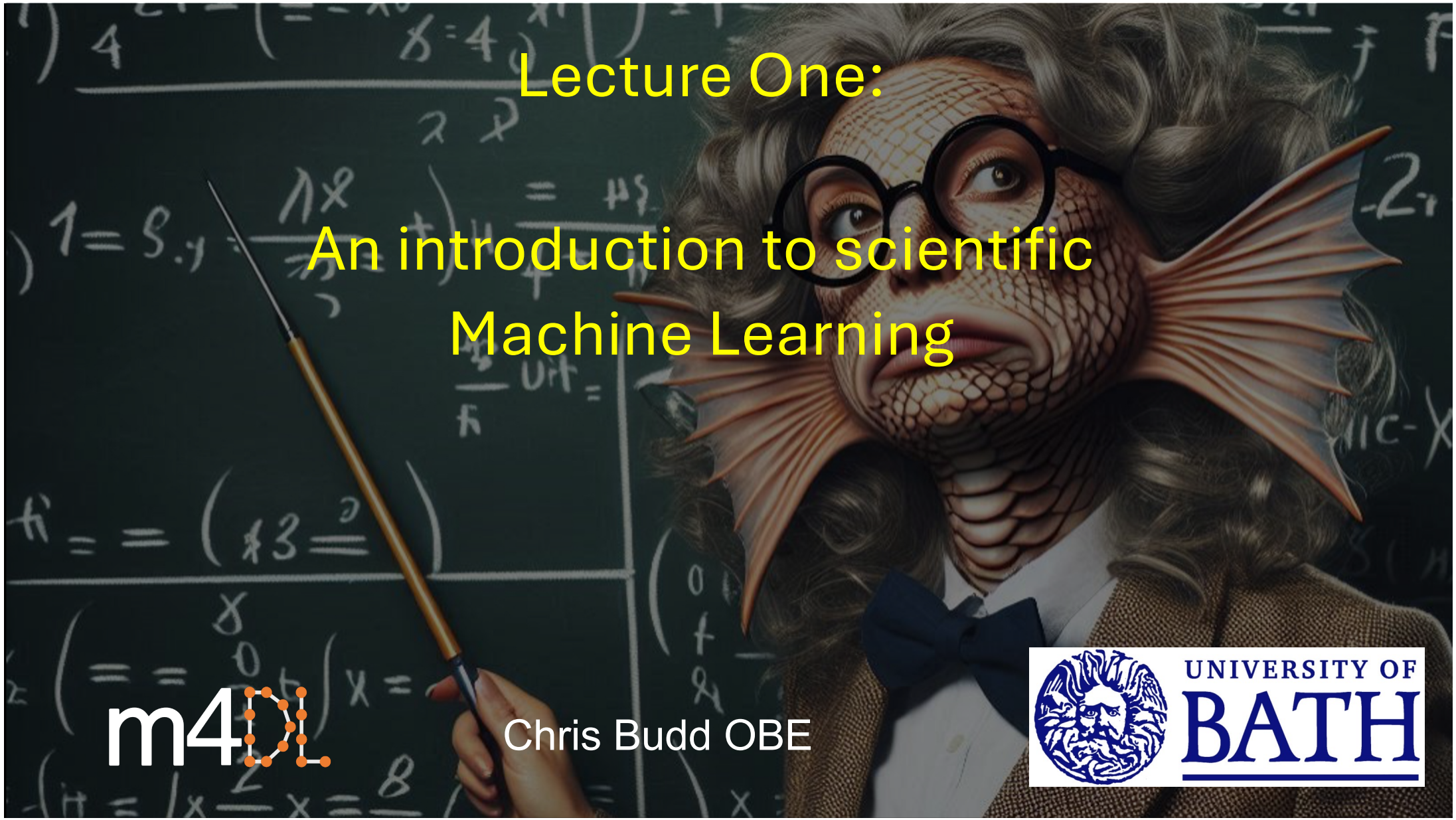

Lecture One:
An introduction to scientific Machine Learning
m4DL
Chris Budd OBE
UNIVERSITY OF BATH

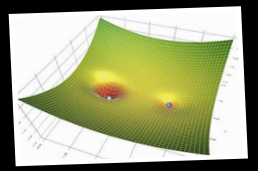
Mathematics lies at the heart of machine learning based AI

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

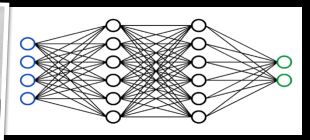
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

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

This mathematics is used in medical imaging

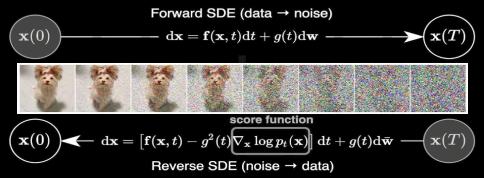
Algebra and Calculus

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

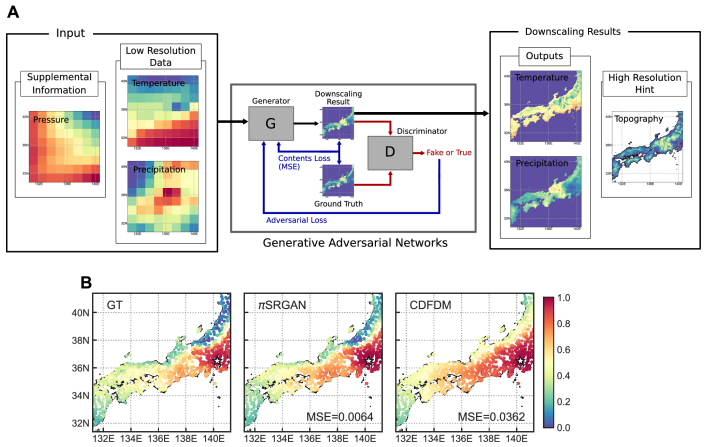


$$\boldsymbol{x}_{k+1} = \sigma\left(A_k \boldsymbol{x}_k + \boldsymbol{b}_k\right) \in \mathbb{R}^W$$

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

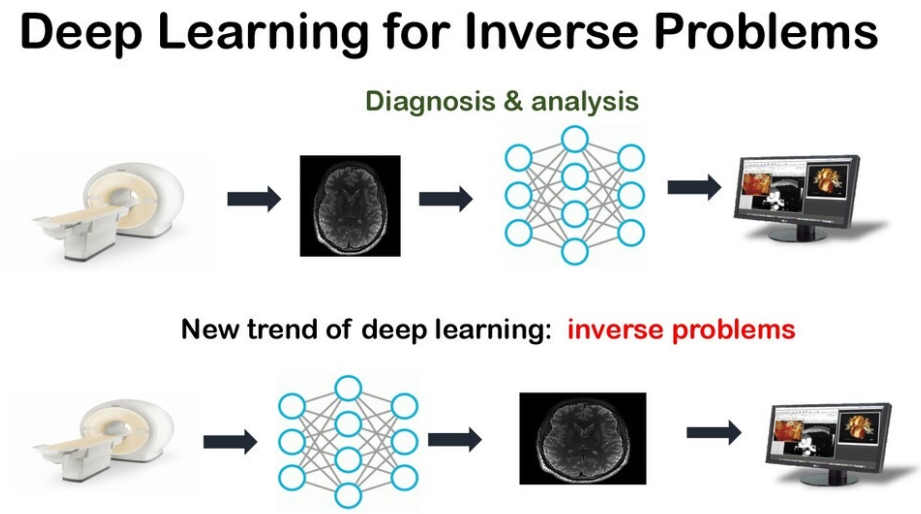


- Function approximation and classification
- Solution of differential equations (ODEs and PDEs)
- Efficient simulation of differential operators
- Solution and regularization of inverse problems
- Prediction and time series analysis
- Classification and generation of images (down scaling)
- CAD
- Forecasting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

Call this Scientific Machine Learning

Combination of:

Numerical Analysis
Functional Analysis
Applied maths/physics
Computer Science
Data Science



IDEA:

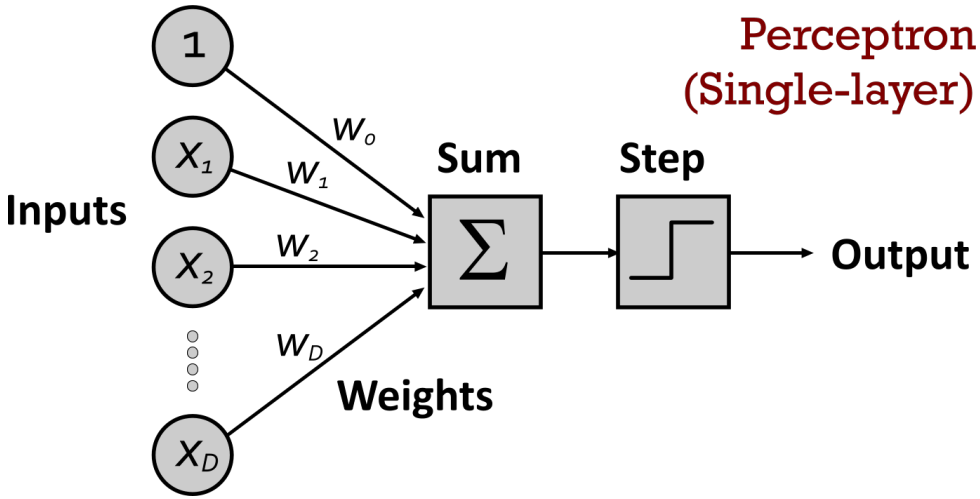
- Derive effective and efficient machine learning methods
- Prove (where possible) rigorous results on their accuracy, stability, and convergence
- Apply them to reliably solve important scientific problems

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 and things



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

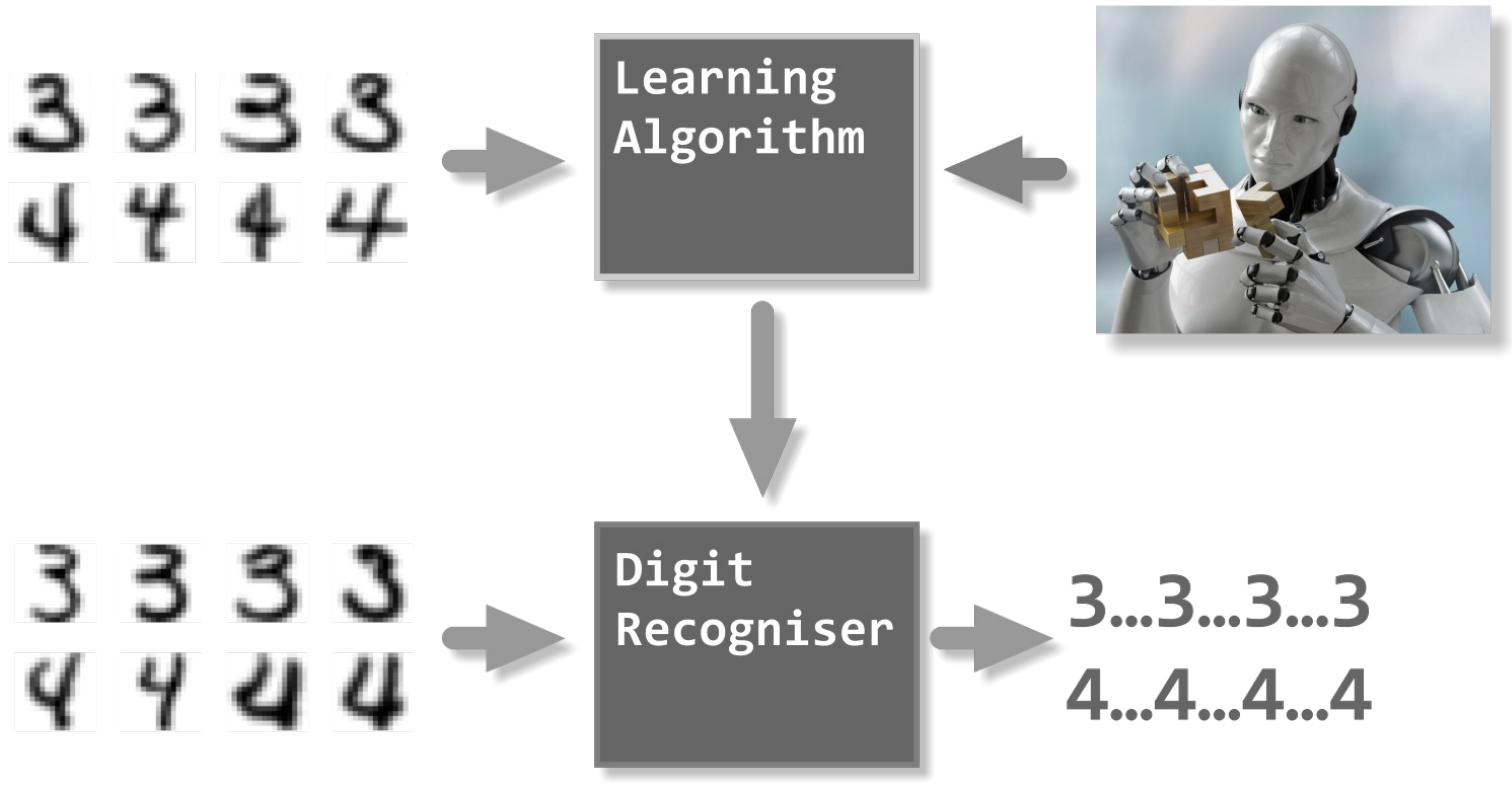
A brief introduction to the world of machine learning

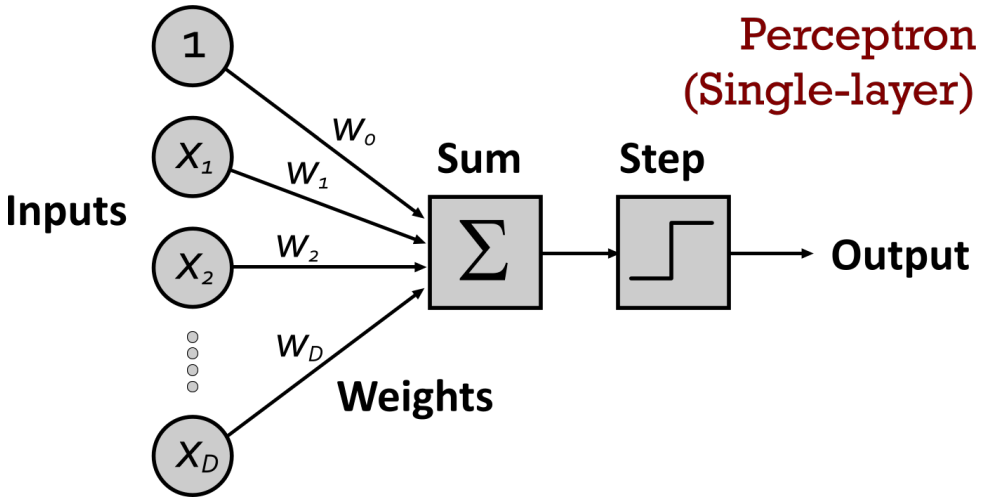
Image classification/Prediction

Probability distribution of images

Neural net trained on samples from the set of images

→ Image label

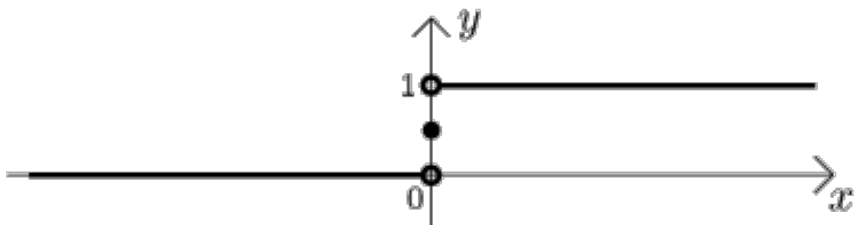

Learning
Algorithm
Digit
Recogniser
3…3…3…3
4…4…4…4



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?

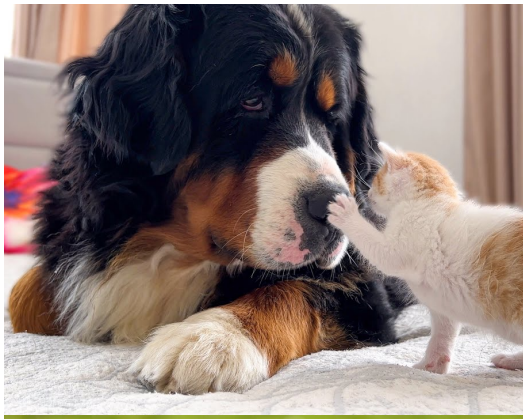
Make

two

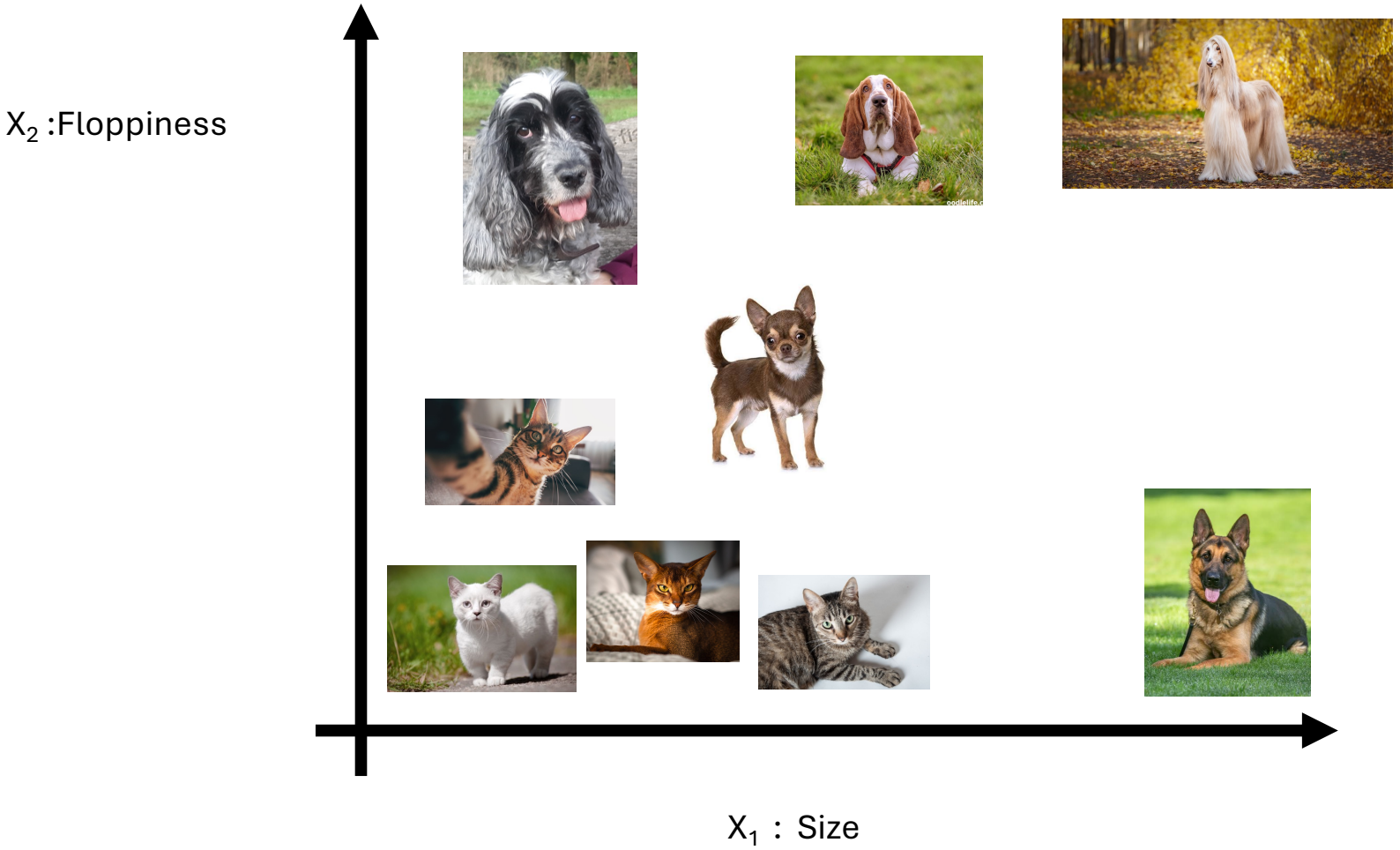
measurements

X_1 : Size

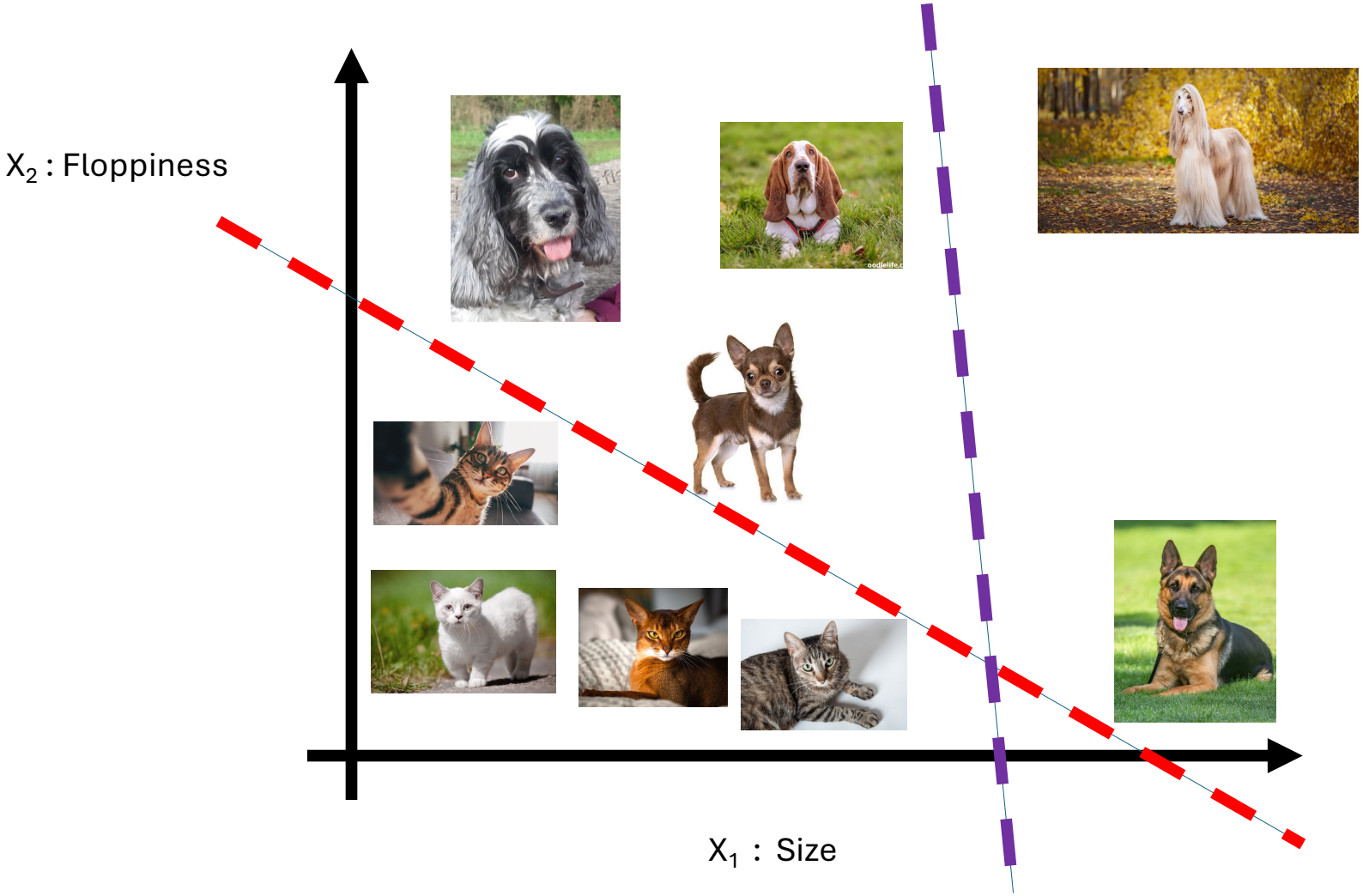
X_2 : Floppiness



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



Draw a line which separates cats from dogs



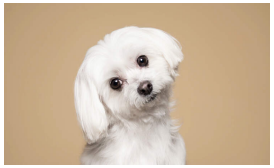
What is going on?

Equation for the line is:

$$w_1X_1 + w_2X_2 - C = 0$$

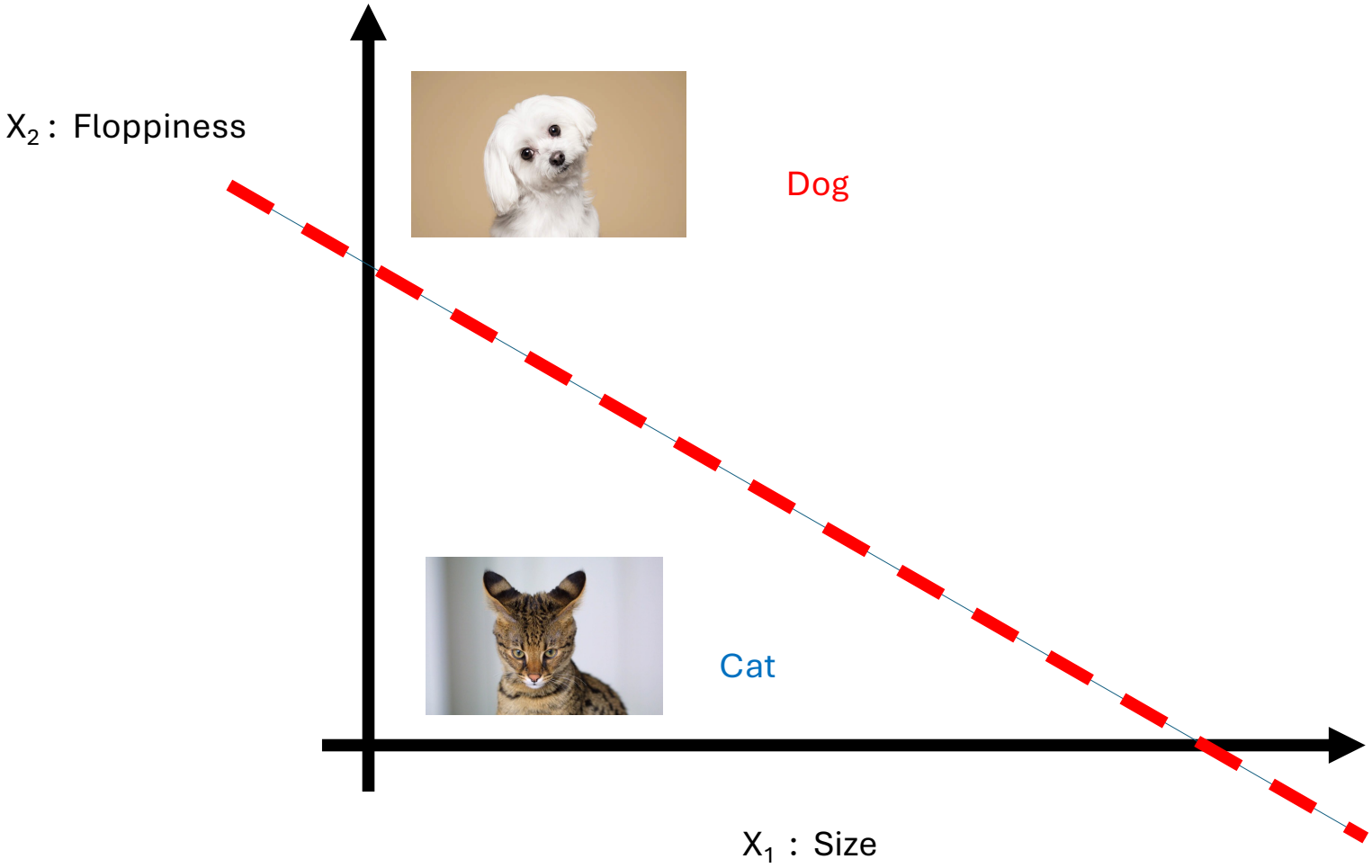
$$w_1X_1 + w_2X_2 - C > 0 \ : \ \text{Dog}$$

$$w_1X_1 + w_2X_2 - C < 0. \ : \ \text{Cat}$$

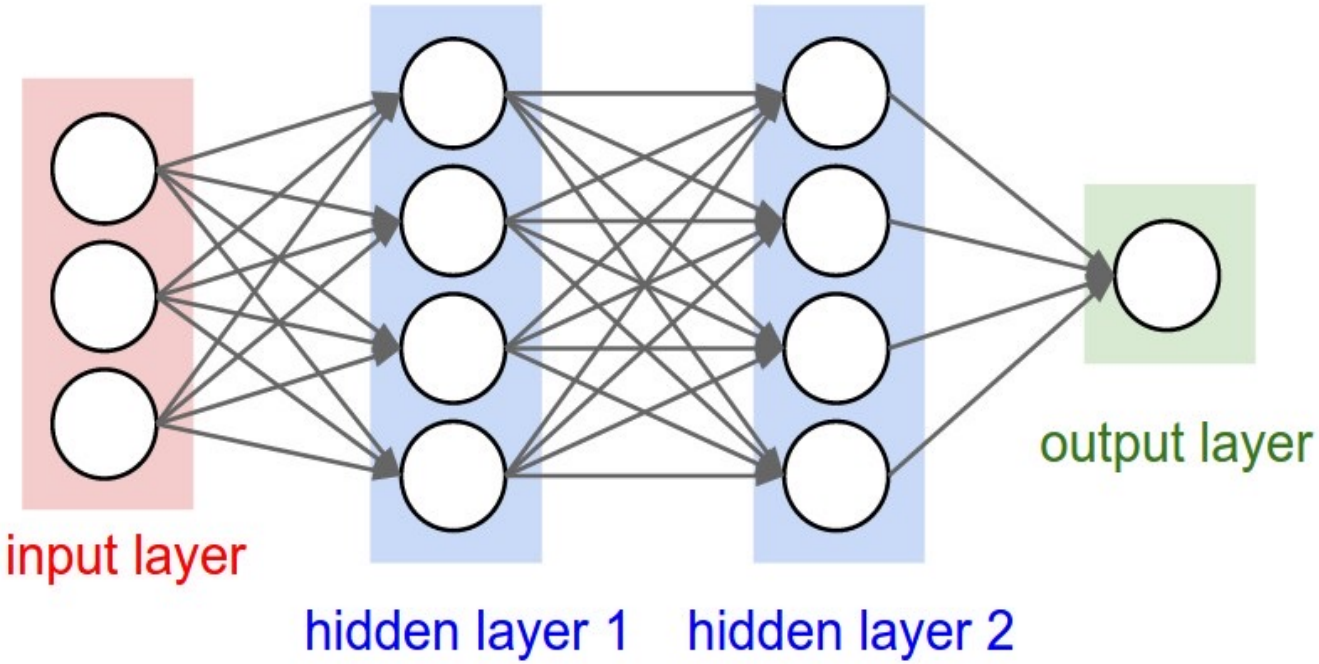


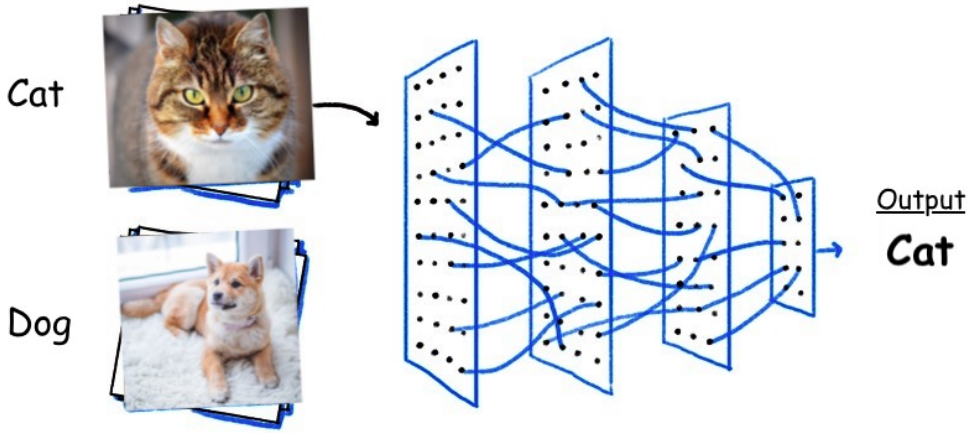
Train the weights w_1, w_2, C on the data to give the right result

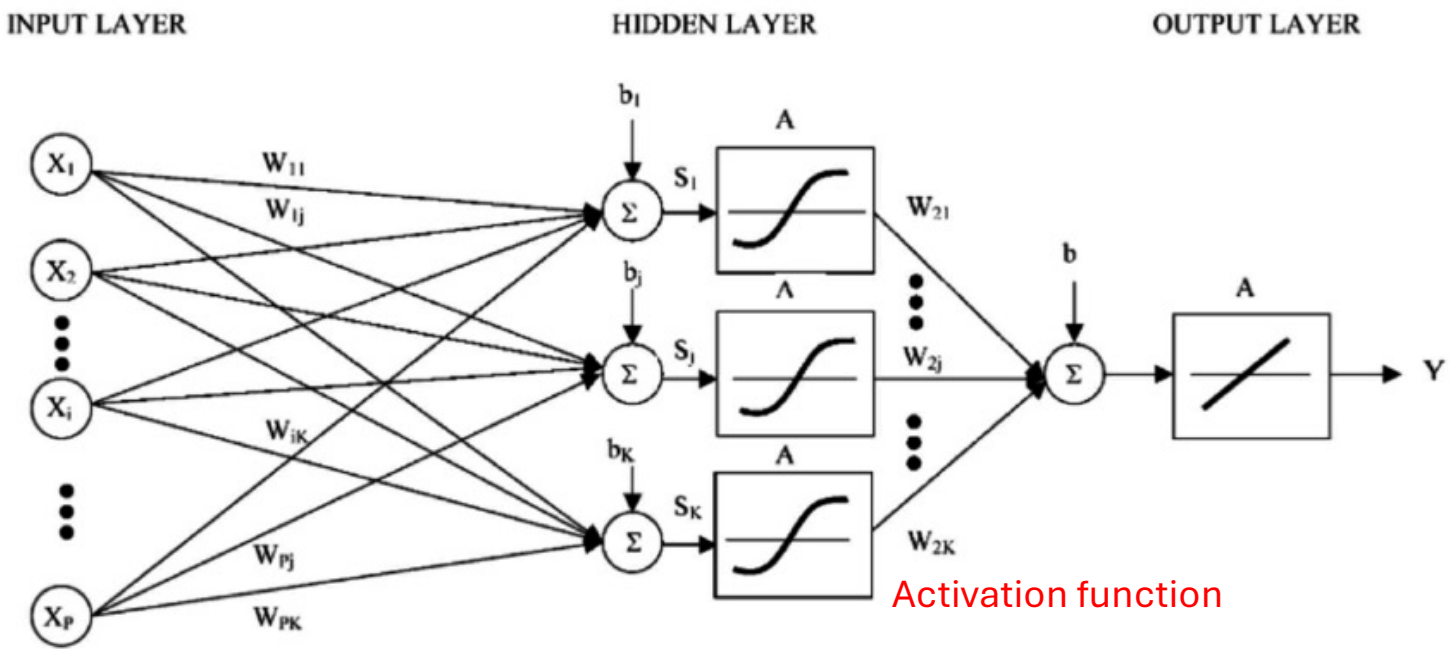
Can now tell a cat from a dog



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




Cat
Dog
Output
Cat



Feed forward neural network

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

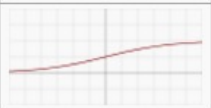
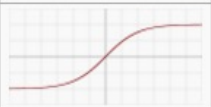
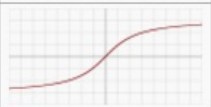
$$y_{j+1} = \sum_{i=0}^{W-1} c_i \; \sigma_{i,j} \left(a_{i,j} \; y_j + b_{i,j} \right)$$

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

W: Width L: Depth

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

θ: 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) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
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!!

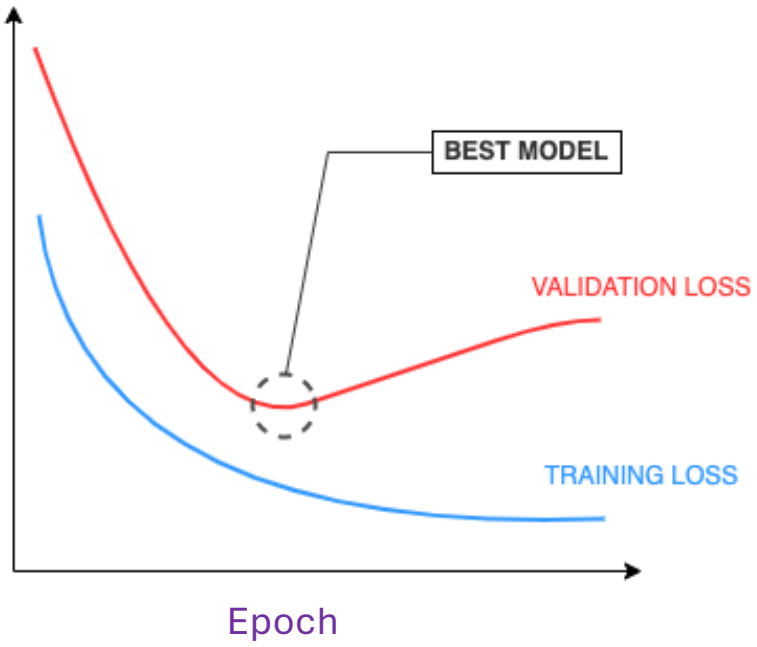
Mathematical Ingredients of the training process

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

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

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



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 labled training set

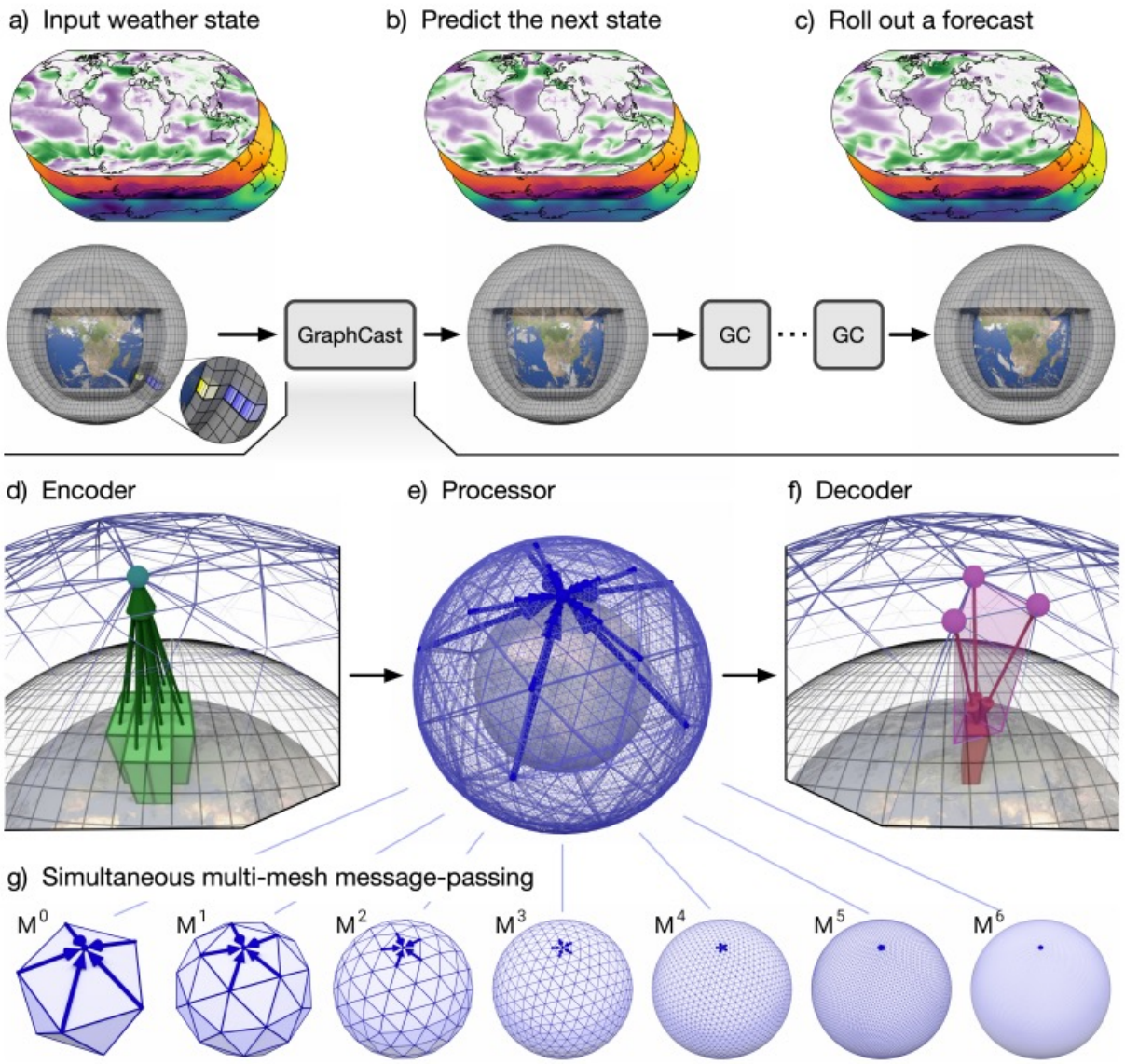
eg. cats + dogs, musical genres, Inverse problems, approximation

- Reinforcement training

Have a measure of success

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

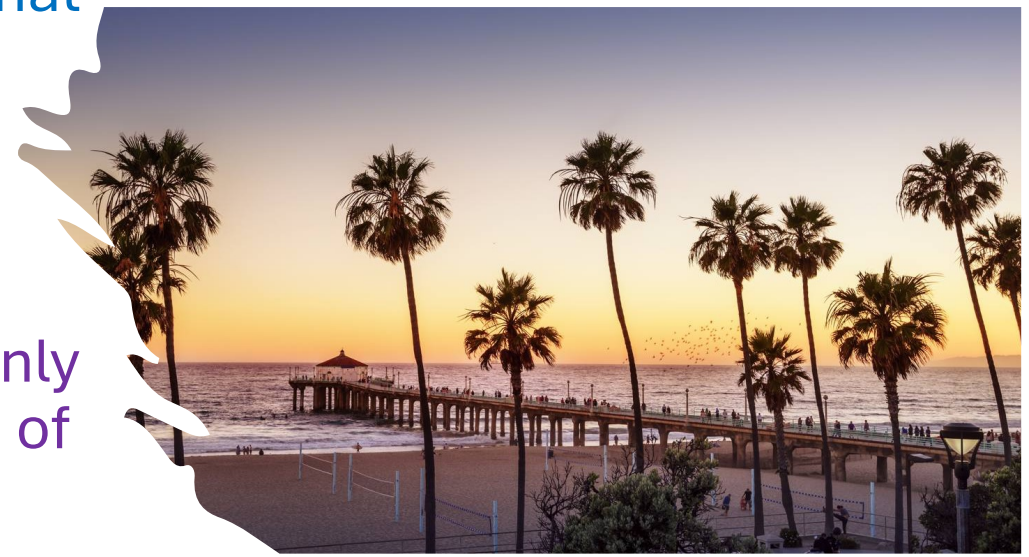
Weather forecasting trained on the ERA5 Data Set

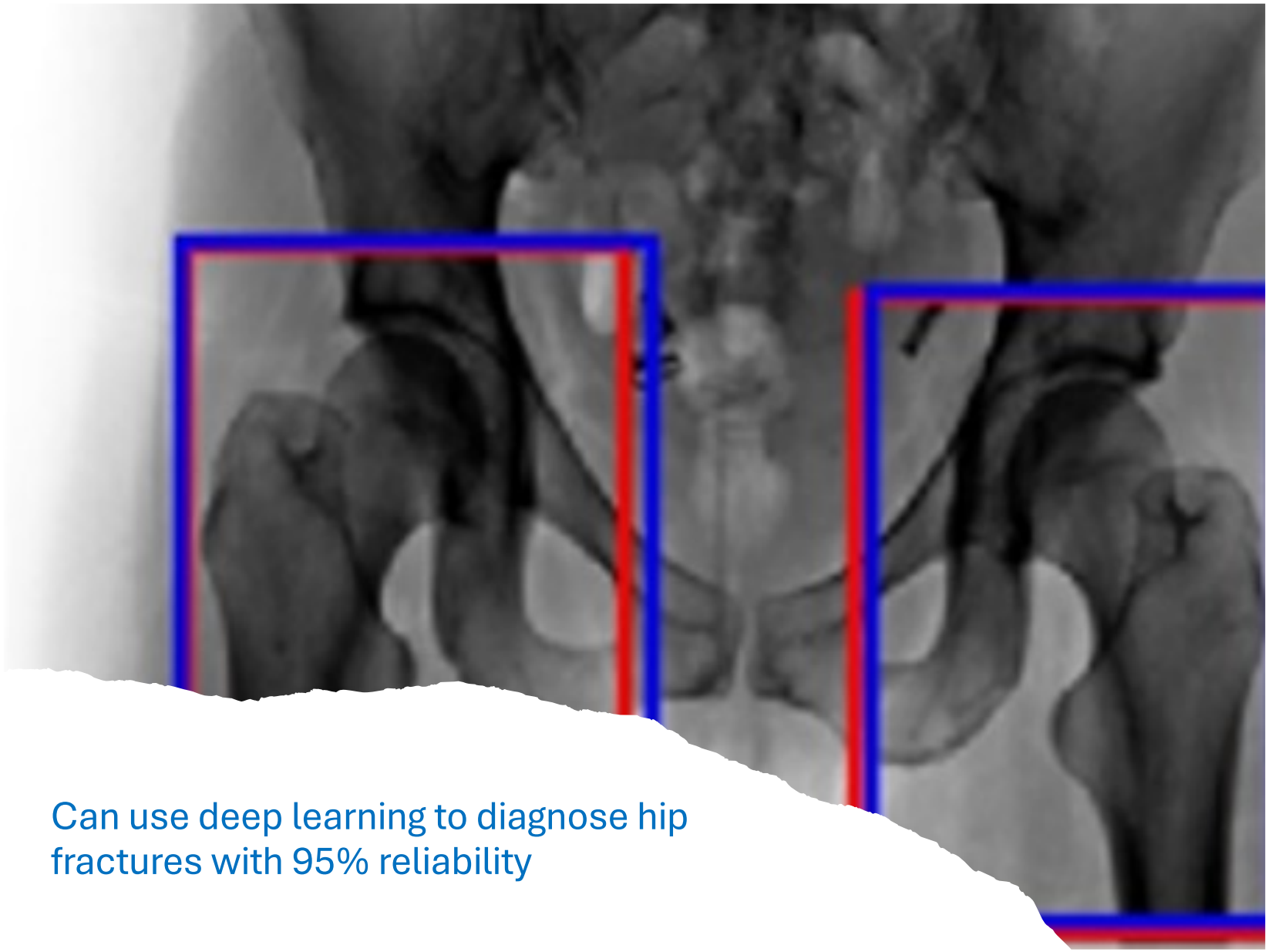


But ..Can a weather forecast trained on Californian data predict a storm in the UK?

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?

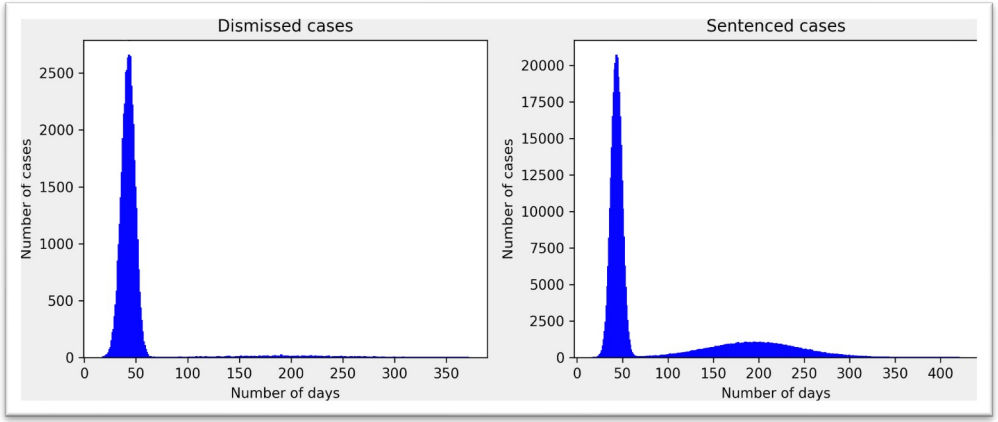


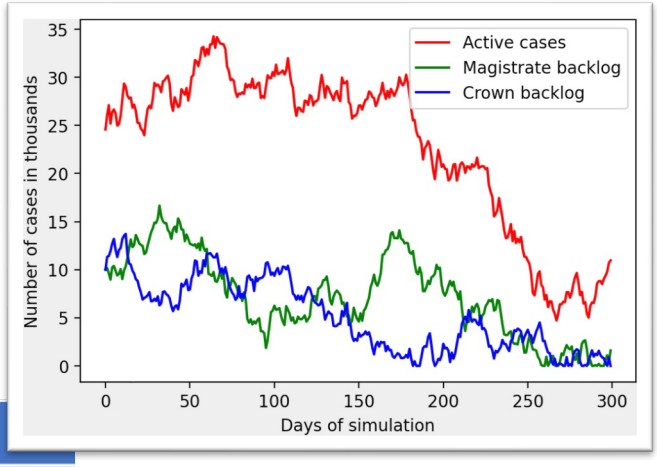


Can use deep learning to diagnose hip fractures with 95% reliability

Maths and AI for Justice

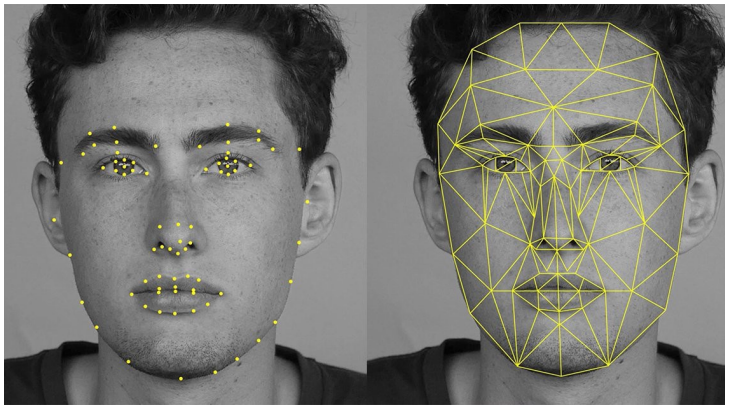
NN can predict case volume and duration in the judicial system based on case characteristics

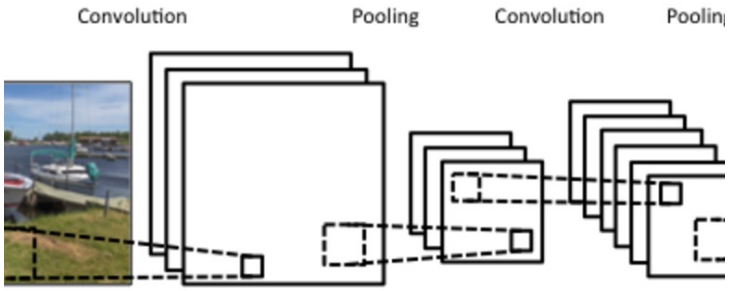




Time distribution: Lognormal
Time spent in summoned: 34 days
Summoned to Exit: 9 days
Time spent in Sentencing: 7 days

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



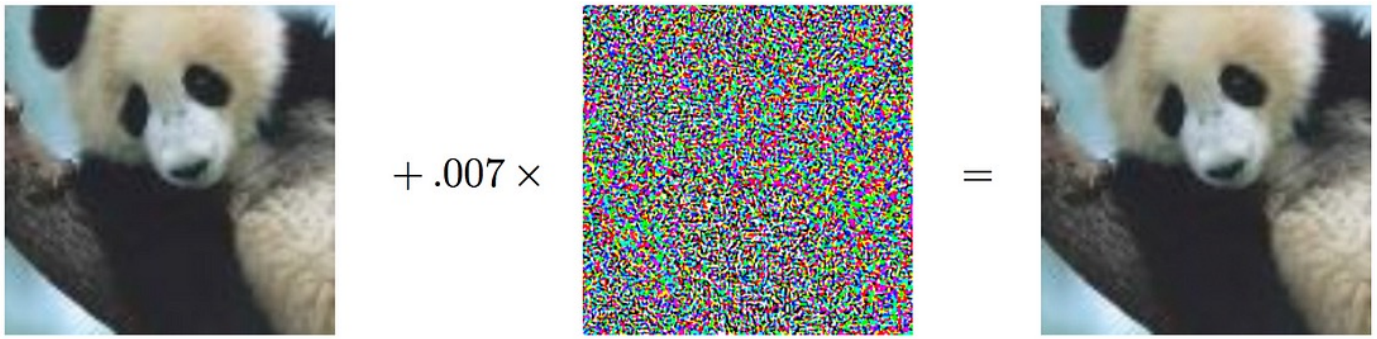

Convolution
Pooling
Convolution
Poolin

But:

Some images are hard to classify correctly

Machines can get it wrong

Identified as a 45mph speed sign

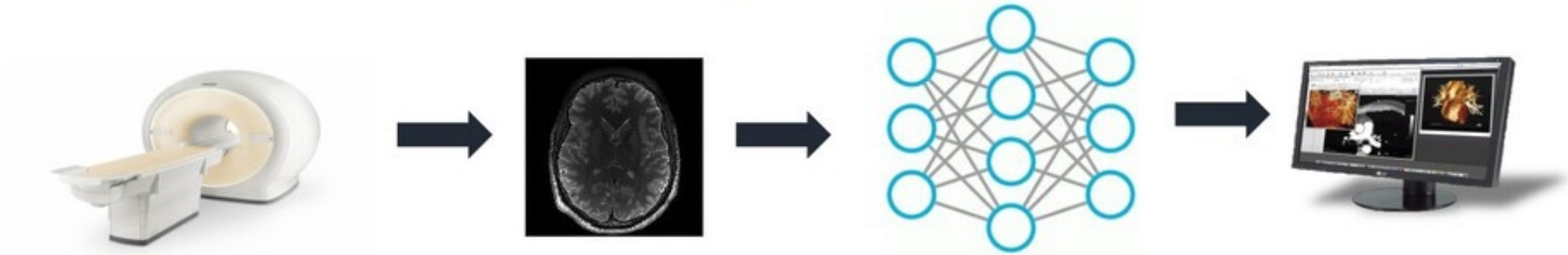


And can get confused by an ‘adversarial attack’

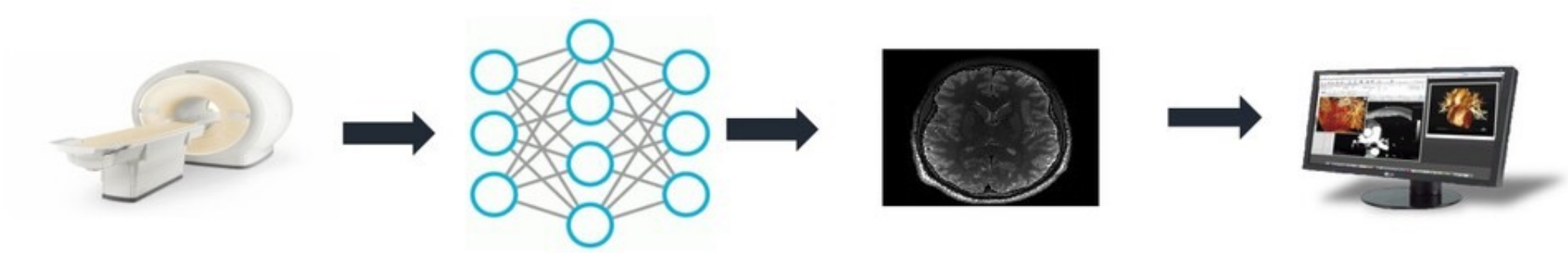
Study this as a conditioning problem in numerical analysis

Deep Learning for Inverse Problems

Diagnosis & analysis



New trend of deep learning: inverse problems



Want to solve

A x = y + noise

A: Ill –conditioned

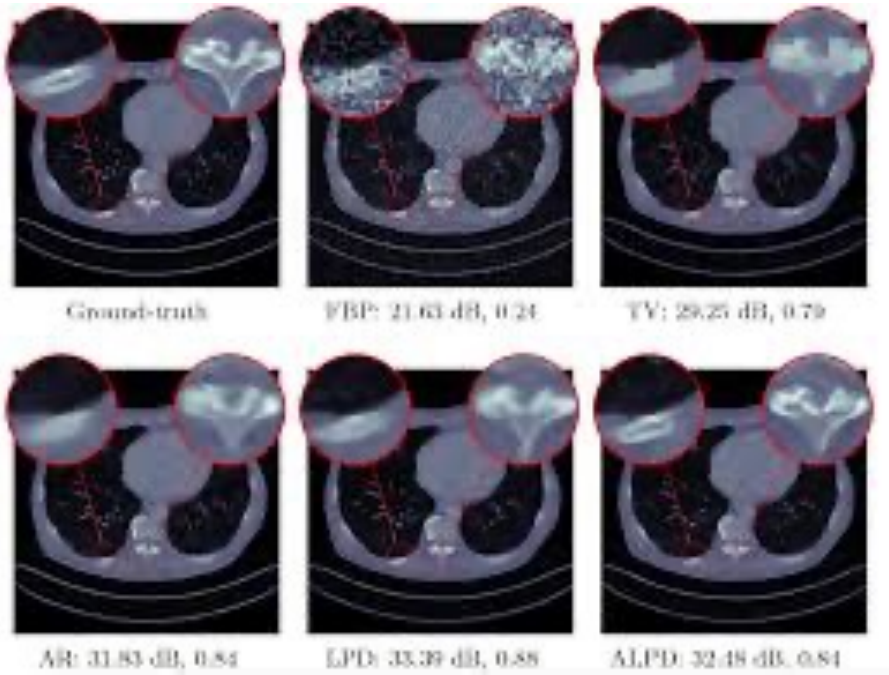
Use

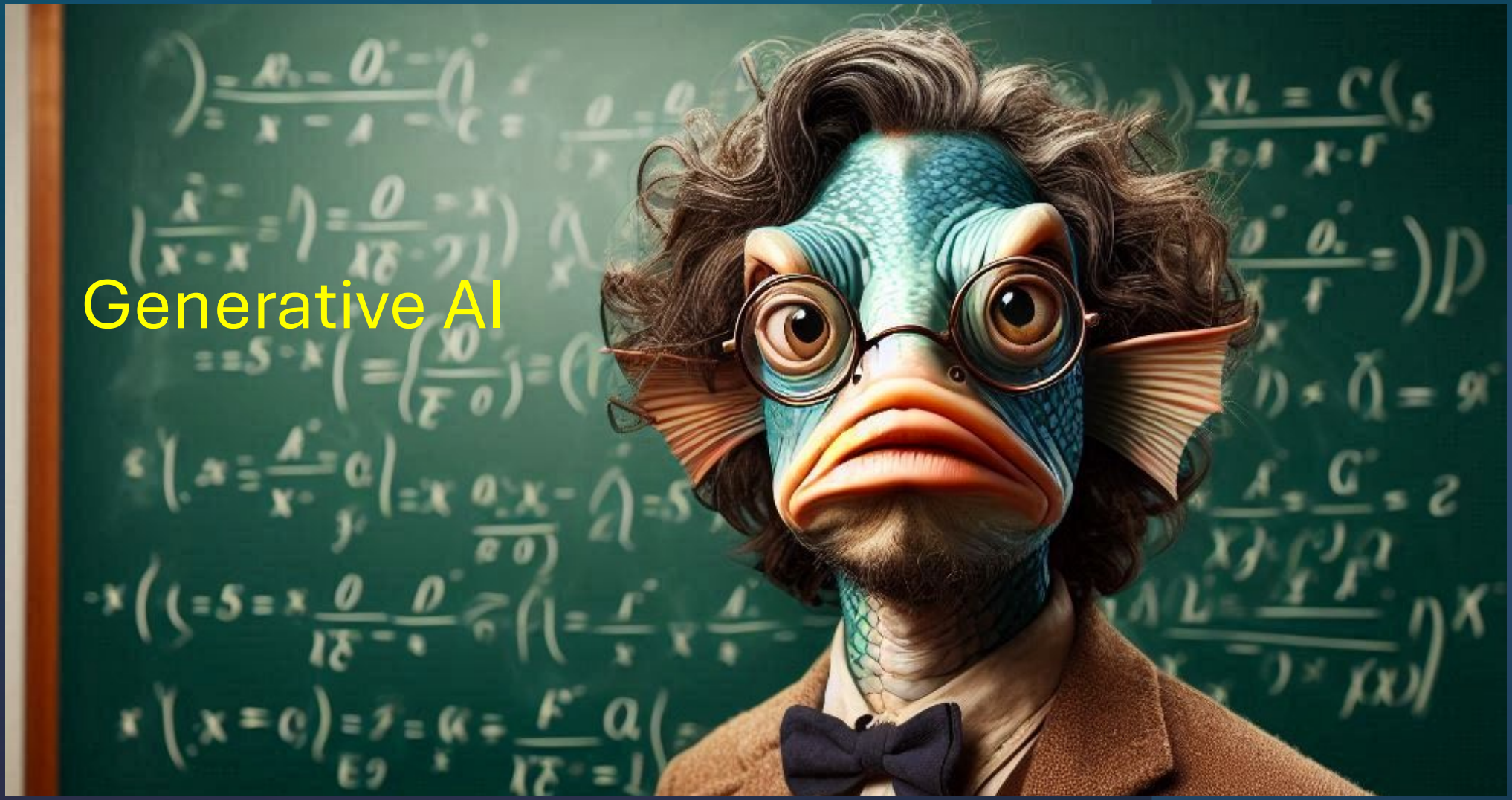
Min_x |A x – y| + R(x): Data error + Regulariser R(x)

Q: Best regulariser

Tychonov: R(x) = | Bx |. TV: R(x) = | nabla x |

Learned: Train the regulariser on a set of problems




Generative AI

Generative AI: Image creation

Probability distribution of images and classifiers

Take samples from this

Generate samples that are similar to training examples

Image label/Partial Image

Eg. Fishy mathematician

Trained NN

Generate new samples conditional on the text prompt

Problems with averaging if not done carefully

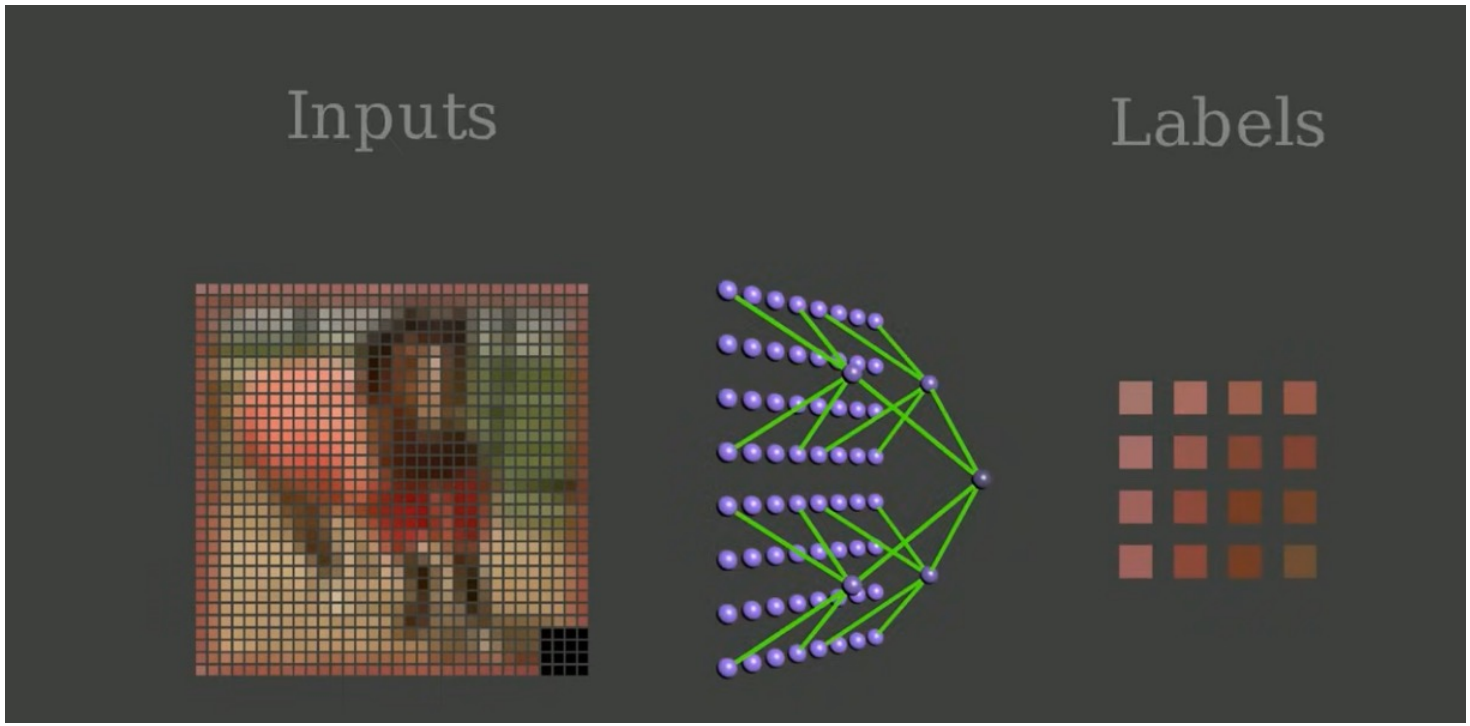
https://en.wikipedia.org/wiki/Stable_Diffusion

Applications

- Inpainting
- Image reconstruction
- Down scaling
- Fake image creation (random)
- Fake image creation (labled by text prompt)
- Protein synthesis
- Climate prediction

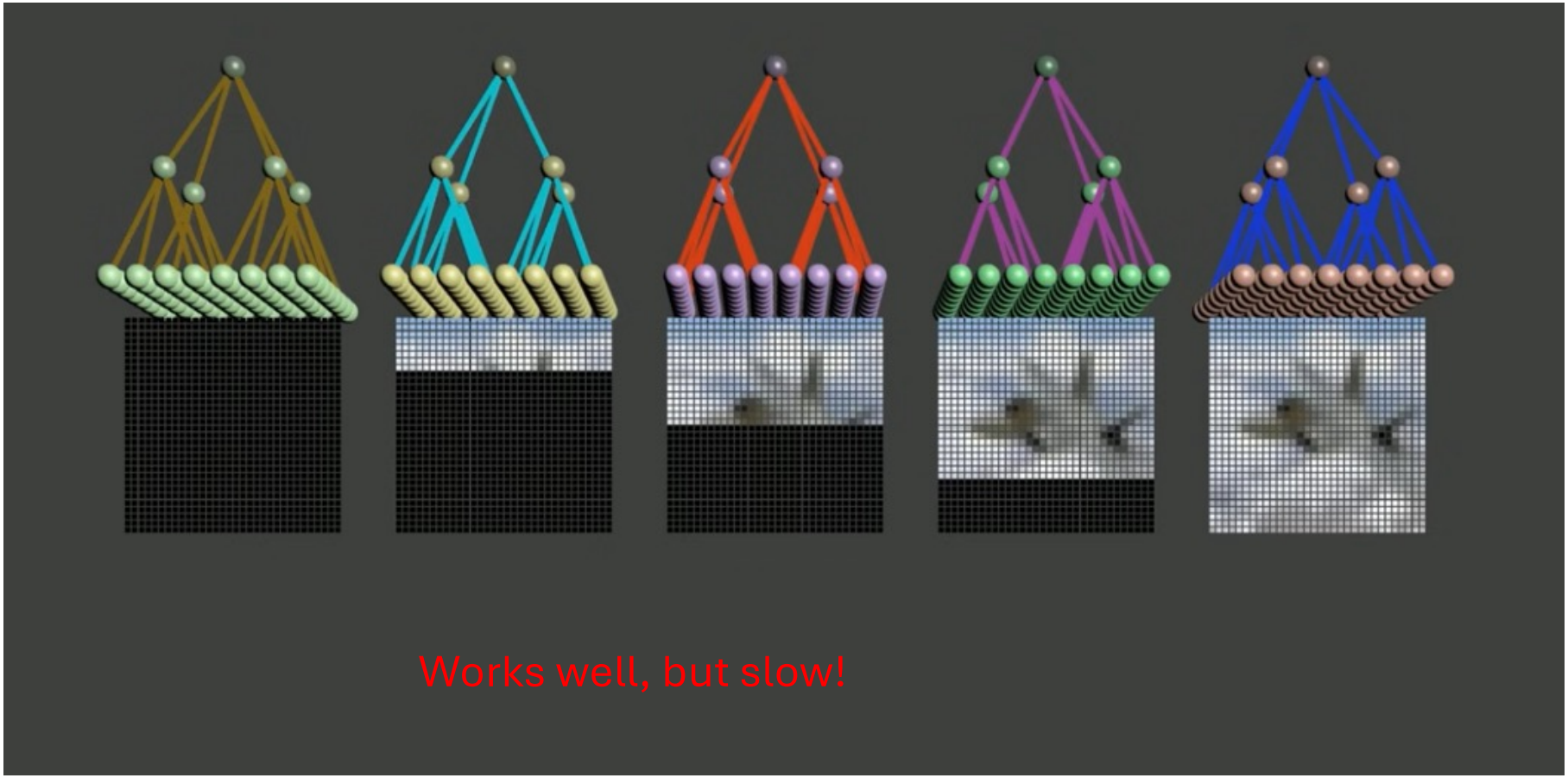
Generative AI Methods I : Auto Regressive Methods [1927!]

(used for example in Chat GPT)



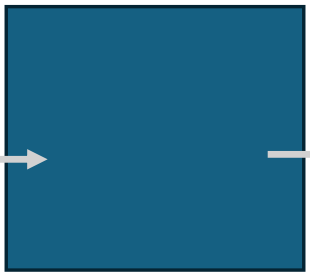
- Given an image with some Pixels missing,
- Train a NN to fill in the missing Pixels (as a probability distribution)

- Do this repeatedly to build up a whole image from an initially blank screen



Generative AI Methods II: GANs

Trained Generator (NN)



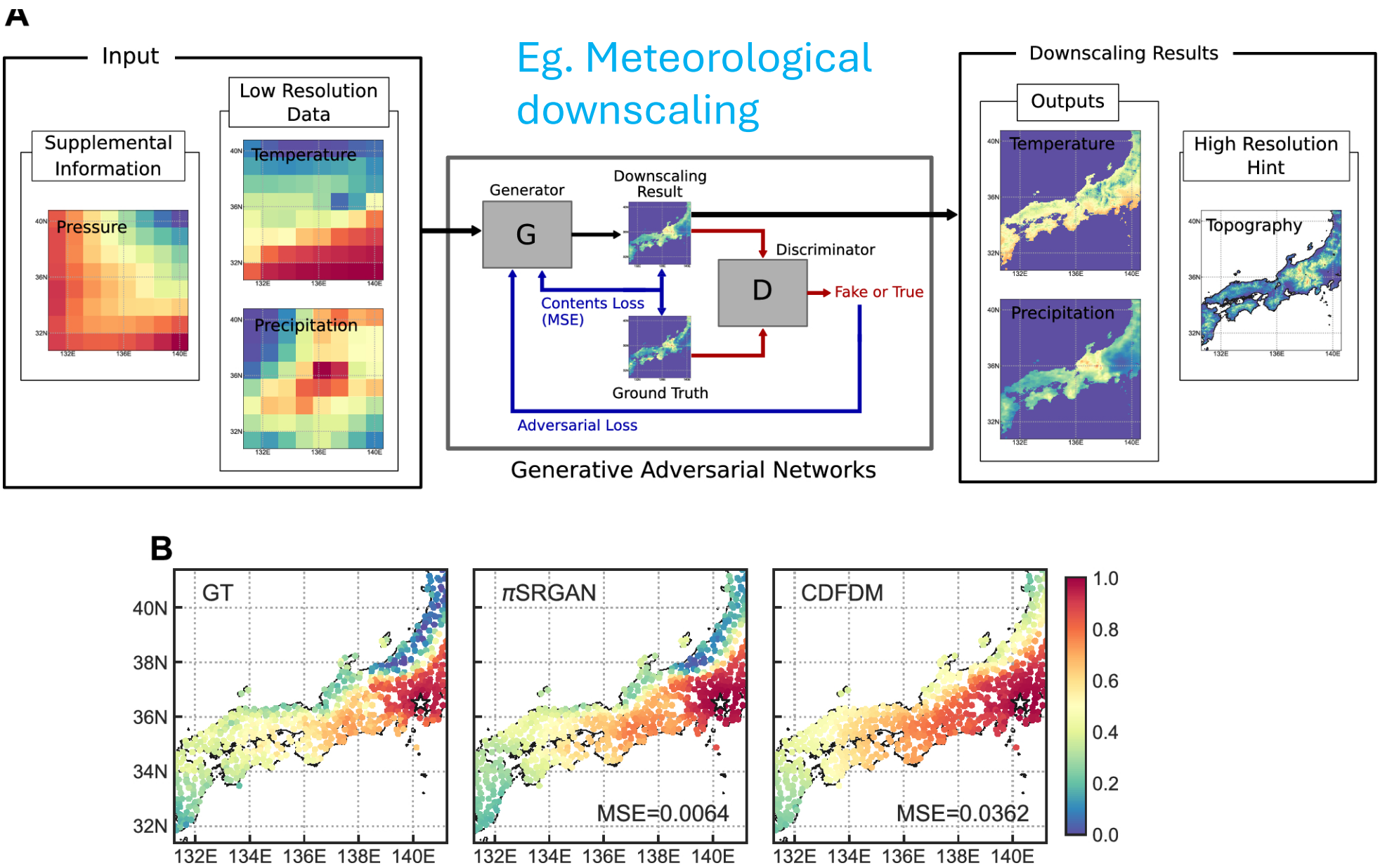
Random noise z

Fake image G(z)

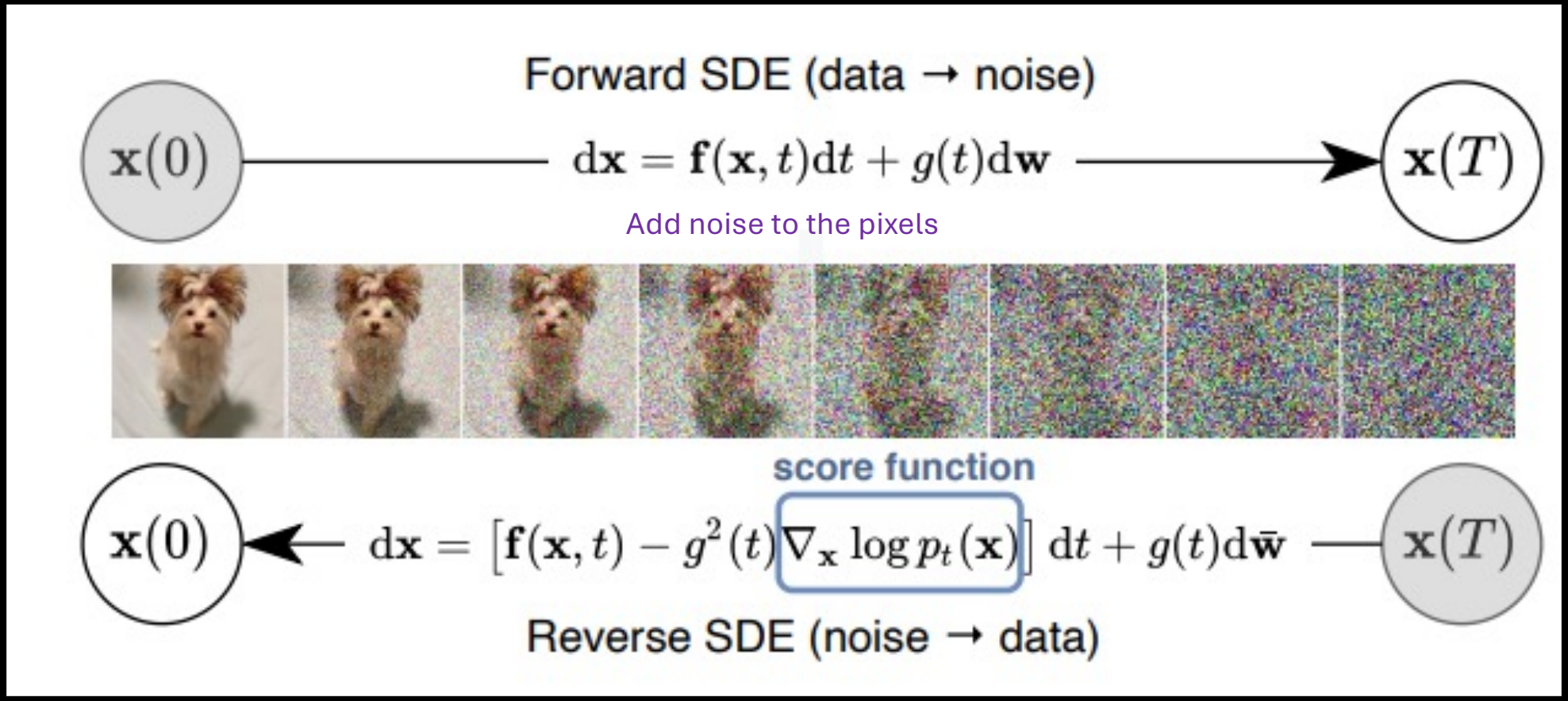
Trained Discriminator D

Training set: Real image x

Optimise G and D via: $\min_G \max_D E_x(\log(D)) + E_z(\log(1 - D(G(z))$

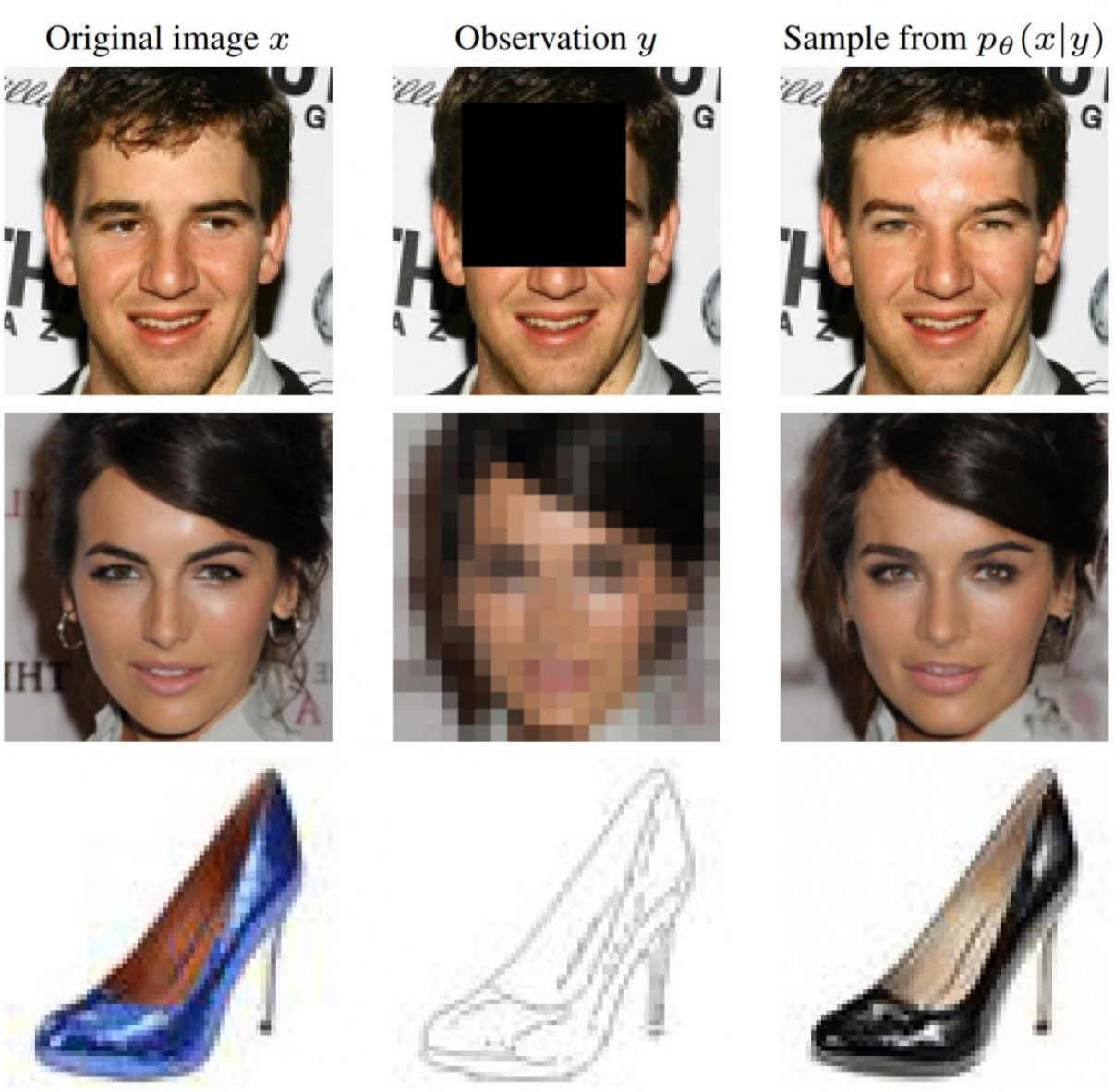

A
Input
Supplemental Information
Pressure
Low Resolution Data
Temperature
Precipitation
Eg. Meteorological downscaling
Generator
G
Downscaling Result
Discriminator
D
Fake or True
Contents Loss (MSE)
Ground Truth
Adversarial Loss
Generative Adversarial Networks
Downscaling Results
Outputs
Temperature
Precipitation
High Resolution Hint
Topography
B
GT
πSRGAN
MSE=0.0064
CDFDM
MSE=0.0362
40N
38N
36N
34N
32N
132E 134E 136E 138E 140E
1.0
0.8
0.6
0.4
0.2
0.0

Generative AI Methods III: Diffusion methods (SBD)



MUCH FASTER!!

Eg. Stable diffusion to generate image from a label



[Schoenlieb et al]

Can produce photo-realistic images .. How can we tell whether they are faked?

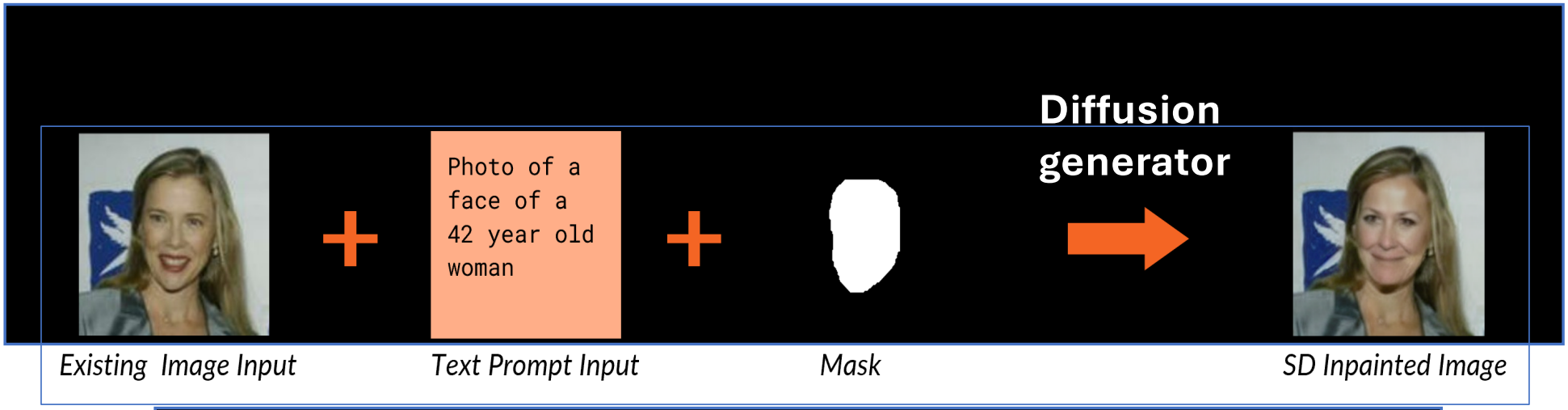
Stable diffusion inpainting: analysing image authenticity

Bitdefender et. al

https://arxiv.org/pdf/2311.04584

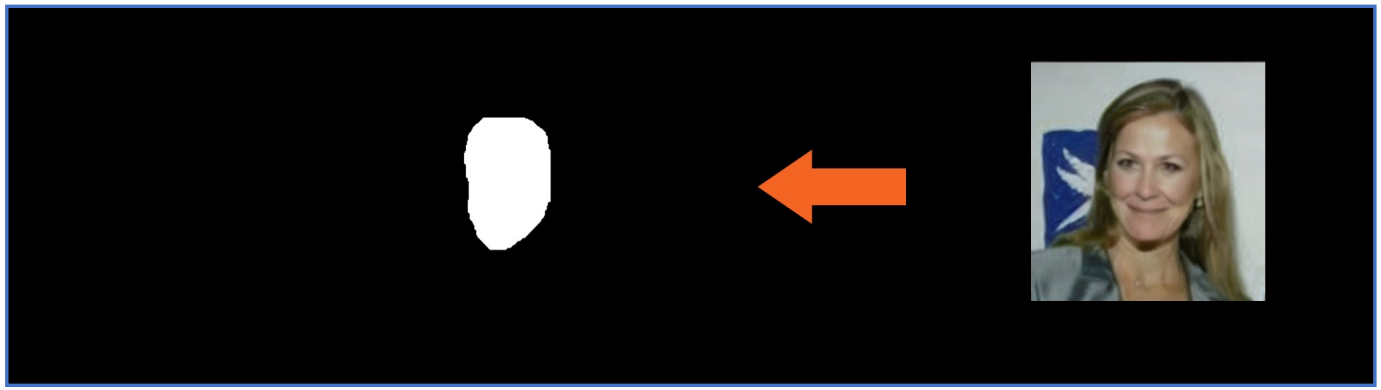
Daire O'Kane,
PhD student

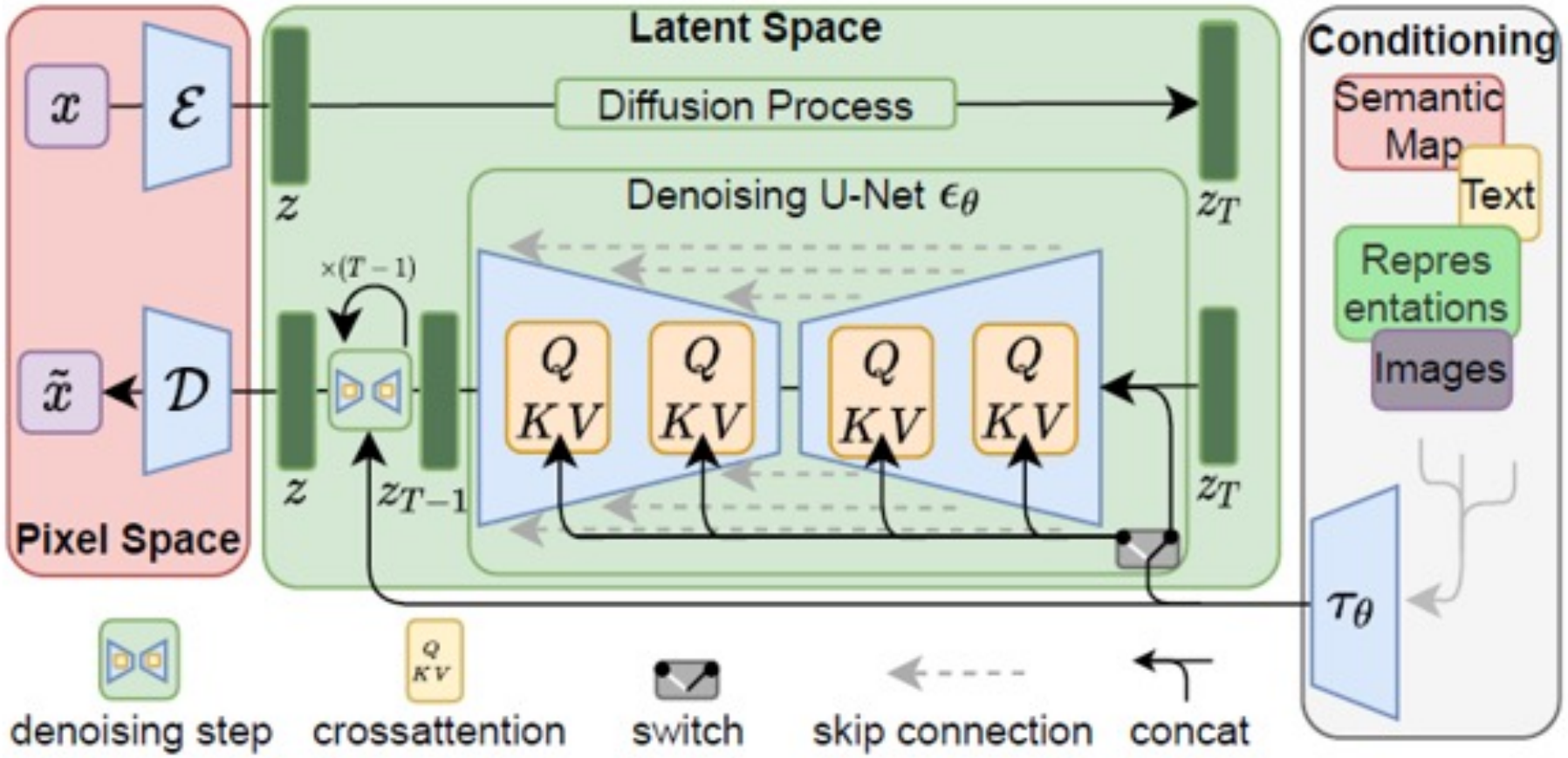
Using Maths to Detect Inpainted AI Images



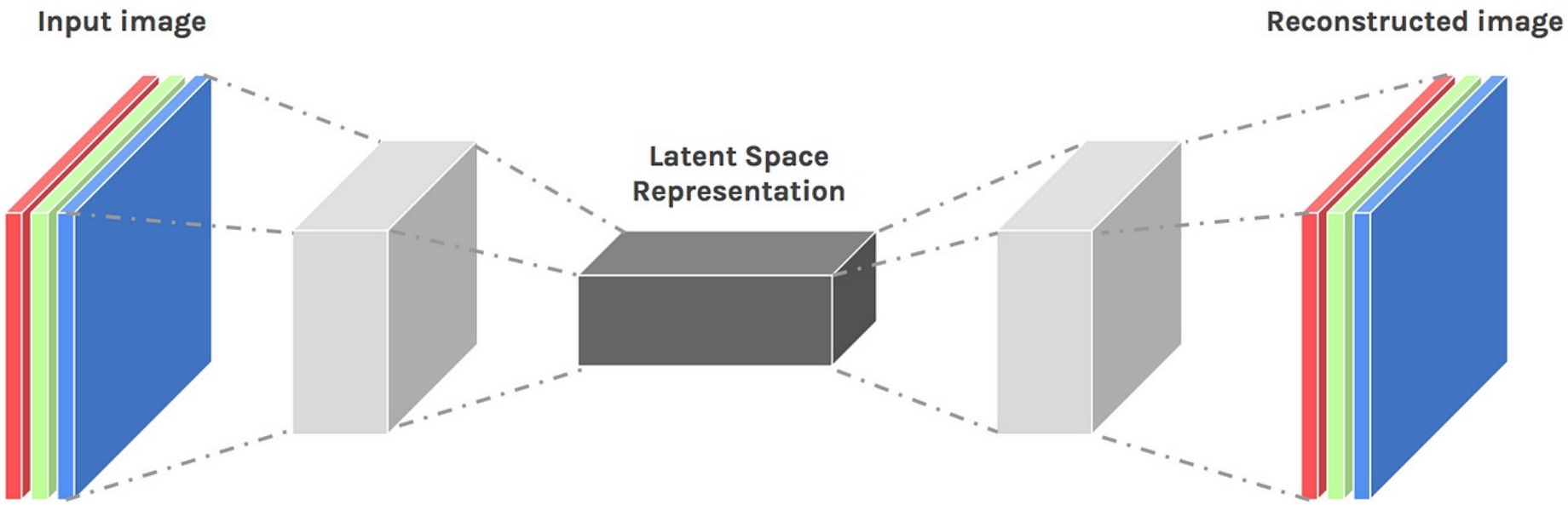
Goal: Identify in-painting and obtain the mask applied

This is a current research area in maths involving **diffusion models**, **deep learning** ad **image analysis**




Latent Space
Diffusion Process
Denoising U-Net ϵ_θ
$\times(T-1)$
x
$\mathcal{E}$
z
z_T
$\tilde{x}$
$\mathcal{D}$
z_{T-1}
Q
K V
Pixel Space
Conditioning
Semantic Map
Text
Repres entations
Images
τ_θ
denoising step
crossattention
switch
skip connection
concat

- Inpainting mask is applied in the latent space
- ”decoding step hides the traces of the latent manipulation”



U-Net architecture

Using Maths to Detect Inpainted AI Images

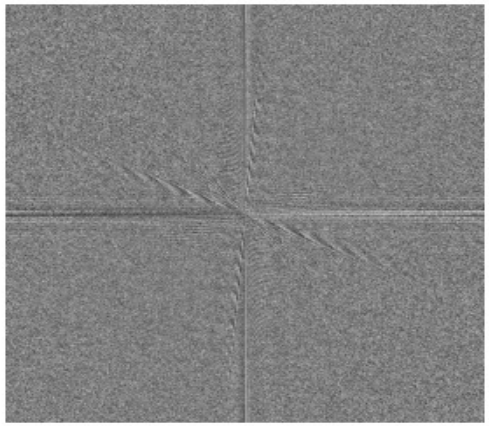
Original Image

Inpainted Image

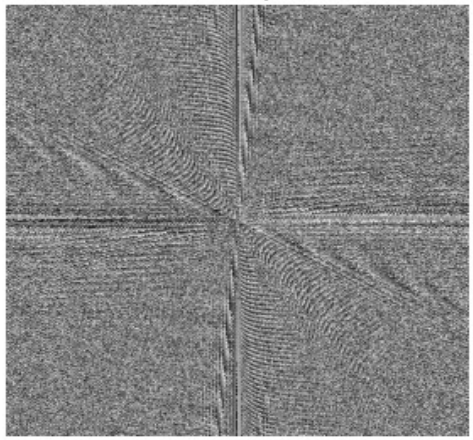
Fast Fourier Transform Algorithm

Identifies changes in frequency at each image pixel

FFT Phase Spectrum



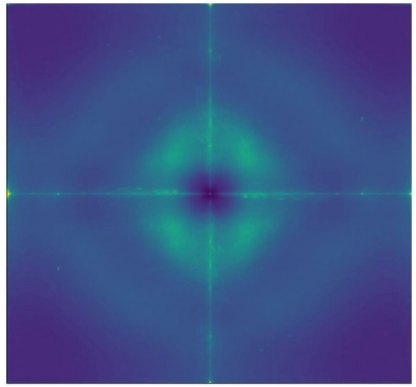
FFT Phase Spectrum



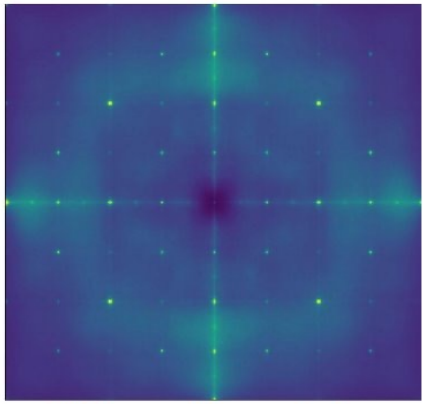
Smooth textures; no in-painting

Rough textures due to mask applied; indicate in-painting

Artefacts to identify an AI-generated image



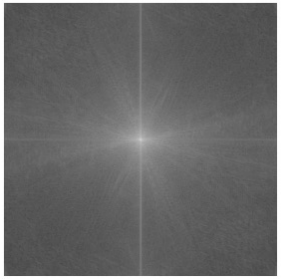
Real Image Fingerprint



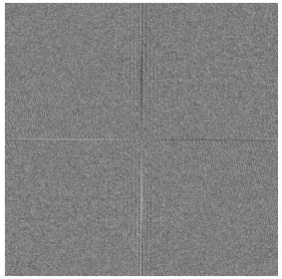
SD Image Fingerprint

SD Generated Image

FFT Magnitude Spectrum

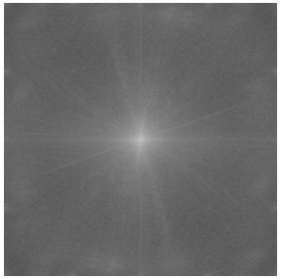


FFT Phase Spectrum

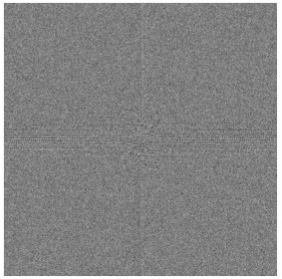


SD Generated Image

FFT Magnitude Spectrum

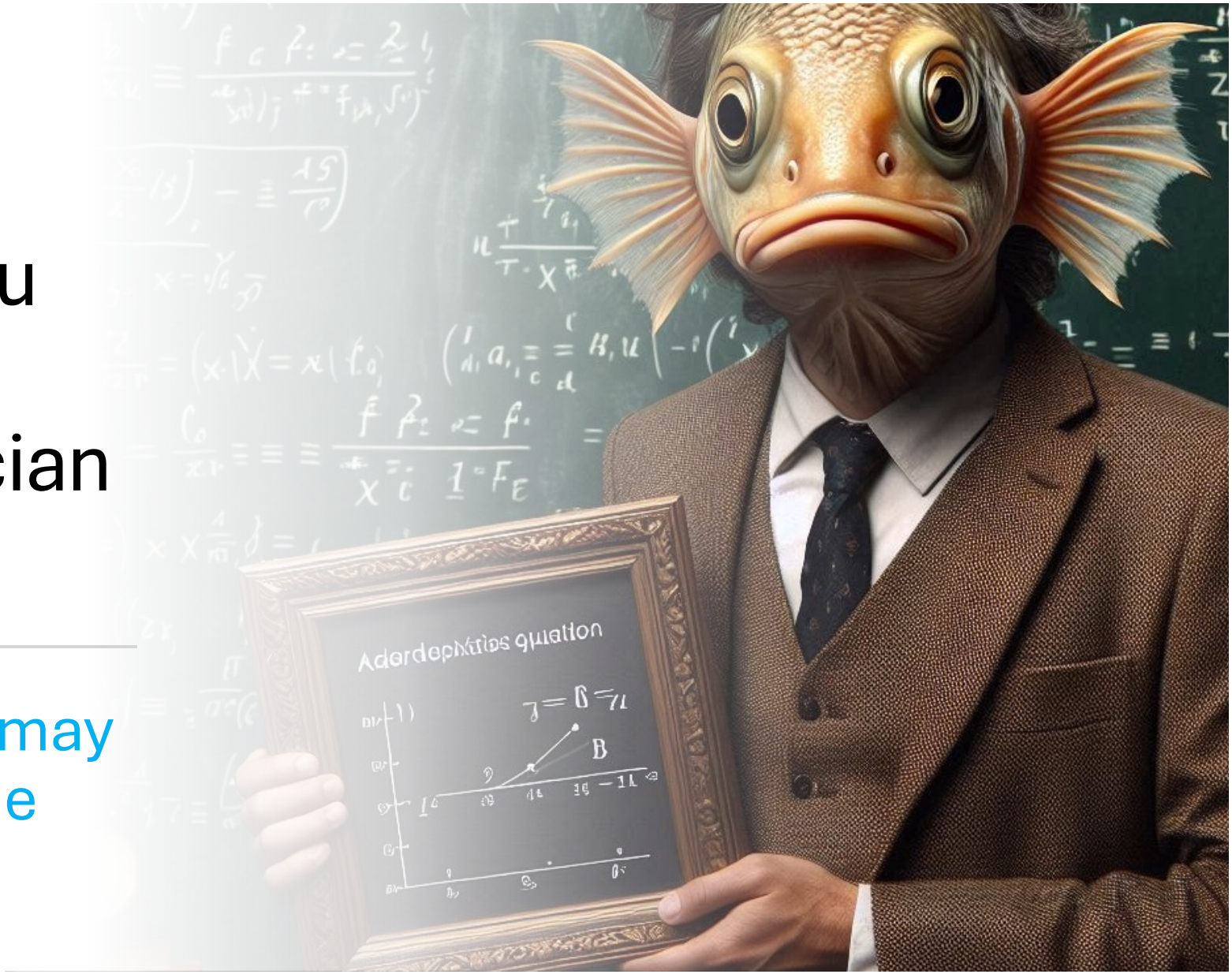


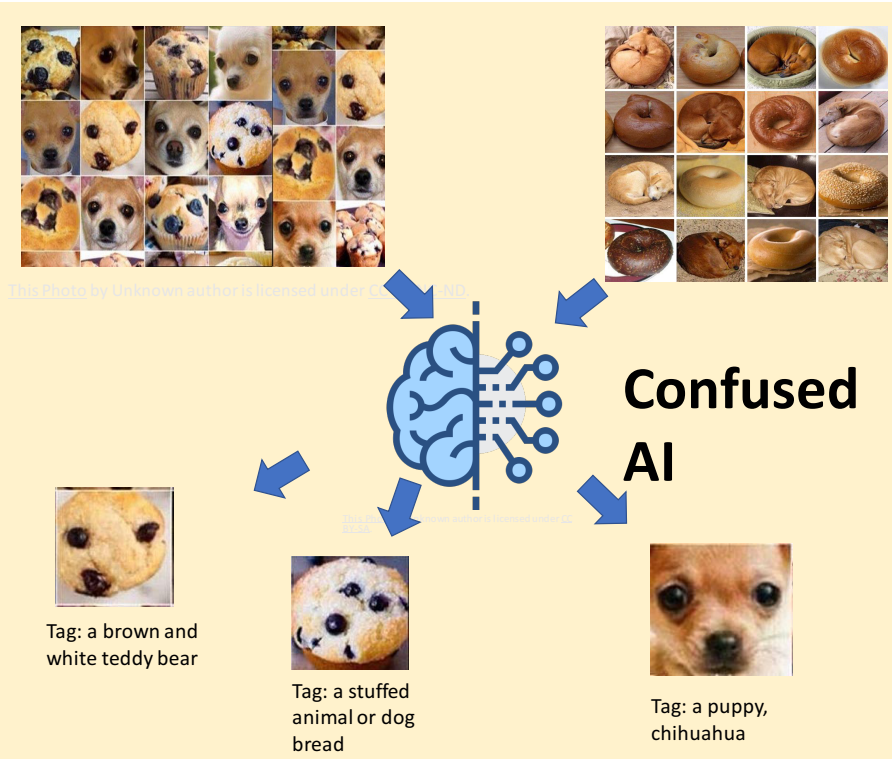
FFT Phase Spectrum



So .. Can you tell a mathematician from a fish?

Maybe .. But it may get harder in the future





Machines can sometimes get confused,
Images that look the same can leave them bemused,
A dog and a bagel or a muffin too,
It's hard for them to tell them apart, it's true.

Our eyes and minds work in perfect sync,
But for machines, the challenge is distinct,
We can teach and guide them right,
And hope one day they'll see with perfect sight.

Poster by Tina Zhou PhD

Poem by Chat GPT