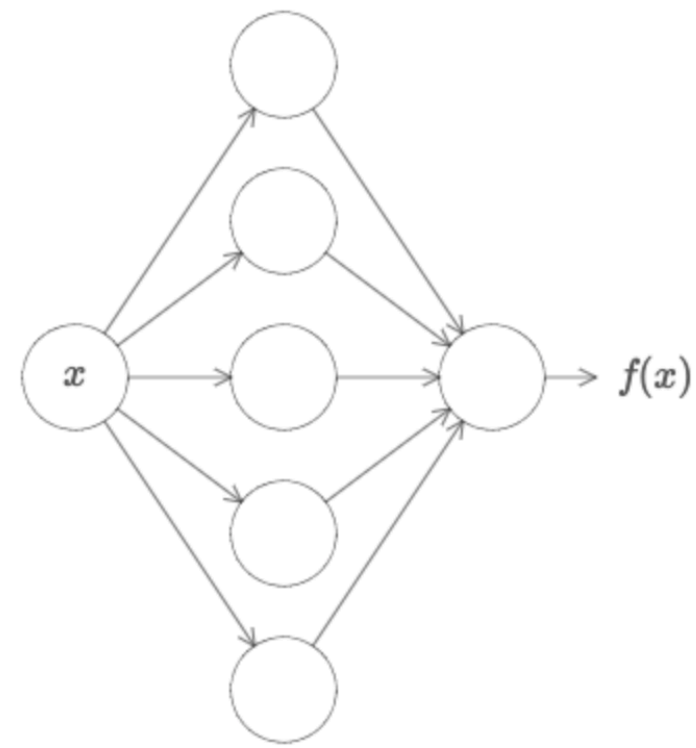
the choice of non-linear activation function affects the **computational and training properties** of the neural nets, since they modify the output gradients required for GD training

NN nonlinearly only via activation functions!

The Universal Approximation Theorem

theorem: a neural network with a single hidden layer and enough neurones can **approximate any continuous, multi-input/multi-output function** with arbitrary accuracy





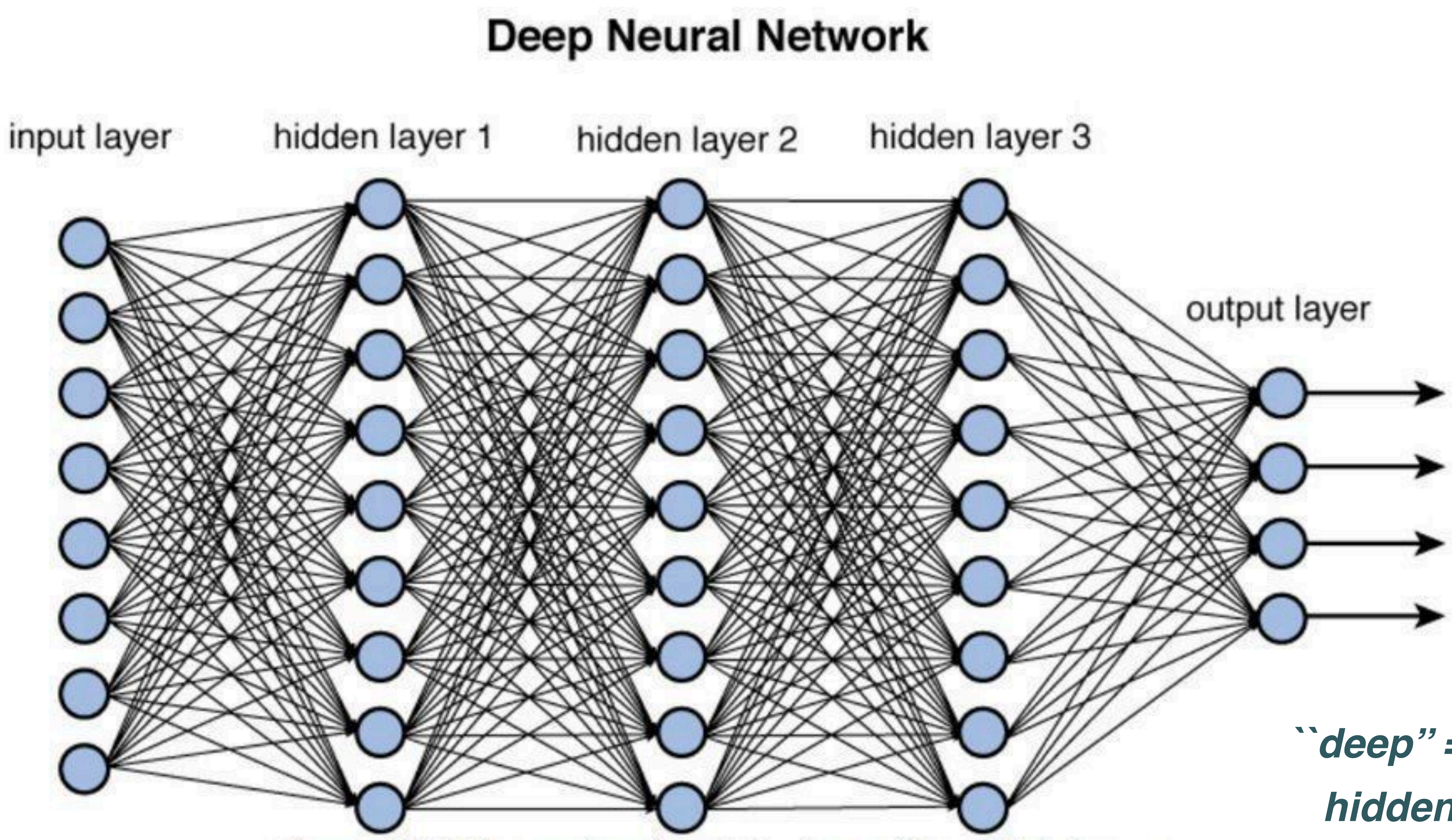
neural networks exhibit *universality properties:* no matter what function we want to compute, we know (theorem!) that there is a neural network which can carry out this task

See M. Nielsen, *Neural Networks and Deep Learning*: http://neuralnetworksanddeeplearning.com/chap4.html

Deep Networks

A neural network can thus be thought of a **complicated non-linear mapping** between the inputs and the outputs that depends on the parameters (weights and bias) of each neuron

We can make a NN **deep** by adding hidden layers, which greatly expands their **representational power**, also known as **expressivity**



``deep'' = several hidden layers

NN training

As standard in **Supervised Learning**, the first step to train a NN is to specify a **cost function**

$$\left(\boldsymbol{x}_i, y_i\right), \quad i = 1, \ldots, n \longrightarrow \hat{y}_i\left(\boldsymbol{\theta}\right), \quad i = 1, \ldots, n$$

for each of the n data points … ***… the output of the NN provides the model prediction***

the loss function depends on whether the NN should provide **continuous or categorical** (discrete) predictions. For continuous data we can have the **mean square error**

$$E(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\left(\boldsymbol{\theta}\right)\right)^2$$

for categorical data we use the **cross-entropy**, which for binary (true/false) classification is

$$E(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i \ln \hat{y}_i(\boldsymbol{\theta}) + (1 - y_i)\ln(1 - \hat{y}_i(\boldsymbol{\theta})\right)$$

where the true labels satisfy $y_i \in \{0,1\}$

NN training are based on a specific version of GD methods: **backpropagation**

Unsupervised Learning: Classification

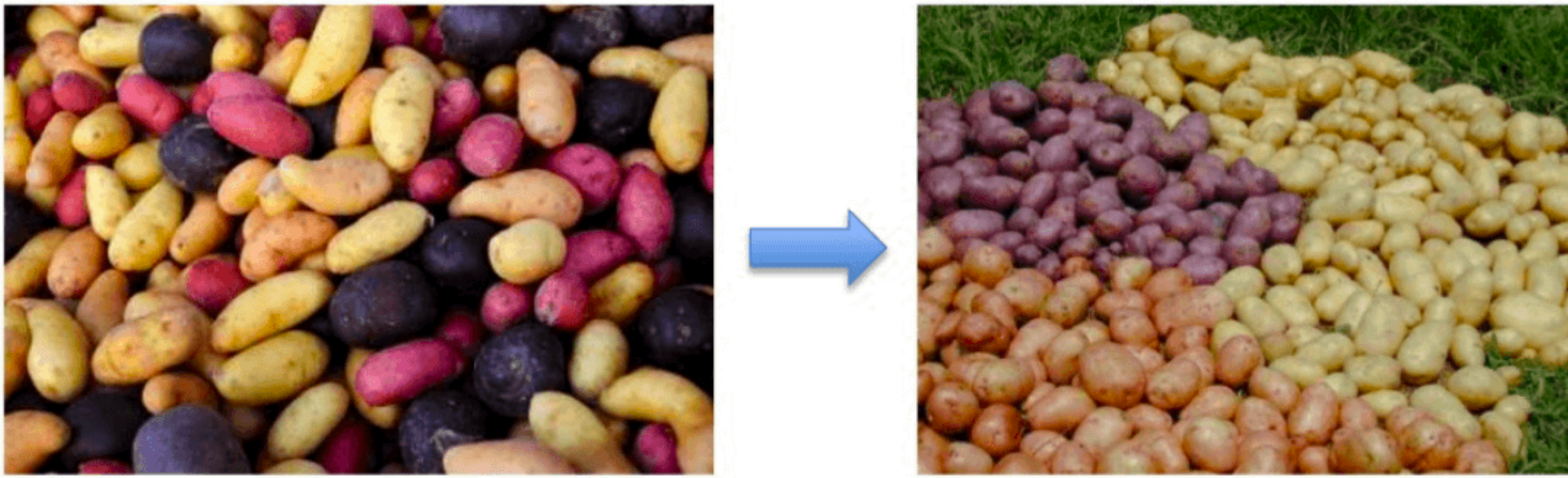
Potato Unsupervised Learning

sample

Cluster/group

what are we learning here? and is this the unique pattern that we could have identified in the data?

Potato Unsupervised Learning



sample

Cluster/group

here learning is carried out done in ``potato color'' space, but one could also consider ``potation size'' or ``potato shape'' spaces …

i) identify features + ii) use them to group the data points

Unsupervised Learning

What are the **benefits** of unsupervised learning?

- ☑ Identify unknown patterns in unlabelled data.

  e.g. potatoes come in different colours, sizes, and shapes

- ☑ Find features which can be useful for categorisation.

  e.g. sort out my potatoes as a function of their type

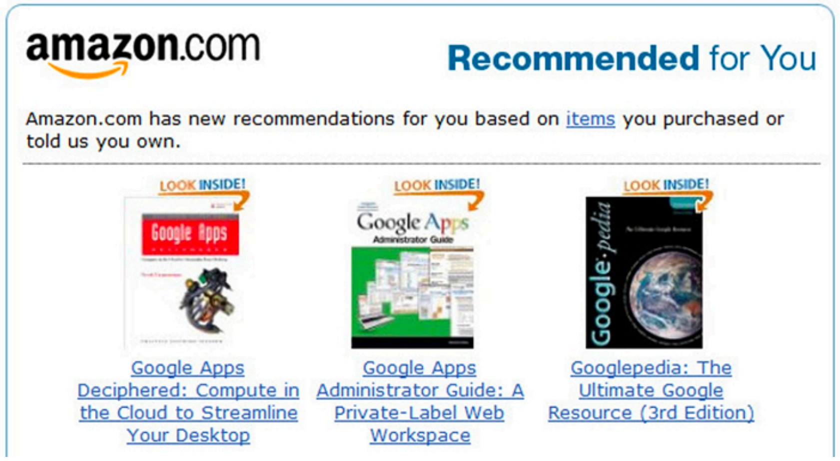
- ☑ It is easier to get automatically unlabelled data than labeled data, which often needs manual intervention.

  e.g. I don't need to label my potatoes with their variety!

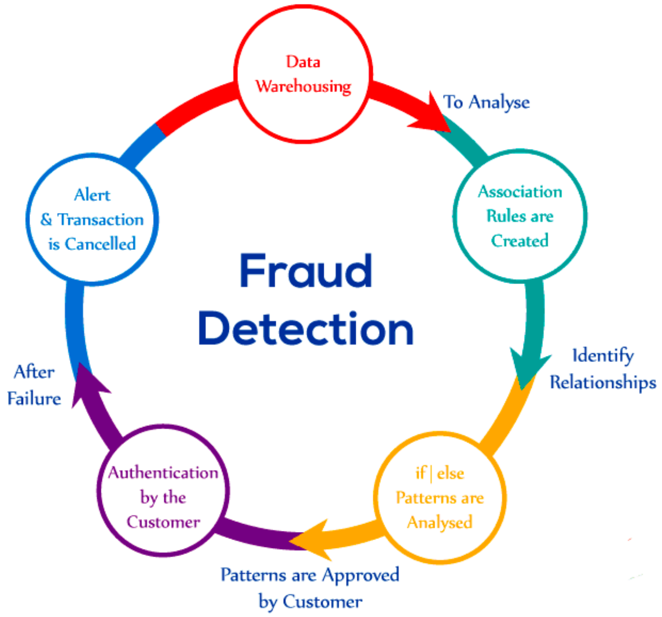
Unsupervised Learning

Some representative applications of unsupervised learning

- ☑ **Group customers** as a function of previous purchase and browsing history, to offer tailored recommendations



- ☑ Detect **anomalies** in credit card transactions that may indicate fraudulent use



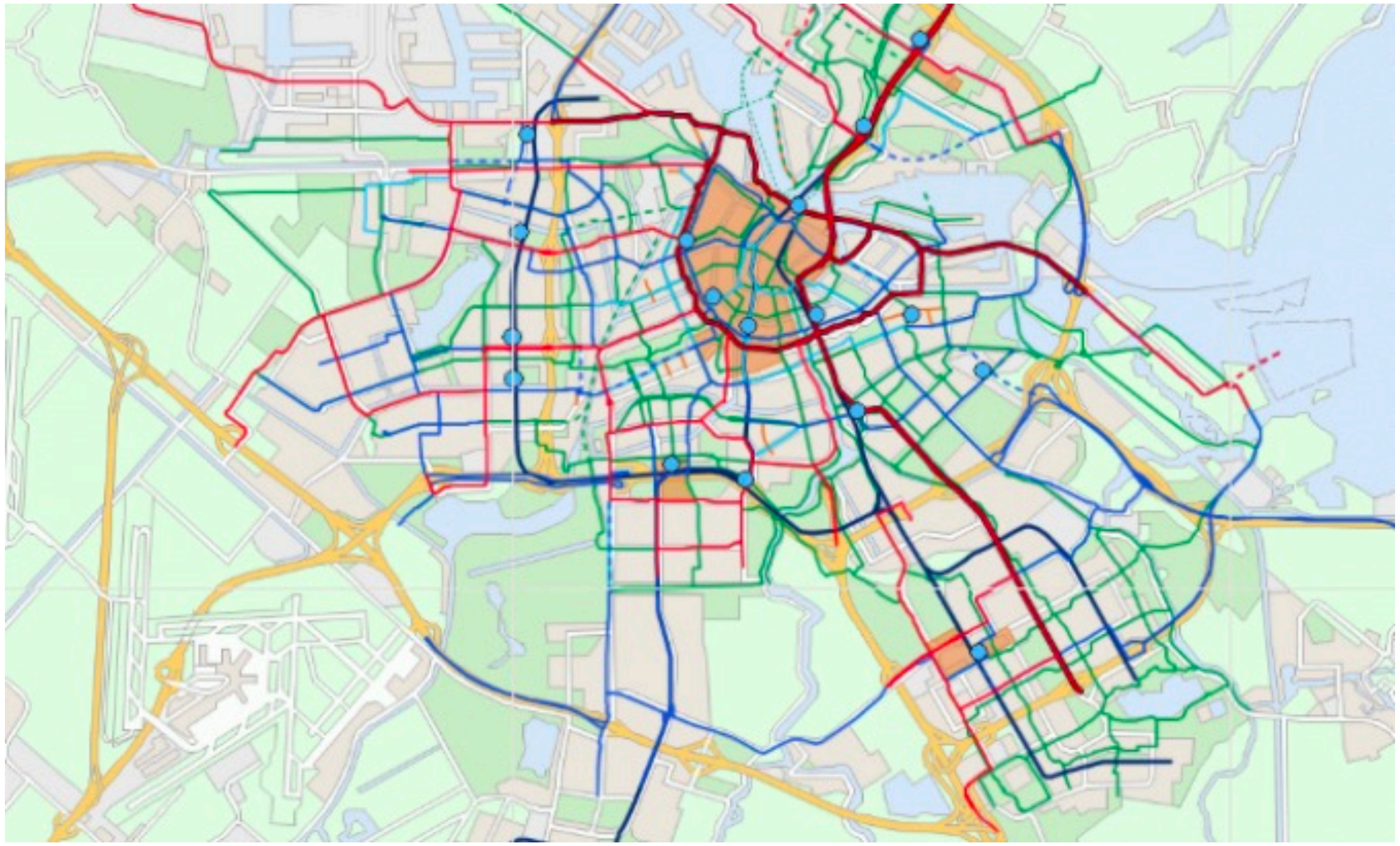
what would be my ``parameter space'' here?

Unsupervised Learning

Some representative applications of unsupervised learning

☑ Identify **travel patterns** e.g. of ride-sharing (bicycle, scooter, car) services

what information we should look for? And why it is important?

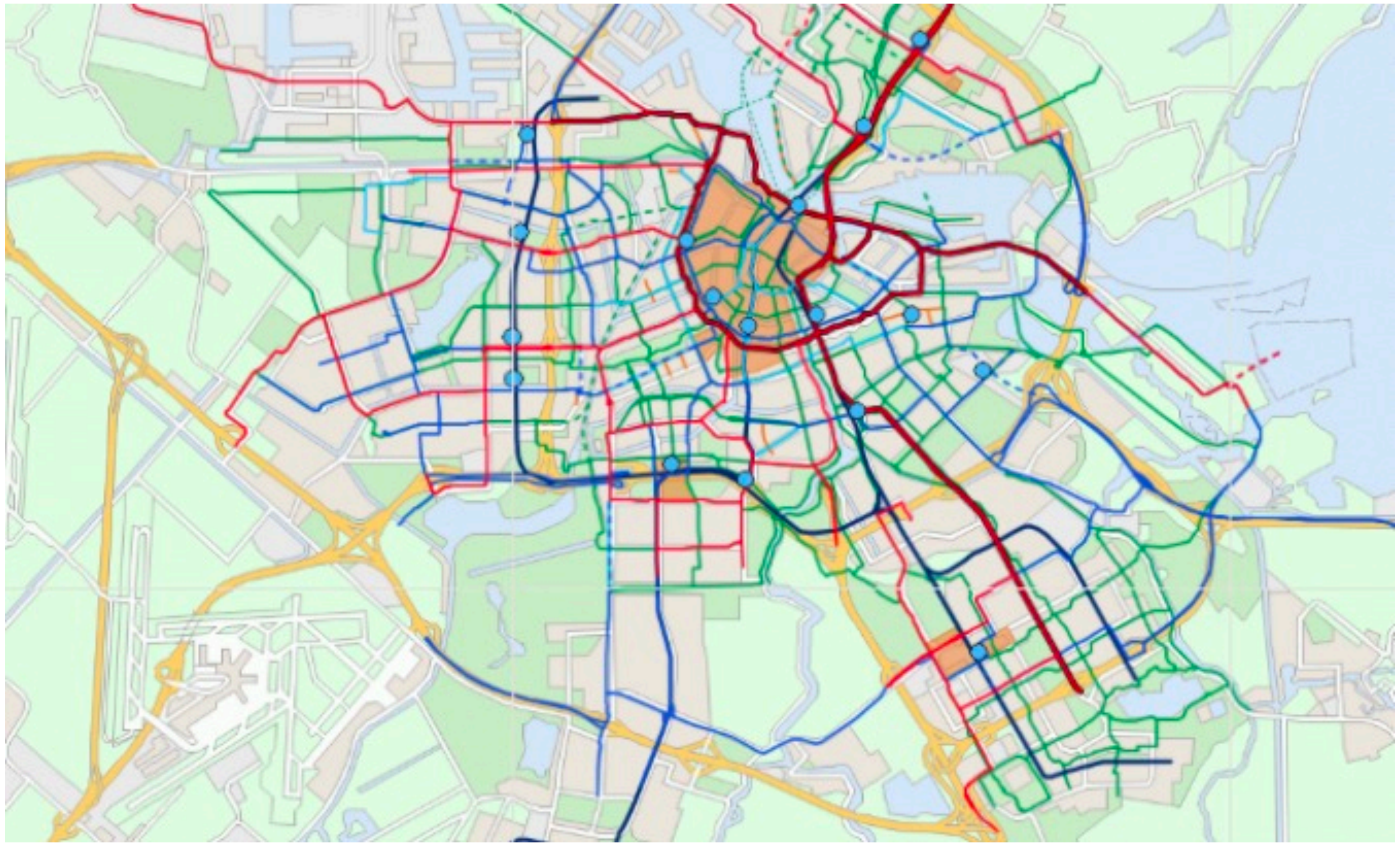


Unsupervised Learning

Some representative applications of unsupervised learning

- ☑ Identify **travel patterns** e.g. of ride-sharing (bicycle, scooter, car) services

where to put the scooters, at what time, how many per hub etc



Unsupervised Learning

What are the **disadvantages** of unsupervised learning?

- ☑ Some **interpretation** work is required after classification

  e.g. what do the identified patterns actually mean?

  a peculiar credit card translation does not always imply fraud!

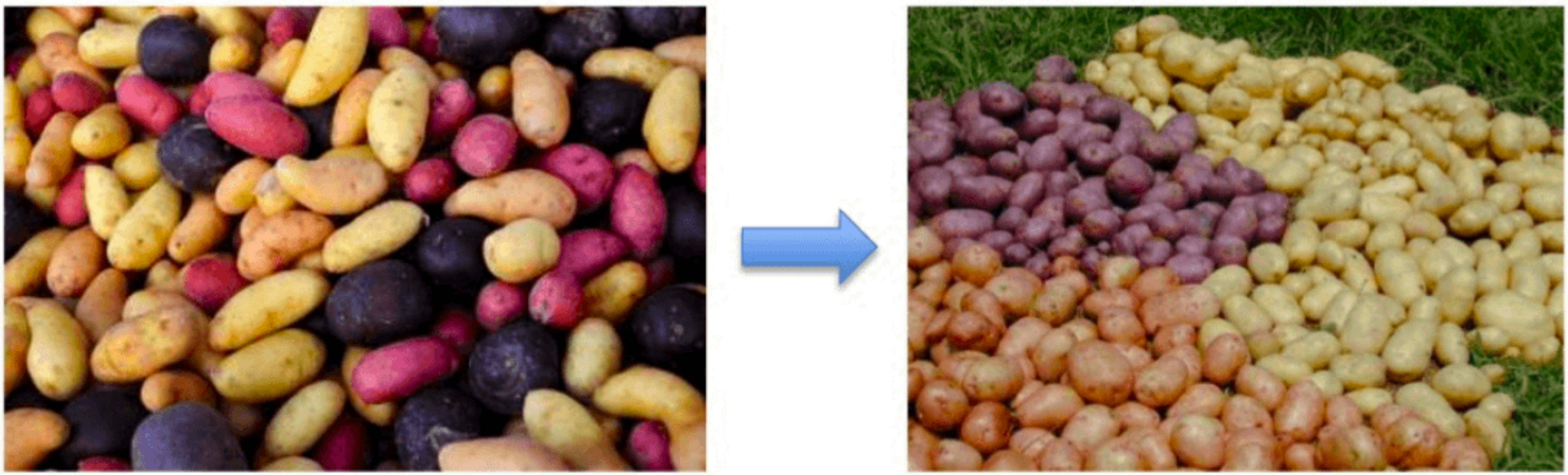
- ☑ The lack of labels makes the process less efficient than in supervised learning

  e.g. is the algorithm identifying the relevant features?

- ☑ Non-trivial to obtain **model prediction**: how to construct a rule that can be applied to identify the same patterns in new data?

  e.g. what happens if I now have new potatoes?

Clustering



sample

Cluster/group

data points: potatoes with different features (size, shape, colour)

desired output: cluster potatoes wrt some relevant feature e.g. potato variety

note: no labels attached to the potatoes!

Clustering

In ML context, **unsupervised learning** is concerned with discovering underlying structures in **unlabelled data**

an important example of unsupervised learning is **clustering:** the aim is to group unlabelled data into clusters using some **distance or similarity measure**

let us illustrate these ideas with ***K*-means clustering**

$$\{\boldsymbol{x}_n\}_{n=1}^N \qquad \boldsymbol{x}_n = (x_{n,1}, x_{n,2}, \ldots, x_{n,p})$$

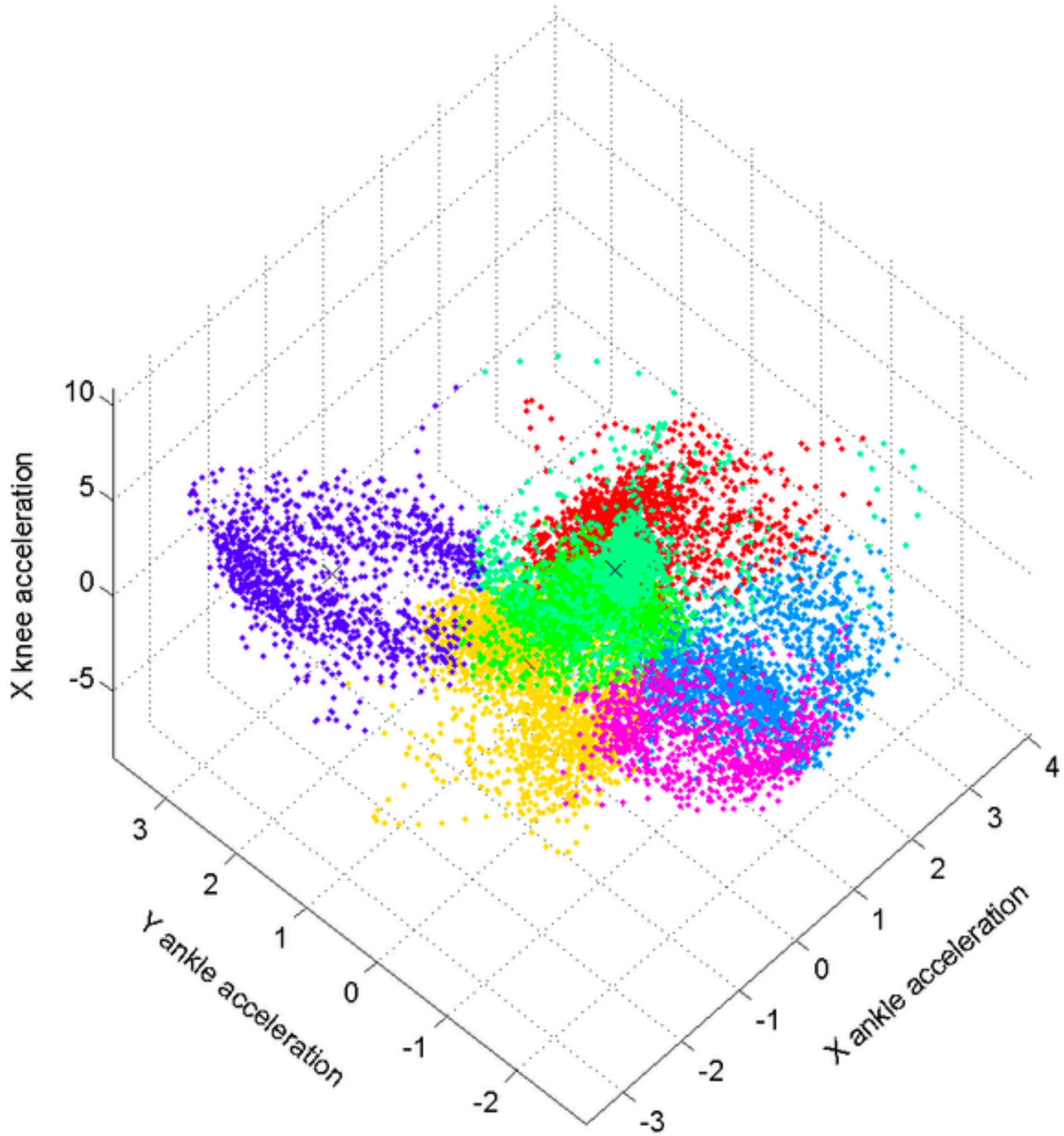
unlabelled dataset: N points with p features each

$$\{\boldsymbol{\mu}_k\}_{k=1}^K \qquad \boldsymbol{\mu}_k = (\mu_{k,1}, \mu_{k,2}, \ldots, \mu_{k,p})$$

***cluster means**: K clusters with p features each*

the intuitive idea is that the cluster means represent the **main features of each cluster**, to which the data points will be assigned in the clustering procedure

Unsupervised Learning



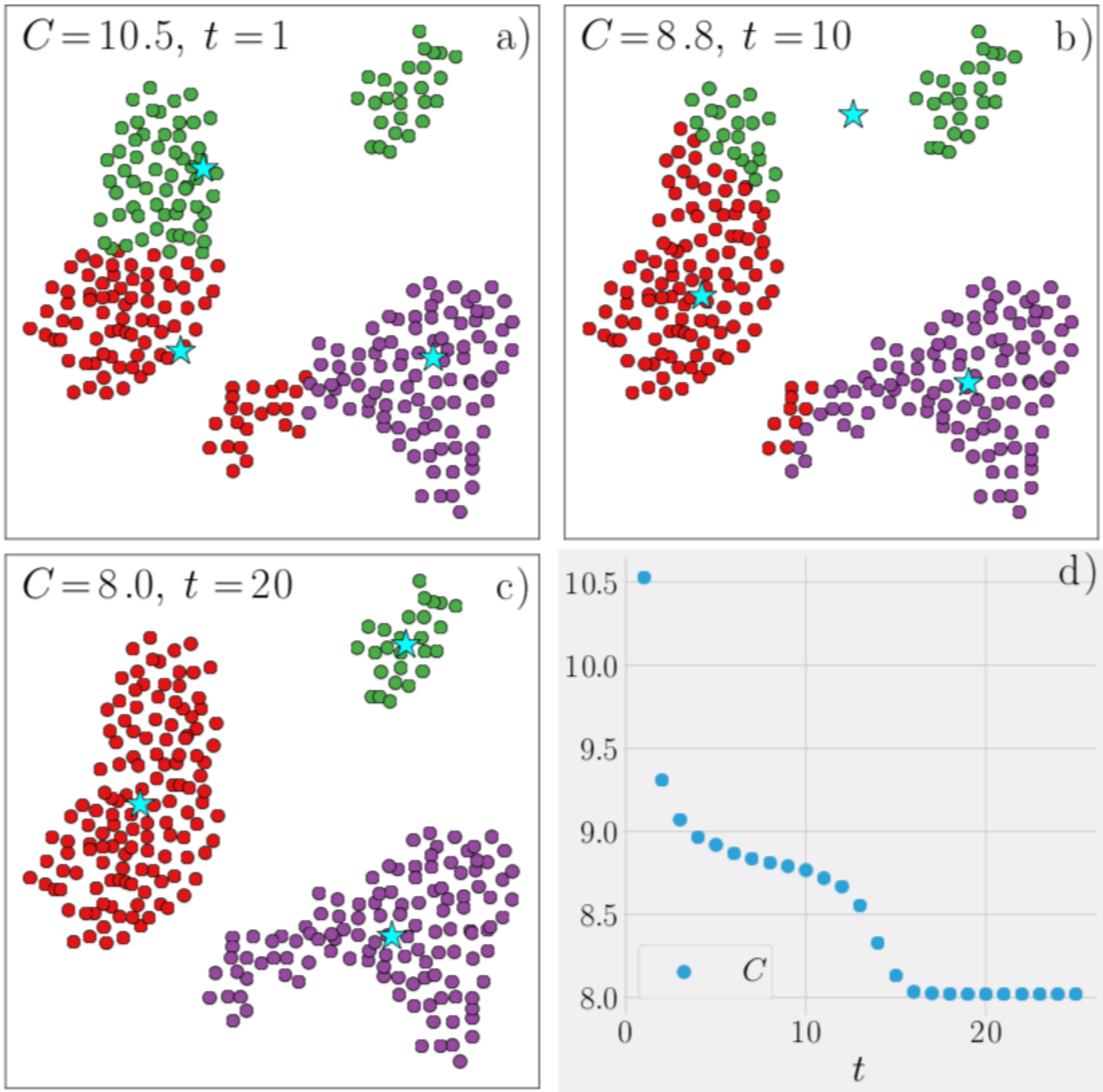
nb color here for visualisation, but not this info not available in real applications!

how many ``**groups of samples”** do we have have in our dataset?

Clustering

2D example of clustering: each colour represents a cluster, with stars indicating their **centers**

how is this clustering achieved **in practice?**



Clustering

Q: what is a suitable ``distance'' or ``metric'' between two data points of the training set?

- *training set: points in n-dimensional Cartesian space?*
- *training set: state vectors in QM Hilbert space?*
- *training set: pictures of cats and dogs?*
- *training set: four-momenta of particles produced in a high-energy collision?*

defining ``closeness'' in unsupervised learning often non-trivial

Clustering

in K-means clustering, the **cluster means** and the **data point assignments** are determined from the minimisation of a cost function:

$$C(\boldsymbol{x};\boldsymbol{\mu}) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\left(\boldsymbol{x}_n - \boldsymbol{\mu}_k\right)^2$$

other distances possible: the user needs to define what ``closeness" means!

binary assignment variable

Euclidean distance between n-th data point and k-th cluster centre

$r_{nk} = 1 \longrightarrow$ *the n-th point is assigned to the k-th cluster*

$r_{nk} = 0 \longrightarrow$ *the n-th point is not assigned to the k-th cluster*

furthermore since **clustering is exclusive** one needs to impose: $\displaystyle\sum_{k=1}^{K} r_{nk} = 1 \quad \forall n$

one sees that K-means clustering aims to **minimise the variance within each cluster**

Clustering

Let's describe an algorithm that implements *K*-means clustering by minimising the cost function

$$C\left(\boldsymbol{x};\boldsymbol{\mu}\right) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\left(\boldsymbol{x}_n - \boldsymbol{\mu}_k\right)^2$$

this algorithm alternates iteratively between two main steps:

- ***(1) Expectation***: starting from set of cluster assignments ***{r_{nk}}*** minimise *C* wrt cluster means

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} C\left(\boldsymbol{x};\boldsymbol{\mu}\right) = 0 \quad \rightarrow \quad \boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n \qquad N_k = \sum_{n=1}^{N} r_{nk}$$

number of points in k-th cluster

- ***(2) Maximization***: given the *K* cluster centers, the assignments ***{r_{nk}}*** should minimise *C*. This can be achieved by assigning each data point to its closest cluster-mean

$$r_{nk} = 1 \quad \text{if} \quad k = \arg\ \min_{k'}(\boldsymbol{x}_n - \boldsymbol{\mu}_{k'})$$
$$r_{nk} = 0 \quad \text{if} \quad k \neq \arg\ \min_{k'}(\boldsymbol{x}_n - \boldsymbol{\mu}_{k'})$$

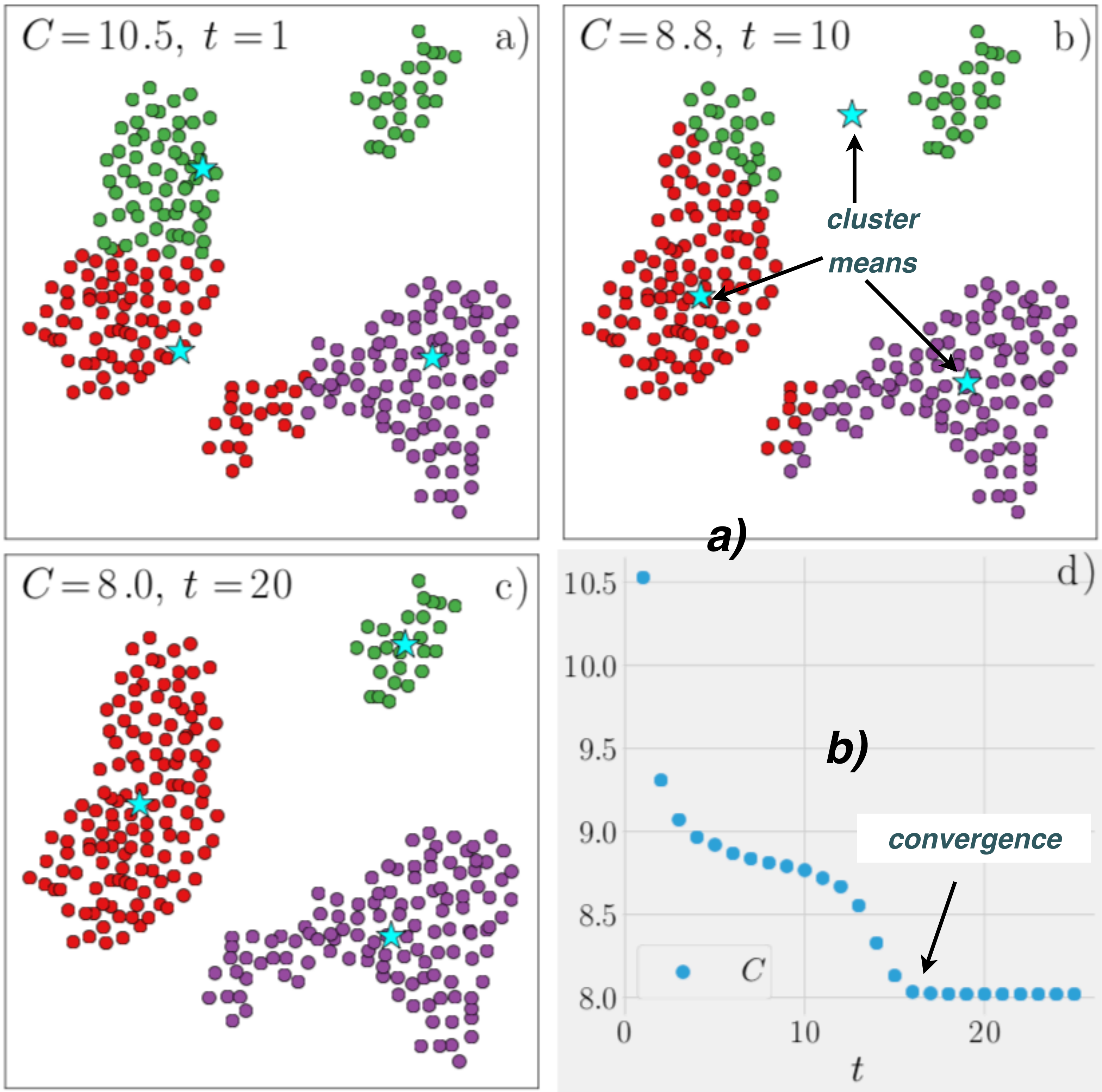
iterate until convergence achieved!

note that here GD not required, optimisation is semi-analytical

Clustering

these two steps are iterated until some **convergence criterion** is achieved, *e.g.* when the change in the cost function between two iterations is below some threshold

each colour: different cluster

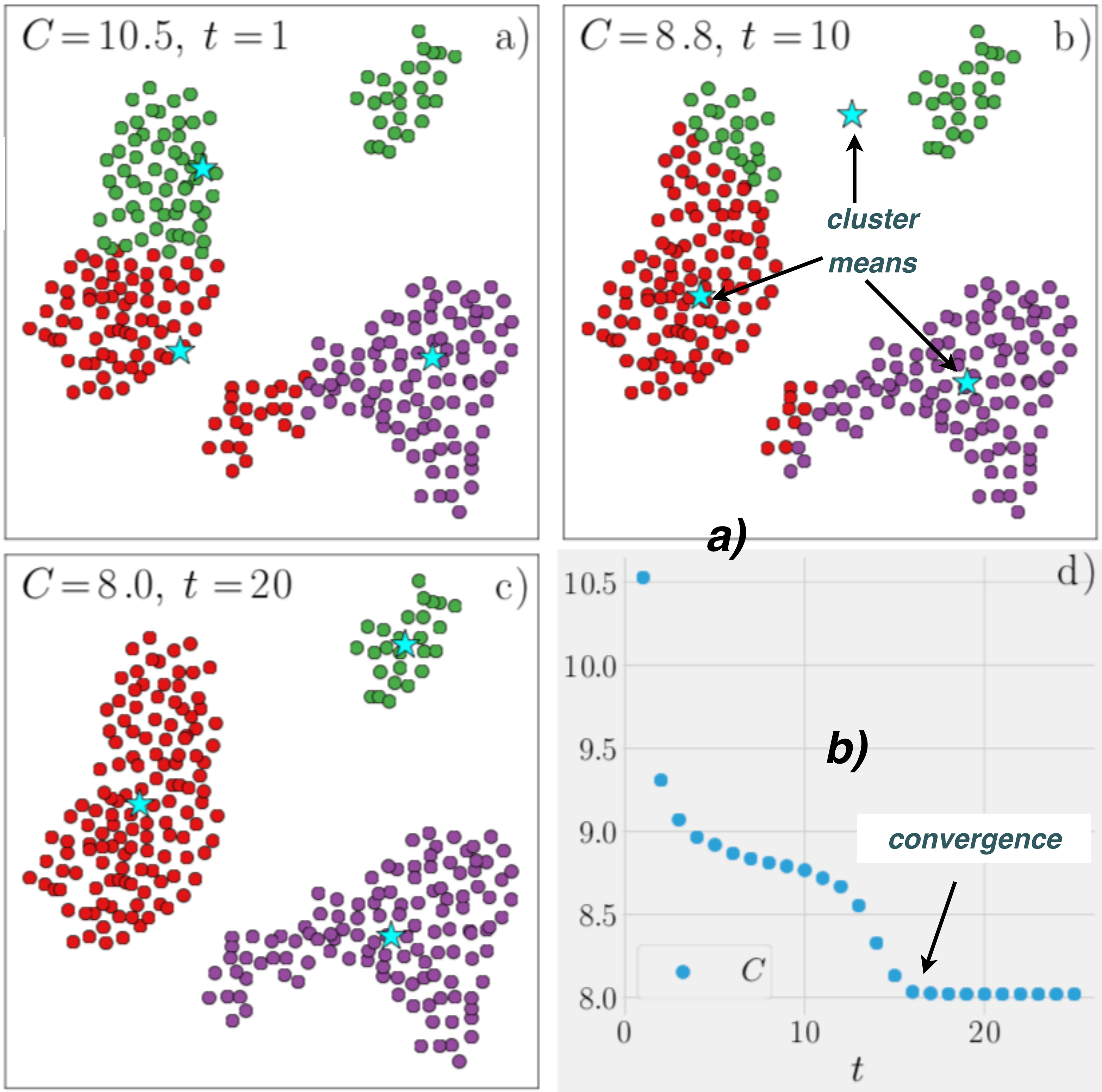


here overfitting not possible: there exists a unique assignment that minimises the cost function

Clustering

these two steps are iterated until some **convergence criterion** is achieved, *e.g.* when the change in the cost function between two iterations is below some threshold

each colour: different cluster

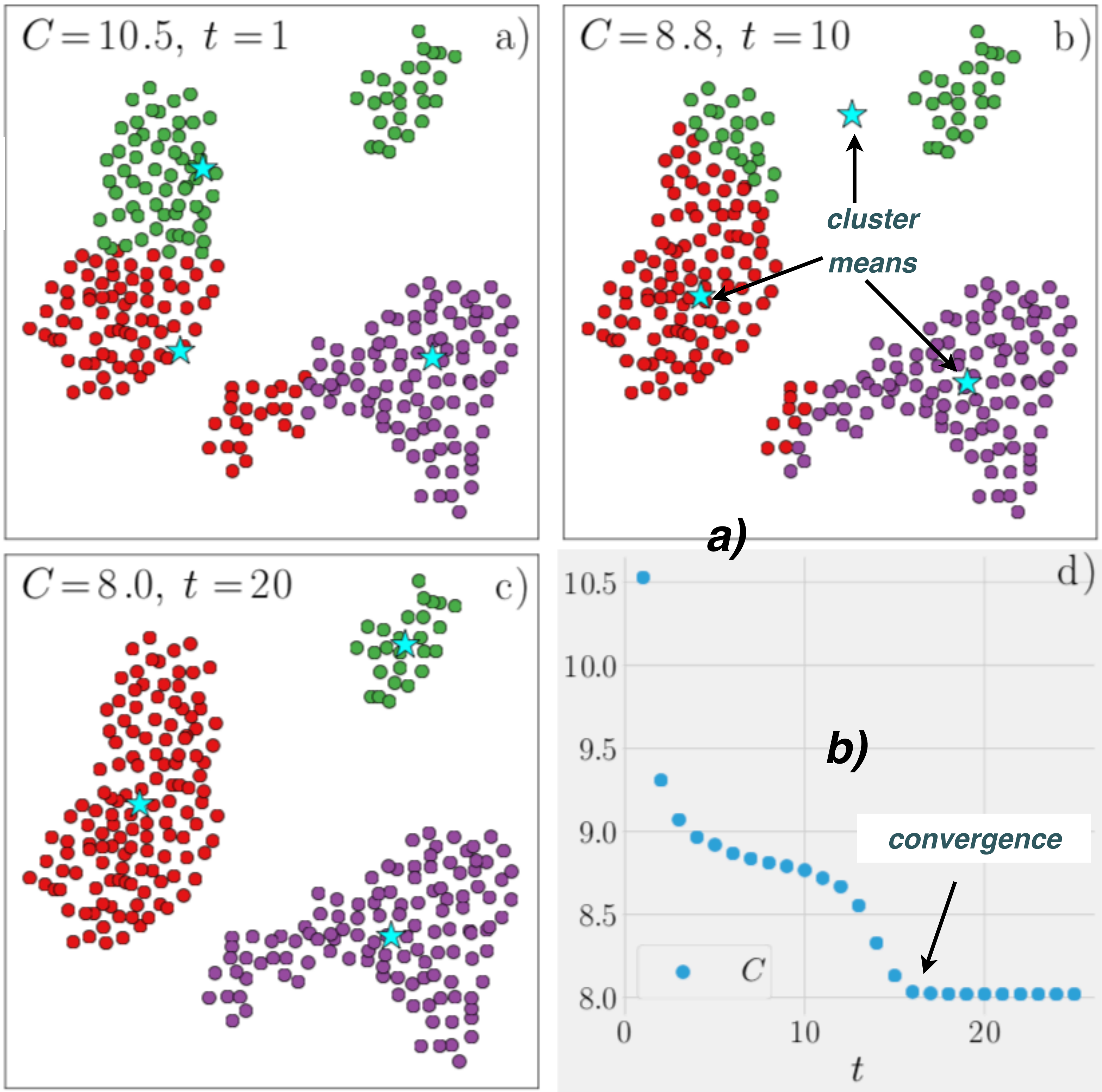


what is the underlying assumption in K-means clustering? And when it would not be justified?

Clustering

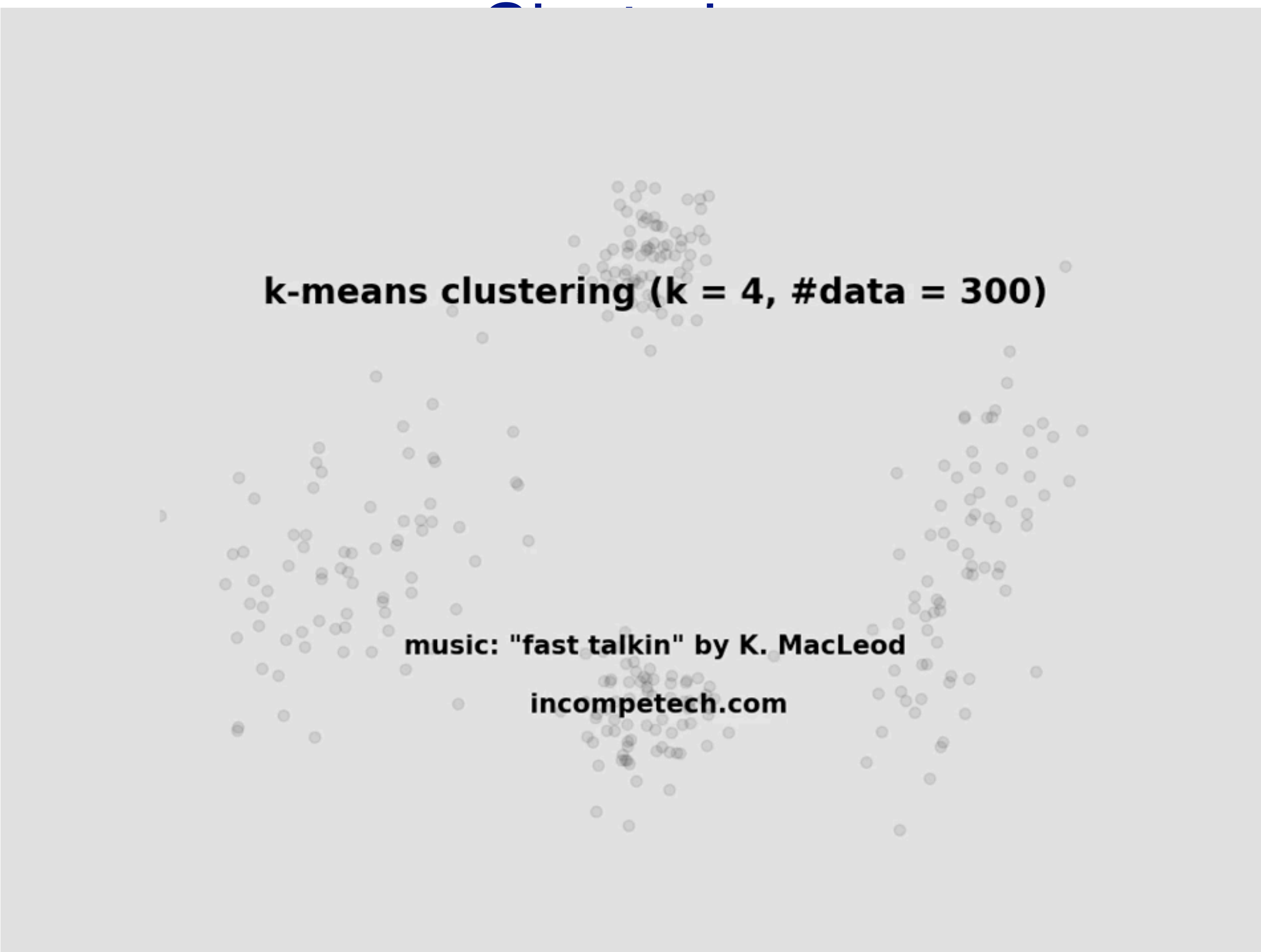
these two steps are iterated until some **convergence criterion** is achieved, *e.g.* when the change in the cost function between two iterations is below some threshold

each colour: different cluster



*K-means clustering can lead to spurious results since the **underlying assumption** is that the latent model has uniform variances*

fails if the underlying clusters have different variances!


k-means clustering (k = 4, #data = 300)
music: "fast talkin" by K. MacLeod
incompetech.com

Convolutional Neural Networks

Supervised learning and classification

The goal is to **predict a class label** from a pre-defined list of possibilities

the simplest type of problem is **binary classification** (yes/no problems)

e.g. should I put this email in the spam folder?

but in general one considers **multiclass classification** (> 2 categories)

e.g. which type of bird is the one I just photographed?

In the context of ML applications there exist a large number of approaches to classifications tasks. The most basic one is based on assembling a **discriminant function** that maps each input data point to its specific class

model parameters

$$\boldsymbol{x}_i \rightarrow g\left(\boldsymbol{x}_i, \boldsymbol{\theta}\right)$$

input data point

discriminant

(aka activation function)

$$g = 0 \,\text{if}\, \boldsymbol{x}_i \in \mathscr{C}_1$$

$$g = 1 \,\text{if}\, \boldsymbol{x}_i \in \mathscr{C}_2$$

$$g = 2 \,\text{if}\, \boldsymbol{x}_i \in \mathscr{C}_3$$

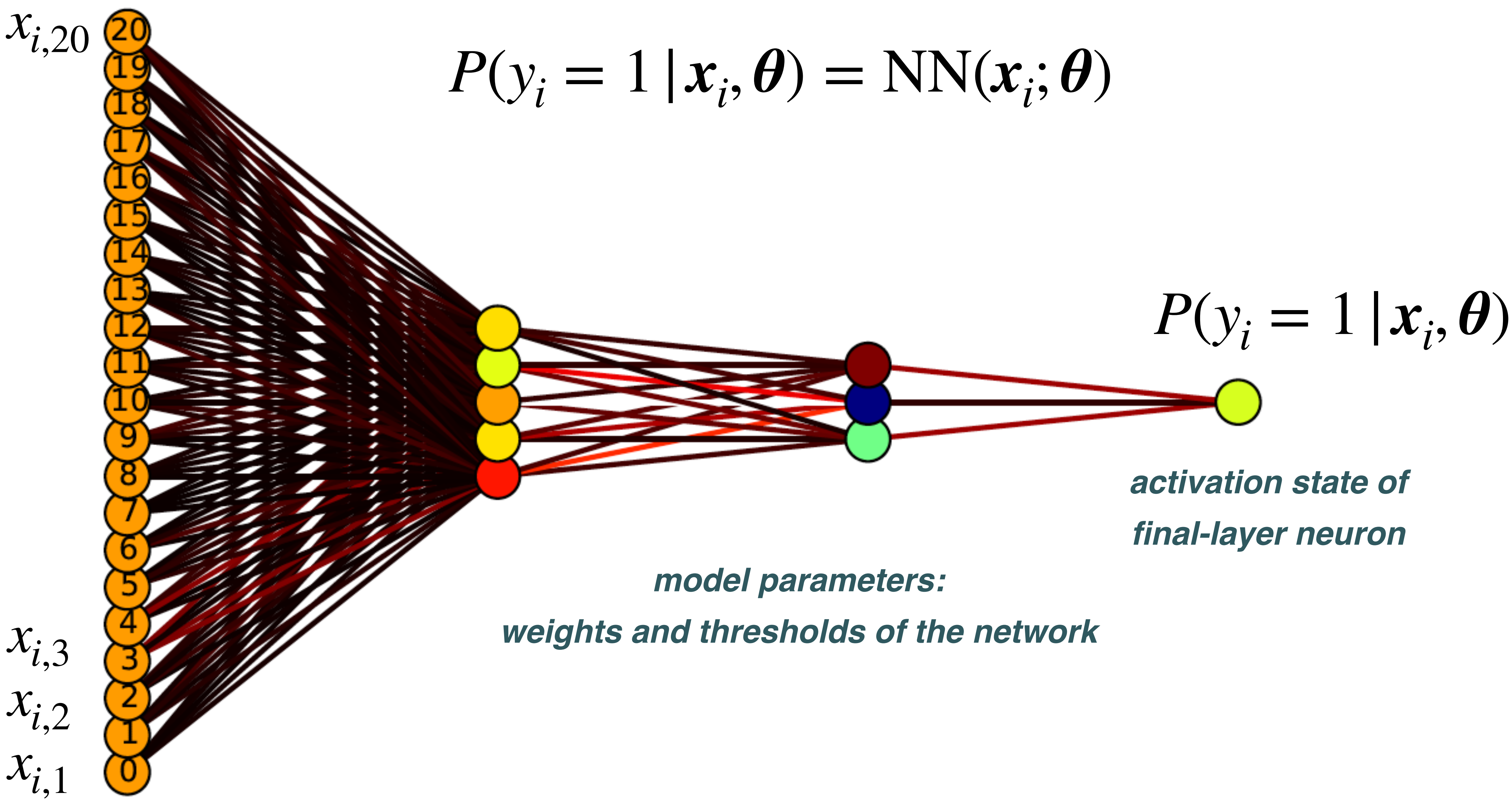
Supervised learning and classification

instead of modelling the classification probability by a **simple logistic function**

$$P(y_i = 1 \,|\, \boldsymbol{x}_i, \boldsymbol{\theta}) = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \boldsymbol{\theta}}}$$

one can adopt more complex models, such as deep neural networks

$$P(y_i = 1 \,|\, \boldsymbol{x}_i, \boldsymbol{\theta}) = \mathrm{NN}(\boldsymbol{x}_i; \boldsymbol{\theta})$$



Learning with symmetry

Like physical systems, many datasets and supervised learning tasks also possess additional **symmetries and structure** what can (and should) be exploited

e.g. we want to train a classifier to identify pictures of cats. What **high-level features** must one learn first?

Learning with symmetry

Like physical systems, many datasets and supervised learning tasks also possess additional **symmetries and structure** what can (and should) be exploited

e.g. we want to train a classifier to identify pictures of cats. What **high-level features** must one learn first?

- *The features that define ``cat" are local in the picture: whiskers, tail, paws ….: **locality***
- *Cats can be anywhere in the image: **translational invariance***
- *Relative position of features must be respected (eg whiskers and tail shoaled appear in opposite sides of ``cat"): **rotational invariance***

O***ur classifier should exhibit all these high-level features***

Learning with symmetry

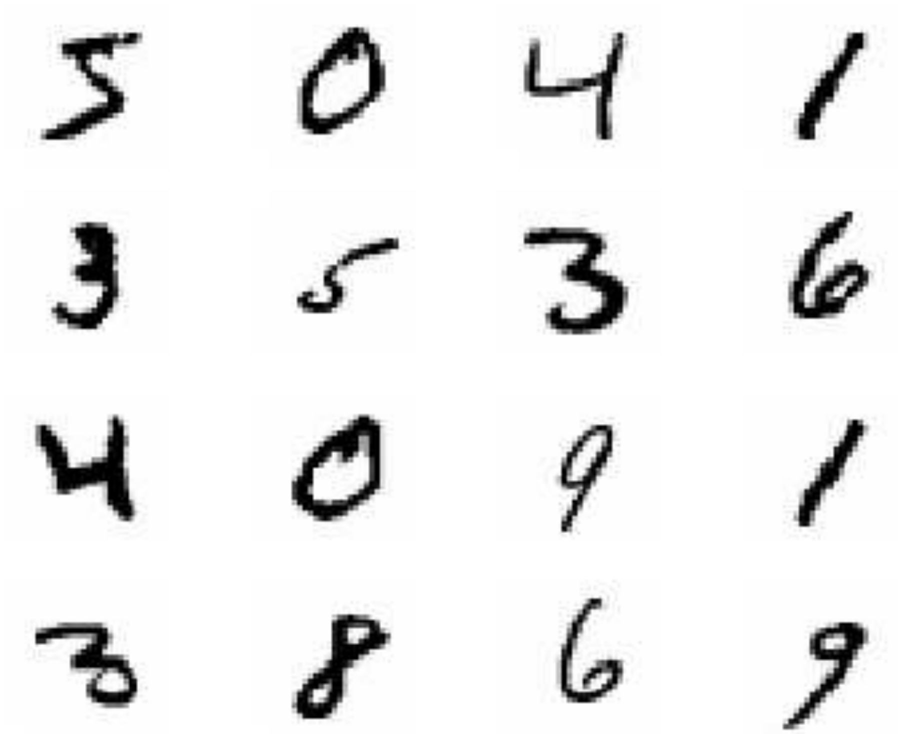
our goal is to create models which are **invariant** wrt certain transformations of the inputs

CNNs hard-code these invariance properties into the **structure of the network**

extensively used for applications in **pattern recognition**

e.g. classify handwritten digits

Inputs: set of pixel intensity values of each image

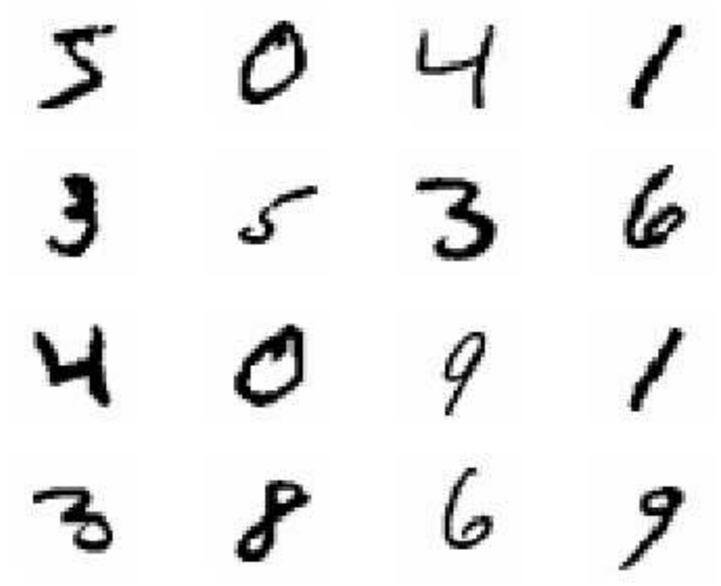


Output: posterior probability distribution over the 10 digit classes

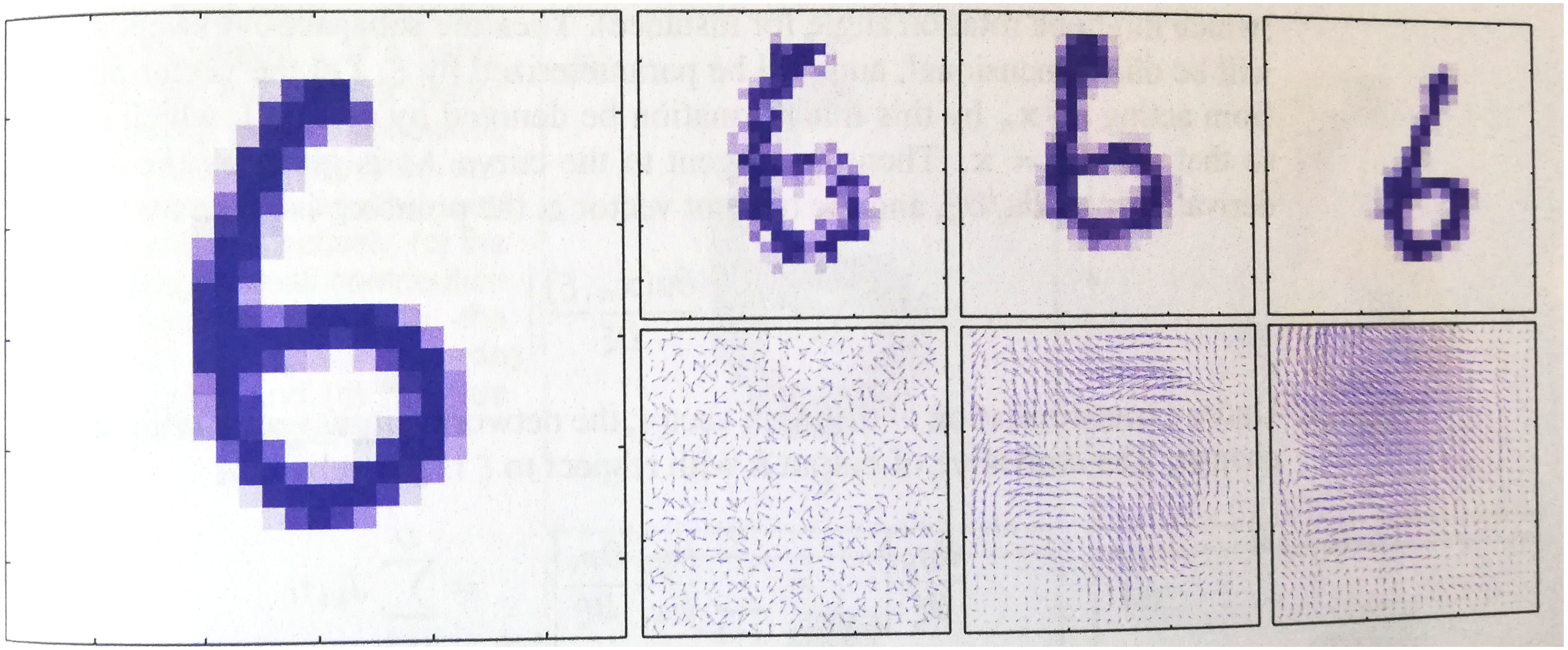
what kind of **symmetries** must we built-in in our ML classifier model?

Learning with symmetry

what kind of **symmetries** must we
built-in in our ML classifier model?



- Invariance under **translations**
- Invariance under **scaling**
- Invariance under **small rotations**
- Invariance under **smearing**
- Invariance under **elastic deformations**



Convolutional Neural Networks

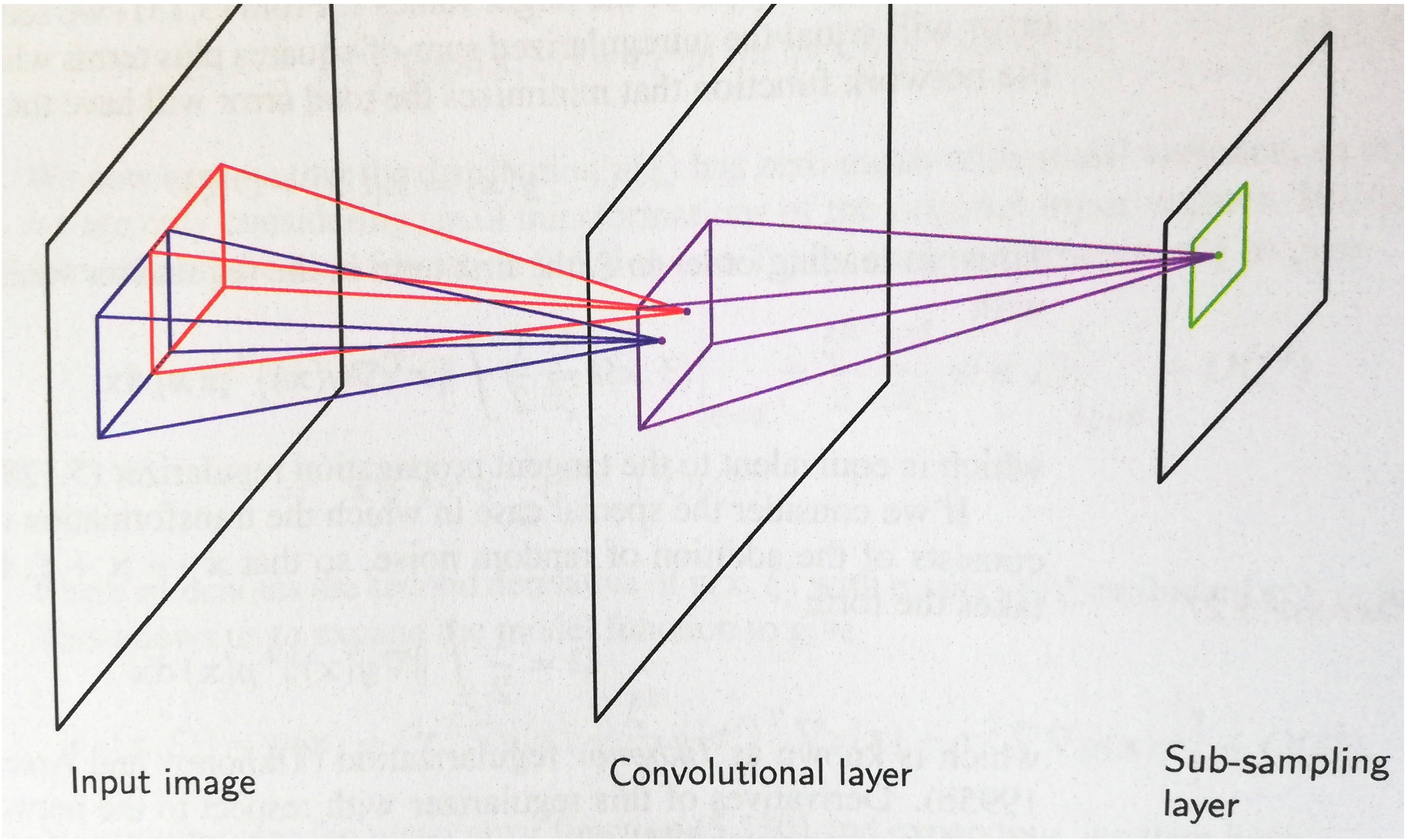
the simplest approach would be to input the images to a **fully connected NN** which given enough training data (and time) would **learn the symmetries by example**

however this way a crucial property is ignored: **nearby pixels are strongly correlated**

we should aim instead first to **identify local features** that depend on small subregions

afterwards such local features can be combined into **higher-level ones**

Convolutional Neural Networks (CNNs) are architectures that take **advantage of this additional high-level structures** that all-to-all coupled networks fail to exploit

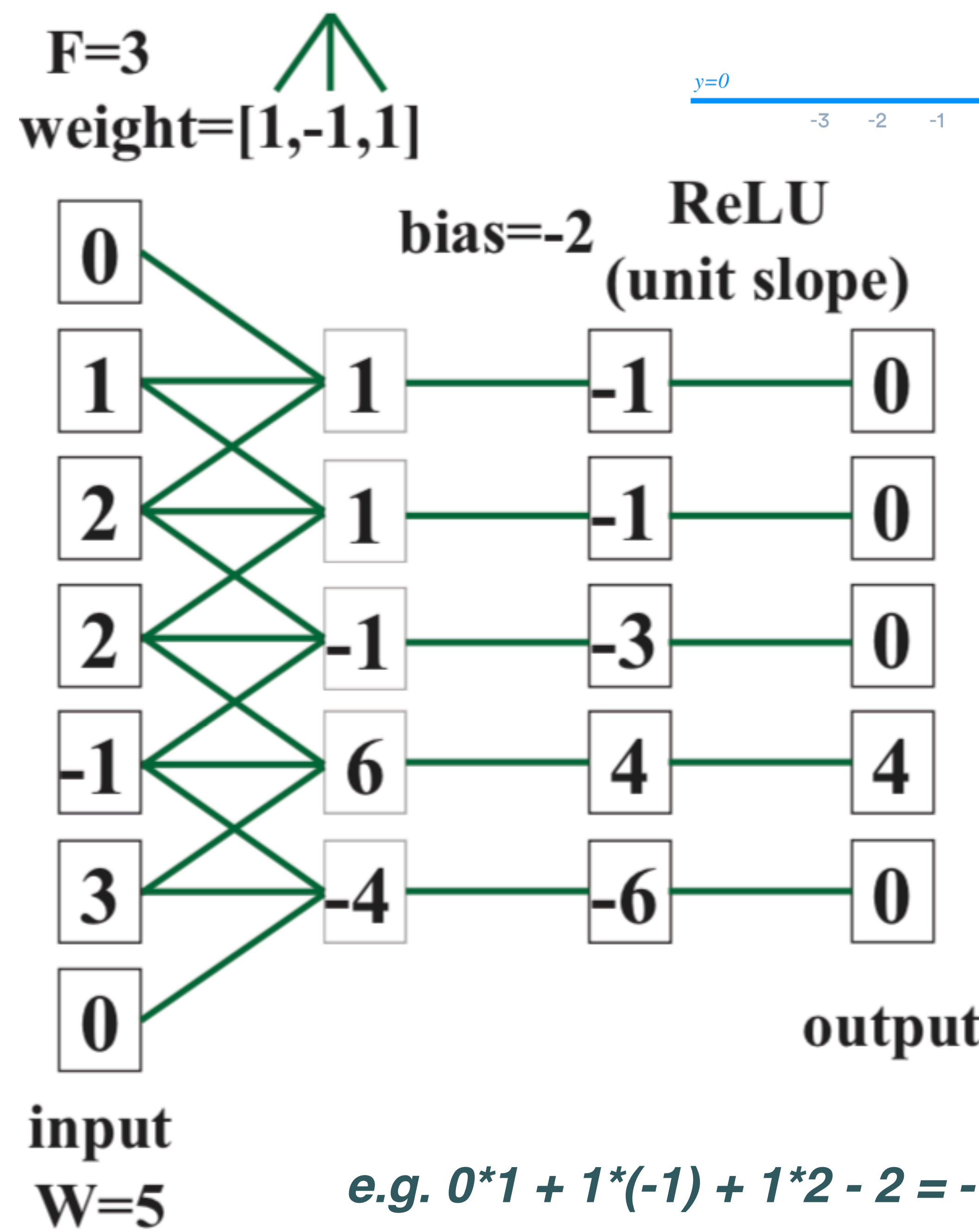


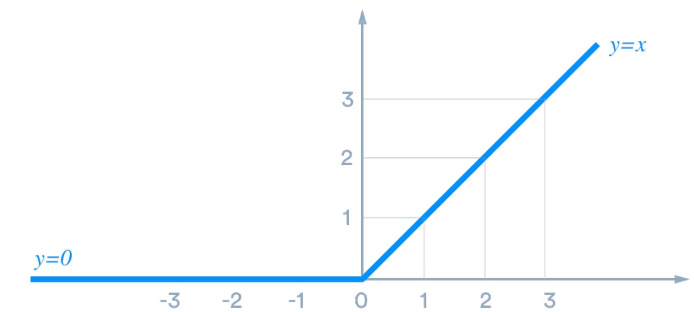
Convolutional Neural Networks

CNNs are composed by two kinds of layers

Convolution of input with **filters**

size-3 Filter





example of convolutional layer

note that convolution changes the depth, but not the height and width of the network

*e.g. 0*1 + 1*(-1) + 1*2 - 2 = -1*

Convolutional Neural Networks

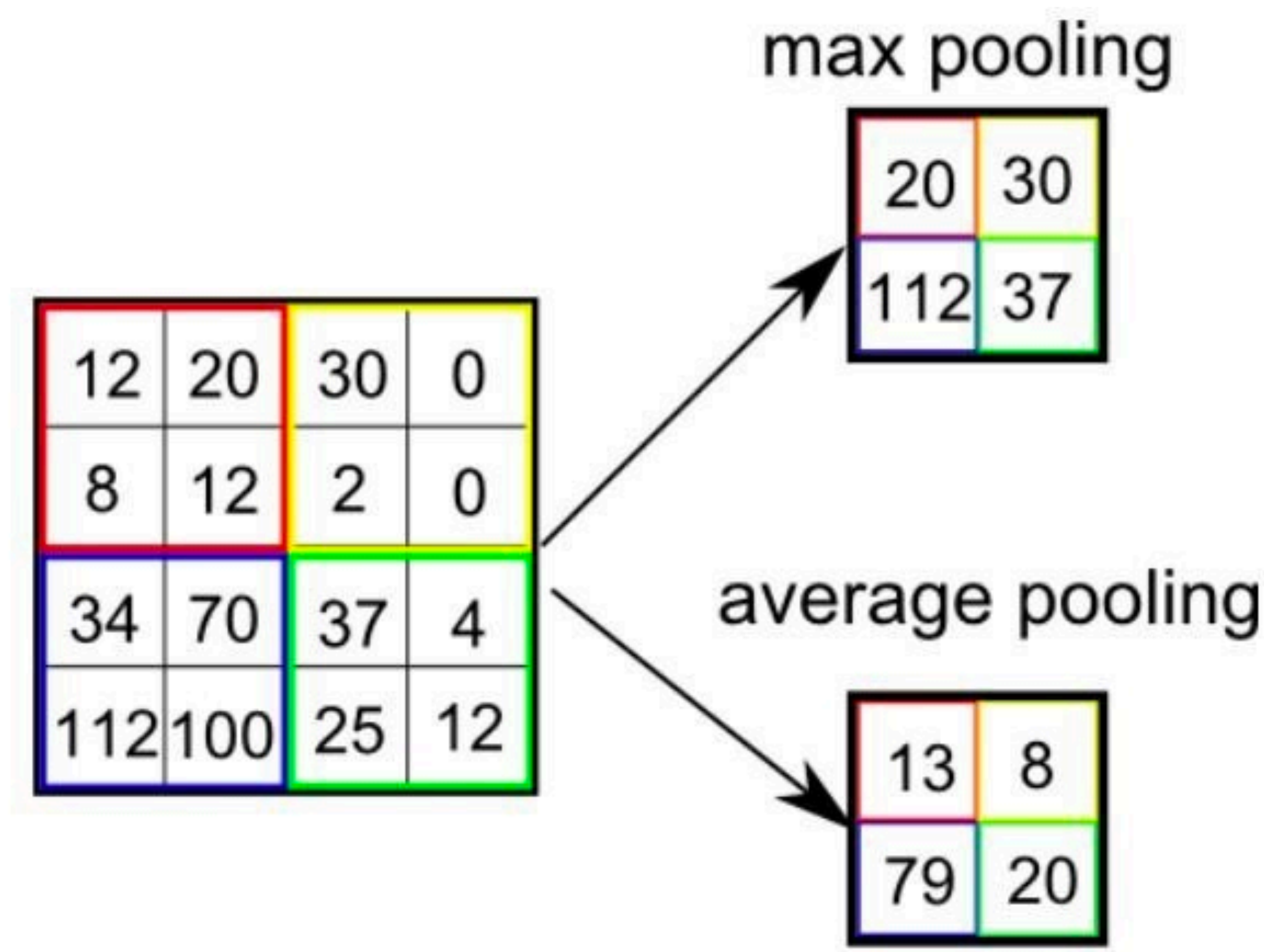
CNNs are composed by two kinds of layers

Convolution layer of input with **filters**

Pooling layer that coarse-grains the input while maintaining locality and spatial structure

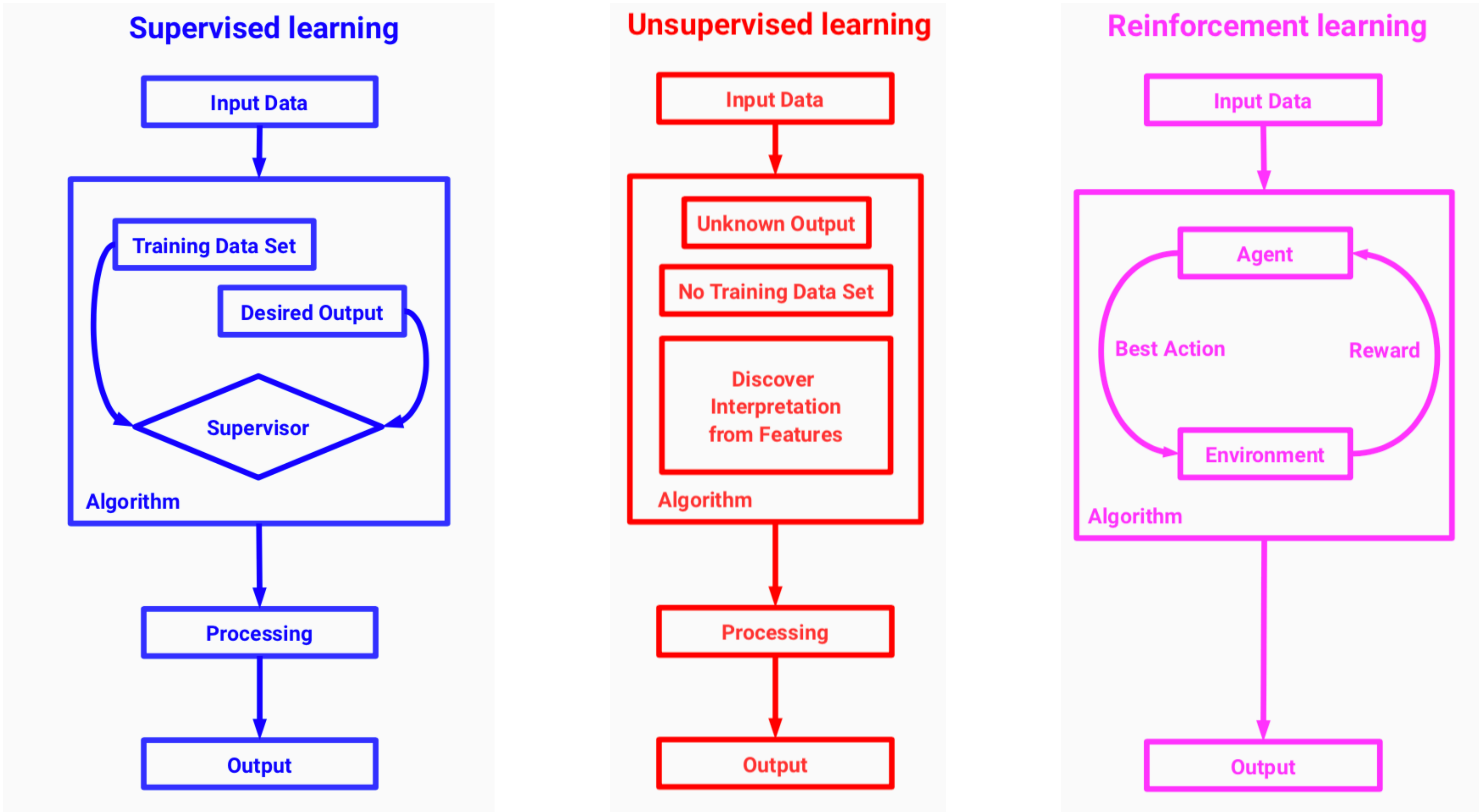
e.g. **MaxPool**, the spatial dimensions are coarse-grained by replacing a small region by single neuron whose output is maximum value of the output in the region

in average pooling, one averages over output in region



Reinforcement Learning

Supervised vs Unsupervised Learning



Reinforcement Learning

So far we have considered **two main paradigms** in Machine Learning problems

Supervised Learning: starting from a training dataset with **labelled examples, $\{x_i, y_i\}_{i=1,N}$** , produce a **model f(x)** that predicts and generalises the info in the training sample. The labels **y_i** can be continuous (underlying law is function) or discrete (classification)

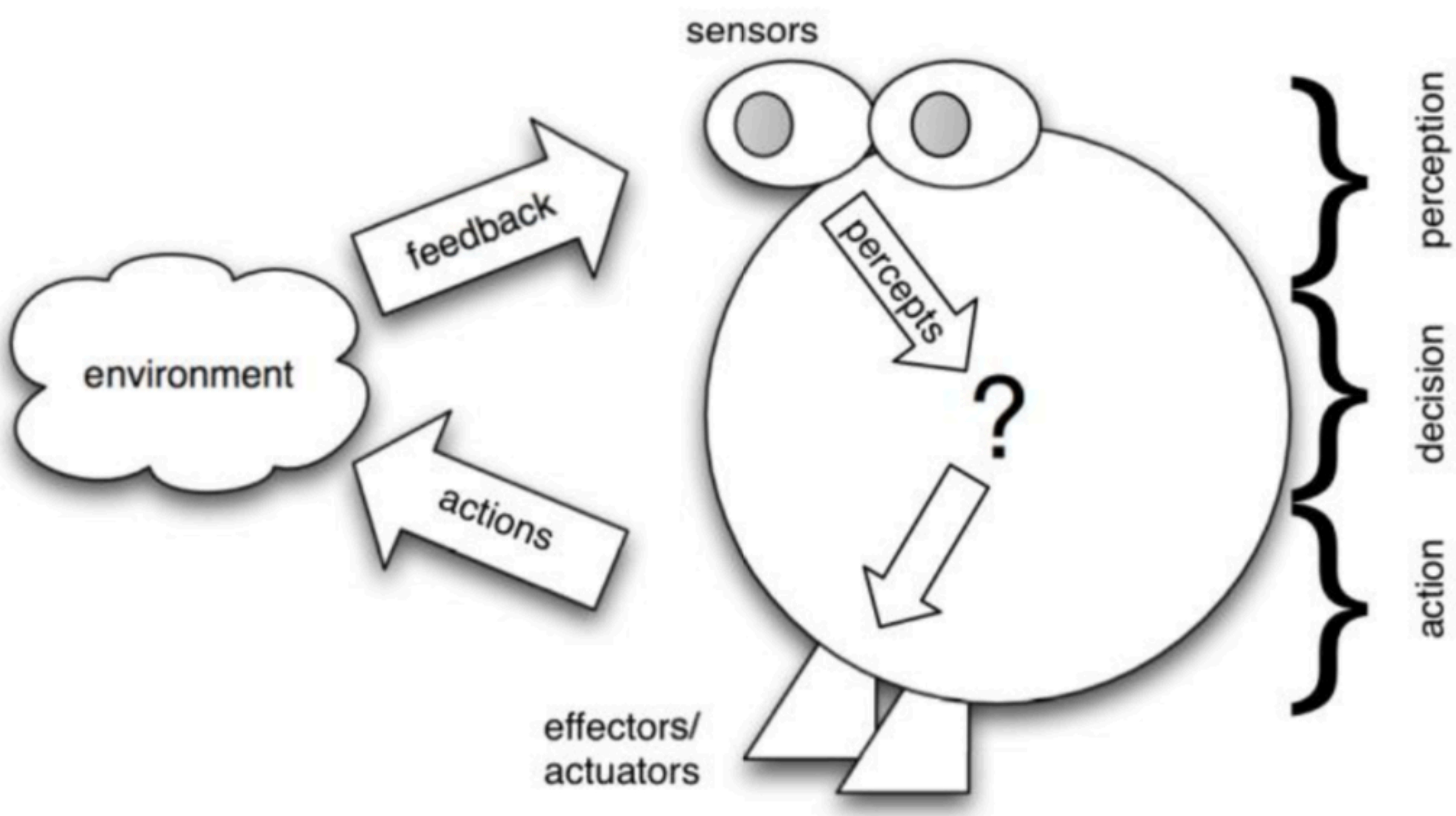
Unsupervised Learning: starting from a training dataset with **unlabelled examples, $\{x_i\}_{i=1,N}$** , produce a **model** that takes a sample as input and as output produces the solution of a practical problem, such as **clustering**, **dimensional reduction**, or **outlier detection**

now we want to discuss a **third ML paradigm**

Reinforcement Learning: given a complex task in a complex environment (dynamic, non deterministic, only partly accessible) train an **agent** that carry out **autonomous action** in this environment and complete the requested task

Agents in Reinforcement Learning

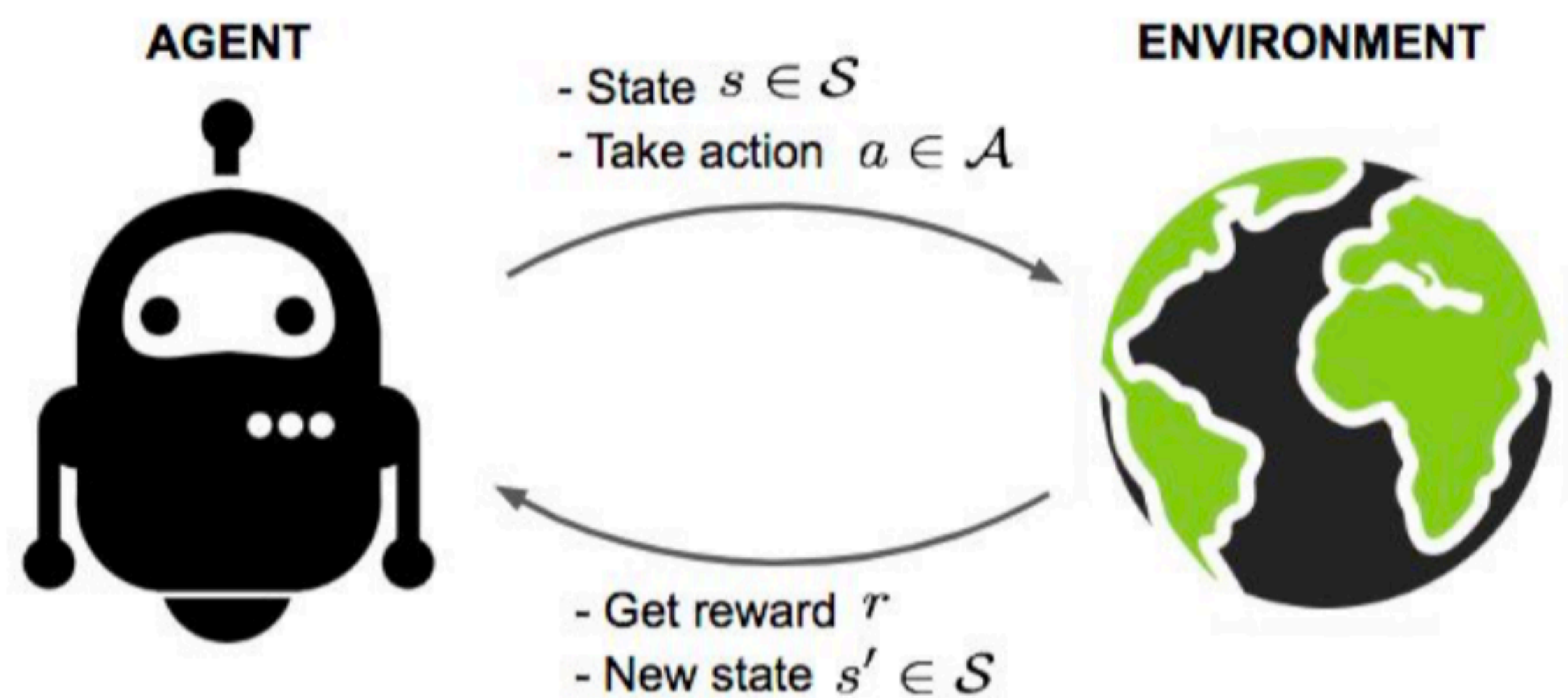
In the context of **Reinforcement Learning,** an **agent** is a computer system capable **autonomous action** in some environment, in order to achieve its delegated goals



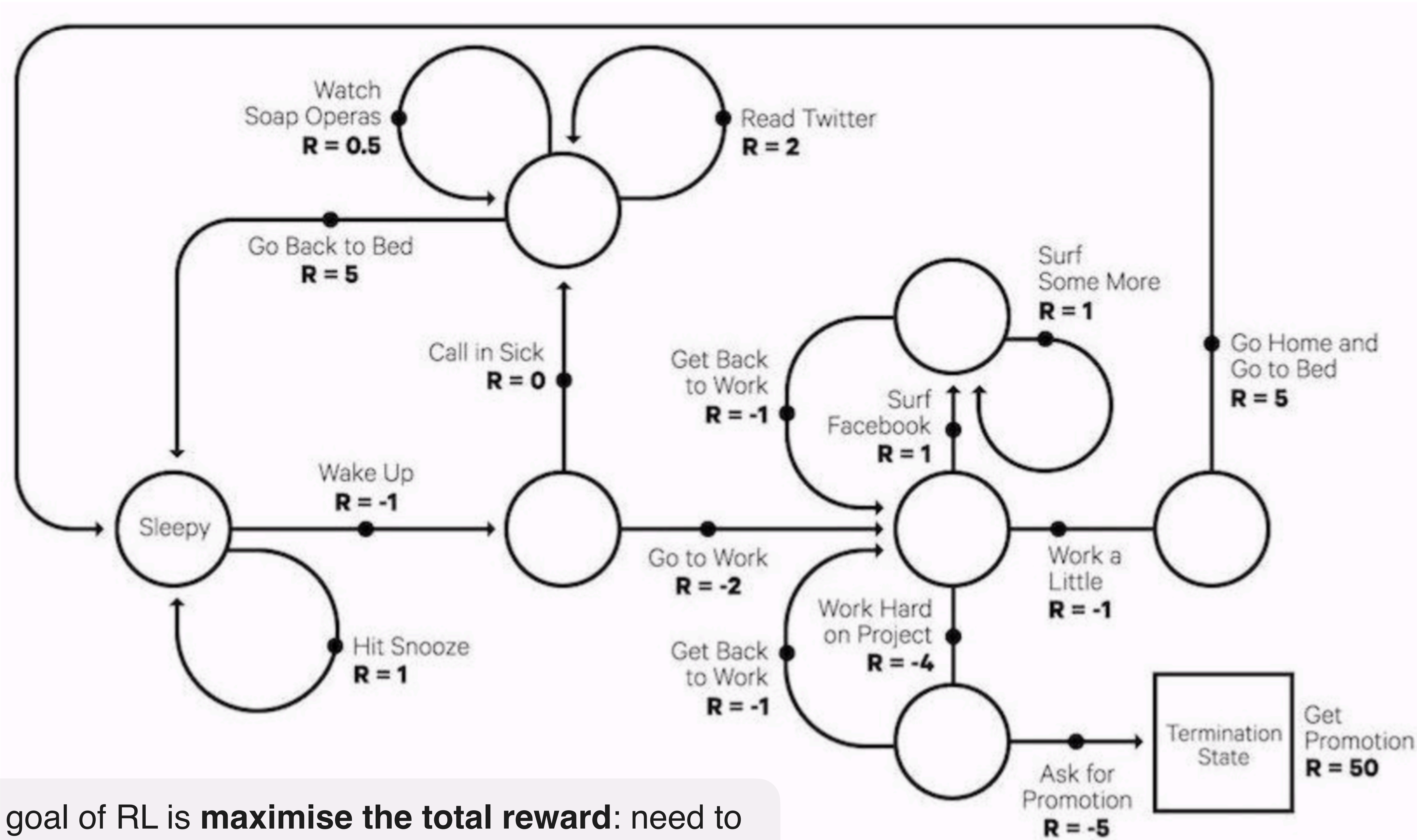
Agents in Reinforcement Learning

The ultimate goal of **Reinforcement Learning** is to

design an agent that **performs complex tasks** and **takes autonomous action** to fulfil its design goals, in an environment that is: partly inaccessible, non-deterministic, non-episodic, dynamic and continuous (*i.e.* the real world!).

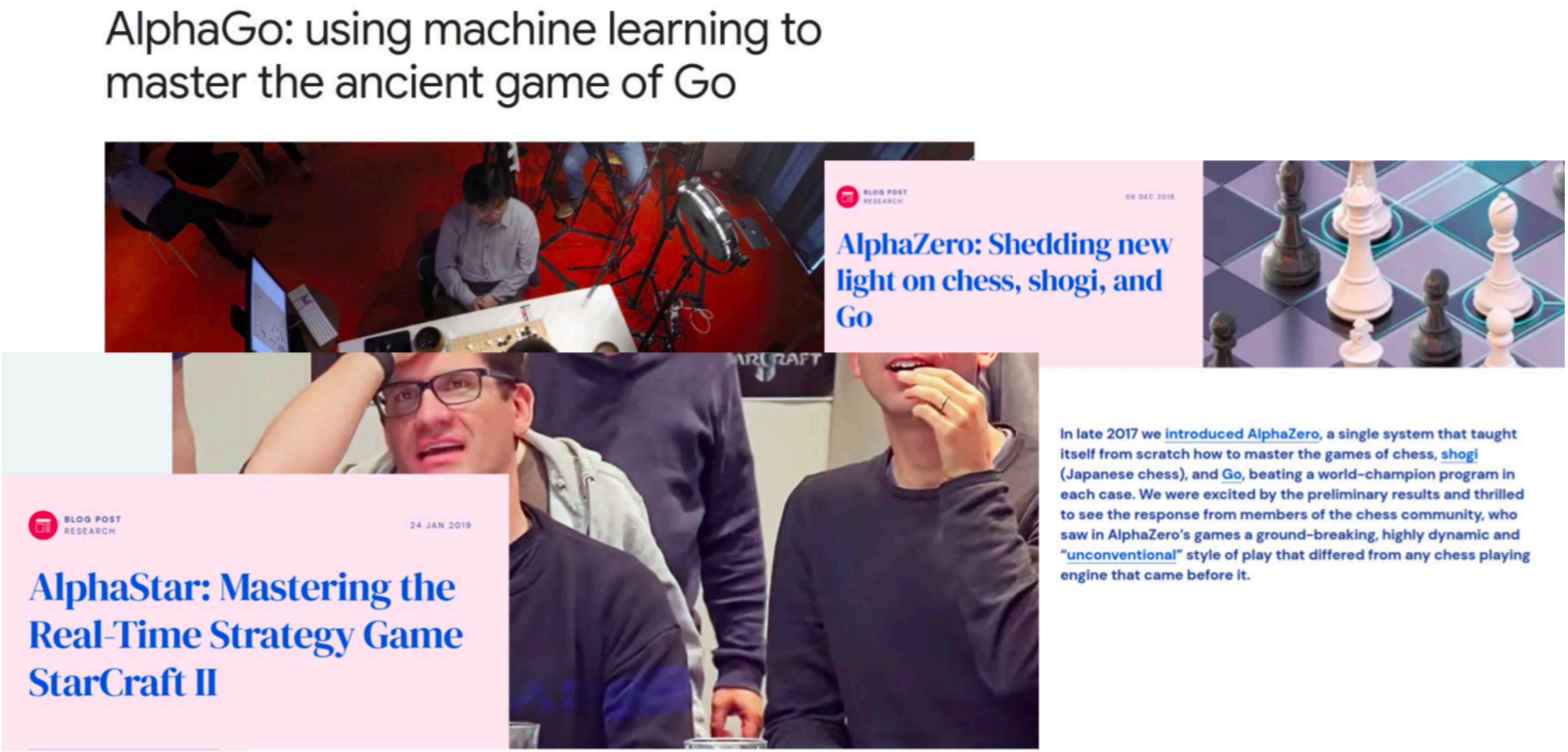


A Reinforcement Learning system



the goal of RL is **maximise the total reward**: need to explore all possible options to determine the best policy for each action that it might need to carry

Reinforcement Learning for Games



Reinforcement Learning for Games

Adversarial Learning and Generative Adversarial Networks

Generative adversarial networks

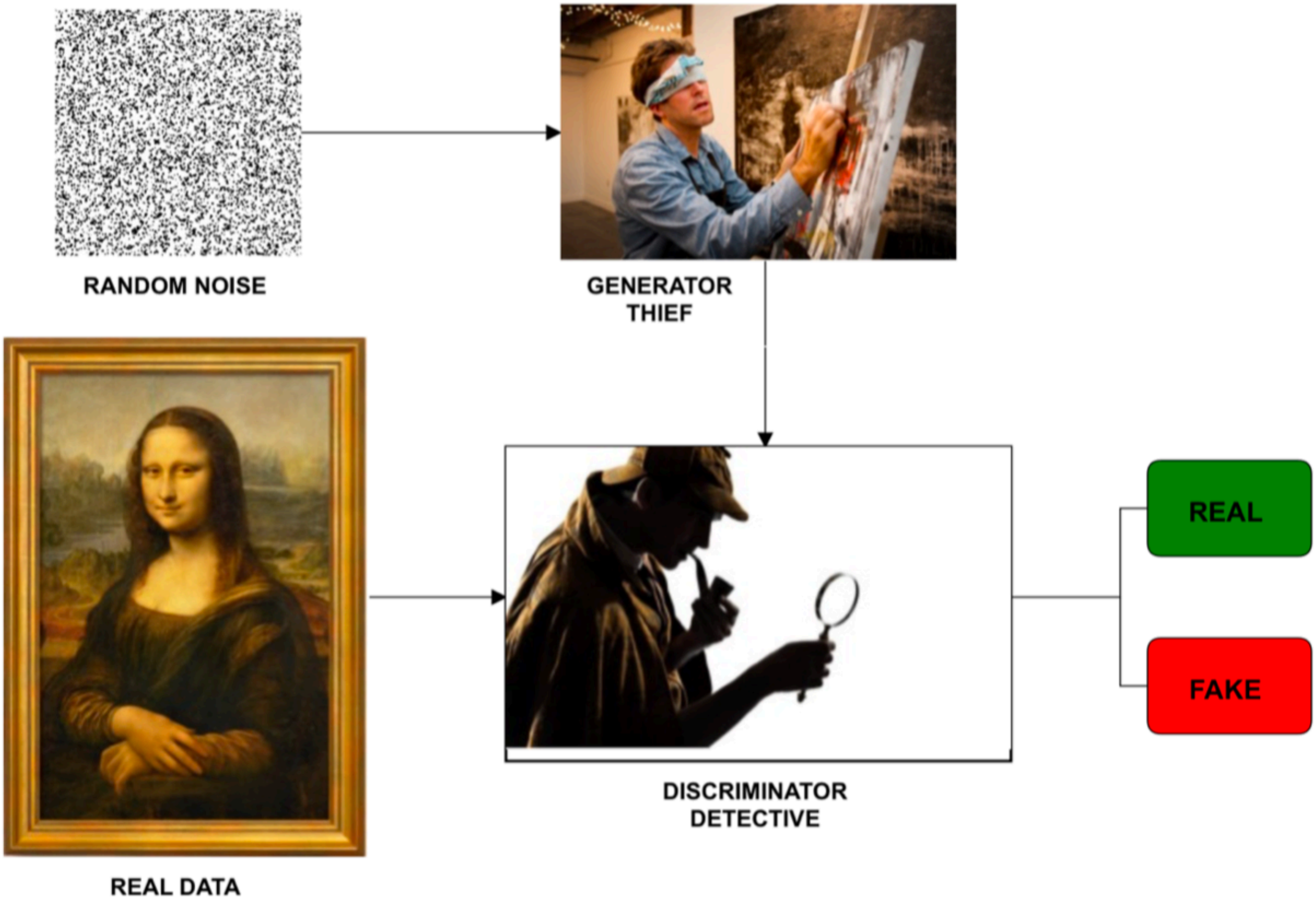
Generative Adversarial Networks (GANs) are deep neural network architectures, composed by two independent NNs which **compete against each other**

- ***(1)*** A **generator G** NN that creates (samples) pseudo-data by inferring the probability distribution associated to the training dataset

- ***(2)*** A **discriminator D** NN which determines the probability of a given sample arises from the actual training data rather than having been produced by **G**

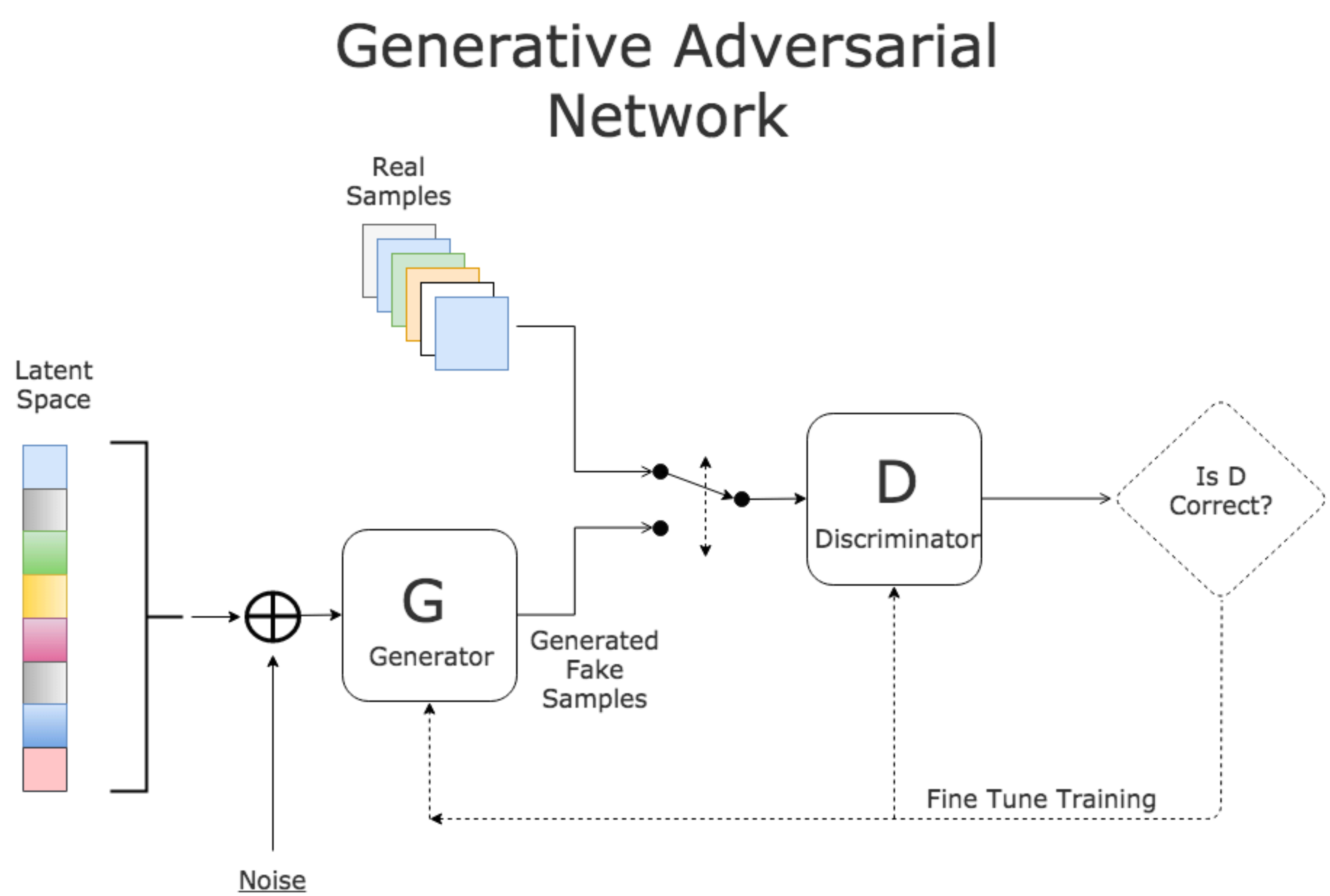
the generator network **G** should be trained to maximise the probability that the discriminator network **D** makes a mistake: that is, **G** should generate pseudo-data samples that are virtually **indistinguishable** from the actual data

Generative adversarial networks



Generative Adversarial Networks

- Architecture for an **unsupervised neural network training** (unlabelled samples)
- Based on two **independent nets** that work separately and act as **adversaries:**
    - the **Discriminator (D)** undergoes training and plays the role of classifier
    - the **Generator (G)** and is tasked to generate random samples that **resemble real samples** with a twist rendering them as fake samples.



GAN training

the generator and discriminator are sequentially trained and iterated until convergence is achieved, at this point **D** cannot tell apart the pseudo-data from **G** from the real data

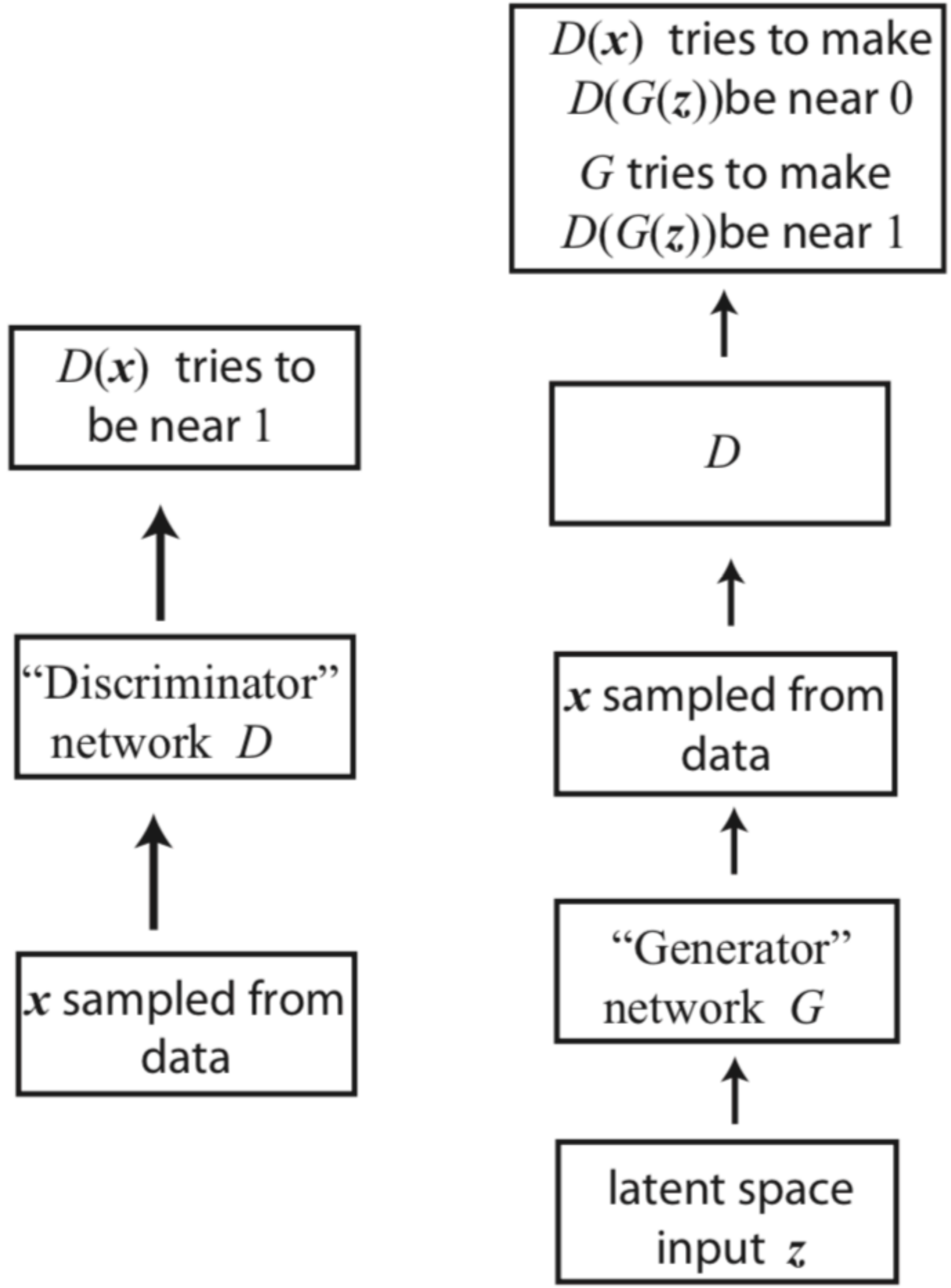


Image generation with GANs

https://thispersondoesnotexist.com/

Generative Adversarial Networks

Part II

ML for Electron Microscopy

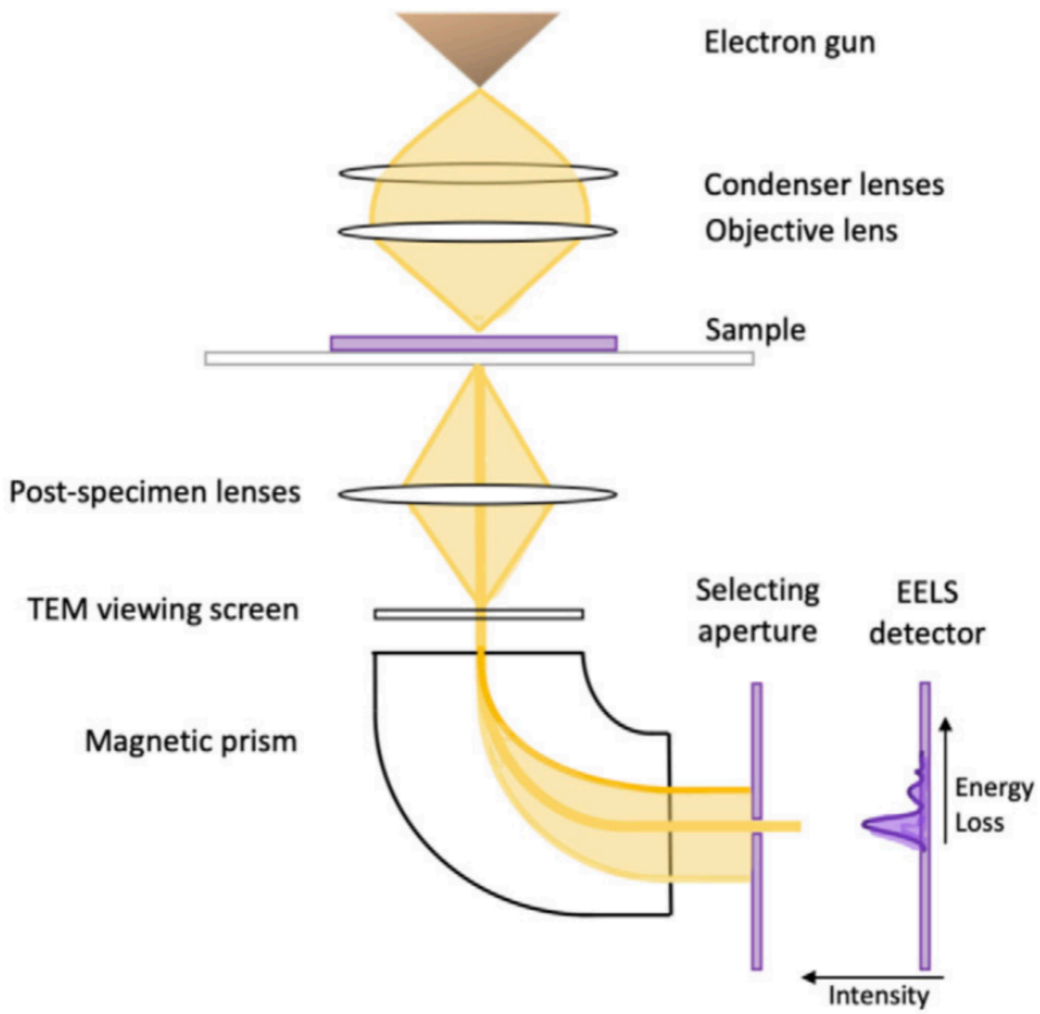
ML for background subtraction in EELS

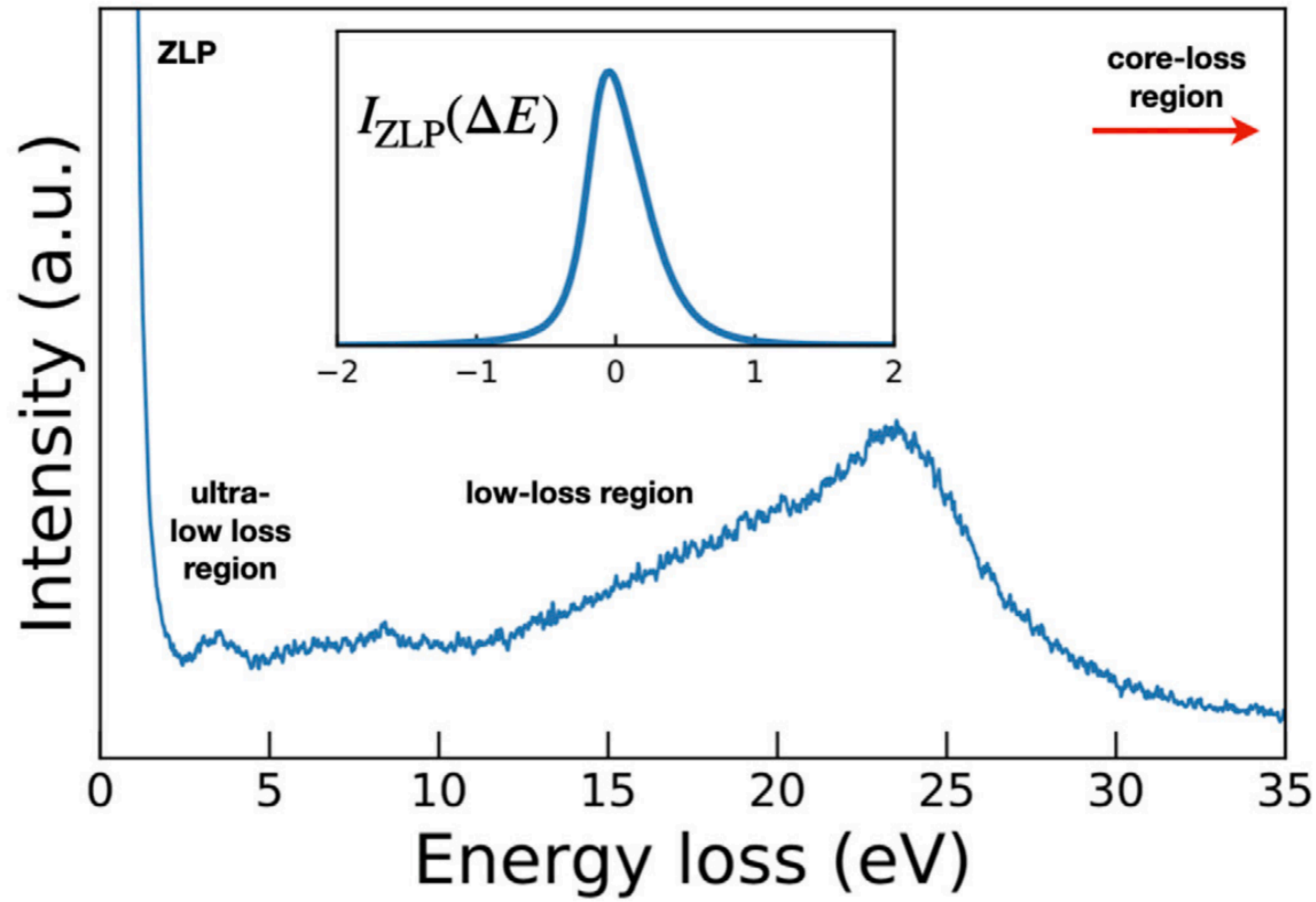
results based on:

- ☑ Roest, van Heijst, Maduro, Rojo, Conesa-Boj, *Ultramicroscopy* (2021)
- ☑ van Heijst, Mukai, Okunishi, Hashiguchi, Maduro, Roest, Rojo, Conesa-Boj, *Annalen der Physiek* (2021)
- ☑ Brokkelkamp, ter Hoeve, Brokkelkamp van Heijst, ter Hoeve, Maduro, Davydof, Kryluyk, Rojo, Conesa-Boj, *Journal of Physical Chemistry A* (2022)

Electron Energy Loss Spectroscopy

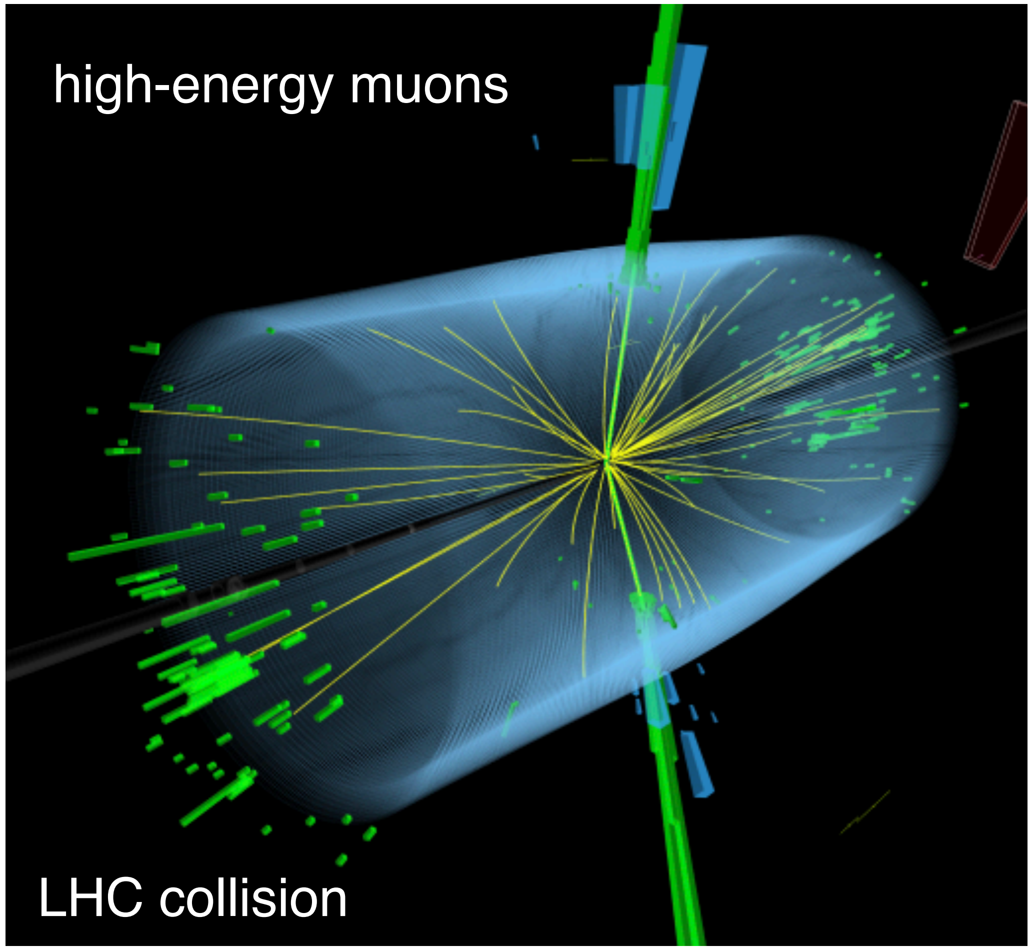
EELS: monitor **energy losses** suffered by the electrons from a Transmission Electron Microscope (TEM) beam upon **interaction with the sample**

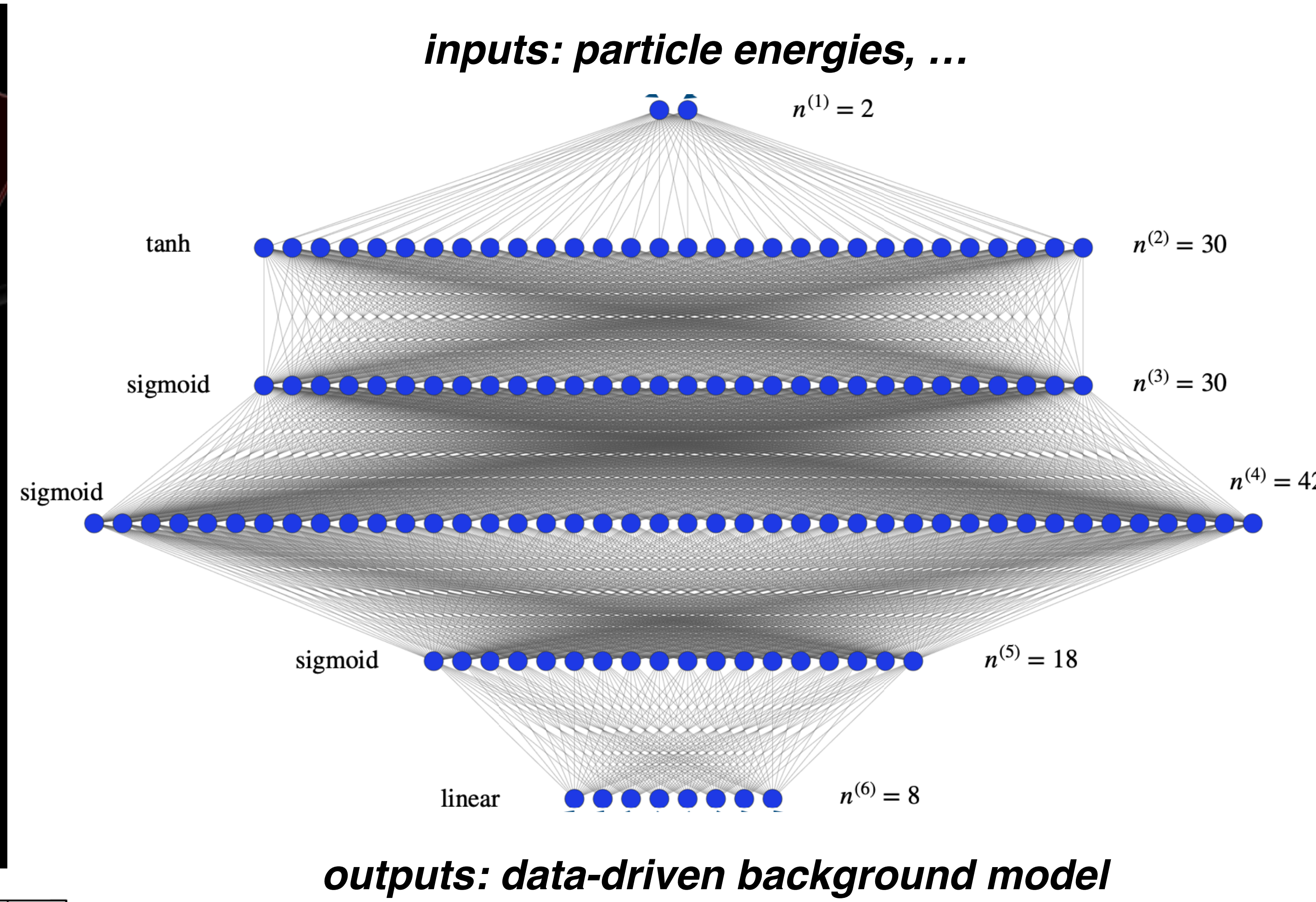


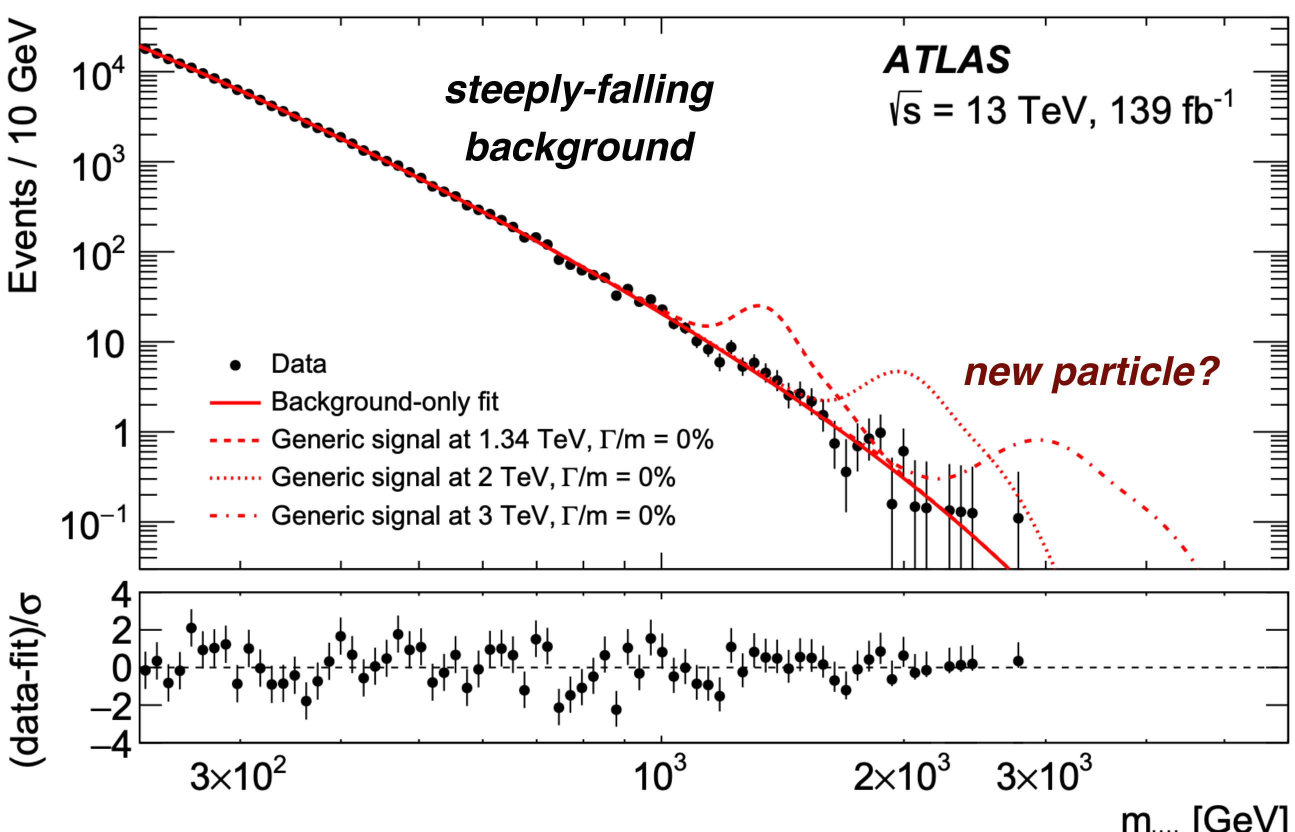


- **Challenge**: EELS measurements affected by **huge background** (Zero-Loss Peak) at low-energy losses from elastic scatterings: complicates interpretation of material properties!
- **Solution:** parametrise backgrounds from data using **Deep Neural Networks** and **Monte Carlo sampling** to remove them in a model-independent manner

ML-driven background subtraction in HEP

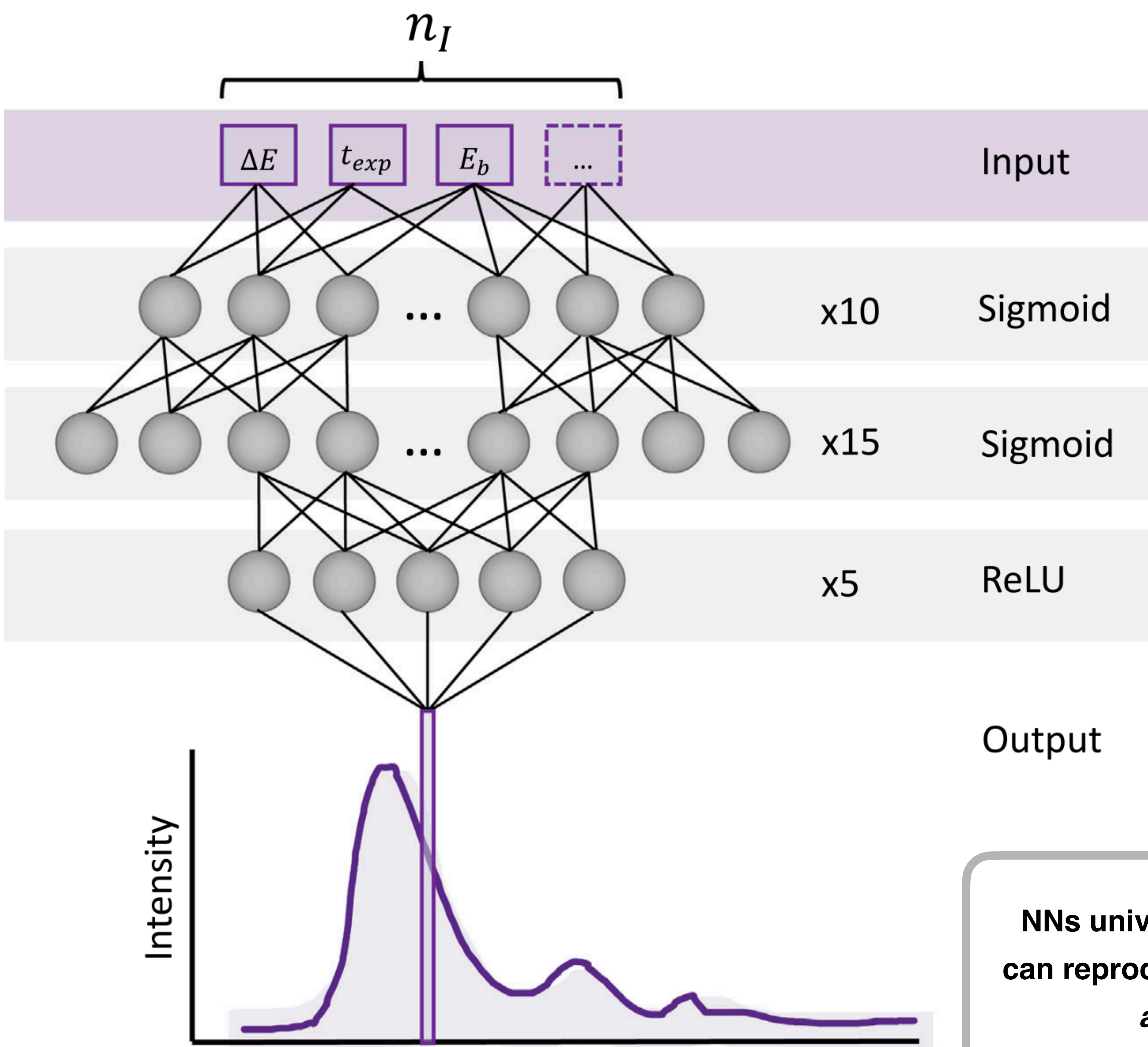






- **Learn from data** underlying physical laws and parametrise them with neural nets
- Estimate **model uncertainties** from Monte Carlo replica method: train a large number of models on *fake replicas* of actual data

A ML model for EELS backgrounds



energy loss, beam energy, exposure time, …

hidden layers

intensity of the Zero-Loss Peak background

NNs universal unbiased interpolants: can reproduce any physical law without *a priori* assumptions

The Monte Carlo replica method

- Generate Monte Carlo **replicas of the original data points** with multi-Gaussian distribution with central values and covariance matrices taken from the input measurements

$$I_{\mathrm{ZLP},i}^{(\mathrm{art})(k)} = I_{\mathrm{ZLP},i}^{(\mathrm{exp})} + r_i^{(\mathrm{stat},k)}\sigma_i^{(\mathrm{stat})} + \sum_{j=1}^{n_{\mathrm{sys}}} r_{i,j}^{(\mathrm{sys},k)}\sigma_{i,j}^{(\mathrm{sys})}\,, \quad \forall i\,, \quad k=1,\ldots,N_{\mathrm{rep}}\,,$$

- Train a NN model on each replica from the **minimisation of the log-likelihood**

$$E^{(k)}\left(\{\theta^{(k)}\}\right) = \frac{1}{n_{\mathrm{dat}}}\sum_{i=1}^{n_{\mathrm{dat}}}\left(\frac{I_{\mathrm{ZLP},i}^{(\mathrm{art})(k)} - I_{\mathrm{ZLP},i}^{(\mathrm{mod})}\left(\{\theta^{(k)}\}\right)}{\sigma_i^{(\mathrm{exp})}}\right)^2,$$

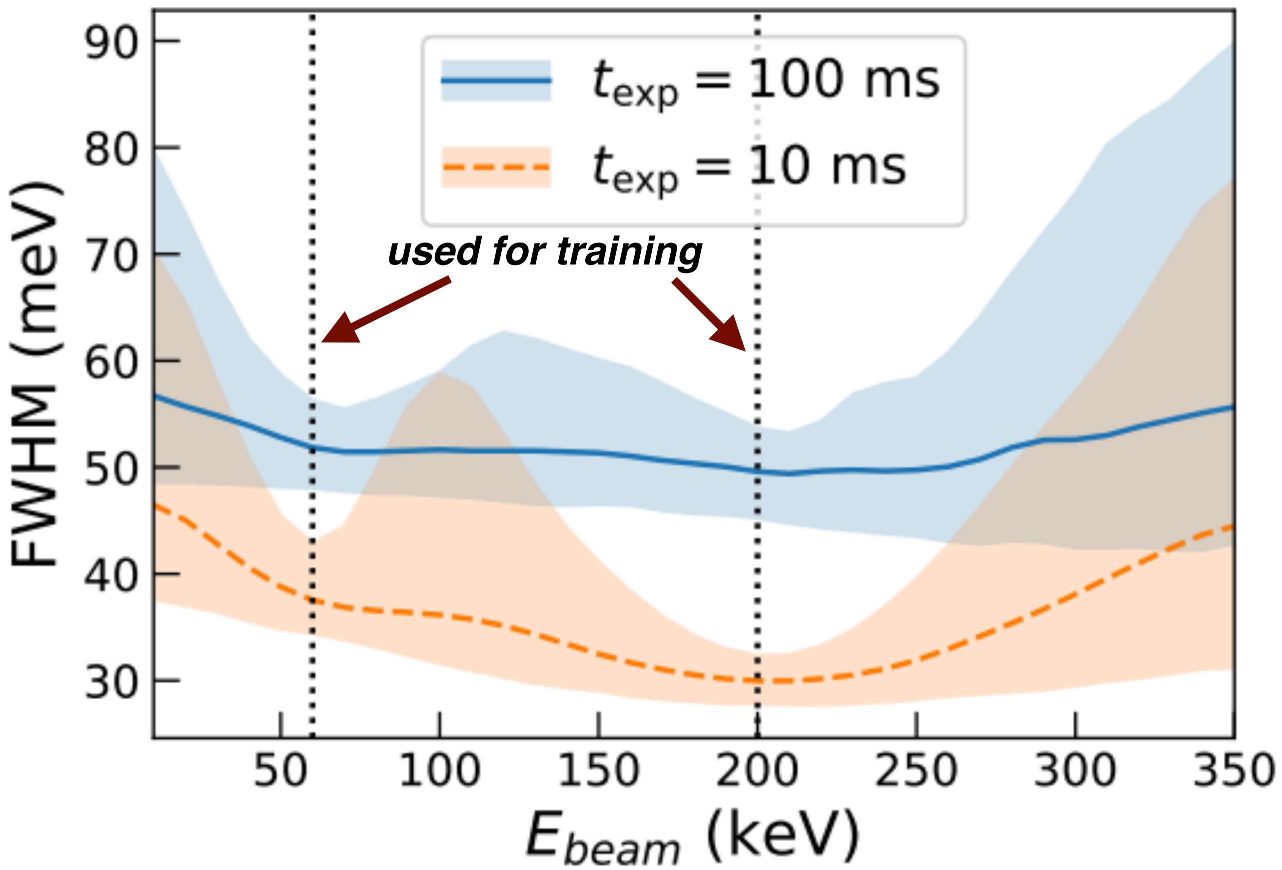
- We end up with a sampling of the **probability density in the space of NN models,** from which we can compute e.g. the variance of the predicted ZLP intensity for arbitrary inputs

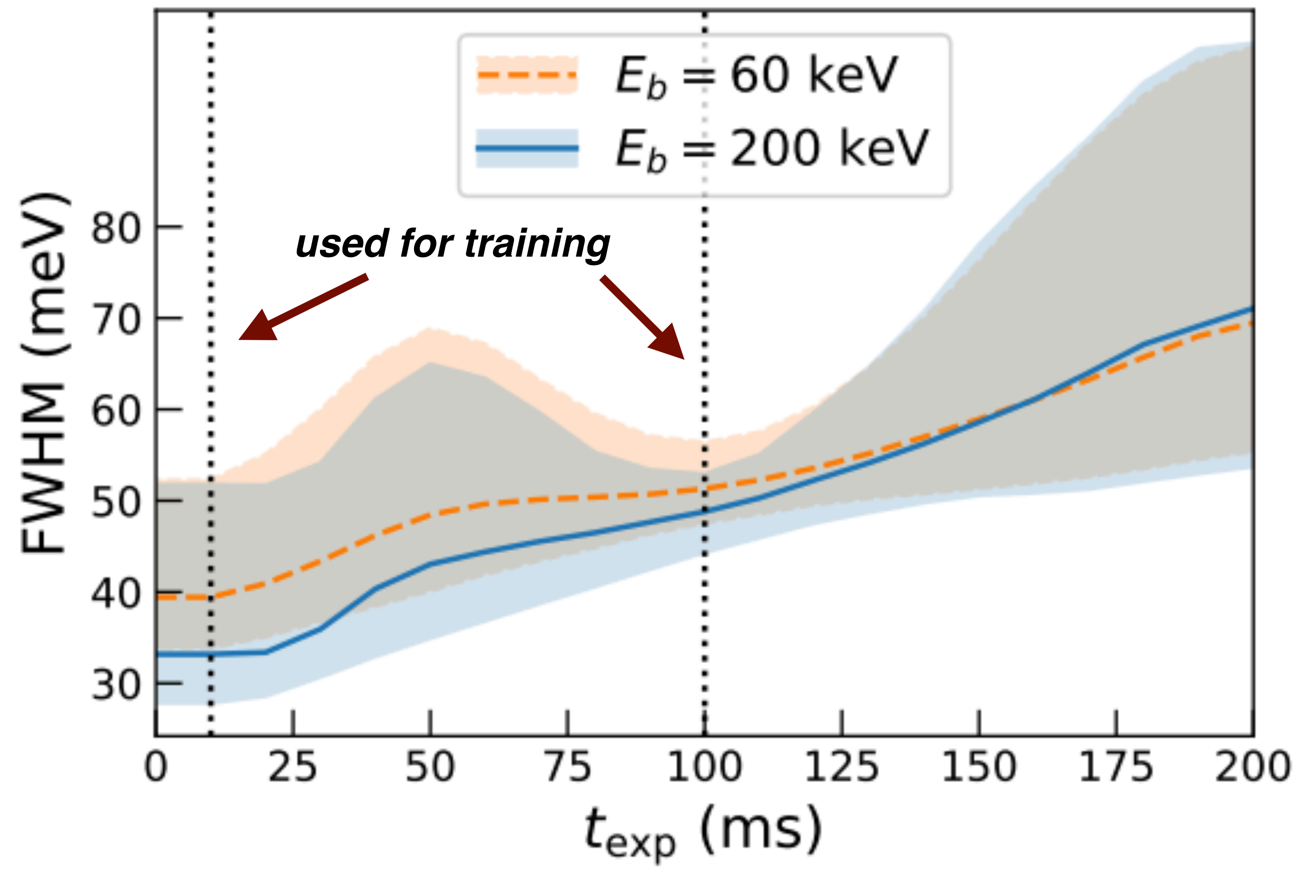
$$\sigma_{I_{\mathrm{ZLP}}}^{(\mathrm{mod})}(\{z_1\}) = \left(\frac{1}{N_{\mathrm{rep}}-1}\sum_{k=1}^{N_{\mathrm{rep}}}\left(I_{\mathrm{ZLP}}^{(\mathrm{mod})(\mathrm{k})} - \left\langle I_{\mathrm{ZLP}}^{(\mathrm{mod})}\right\rangle\right)\right)^{1/2}$$

state of the art for error propagation in deep-learning models

Extrapolation to new TEM operation conditions

- Key property of ML: **prediction** to different ranges of input parameters

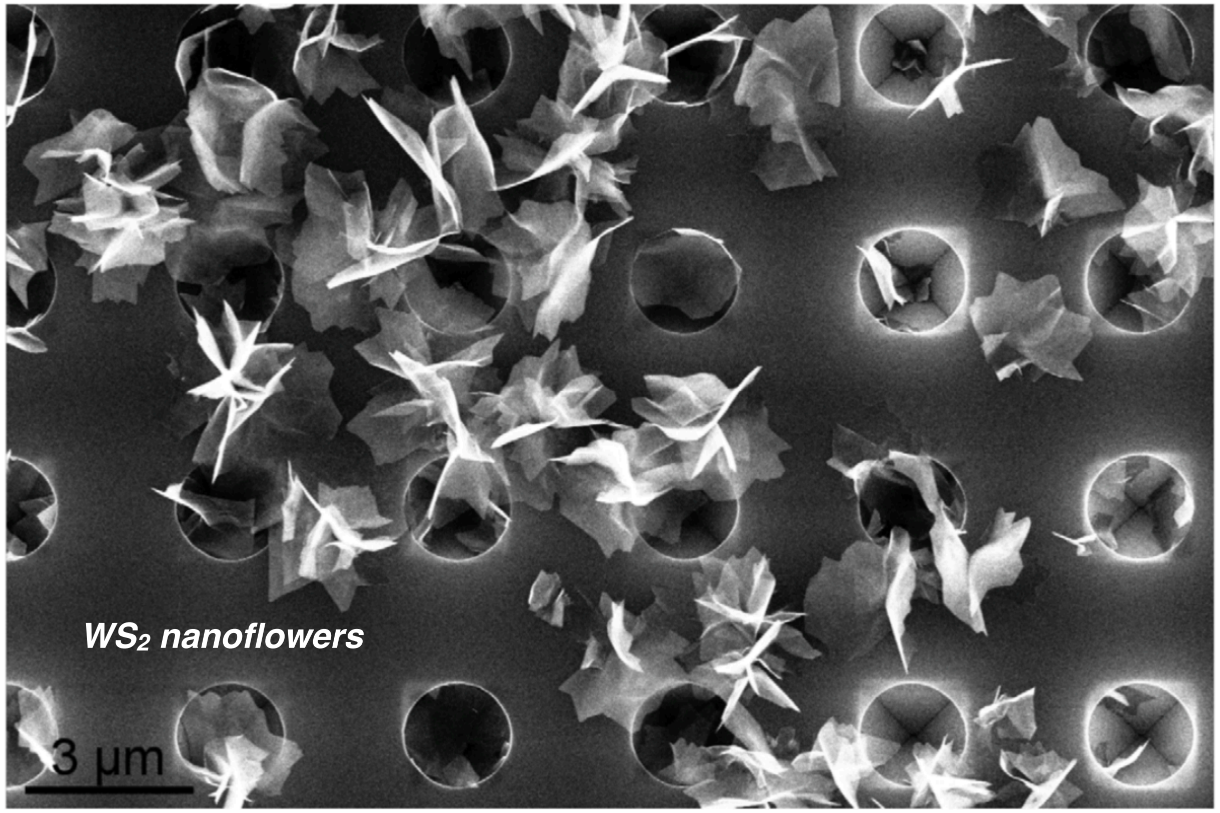


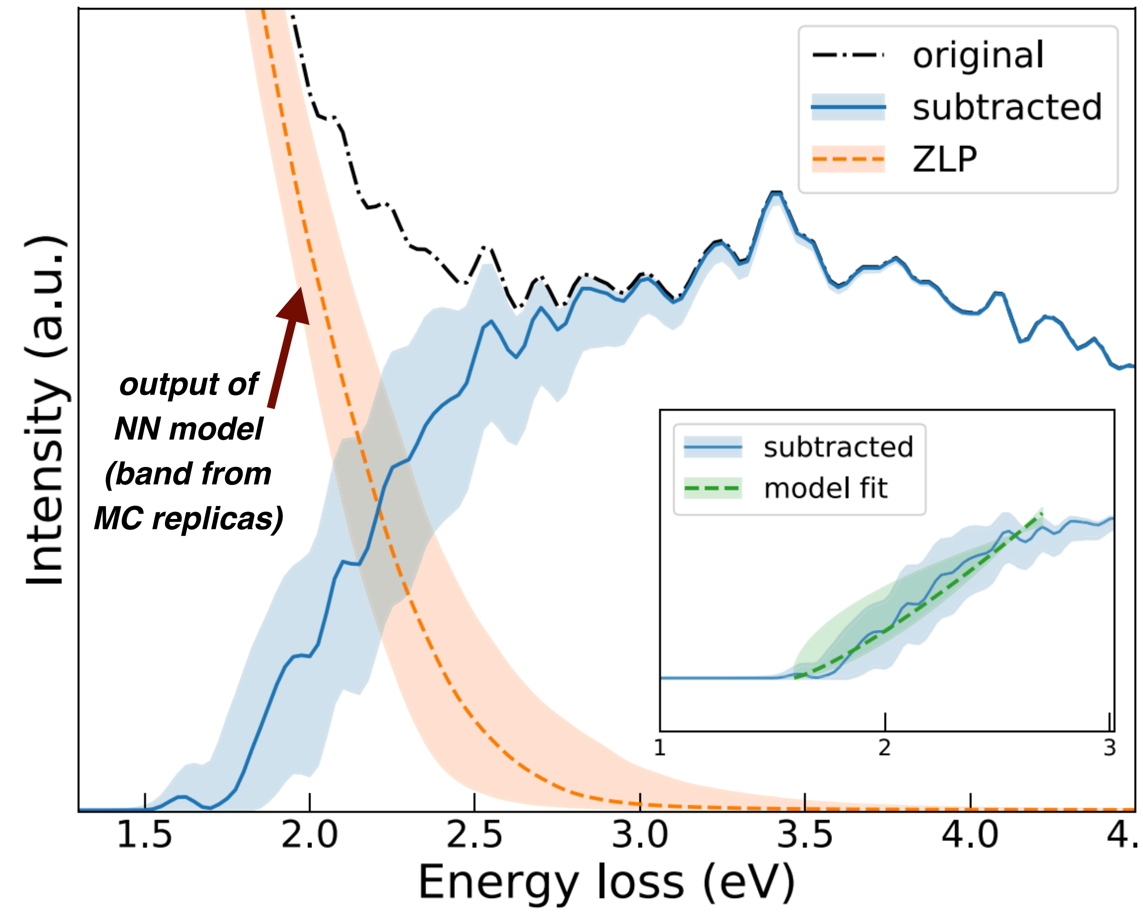


- Train ZLP model for specific values of TEM operation parameters, e.g. **electron beam energy** and then inter/extrapolate outside training range
- The model **uncertainties increase** when our prediction is not reliable and more data needed
- Important: **no assumptions** of functional dependence of background model with input variables

e.g. approx linear dependence with t_{exp} has been learned from data!

Band gap extraction in polytypic WS_2





- ☑ Apply to **nanoflowers** composed by 2H/3R **polytypic WS_2**

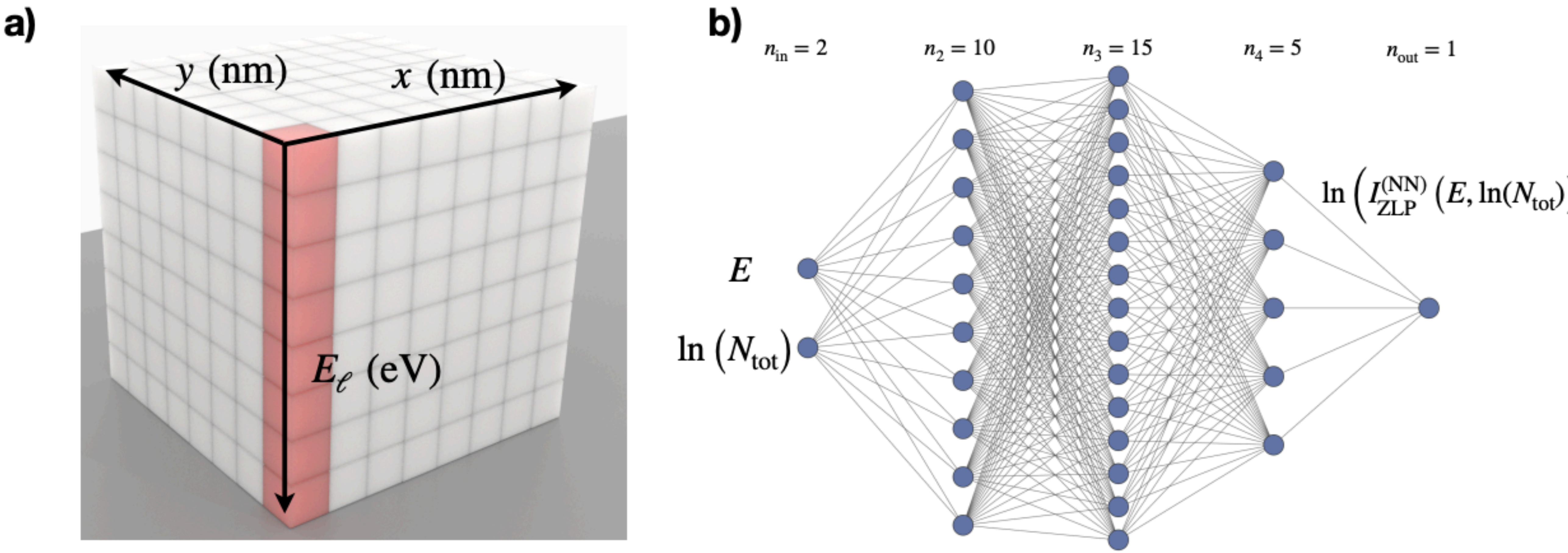
- ☑ First extraction of band gap in this material from fit to subtracted EEL spectra

$$I_{\rm inel}(\Delta E) \simeq A\,(\Delta E - E_{\rm BG})^b \qquad E_{\rm BG} = 1.6^{+0.3}_{-0.2}\,{\rm eV}\,, \qquad b = 1.3^{+0.3}_{-0.7}\,.$$

- ☑ ML-subtracted spectra make possible mapping **exciton transitions** down to 1.5 eV

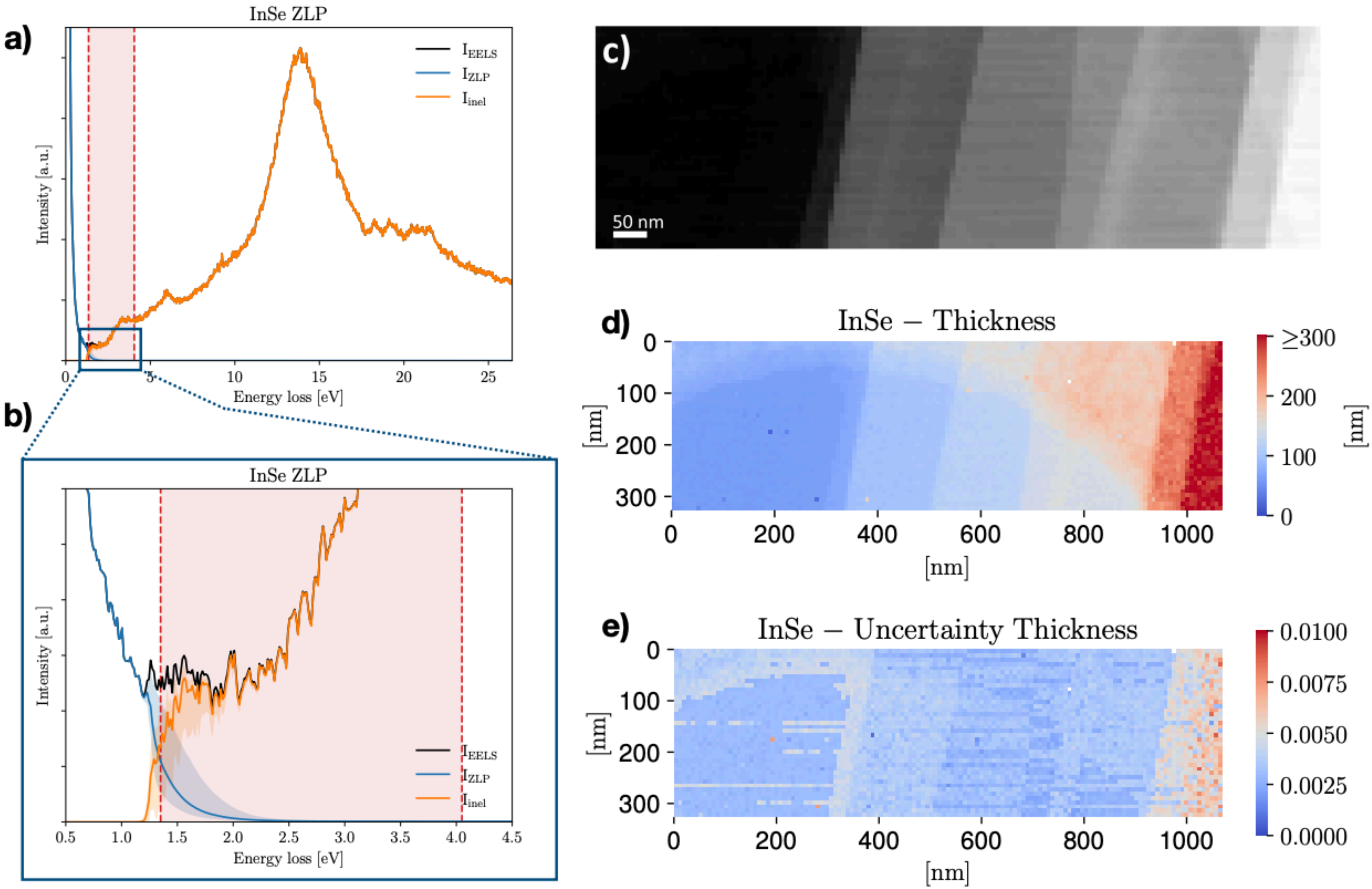
consistent with ab-initio DFT calculations (L. Maduro et al @ TUD)

ML analysis of spectral images

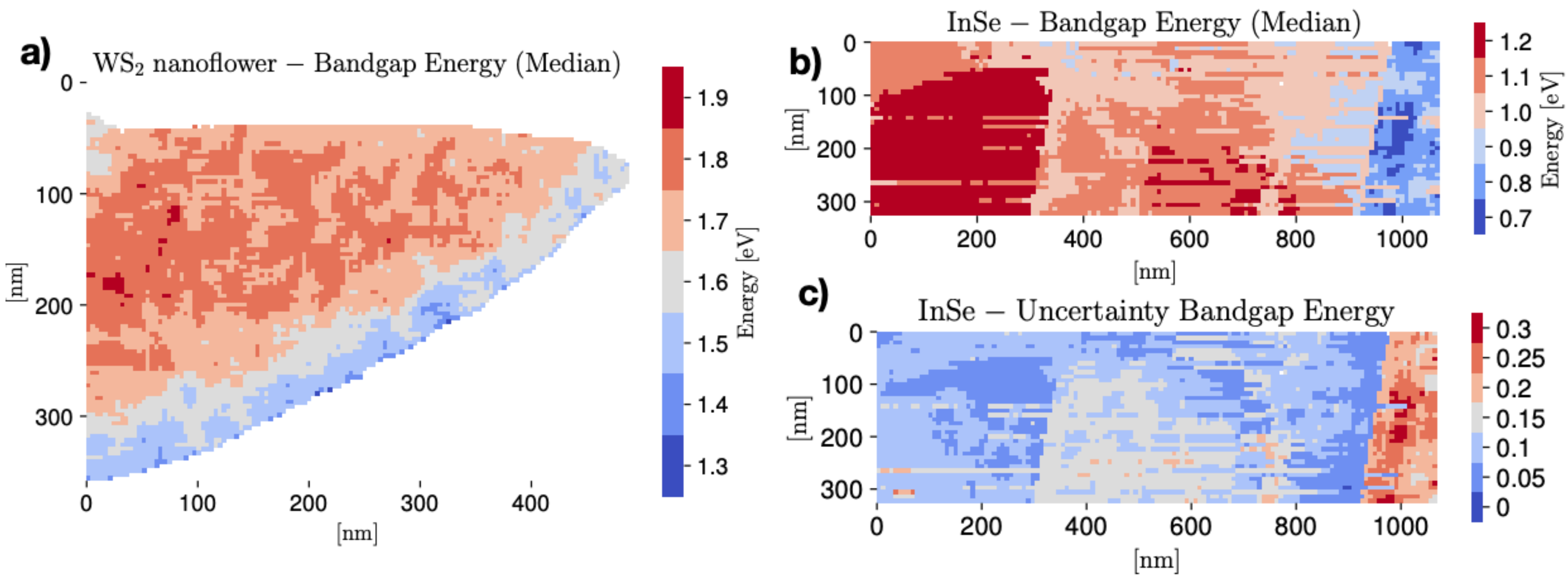


- EELS spectral image contains up to **O(10^5) individual spectra**
- Use **unsupervised learning** (*K-means clustering*) to identify clusters of pixels with comparable sample thickness and combine them for the **(supervised) NN training**
- Simultaneous determination of physical properties across the **whole nanostructure** with their **uncertainties:** thickness, band gap, position and width of plasmonic and excitonic resonances,…

ML analysis of spectral images



ML analysis of spectral images



- ☑ EELS spectral image contains up to **O(10^5) individual spectra**
- ☑ Use **unsupervised learning** (*K-means clustering*) to identify clusters of pixels with comparable sample thickness and combine them for the **(supervised) NN training**
- ☑ Simultaneous determination of physical properties across the **whole nanostructure** with their **uncertainties:** thickness, band gap, position and width of plasmonic and excitonic resonances,…

Other ML application to Electron Microscopy

Automated defect identification

- **Structural defects** and **impurities** in the crystalline lattice of materials can affect properties and performance, for example, degrading or enhancing electric transport.

- Defect identification and classification in TEM measurements usually done ``by eye'', **inspecting EM images** one by one.

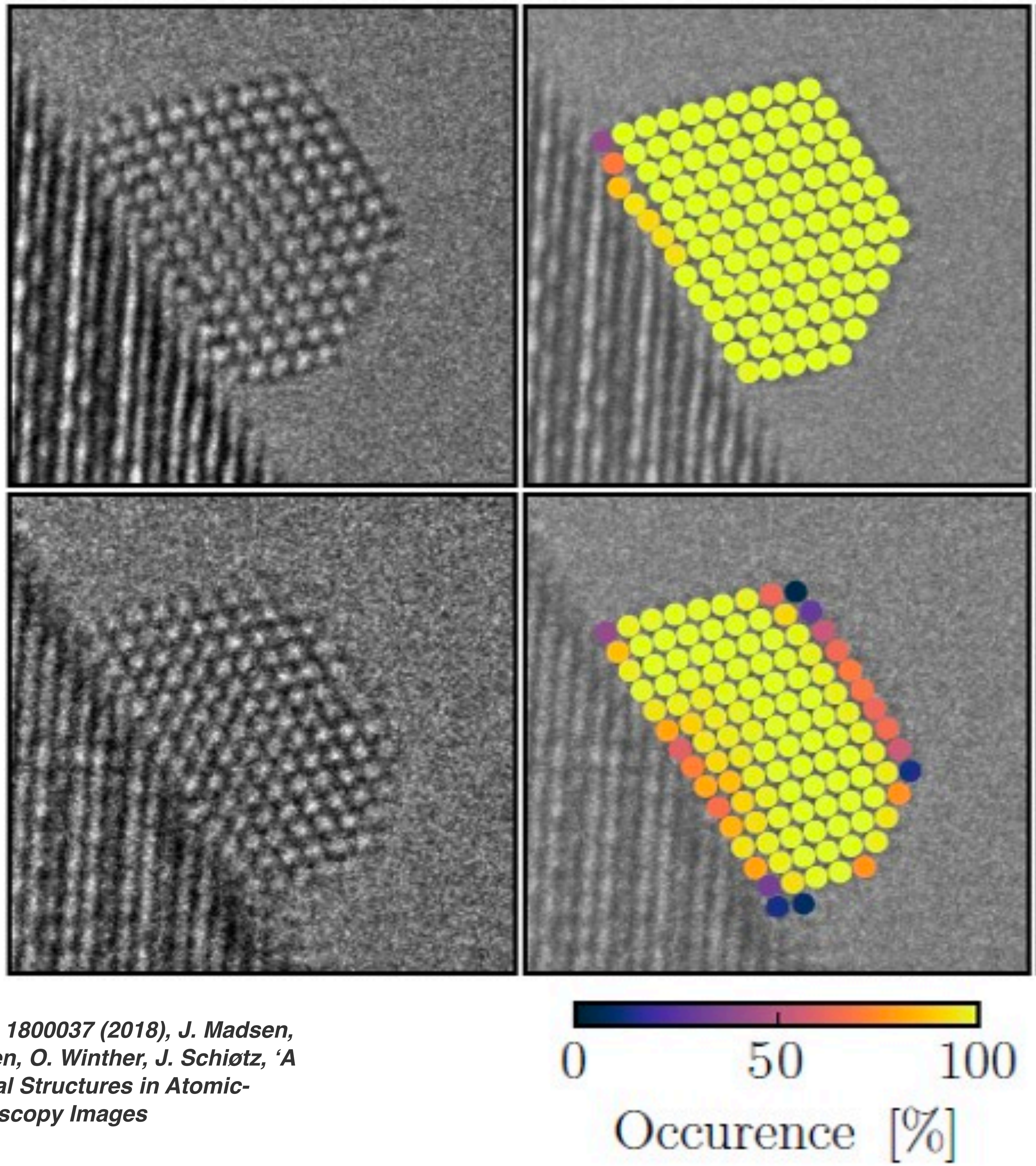
- ML can be used to formulate strategies that make possible automatically identifying defects in TEM images, for example using **Convolutional Neural Networks**

- A key ingredient is the **training dataset,** which can be composed by either labelled TEM images or simulated data using theory calculations

- Carefully monitoring the performance is crucial to demonstrate that defect identification is robust and efficient

Automated defect identification

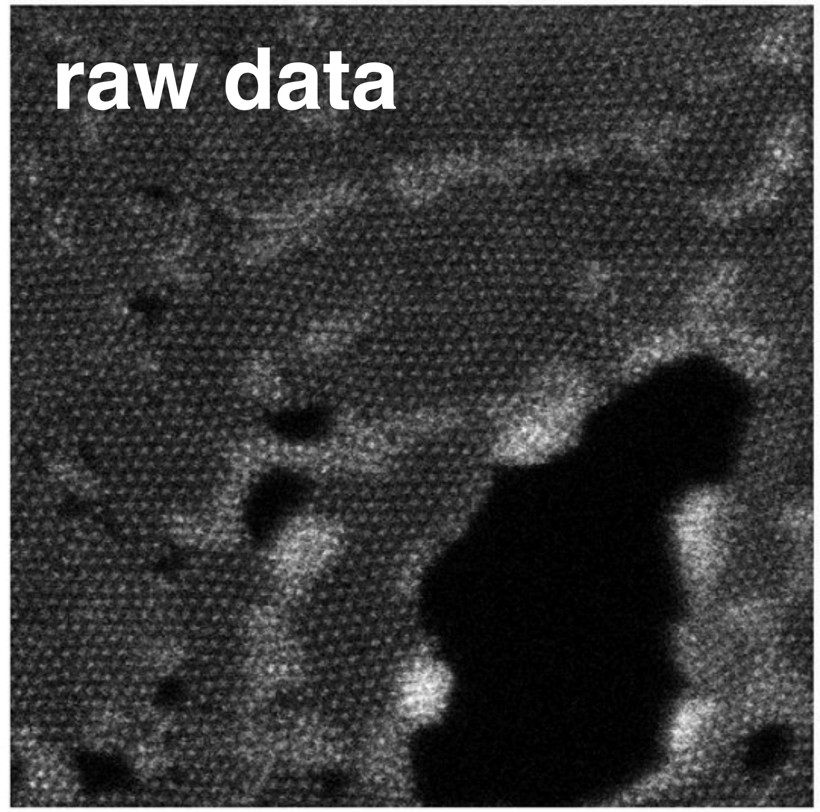
- A **CNN** is used to identify the atomic columns in the bulk of a nanoparticle.
- At the surface of the particle, the atomic columns continuously shift around giving new surface configurations
- The CNN predicts which **local structures at the atomic level** are more likely to appear



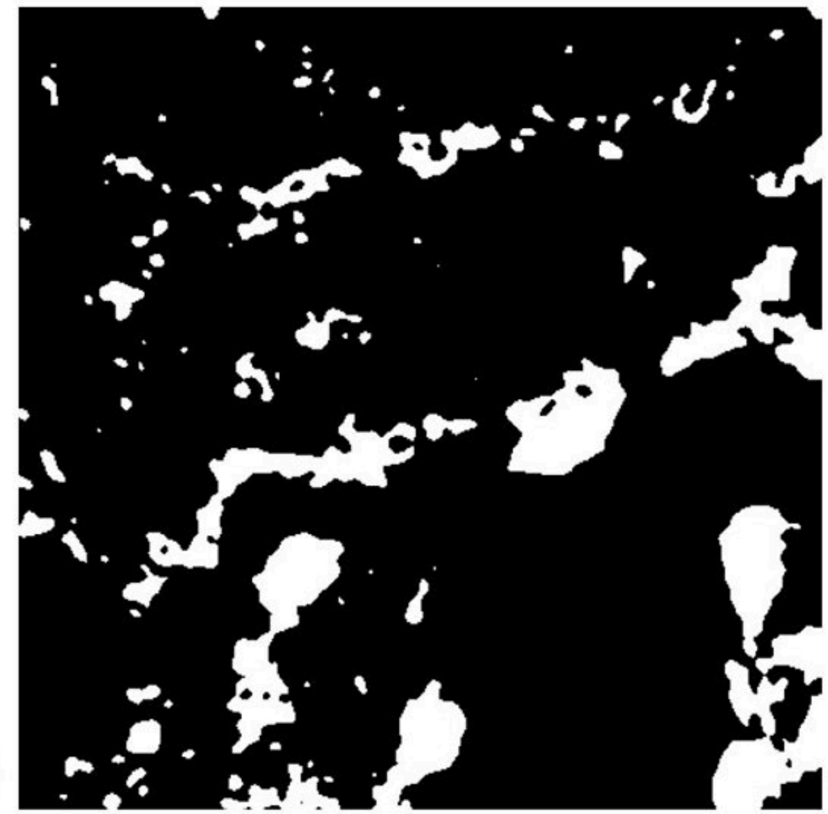
Advanced Theory and Simulations 2018, 1800037 (2018), J. Madsen, P. Liu, J. Kling, J. B. Wagner, T. W. Hansen, O. Winther, J. Schiøtz, 'A Deep Learning Approach to Identify Local Structures in Atomic-Resolution Transmission Electron Microscopy Images

Automated defect identification

a)



b)

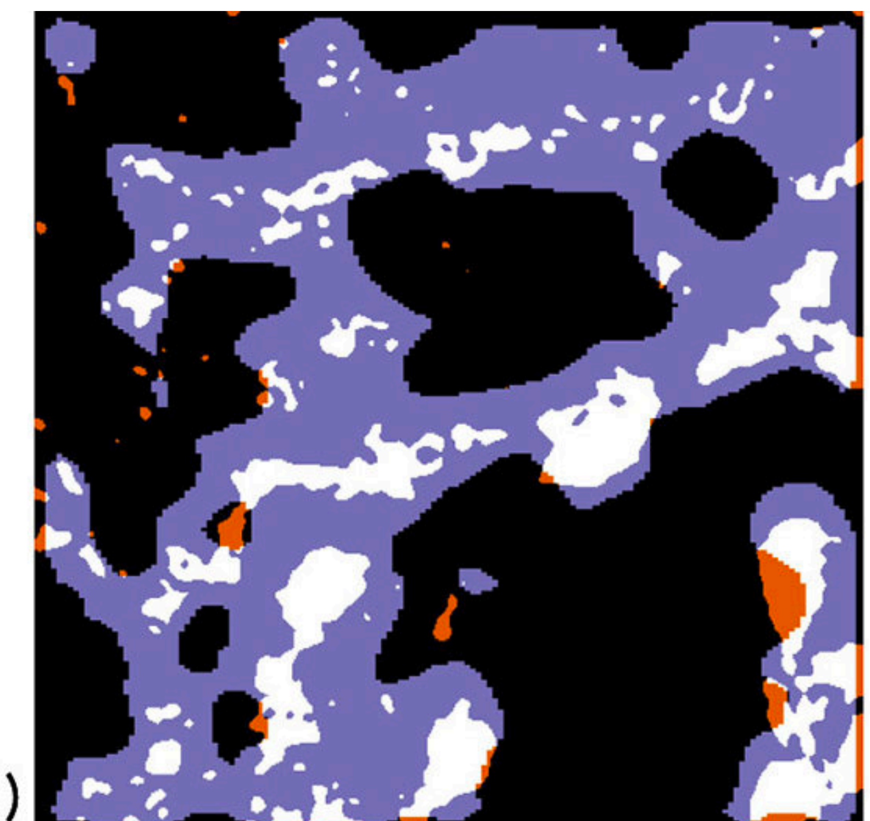


human classification

- ☑ Multi-node **Evolutionary** Neural Networks for Deep Learning (MENNDL): design an optimal deep learning network to extract structural information from raw atomic-resolution TEM data

- ☑ Achieves within a few hours performance comparable with that of human expert inspection

R.M. Patton, J.T. Johnston, S.R. Young, et al., "167-PFlops deep learning for electron microscopy: From learning physics to atomic manipulation." SC'18: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX (2018).



white: defects

orange: false negatives

purple: false positives

ML-based classification

Genetic Algorithms

GAs combine **stochastic and deterministic ingredients** to explore the model parameter space and minimise the cost function

Initial population (random sampling)

$$\boldsymbol{\theta}_1^{(0)}, \boldsymbol{\theta}_2^{(0)}, \boldsymbol{\theta}_3^{(0)}, \ldots$$

Mutations and cross-over

Selection

Evaluate cost function + Sorting

$$C_i\left(\boldsymbol{Y}; f(\boldsymbol{X}; \boldsymbol{\theta}_i^{(t)})\right)$$

$$C_1 \leq C_2 \leq C_3 \ \ldots$$

Convergence criterion

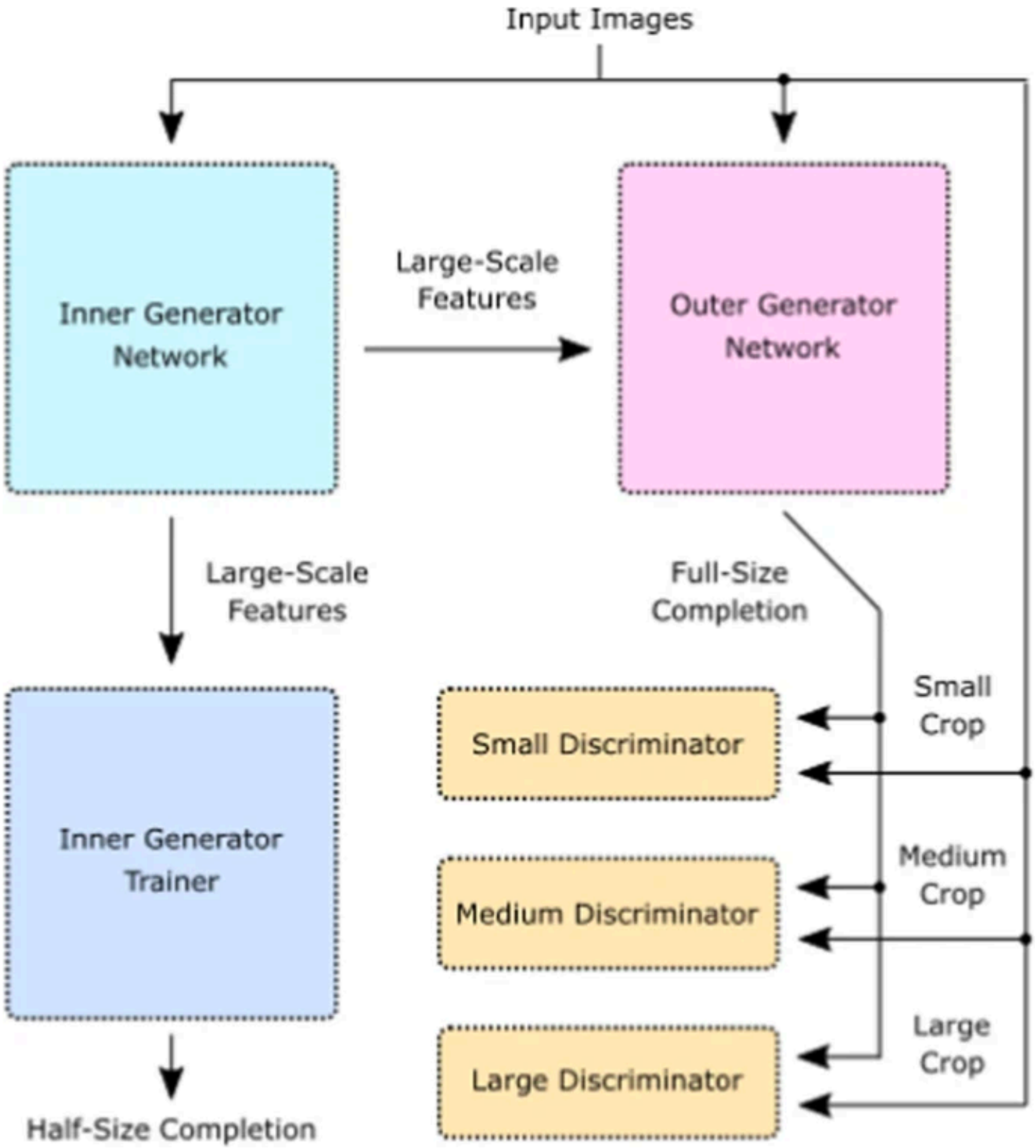
e.g. $C_1 \leq C_{\rm thres}$

No: new generation

Yes: done

Model parameters $\hat{\boldsymbol{\theta}}$

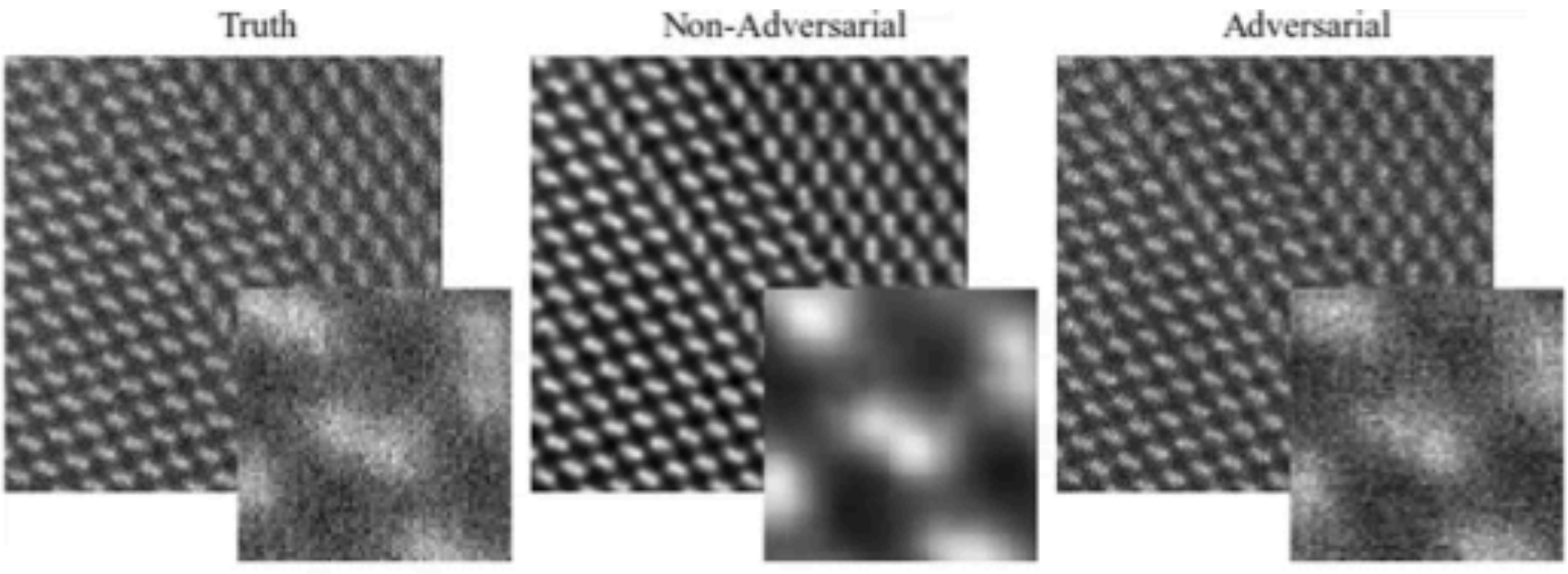
advantages of GA; only local values of cost function required (**no gradients**), built-in stochasticity, easier to avoid local minima



Partial Scanning Transmission Electron Microscopy with Deep Learning, Jeffrey M. Ede & Richard Beanland , Scientific Reports, 10, Article number: 8332 (2020)

Realistic images from partial scans

- GANs ML architectures can be exploited as **compressed sensing algorithms**
- They allow **decreasing electron microscope scan time** and electron beam exposure
- with at the same time ensuring with minimal information loss.



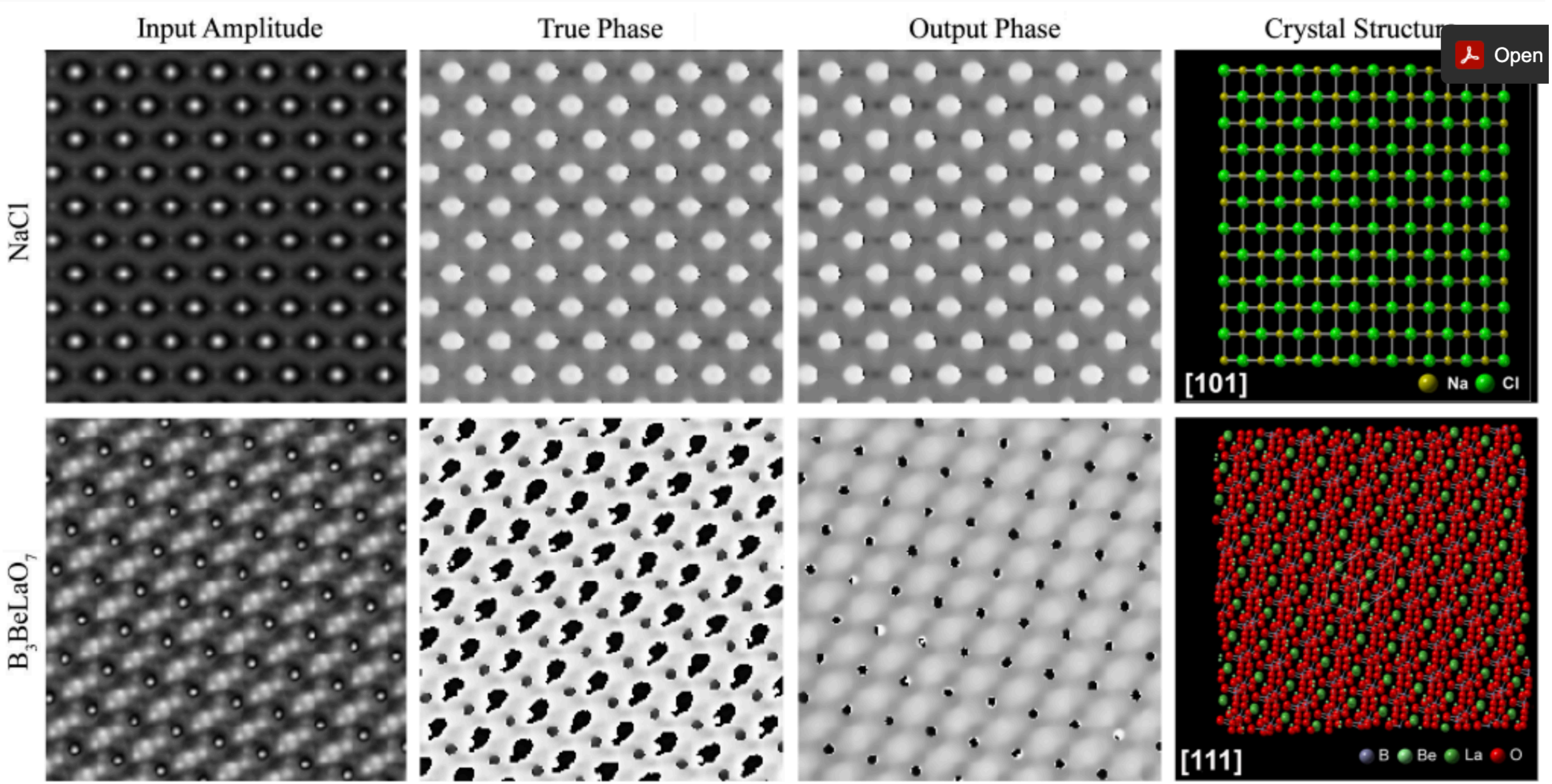
Goal: reconstruct "Truth" (based on full scan) from partial scan. Much better performance using GANs

Complete realistic images from **partial scans** using GANs

Feature prediction

☑ Conventional transmission electron microscopy (CTEM) records **only the intensity**, not the phase, of an image.

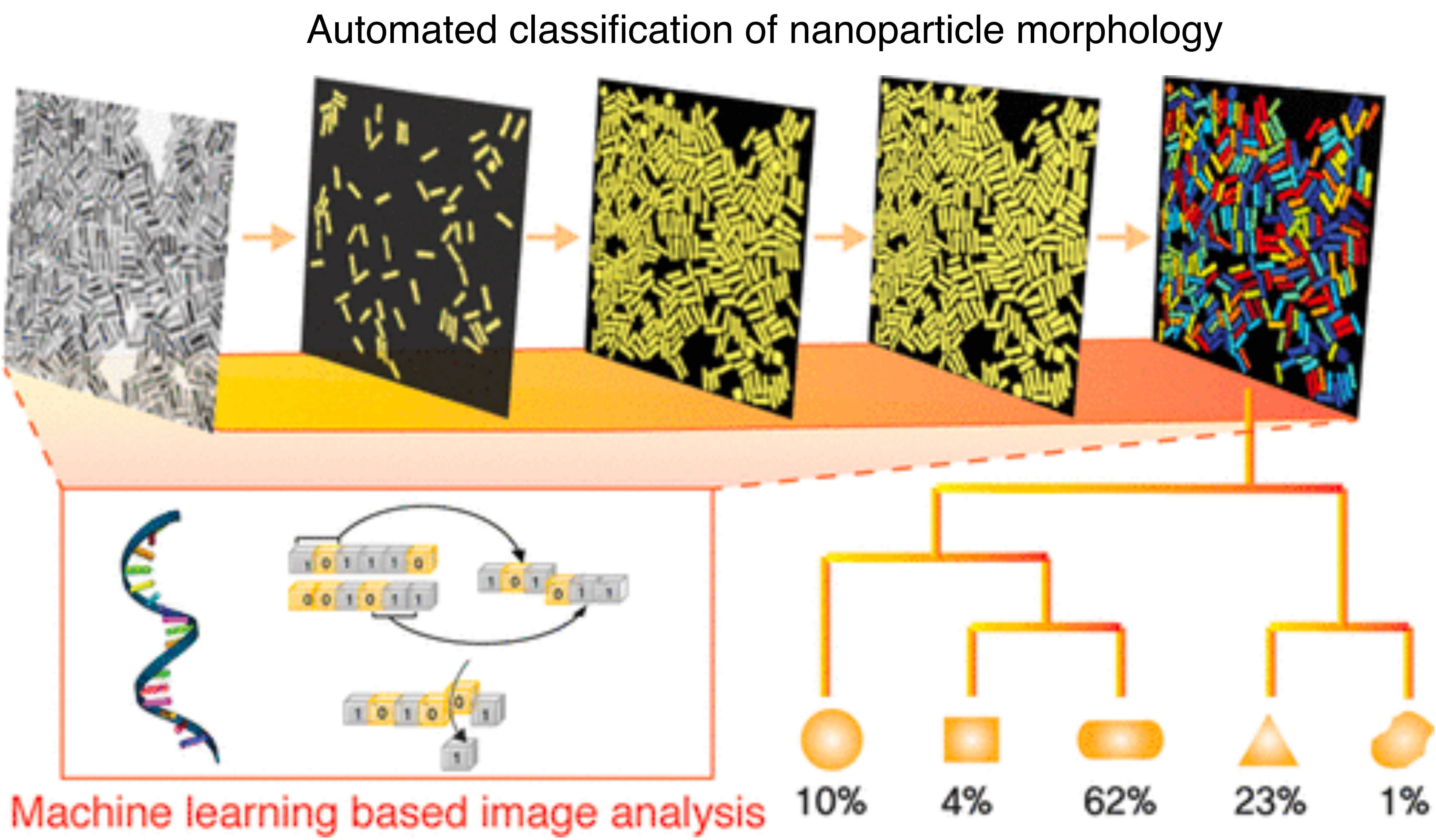
☑ ML can **recover phases from CTEM intensities** for new datasets after training



Ede et al, arXiv:2001.10938v2 [eess.IV]

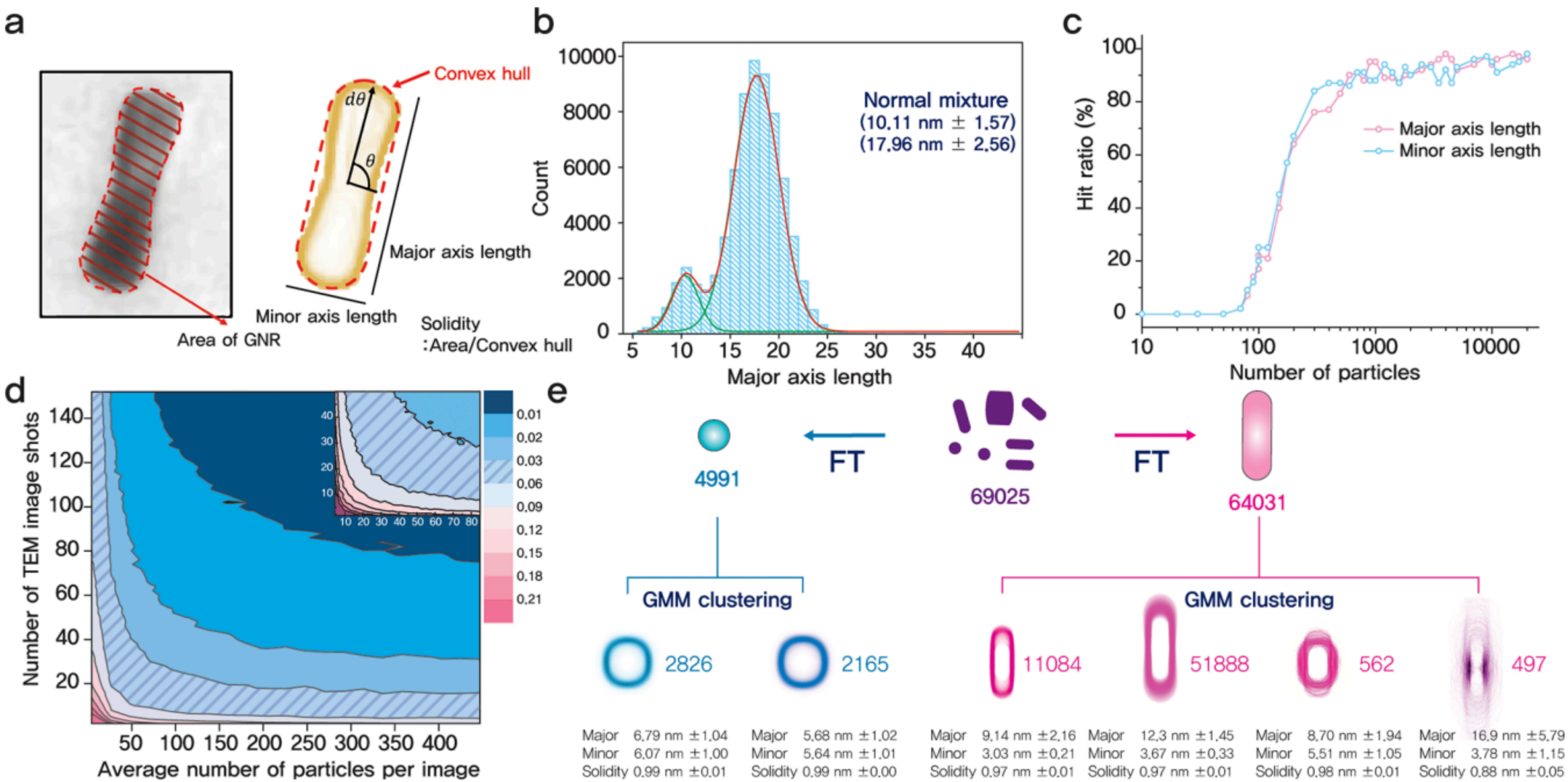
Feature prediction

https://pubs.acs.org/doi/abs/10.1021/acsnano.0c06809



Lee at al, ACS Nano 2020, 14, 12, 17125

Genetic Algorithms combined with Gaussian Mixture Models for classification and categorisation of nanoparticles

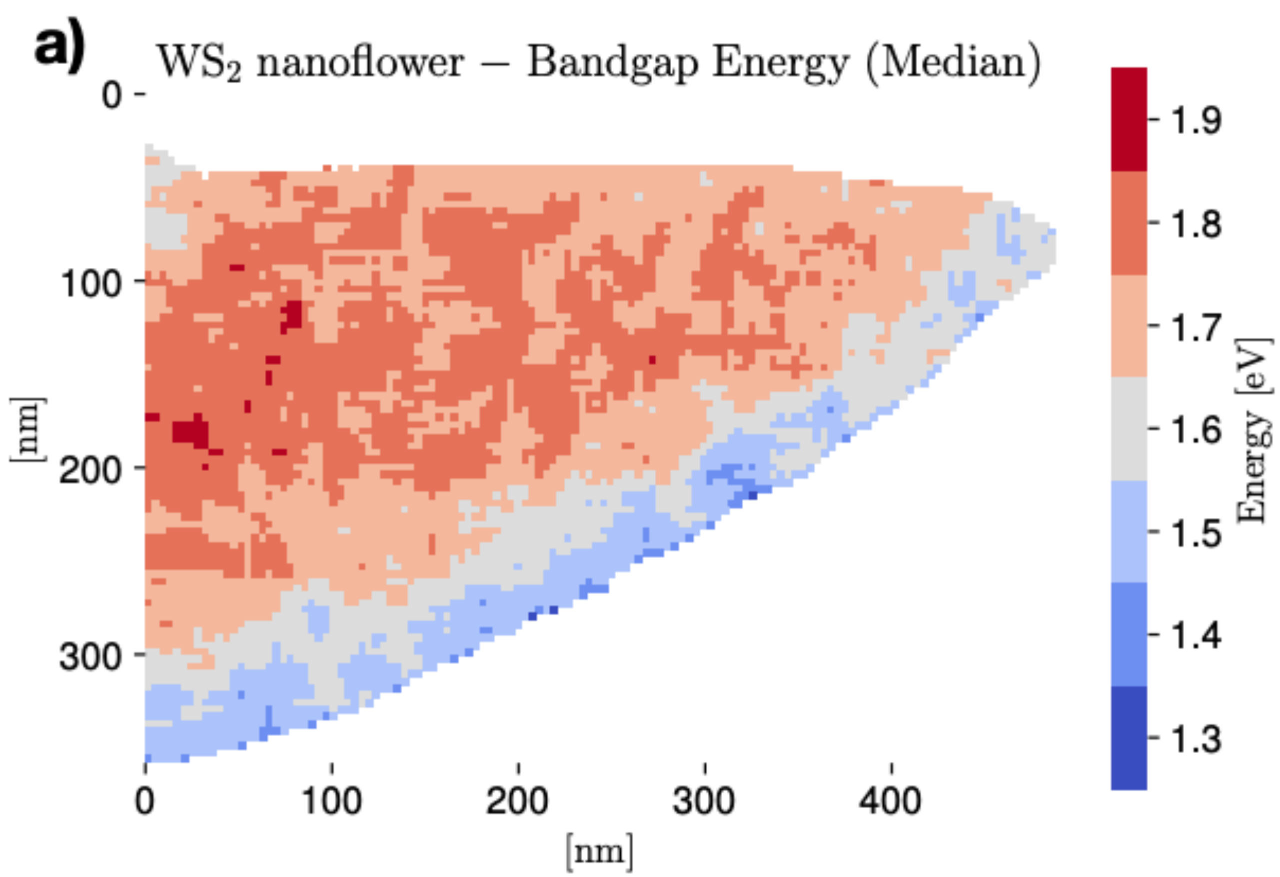


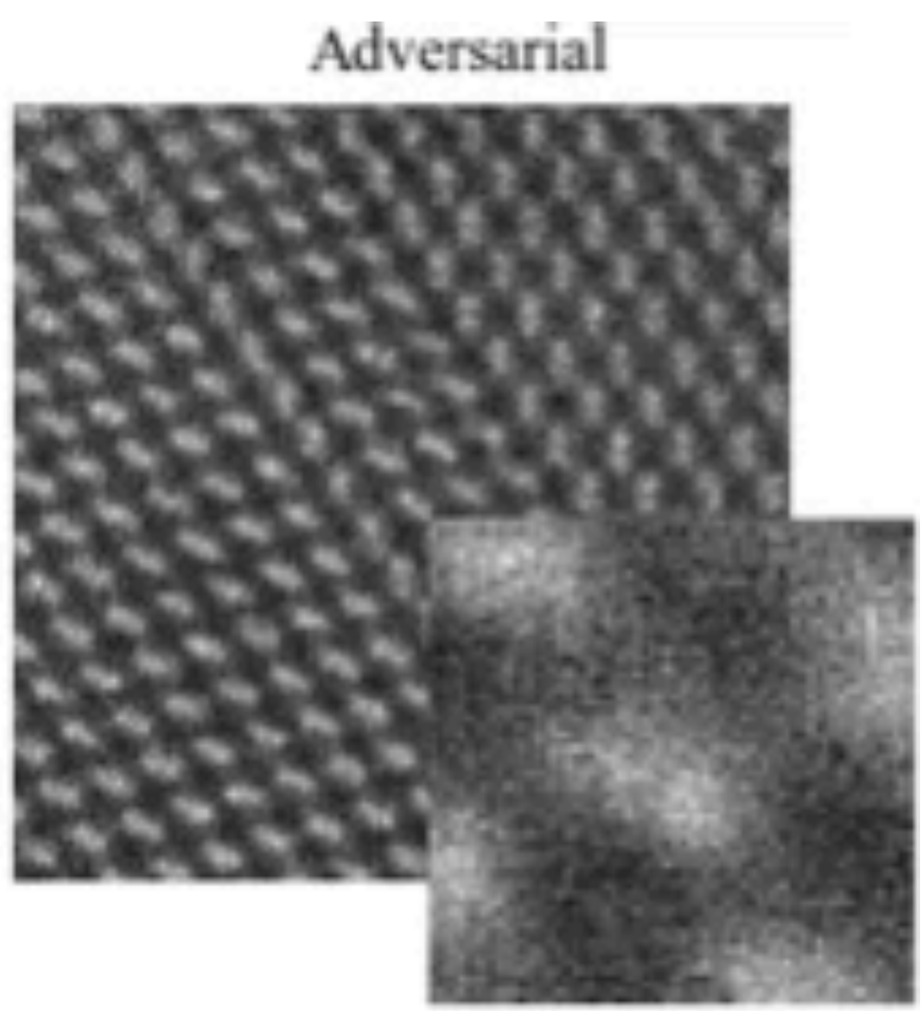
Clustering: example of unsupervised learning, where the training samples do not have labels attached: we need to identify groups of input data into clusters with similar properties

Summary and outlook

- Machine learning makes possible **identifying patterns in the data** whereby one can efficiently solve problems which are difficult of intractable with traditional approaches

- In the context of **applications to electron microscopy**, ML is applied to subtract backgrounds, identify crystalline structures and defects, improve image resolution and automatically classify features in images among many others





Assignment

- Find a nice application of machine learning to electron microscopy and/or related techniques such as electron-based spectroscopy

- Write a short report (2 pages) explaining this application of ML and connecting it with some of the topics that you have covered in the course

- Put the emphasis of what has been achieved by means of ML as compared to traditional techniques, but also mention the possible pitfalls and limitations of the method

You can use some of the examples covered in this lecture or just look for some others by yourselves, just ask me if you need help with this!