



# Lecture One: A brief introduction to Scientific Machine Learning

KE Hub Ai4Sci December 2025

Chris Budd OBE and Aengus Roberts

m4DL



UNIVERSITY OF  
BATH

Mathematics lies at the heart of machine learning based AI

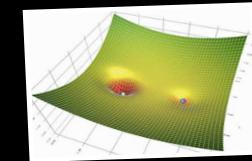
And is our best way of explaining how AI is working and answering questions about its limitations and its possibilities

## Computational Mathematics

Work by 19th century mathematicians **Charles Babbage** and **Ada Lovelace**, and 20th century mathematicians **Alan Turing** and **John Von Neumann** leads to the invention of the modern computer



20th century developments in the **mathematics of optimisation** led to the rapid growth of machine learning



## Statistics and Data Science

19th century mathematicians **Thomas Bayes** and **Carl Friedrich Gauss** transformed the way that we understand and manipulate data



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

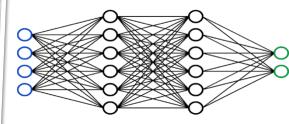
$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} \left[ \sum_{i=1}^N (f_\theta(x_i) - y_i)^2 + R(\theta) \right]$$

This mathematics is used in medical imaging



## Algebra and Calculus

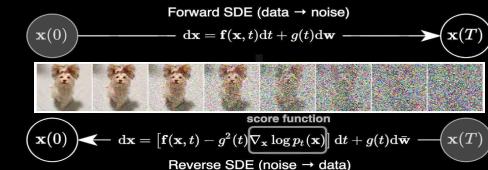
Matrices, invented by 19th century mathematician **Arthur Cayley**, provide the structure for the Internet and all AI architectures



$$\mathbf{x}_{k+1} = \sigma (\mathbf{A}_k \mathbf{x}_k + \mathbf{b}_k) \in \mathbb{R}^W$$

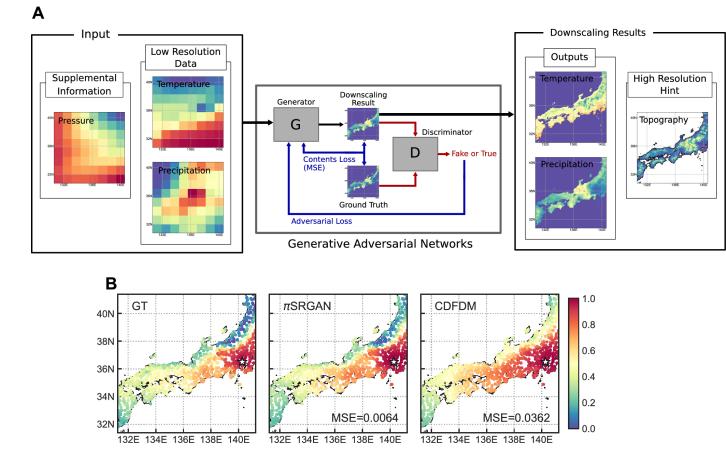


20th century mathematician **Kiyoshi Itô** developed stochastic calculus, the basis of generative AI



Now we are seeing a major impact of machine learning  
on mathematics, physics and engineering

- Function approximation and classification
- Solution of differential equations (ODEs and PDEs)
- Efficient simulation of differential operators
- Solution and regularization of inverse problems
- Prediction and time series analysis
- Classification and generation of images (down scaling)
- CAD
- Forecasting the weather (from data alone!)



# Scientific Machine Learning

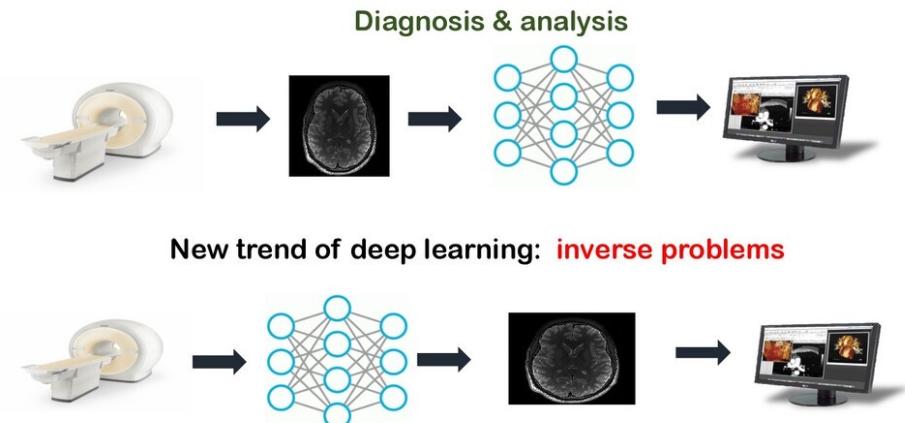
Combination of:

Numerical Analysis  
Functional Analysis  
Applied maths  
Physics  
Computer Science/Scientific computing  
Data Science

IDEA:

- Derive effective and efficient machine learning methods informed by physics
- Prove (where possible) rigorous results on their accuracy, stability, and convergence
- Apply them to reliably solve important scientific problems

## Deep Learning for Inverse Problems



## Structure of the two mini-courses

### Course C

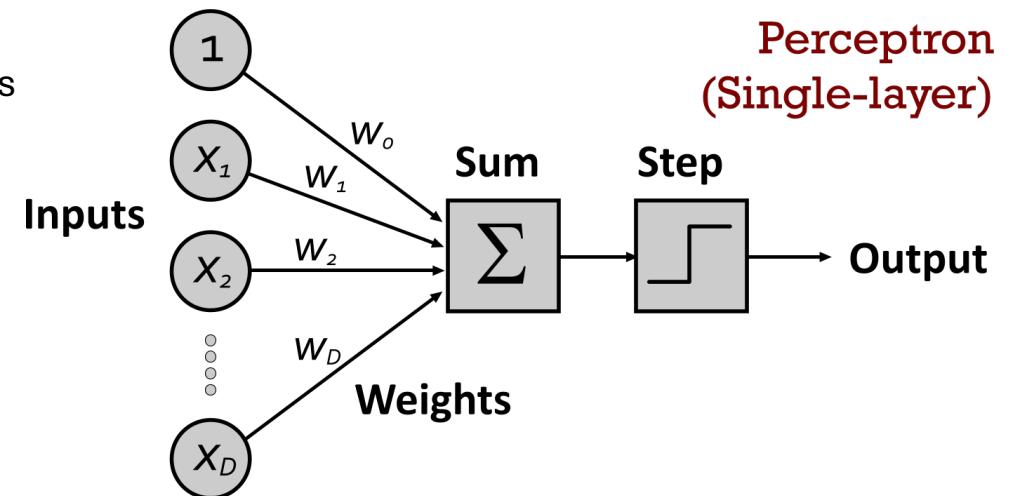
1. Introduction
2. ML for function approximation
3. PINNs for solving differential equations
4. DRMs and variational methods

#### Case Study One: PINNs

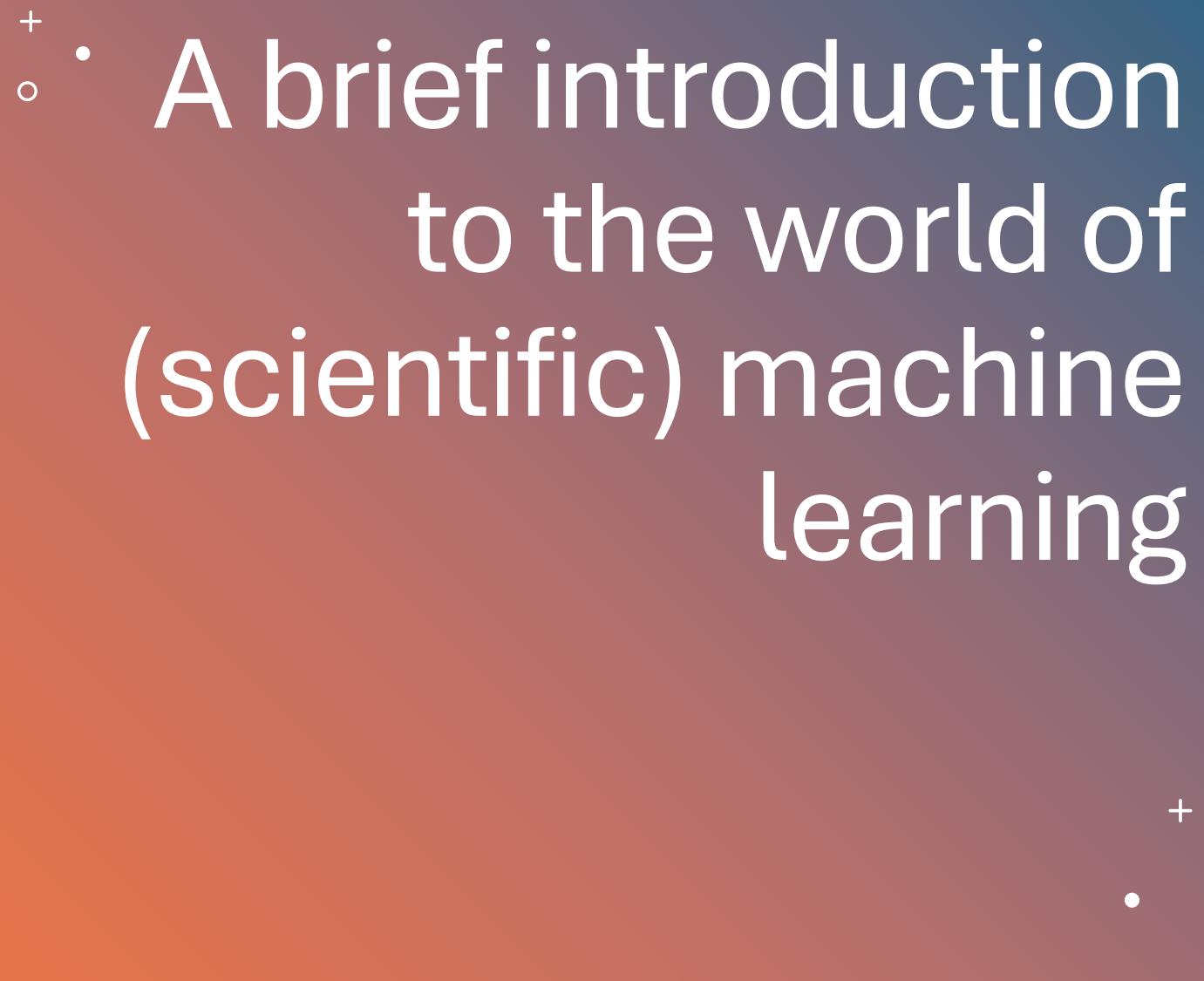
### Course D

1. Differential operators
2. Neural operators
3. Neural ODEs
4. Forecasting the weather

#### Case Study Two: Fourier Neural Operators



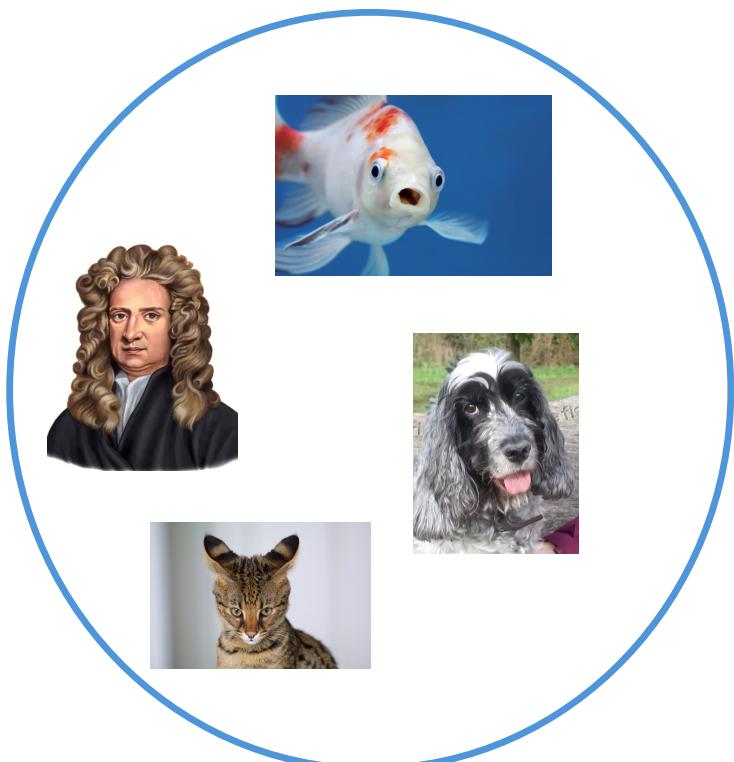
In all cases we will look at the strong links between numerical analysis and machine learning



# A brief introduction to the world of (scientific) machine learning

## Image classification/Prediction

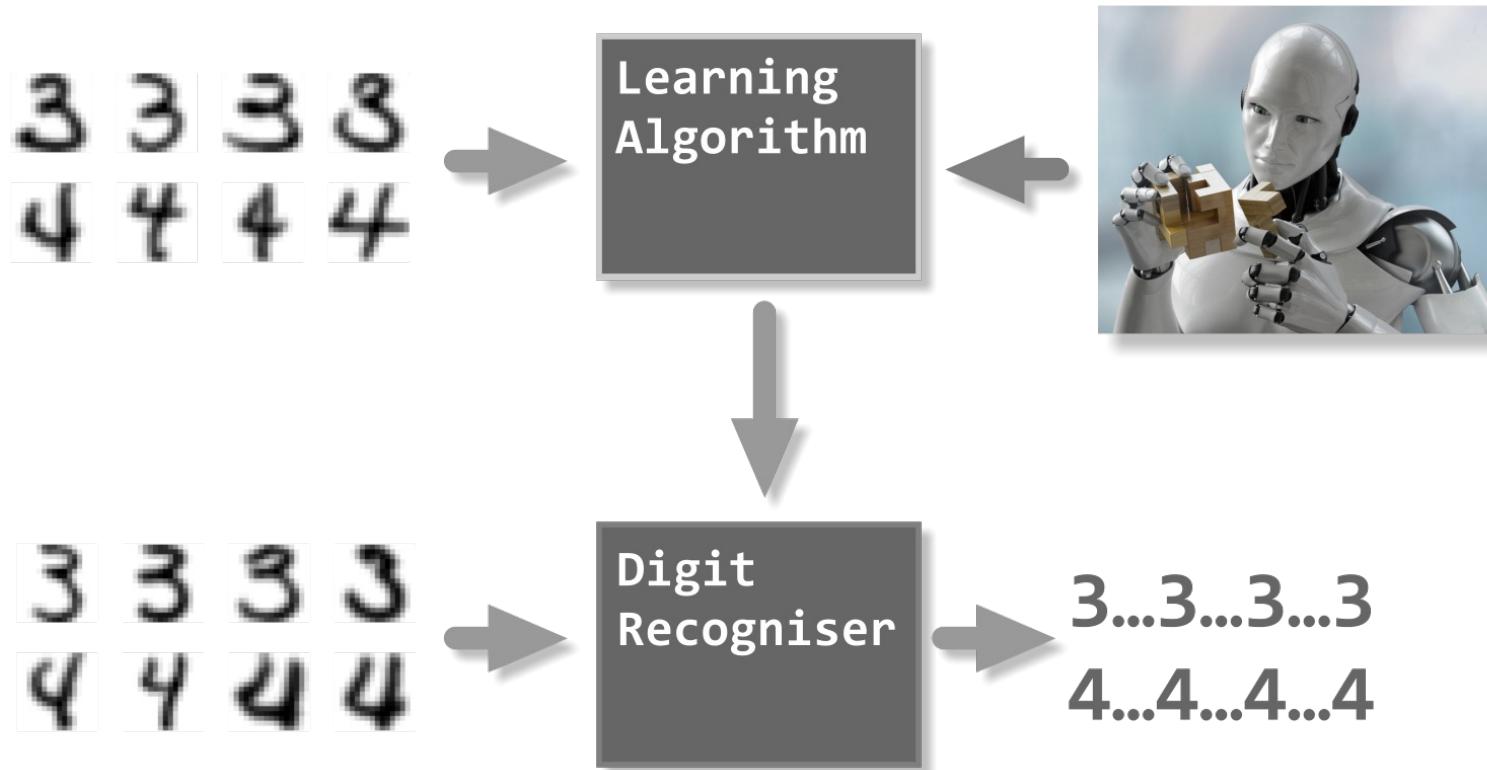
Probability distribution of images

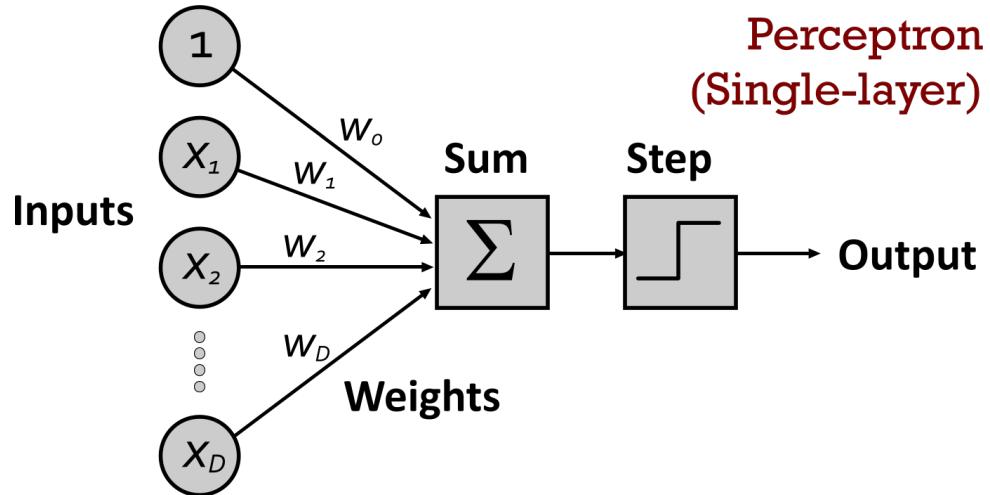


Neural net trained on  
samples from the set of  
images



Image label



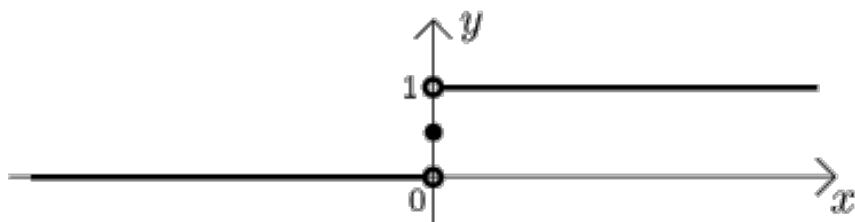


Breakthrough in classification came with the invention of the neural net (1963)

---

$$Out = H \left( \sum_{i=1}^D w_i \ X_i - C \right)$$

$$H(x) = \begin{cases} 1, & x > 0; \\ \frac{1}{2}, & x = 0; \\ 0, & x < 0. \end{cases}$$





Example: Can you tell a cat from a dog?

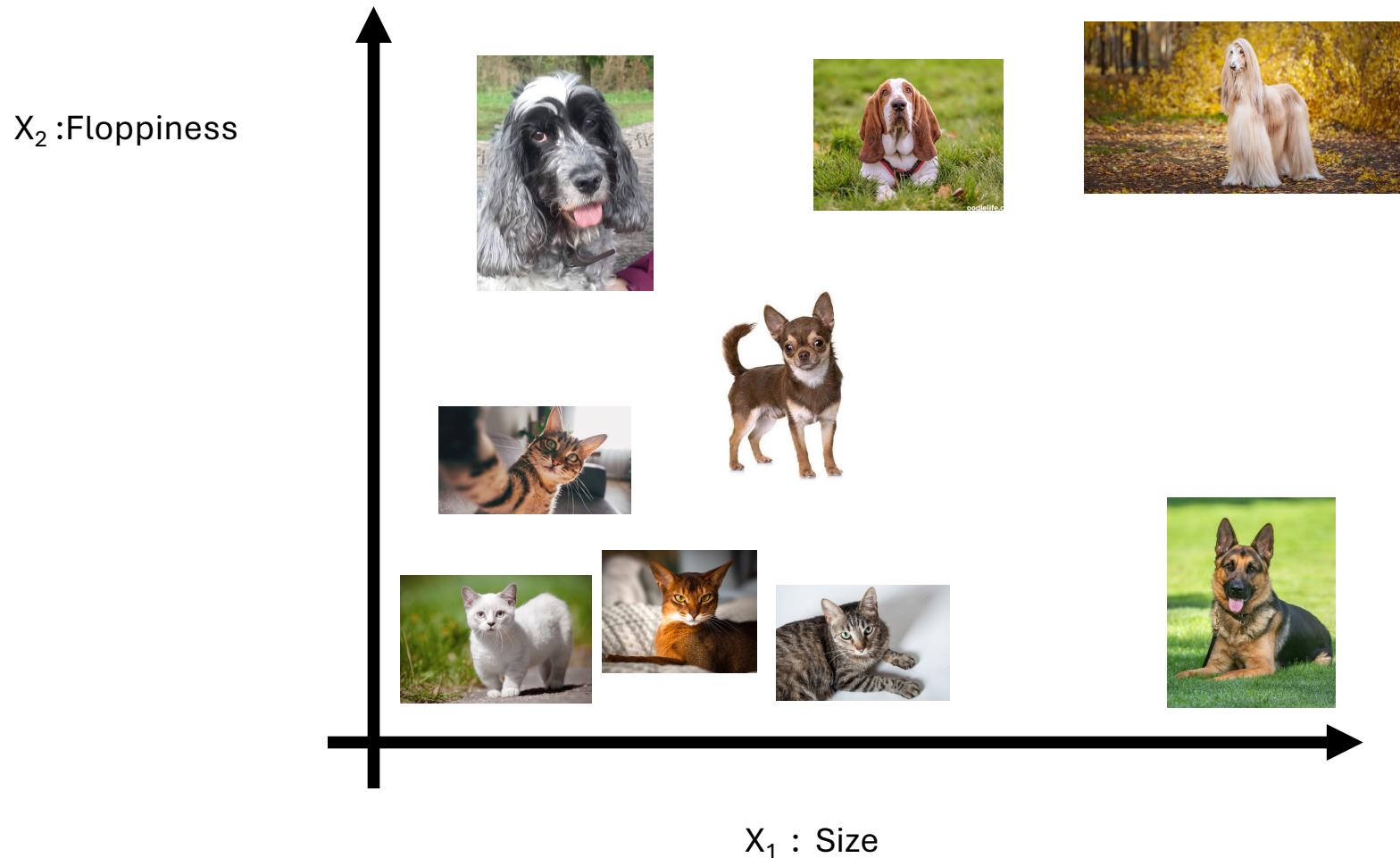
Make  
two  
measurements

$X_1$ : Size

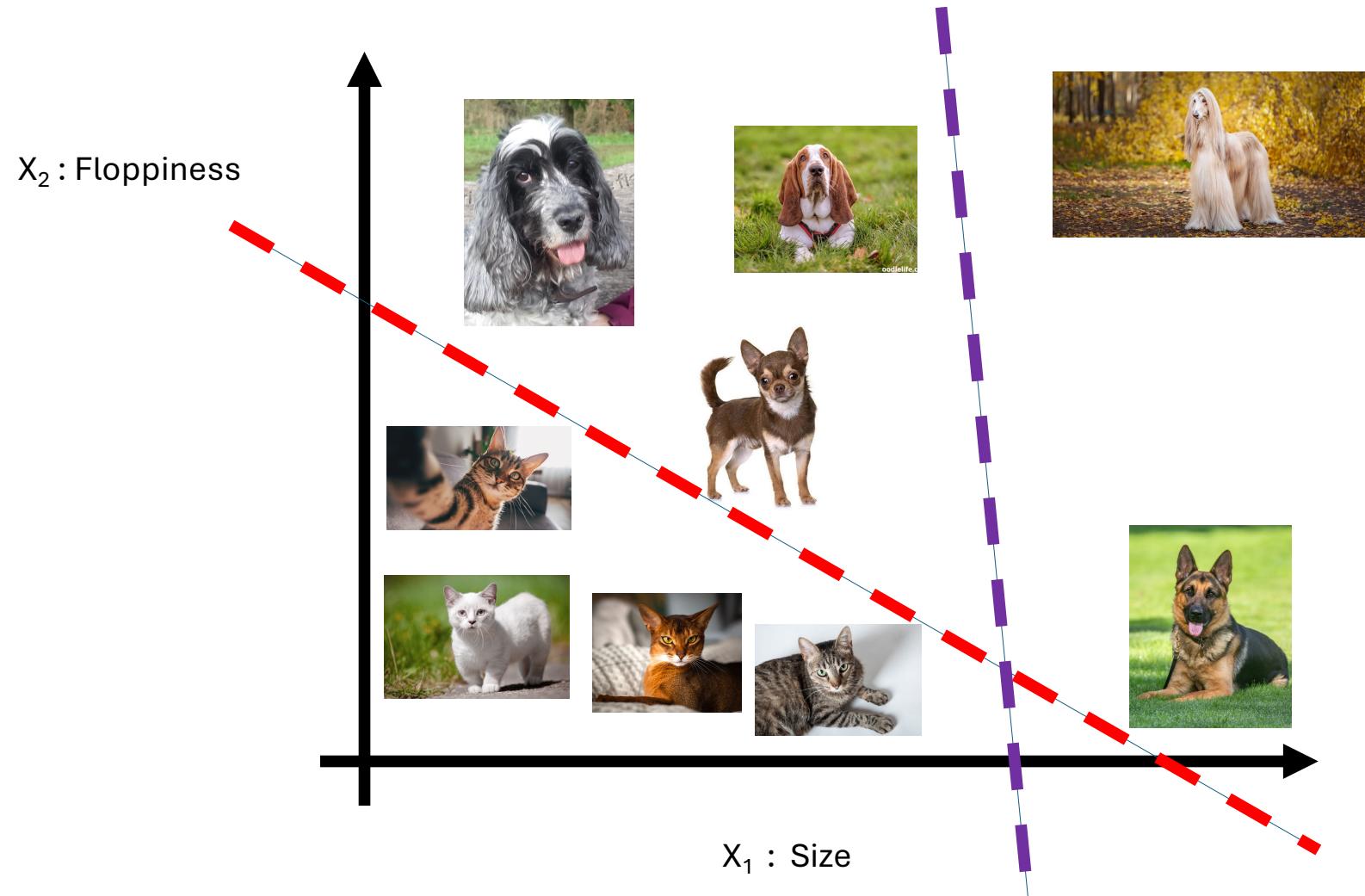
$X_2$ :  
Floppiness



Measure size and floppiness for lots of cats and dogs and plot



Draw a line which separates cats from dogs



# What is going on?

Equation for the line is:

$$w_1X_1 + w_2X_2 - C = 0$$

$w_1X_1 + w_2X_2 - C > 0$  : Dog

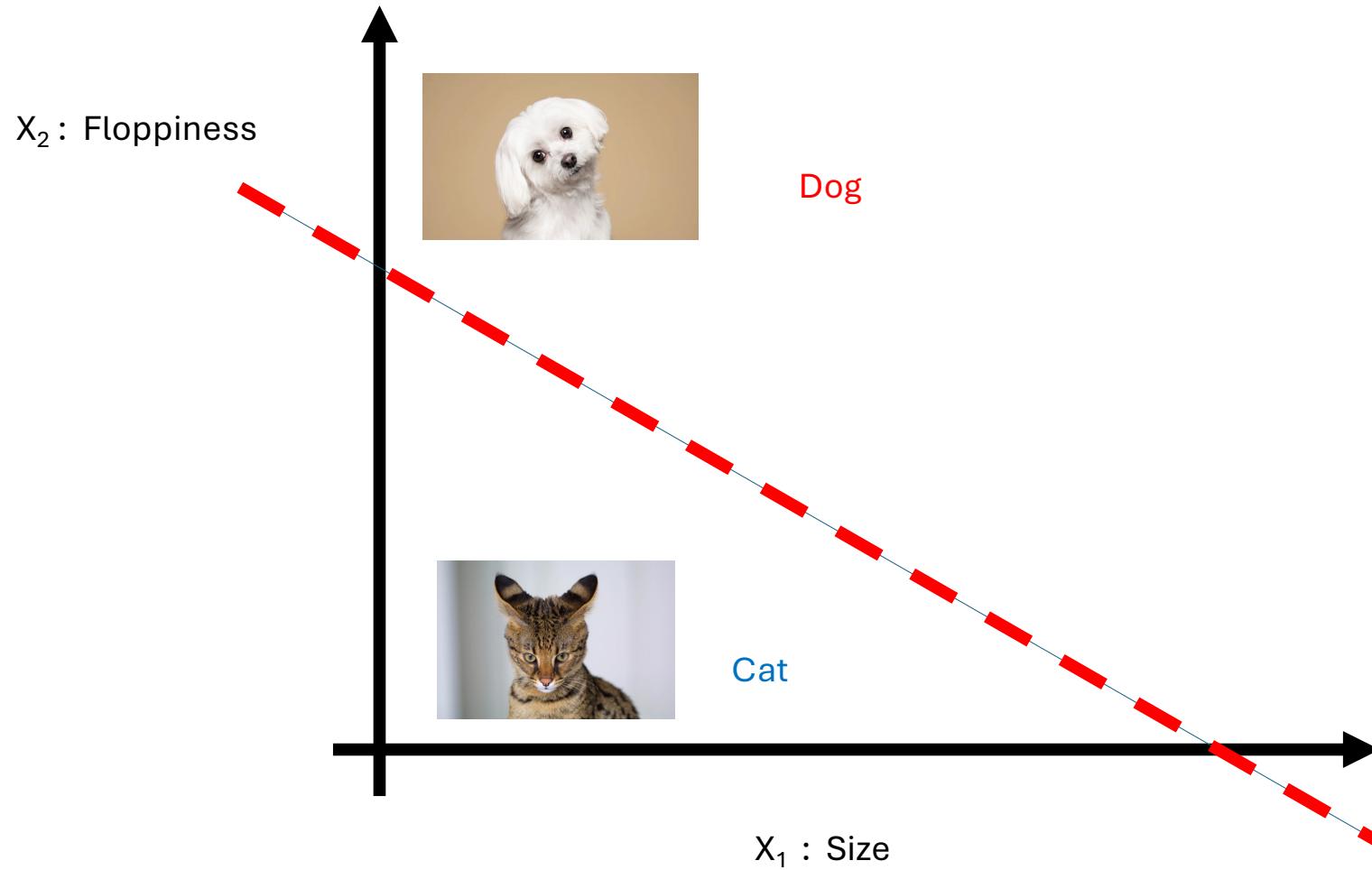


$w_1X_1 + w_2X_2 - C < 0$  : Cat

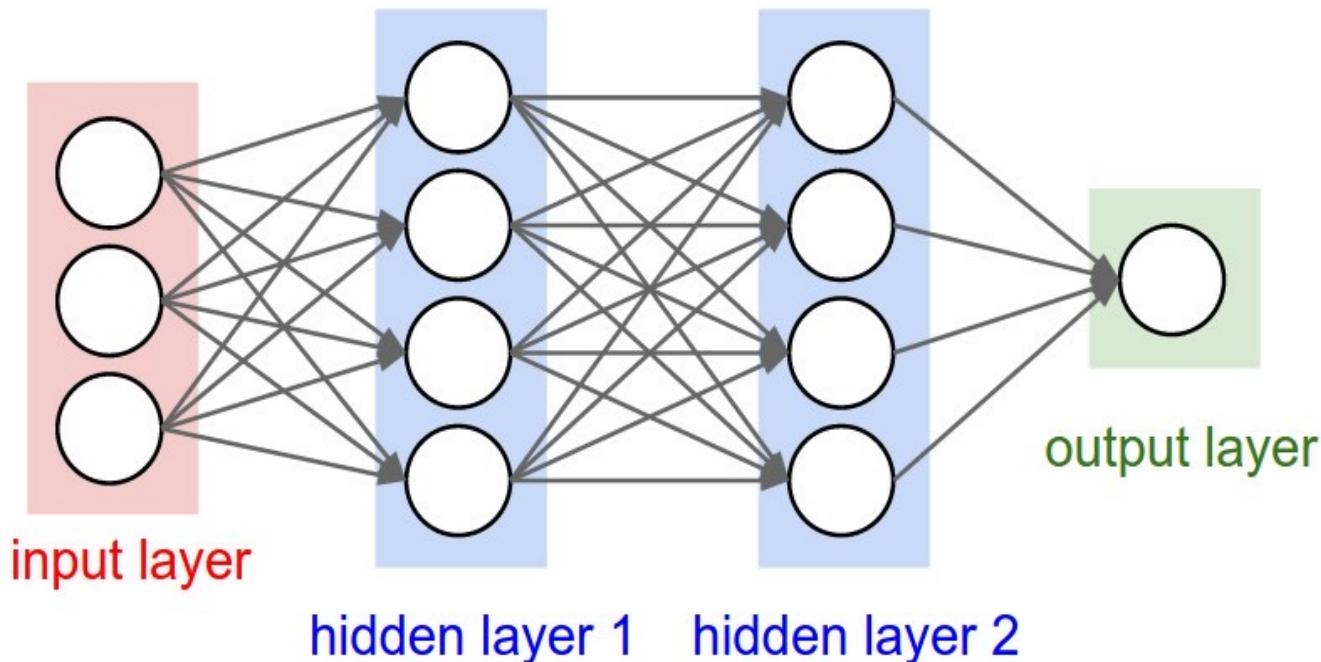


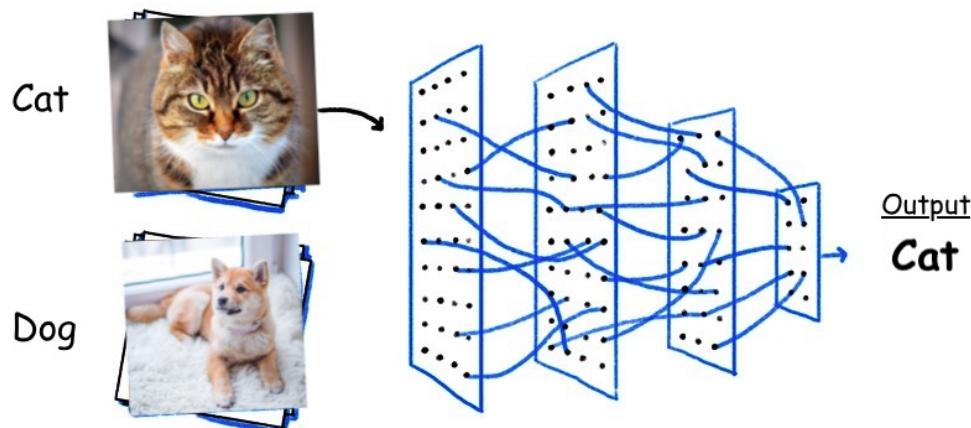
Train the weights  $w_1, w_2, C$  on the data to give the right result

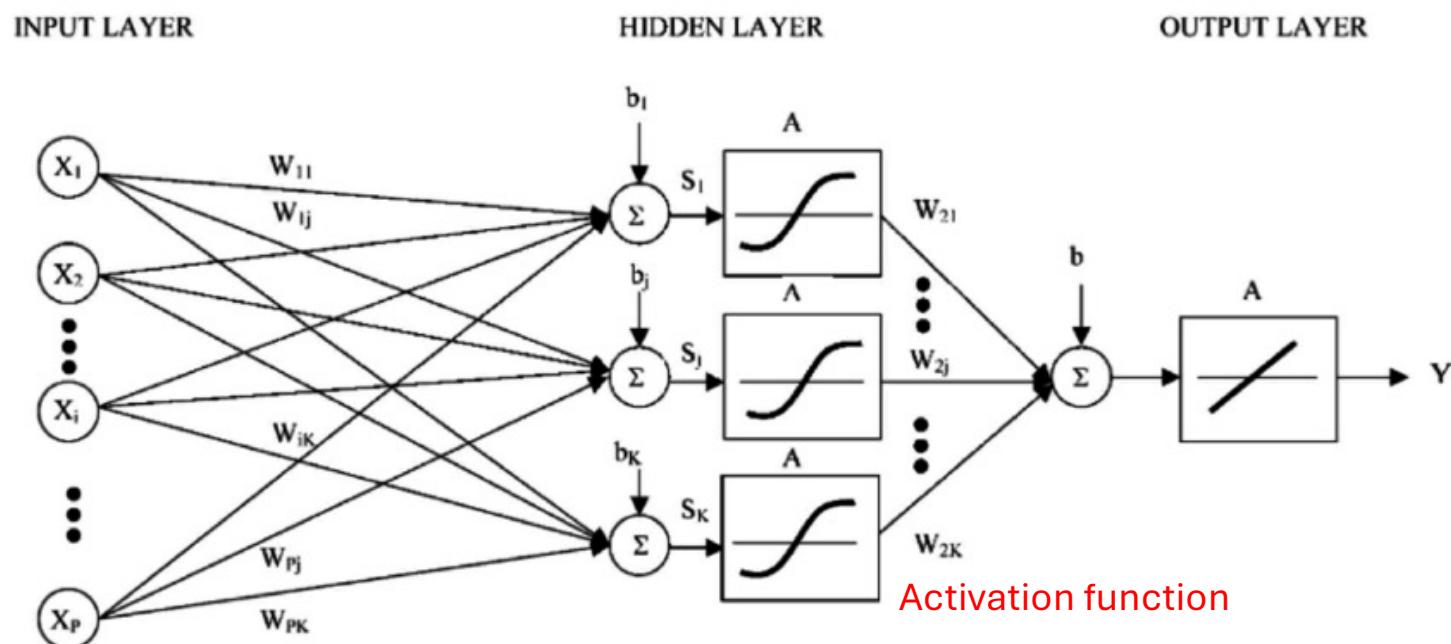
Can now tell a cat from a dog



Deep neural nets (2010) use many more layers and can be used to make much more complex decisions







Feed forward neural network

Or more mathematically, using a notation we use later on:

$$y_{j+1} = \sum_{i=0}^{W-1} c_i \sigma_{i,j} (a_{i,j} y_j + b_{i,j})$$

Input:  $y_0 = x$ .      Output:  $y(x) = y_L$

W: Width      L: Depth

$\sigma_{i,j}$ : Activation functions. (usually nonlinear)

$\theta$ : Set of All of the parameters

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) <sup>[2]</sup>		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) <sup>[3]</sup>		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Need to find the best values for the Parameters

Achieve this by training the NN using a training set of data and a measure of success

Can take many hours of computer time on a GPU.

With a vast expense of energy!!



# Mathematical Ingredients of the training process

Loss function  $L(W)$ : How close is the answer

Back propagation: How does the answer depend  
on changes in the parameters.

Achieved by using **automatic differentiation**  
PyTorch does this with ease!

Optimisation: Choose the parameters to  
minimize the loss over (a subset of the)  
training data

Validation: Test on separate data to avoid  
over fitting to the answer

Epoch

Optimise the weights using optimisation methods:

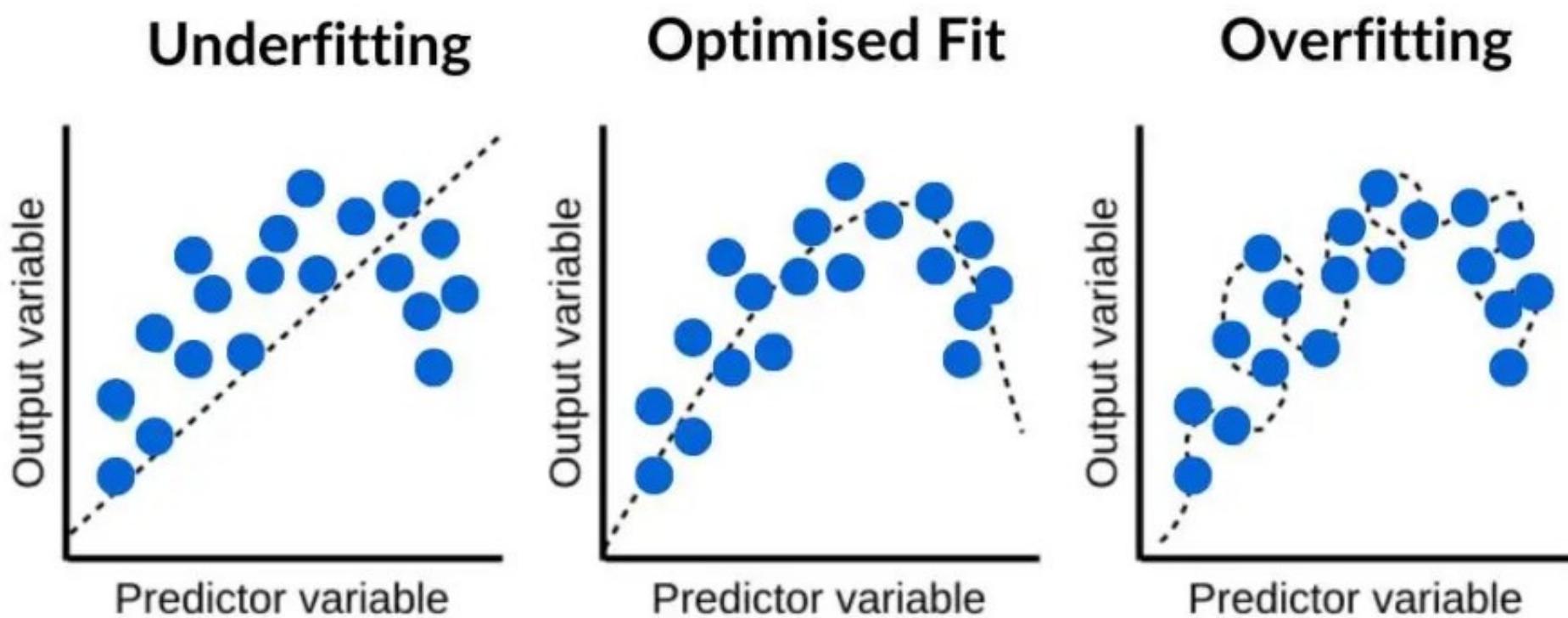
Eg. stochastic gradient descent, **ADAM**, quasi-Newton, simulated annealing, Monte-Carlo Tree Search (MCTS)

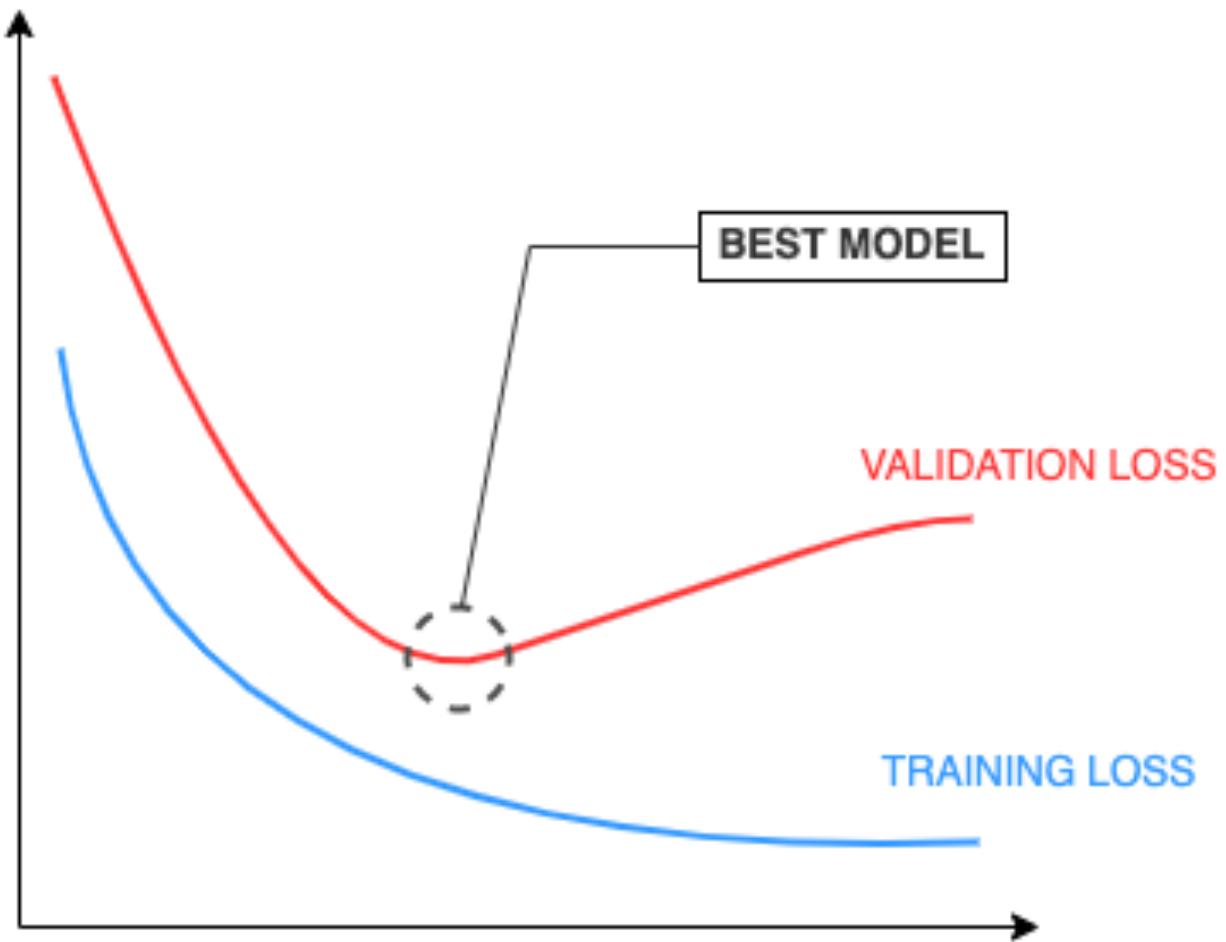
Implemented in eg. **PyTorch**  
(see the Case studies)

Tensor Flow/Keras



AlphaZero used reinforcement learning over 700,000 games with MCTS optimisation





- Supervised training

Have a labeled training set

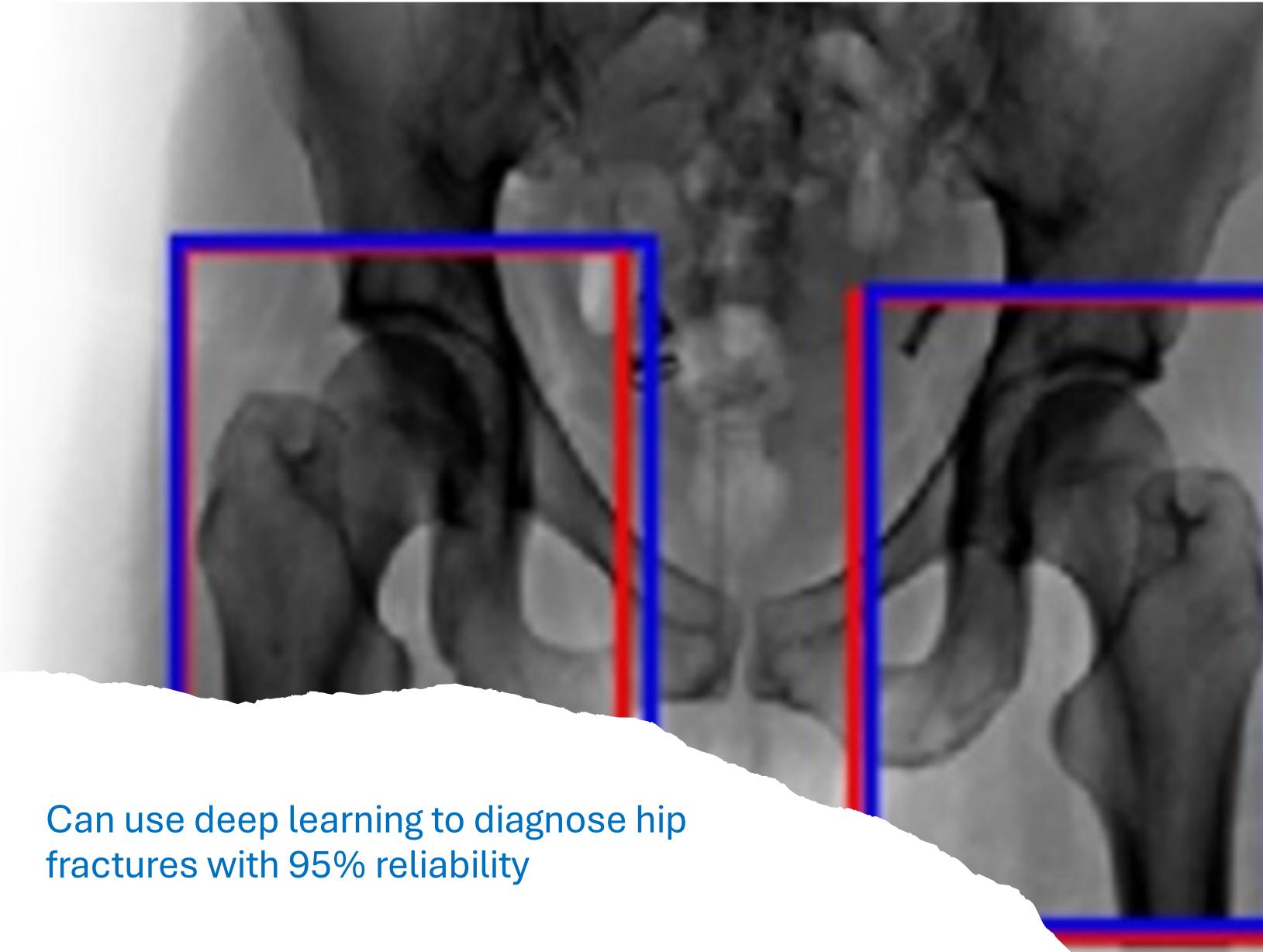
eg. cats + dogs, musical genres,  
Inverse problems, approximation

- Reinforcement training

Have a measure of success

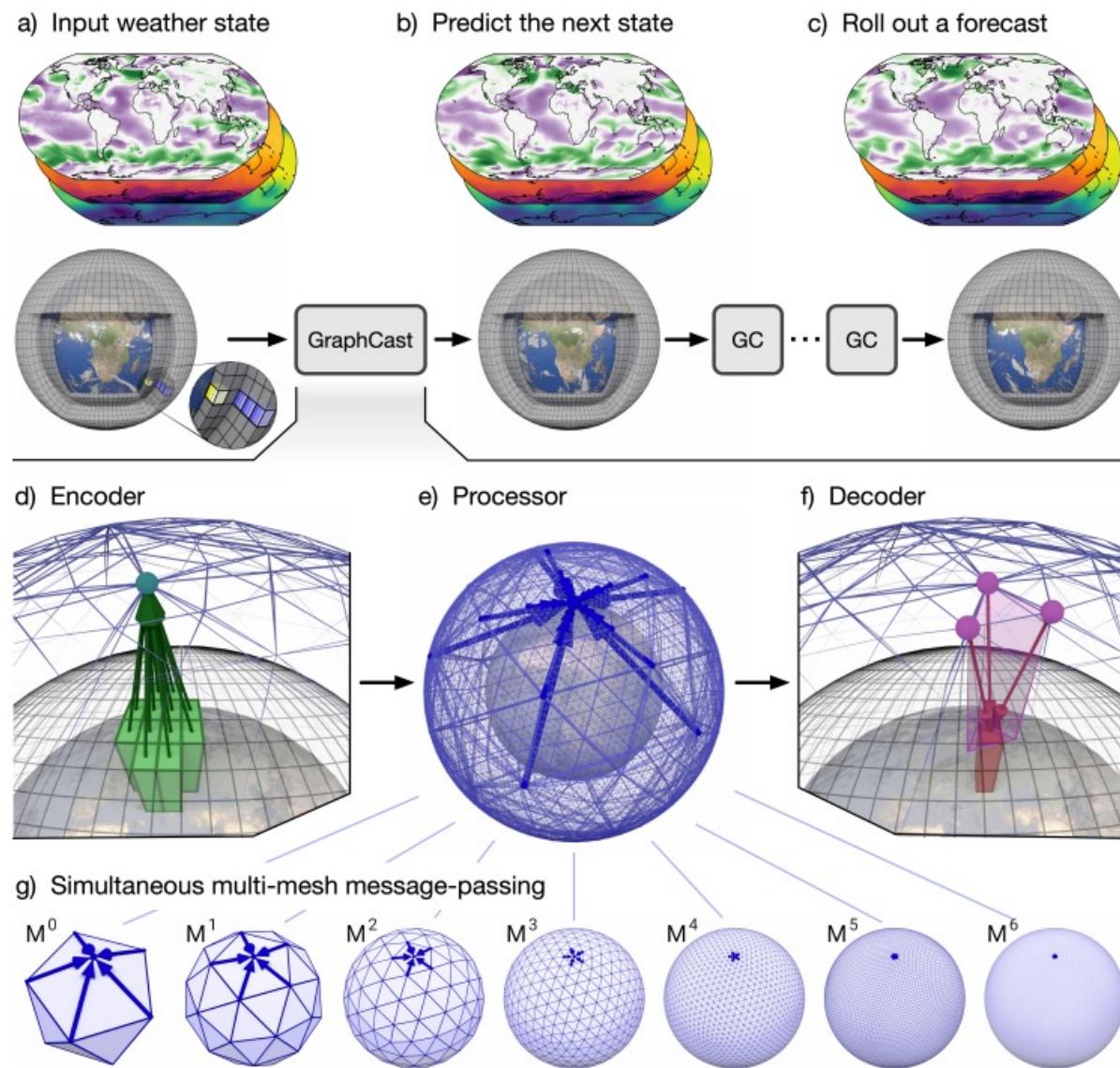
eg. winning GO, solving an equation  
(eg. a PINN), variational method





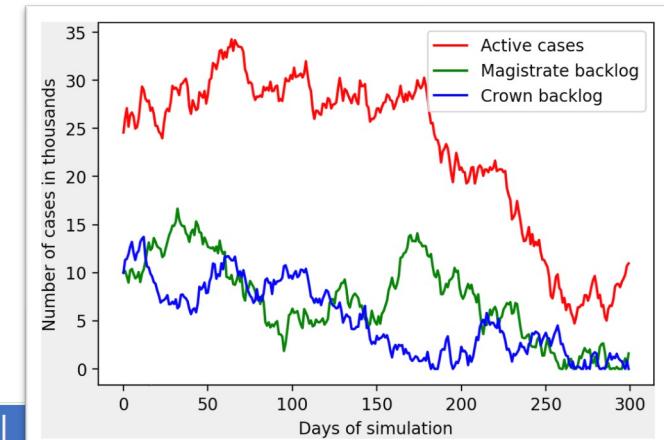
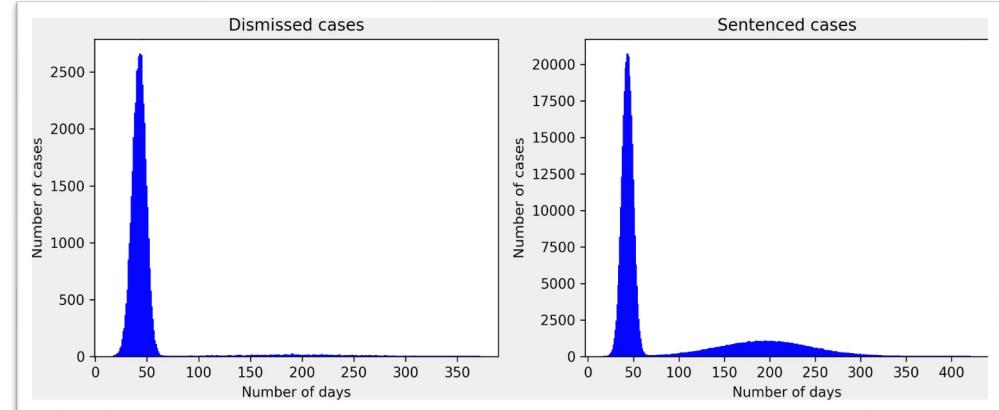
Can use deep learning to diagnose hip fractures with 95% reliability

# Weather forecasting trained on the ERA5 Data Set



# Maths and AI for Justice

NN can predict case volume and duration in the judicial system based on case characteristics



Time distribution: Lognormal

Time spent in summoned: 34 days

Summoned to Exit: 9 days

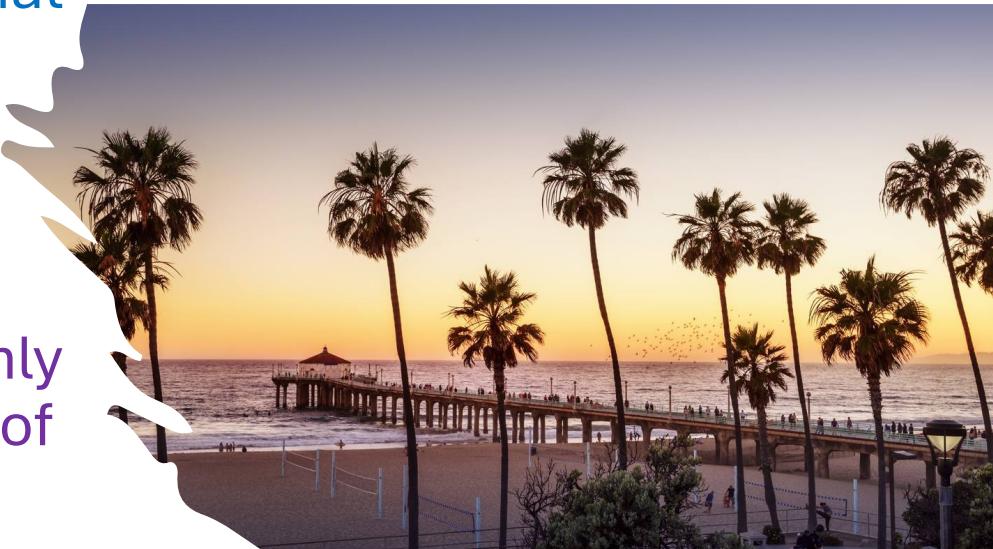
Time spent in Sentencing: 7 days

Data: the national statistics for the Criminal court statistics quarterly (April - June 2023)

But ..Can a weather forecast trained on Californian data predict a storm in the UK?

Learning only from data makes it hard to make decisions away from that data

Would you trust a computer to make a judicial judgement if only trained on some types of cases?





Also:

Some images are hard to  
classify correctly



# Machines can get it wrong

---

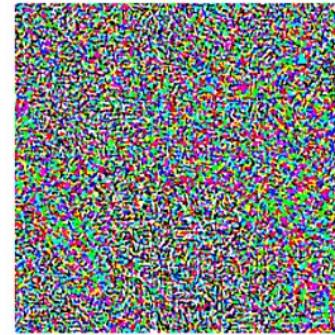
Identified as a 45mph speed sign



“panda”

57.7% confidence

$+ .007 \times$



noise

=



“gibbon”

99.3% confidence

---

And can get confused by an  
‘adversarial attack’

Study this as a conditioning  
problem in numerical analysis

# Machine learning models can memorise data: our privacy is at risk

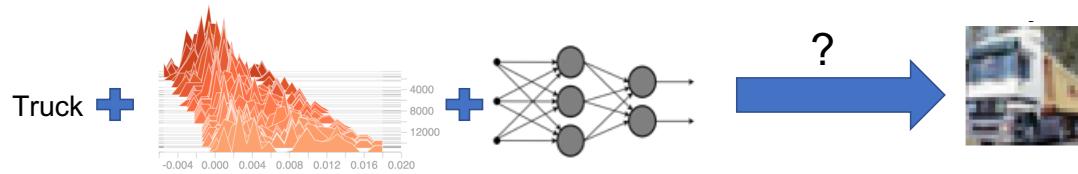
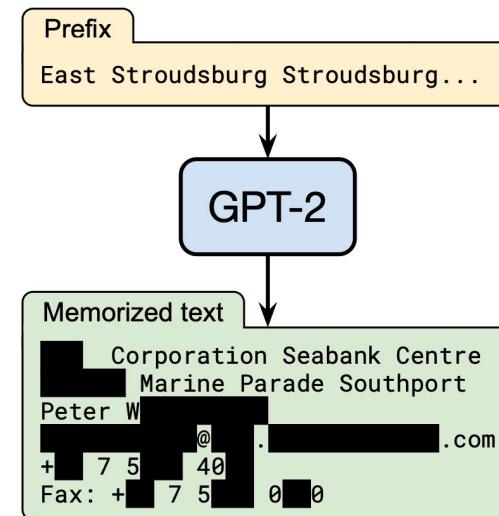


Image classification models can memorise their training images through the training updates (i.e. gradients).



Language models can also memorise their training can be extracted using text prompts.

Chen C, Campbell N. Analysing Training-Data Leakage from Gradients through Linear Systems and Gradient Matching, The 33rd British Machine Vision Conference (BMVC), 2022

Carlini, Nicholas, et al. "Extracting training data from large language models." 30th USENIX Security Symposium (USENIX Security 21). 2021.

One way to mitigate the privacy risk: training with differential privacy

