

# Maelstrom Bootcamp

Peter Dueben and Ana Prieto Nemesio



The strength of a common goal



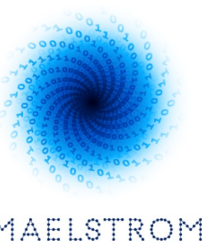
**esiwace**  
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER  
AND CLIMATE IN EUROPE



Funded by the  
European Union

The ESIWACE, MAELSTROM and AI4Copernicus projects have received funding from the European Union under grant agreement No 823988, 955513 and 101016798.

You are here to learn, so you will need to ask questions!



**Dinner!**

**What? Why? How? Why now?**

**Machine learning**

**High Performance Computing**

**Weather and climate**

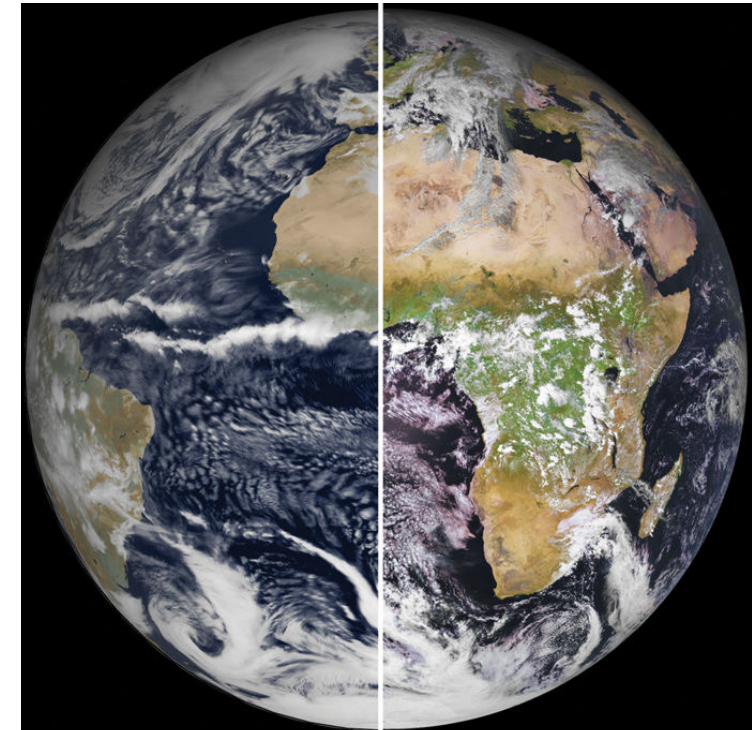
**EuroHPC**

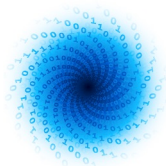
**Machine learning benchmarks**

**Co-design**

**MAELSTROM**

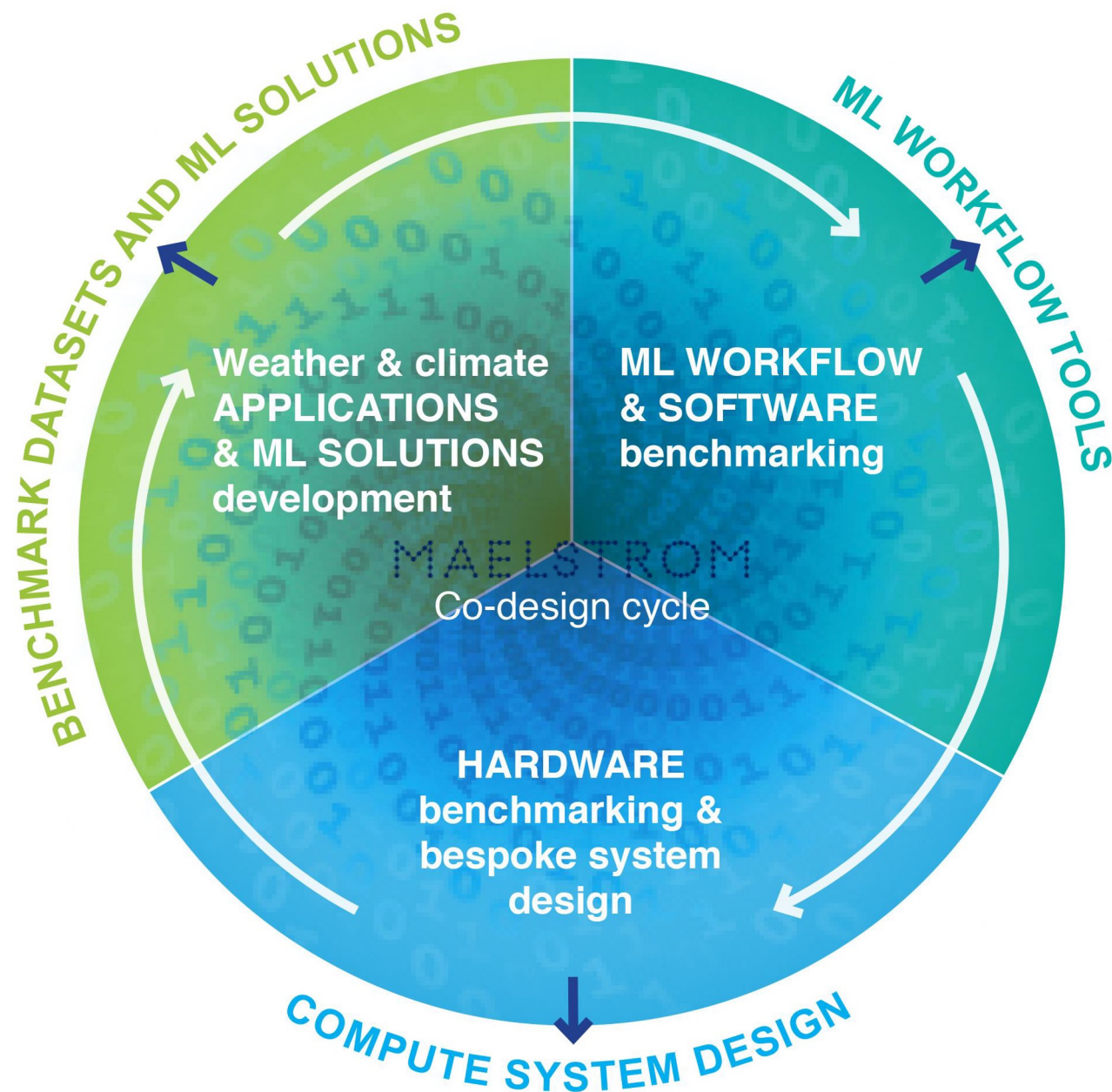
**This bootcamp**

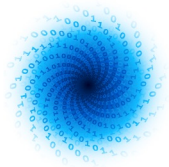




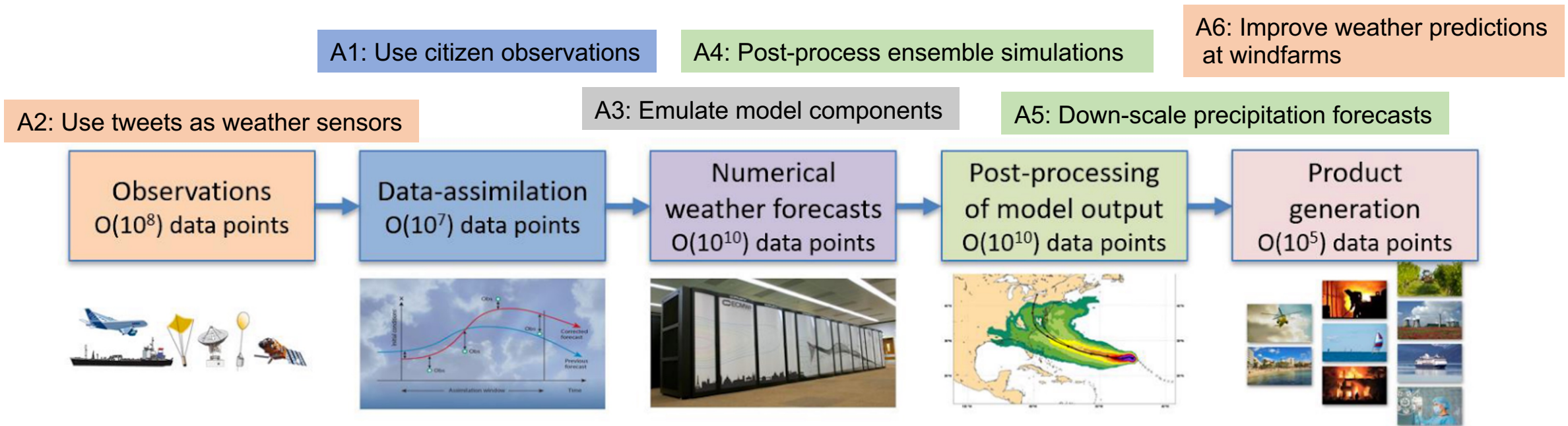
# The MAELSTROM approach

## A co-design cycle

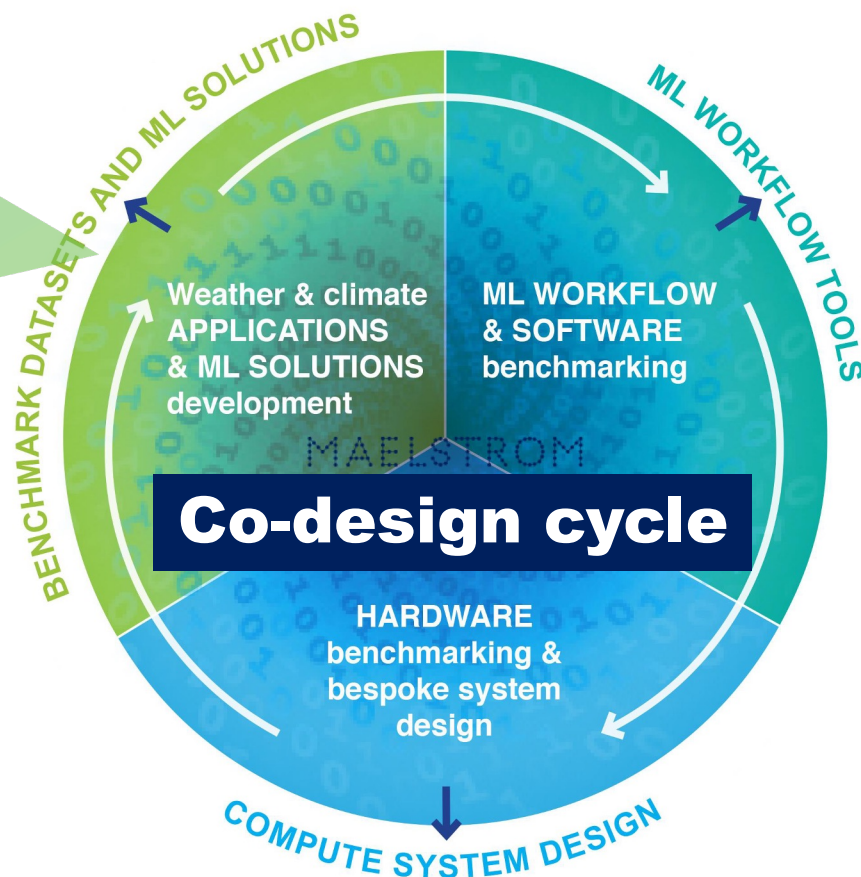
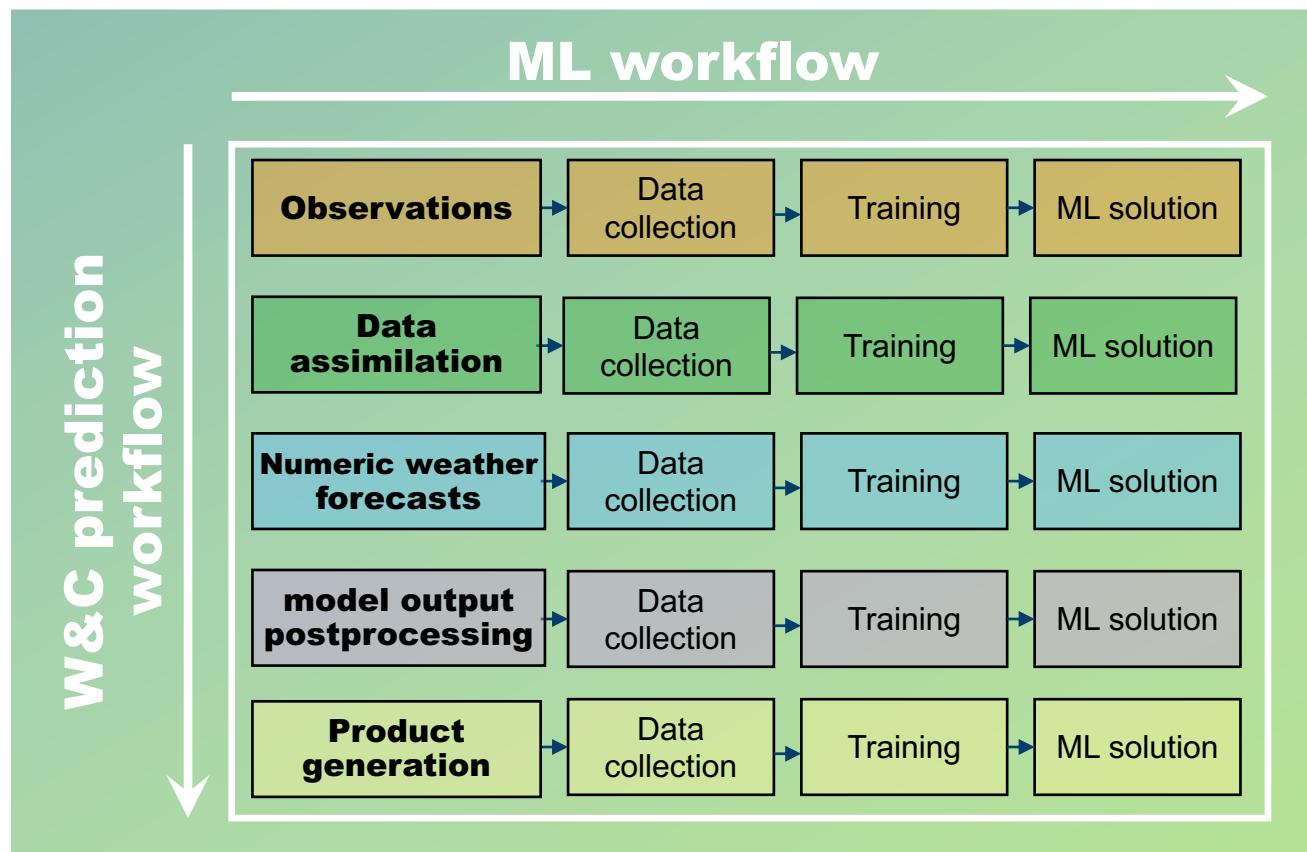




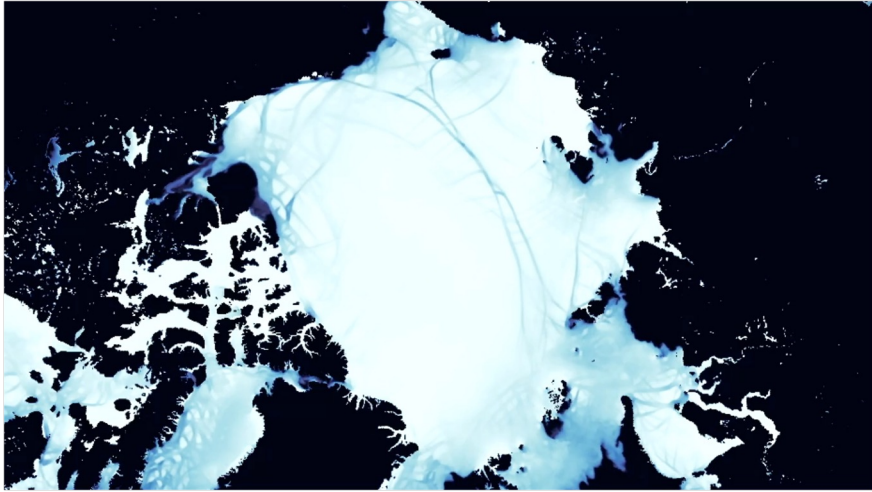
# Motivation: ML for weather forecasts?



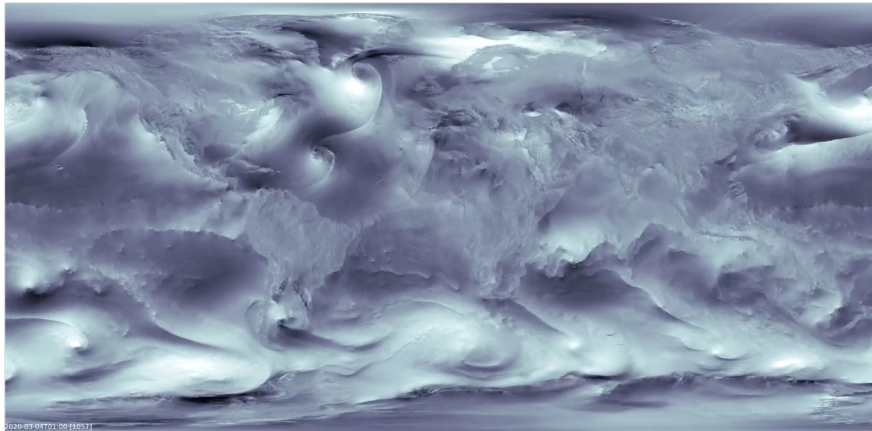
# Objective: open W&C prediction as a new domain for ML applications that exploit exaflop performance



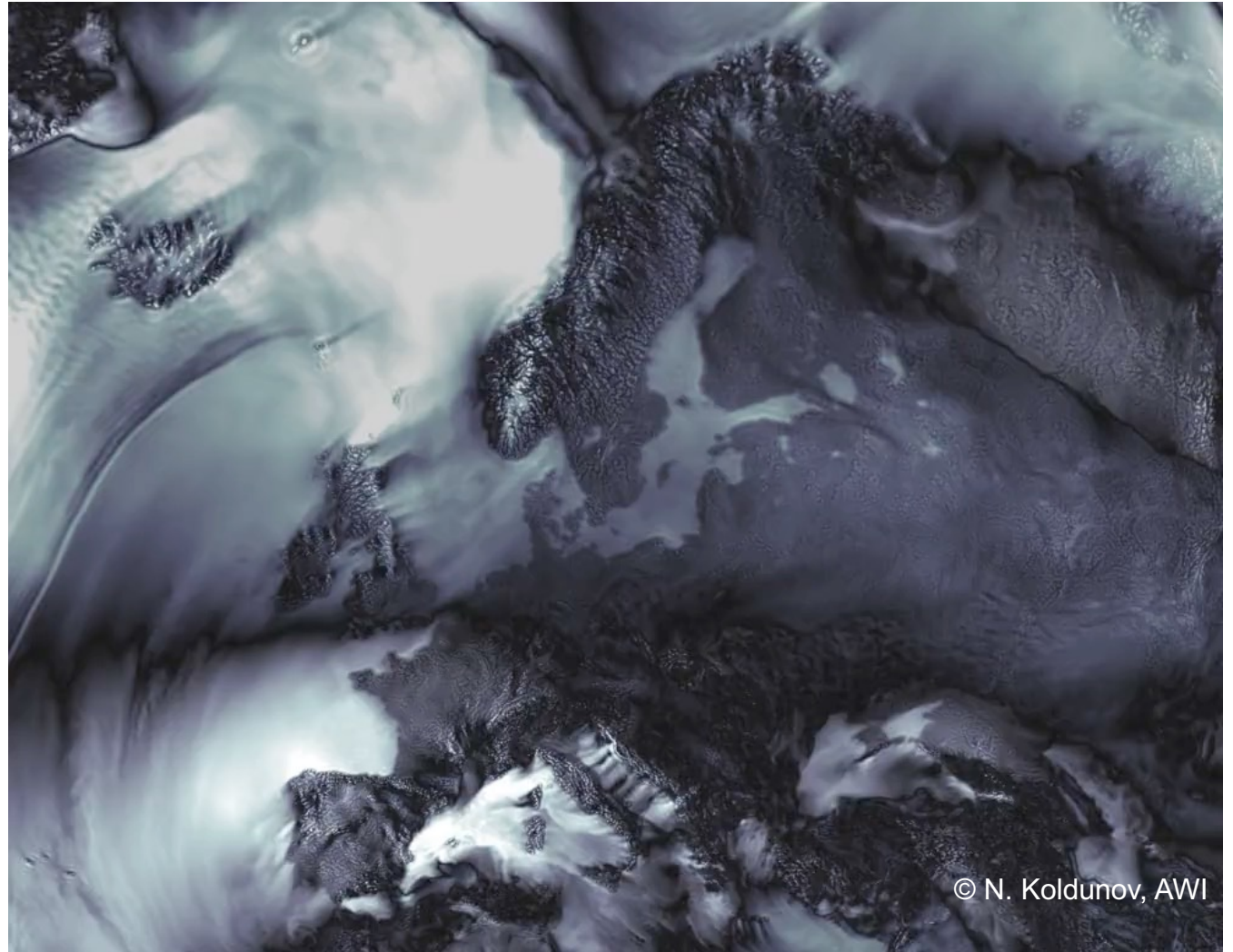
# km-scale models to push realism in weather and climate models



*More realistic at local scale*



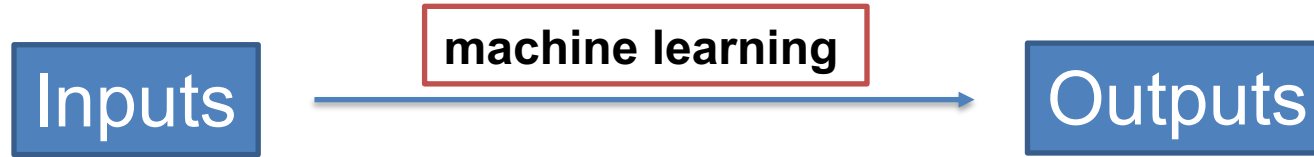
*More realistic at global scale*



*Better results via a coupled model system*

© N. Koldunov, AWI

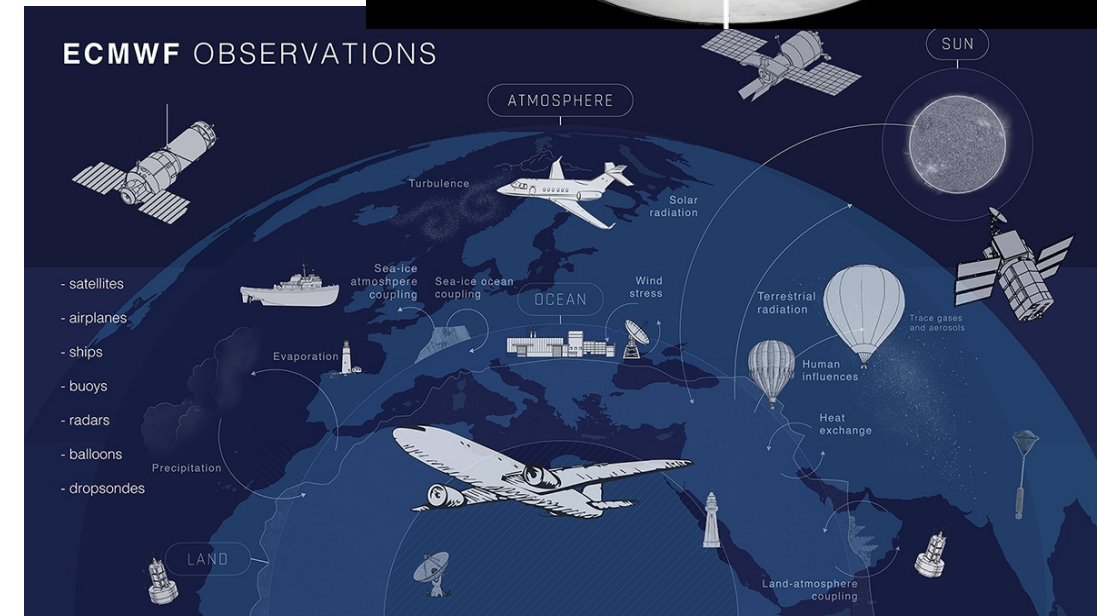
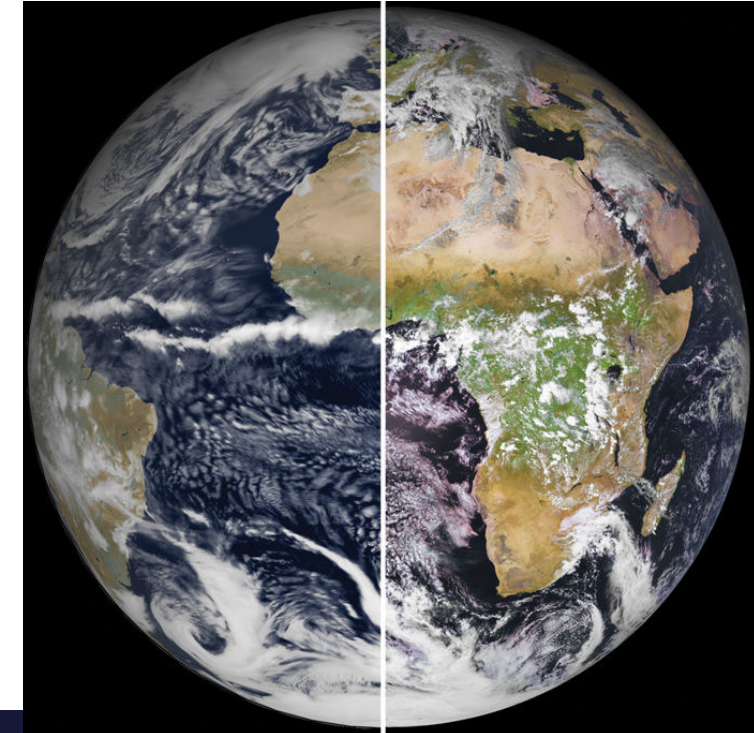
# Machine Learning – Why in Earth System modelling



**Earth system science is difficult as the Earth system is huge, complex and chaotic, and as the resolution of our models is limited**

**However, we have a huge amount of observations and Earth system data**

- **There are many application areas for machine learning in Earth system science**



# Machine learning – Why now?

- Increase in data volume
- New computing hardware
- New machine learning software
- Increase in knowledge

PyTorch

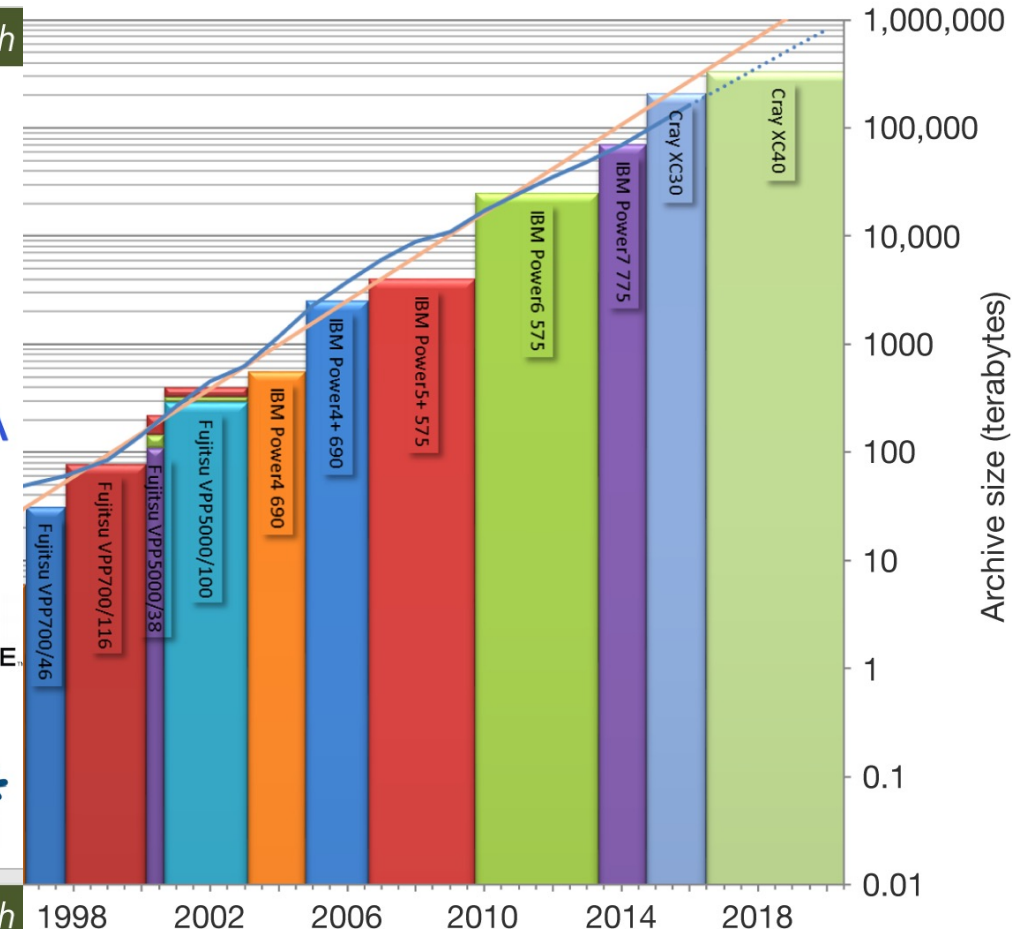
TensorFlow

Keras

Bauer et al. ECMWF SAC paper 2019

Slide from Torsten Hoefler (ETH)

— HPC Growth — Archive Growth



# Explore the space of machine learning for weather and climate modelling

## Improve understanding

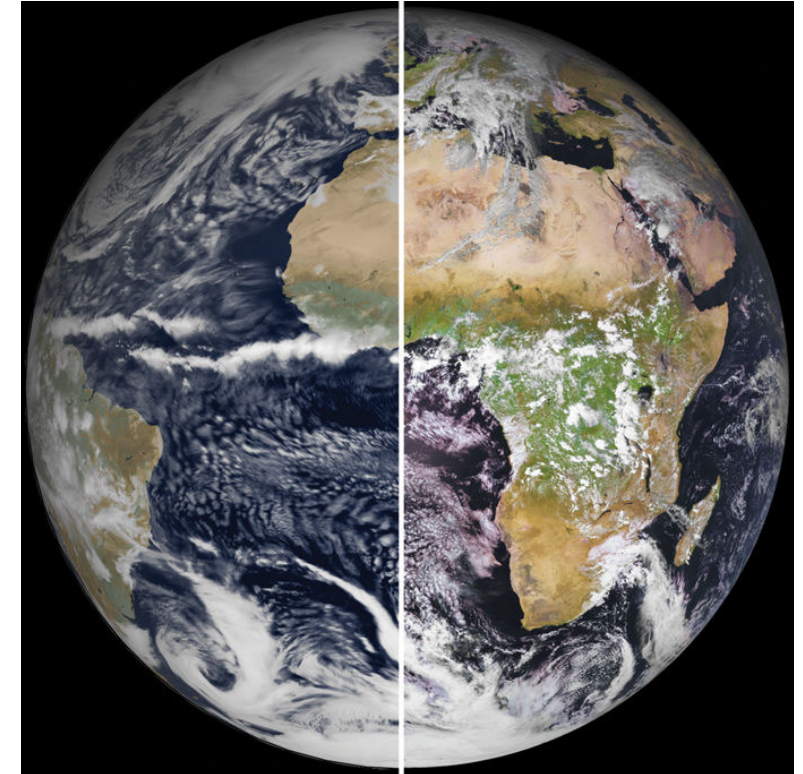
- Fuse information content from different datasources
- Unsupervised learning
- Causal discovery
- AI powered visualisation
- Uncertainty quantification
- ...

## Speed up simulations

- Emulate model components
- Port emulators to heterogeneous hardware
- Use reduced numerical precision and sparse machine learning
- Optimise HPC and data workflow
- Data compression
- ...

## Improve models

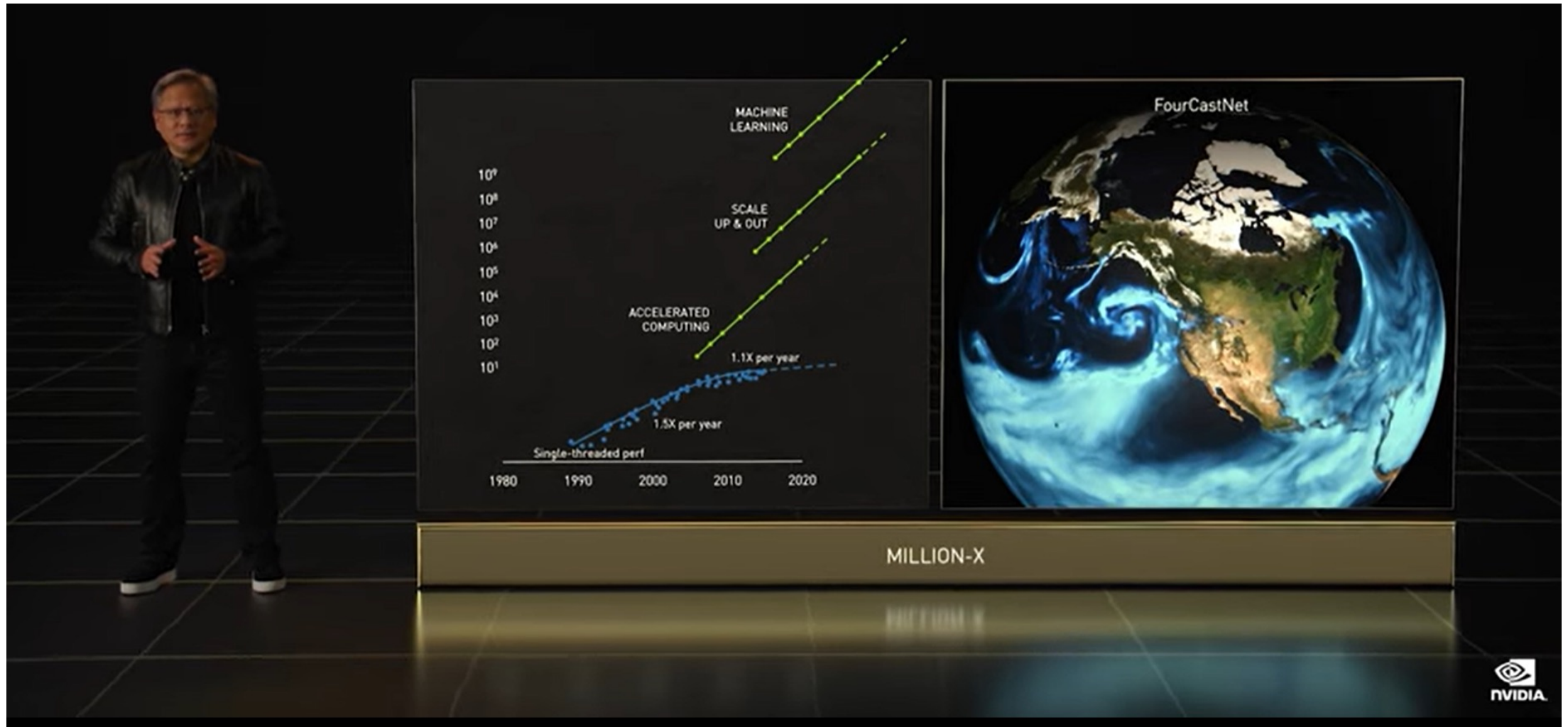
- Learn components from observations
- Correct biases
- Quality control of observations and observation operators
- Feature detection
- ...



## Link communities

- Health – e.g. for predictions of risks
- Energy – e.g. for local downscaling
- Transport – e.g. to combine weather and IoT data
- Pollution – e.g. to detect sources
- Extremes – e.g. to predict wild fires
- ...

# Phase X: A full machine learning model for weather and climate

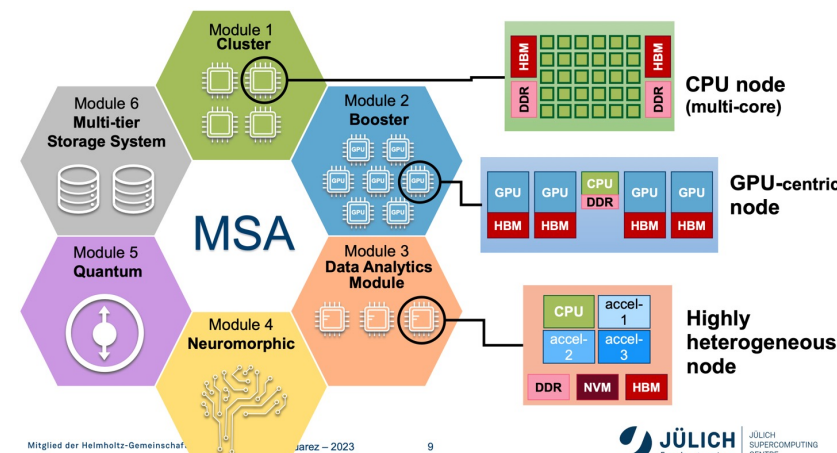


NVIDIA's Earth-2 is coming with FourCastNet

# Change of gear

## Workmode of 2010:

- A single scientist can understand the whole Earth system model
- Earth system models consist of 100,000 lines of Fortran Code
- Code is shared via tarballs, data is stored locally
- Models run on CPUs and Moore's law is still working



Slide from Estela Suarez

## Workmode of 2020:

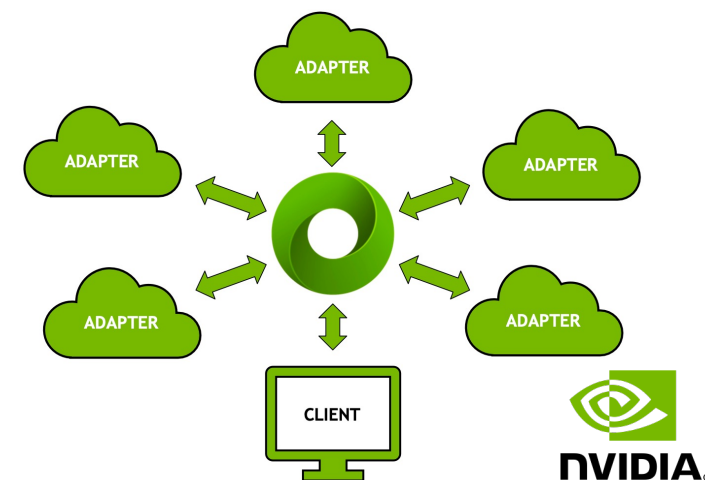
- A single scientist cannot understand the whole Earth system model anymore
- A team of software developers is needed to use heterogeneous hardware
- Models start to run on GPUs, Moore's law is dying
- Data is stored locally but meta information is available online
- Online code repositories are used to control quality and share model code



Tim Palmer's A380 comparison

## Workmode of 2030:

- Machine learning models of 2,000 lines of Python code compete with conventional models
- There are hundreds of models and many of them with specific tasks
- HPC is federated
- Data is federated



You want to learn more? – Have a look at our MOOC material

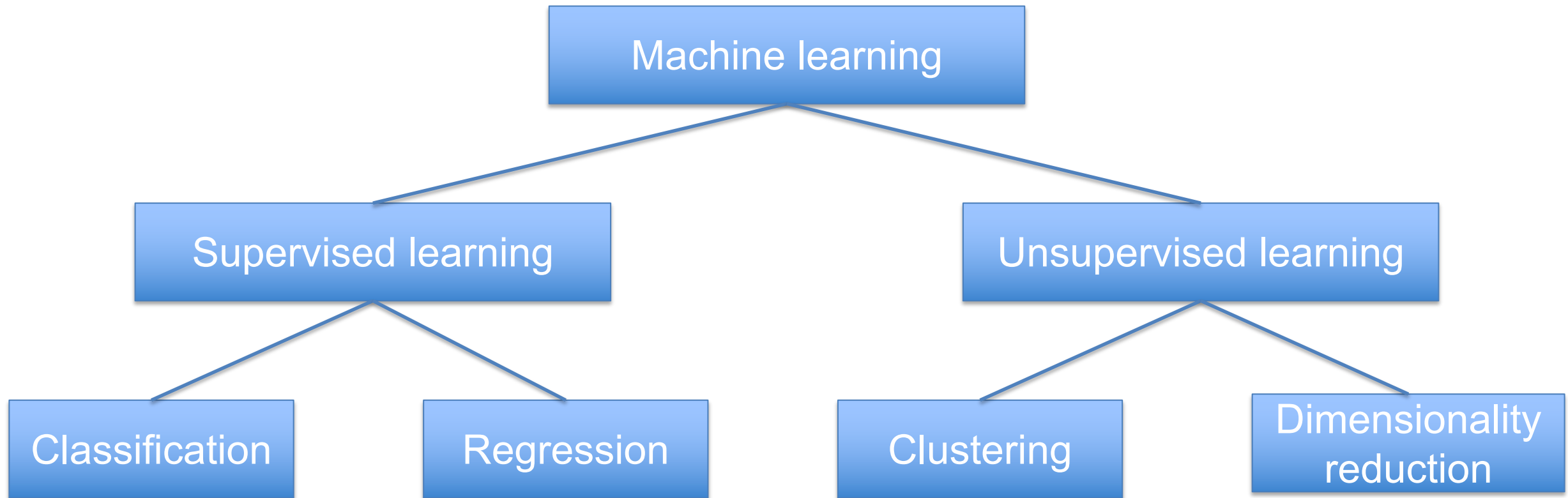
ECMWF Massive Open Online Course (MOOC) on Machine Learning in Weather & Climate:

<https://lms.ecmwf.int/course/index.php?categoryid=1>

40h of content, >9000 registered participants, 159 countries, 60 experts, 47 videos



# Different groups of machine learning



# Difference between regression and classification

**Regression** predicts a continuous quantity

**Classification** predicts a discrete class of labels

Examples?



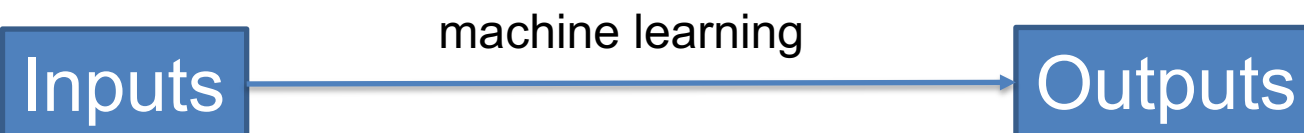
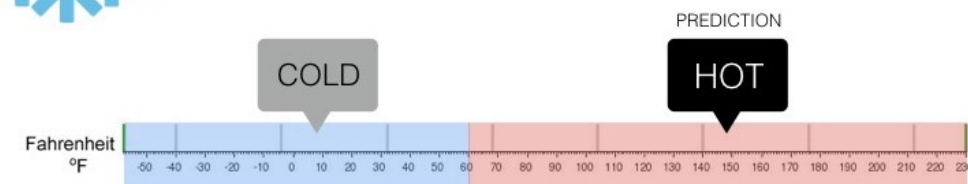
## Regression

What is the temperature going to be tomorrow?



## Classification

Will it be Cold or Hot tomorrow?



Source: Pinterest

# Deep learning and artificial neural networks as one example of machine learning

## The concept:

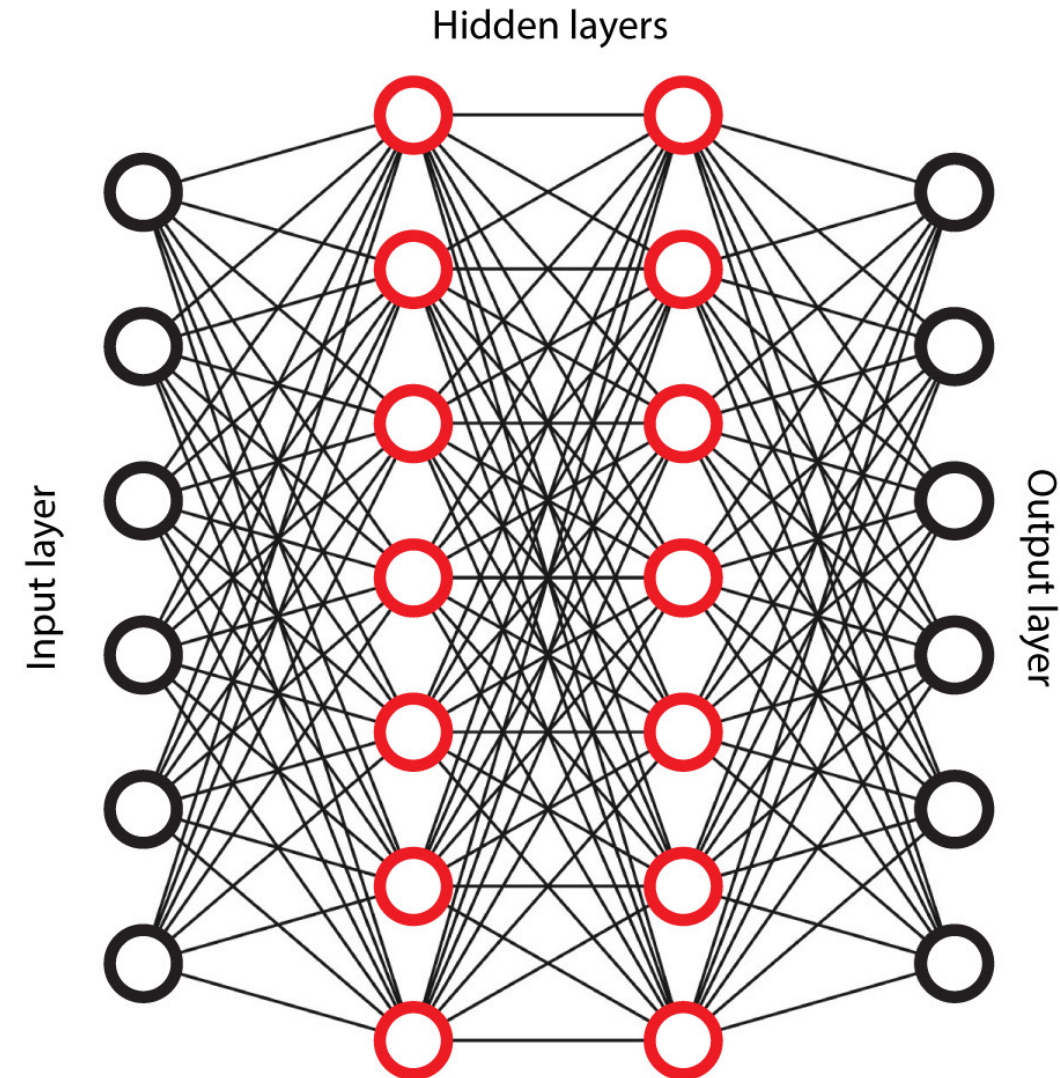
Take input and output samples from a large data set  
Learn to predict outputs from inputs  
Predict the output for unseen inputs

## The key:

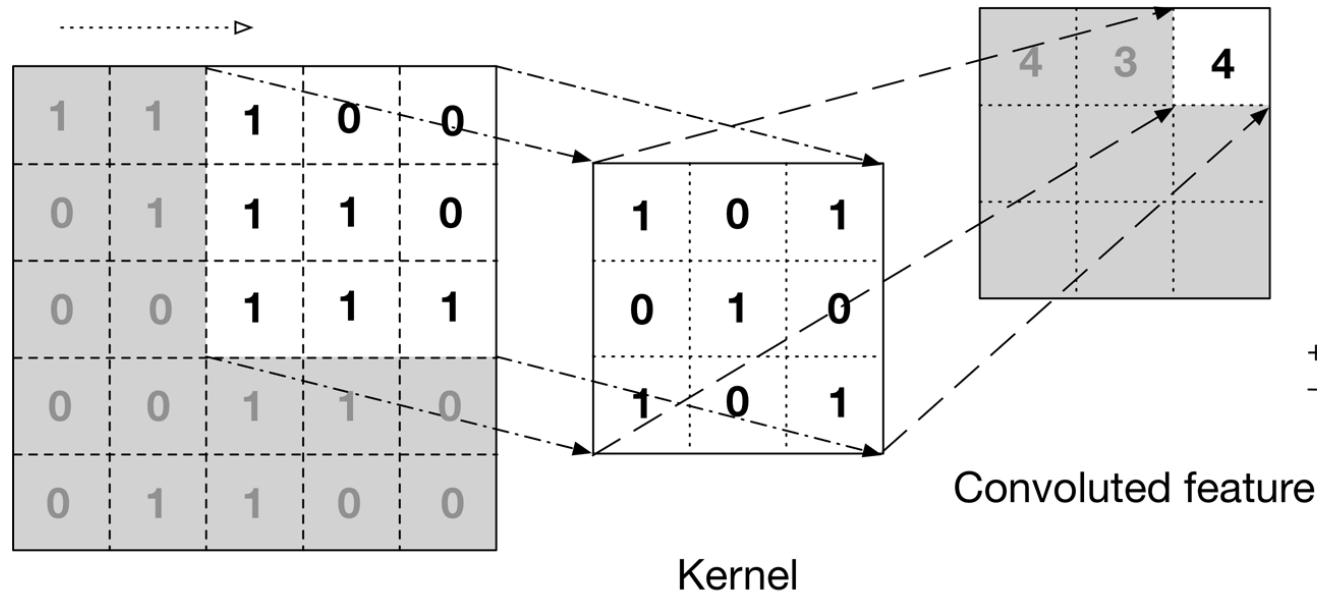
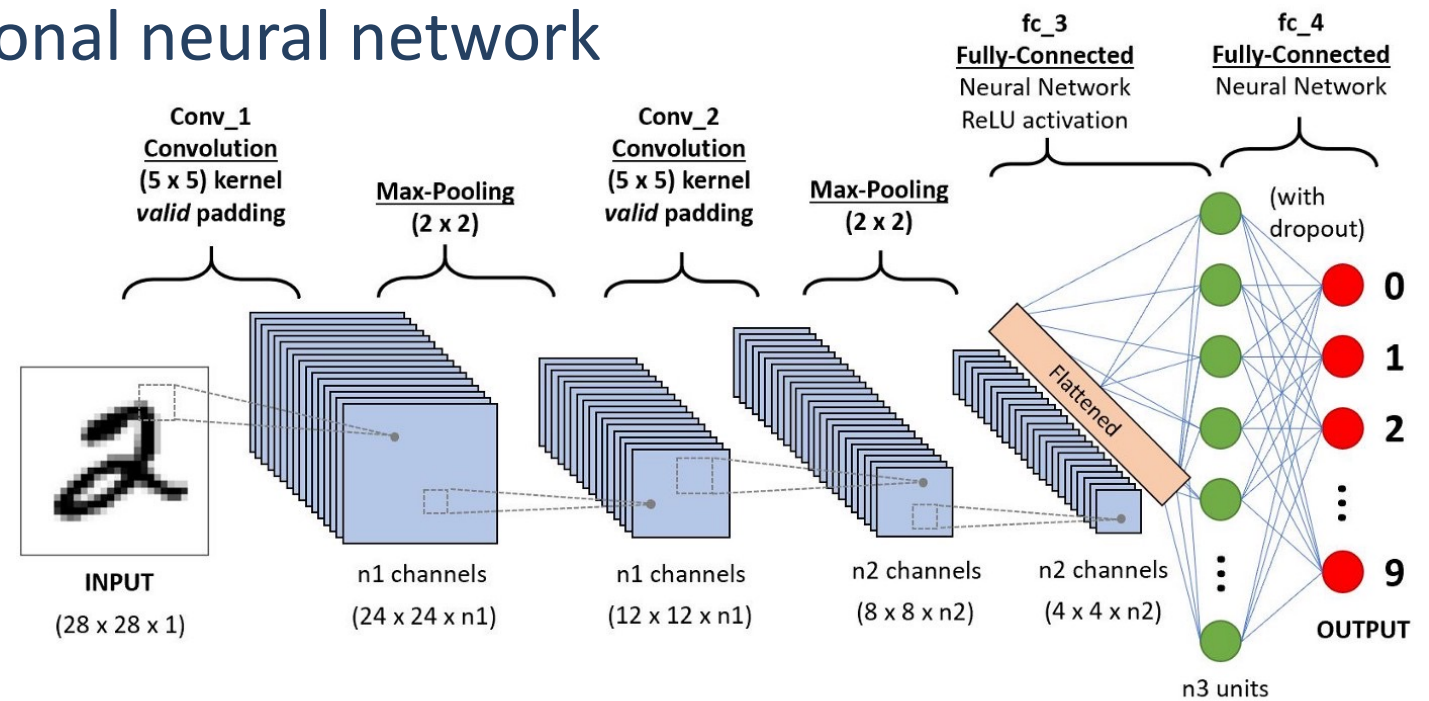
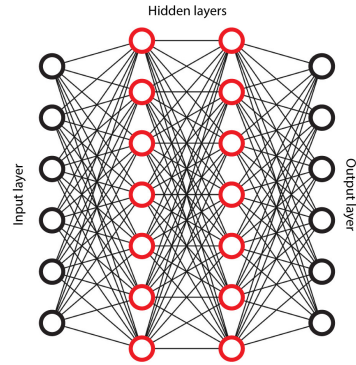
Neural networks can learn a complex task as a “black box”  
No previous knowledge about the system is required  
More data will allow for better networks

## The number of applications is increasing by day:

Image recognition  
Speech recognition  
Healthcare  
Gaming  
Finance  
Music composition and art  
...



# Architecture sketches: Convolutional neural network



Source: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

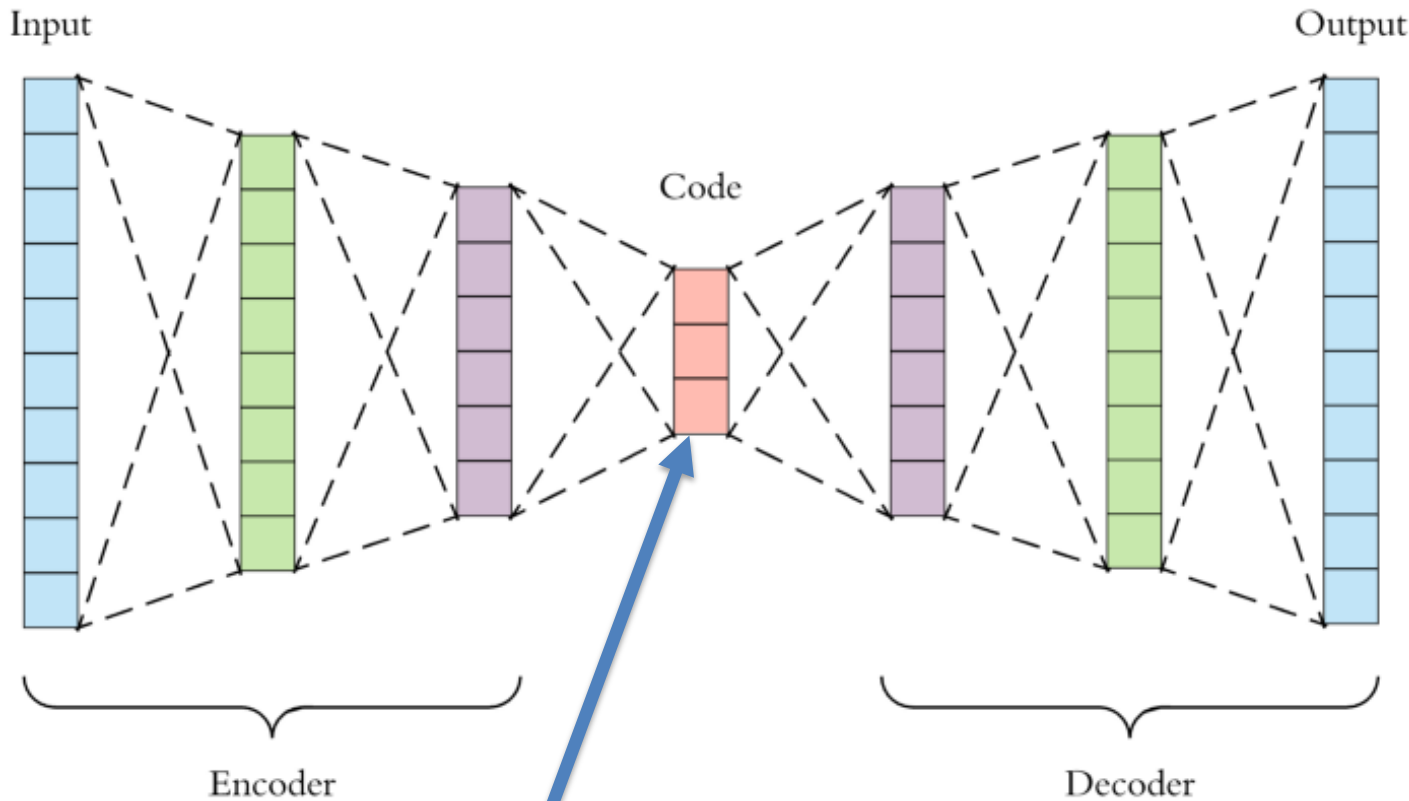
Source: <https://medium.com/machine-learning-for-li/different-convolutional-layers-43dc146f4d0e>

Input data

Kernel

Convolved feature

# Architecture sketches: U-Net / Encoder-Decoder and Variational Autoencoders



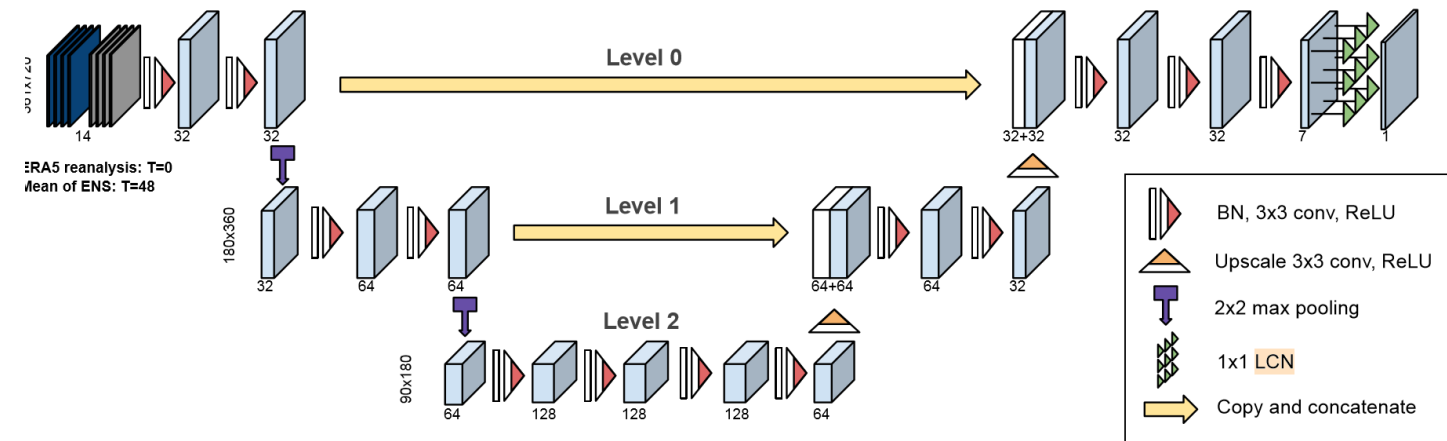
Source: <https://towardsdatascience.com>

Latent space

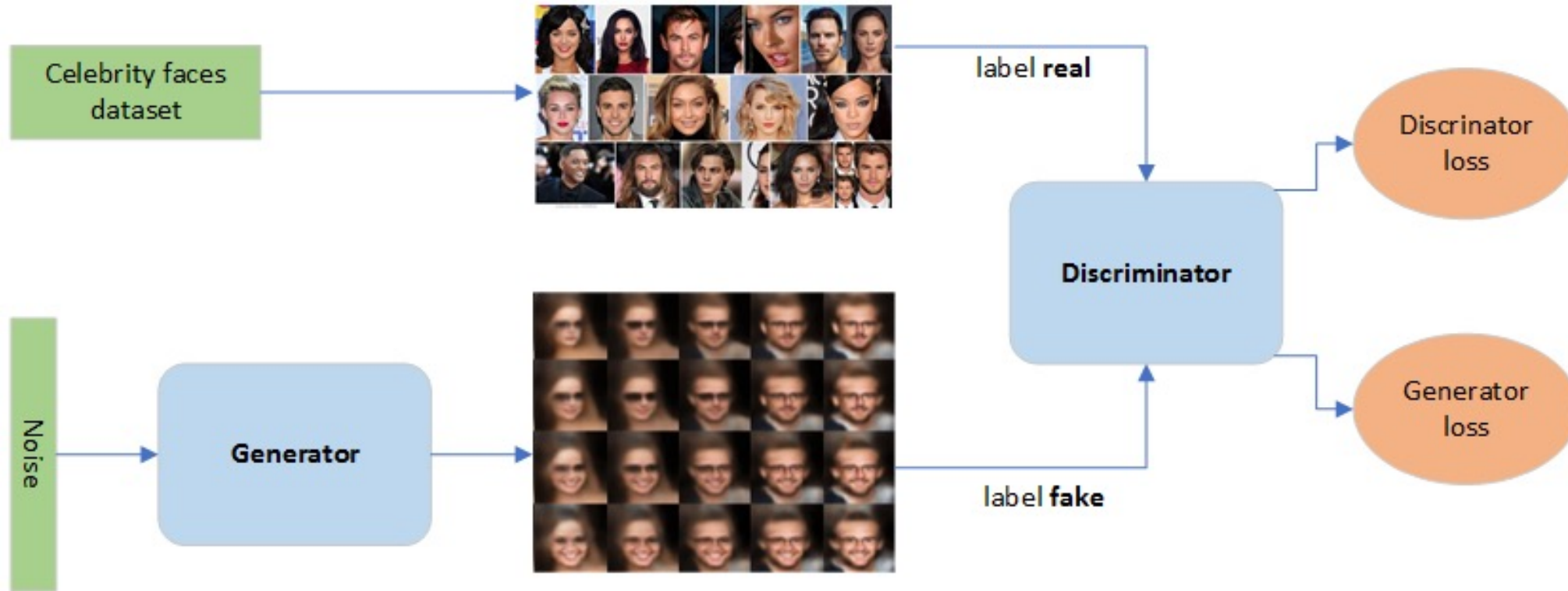
**The latent space can be manipulated to introduce uncertainty.**

**If the output is the same as the input, the information at the latent space can be used as compressed data representation.**

...

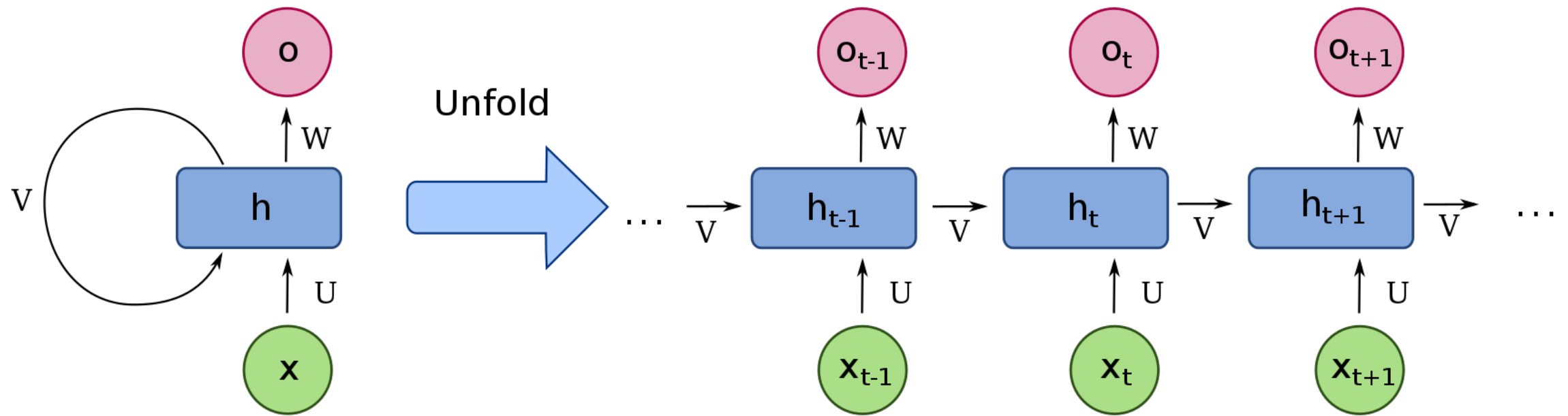


# Architecture sketches: Generative Adversarial Networks



Source: <https://towardsdatascience.com/generative-adversarial-network-gan-for-dummies-a-step-by-step-tutorial-fdeff170391>

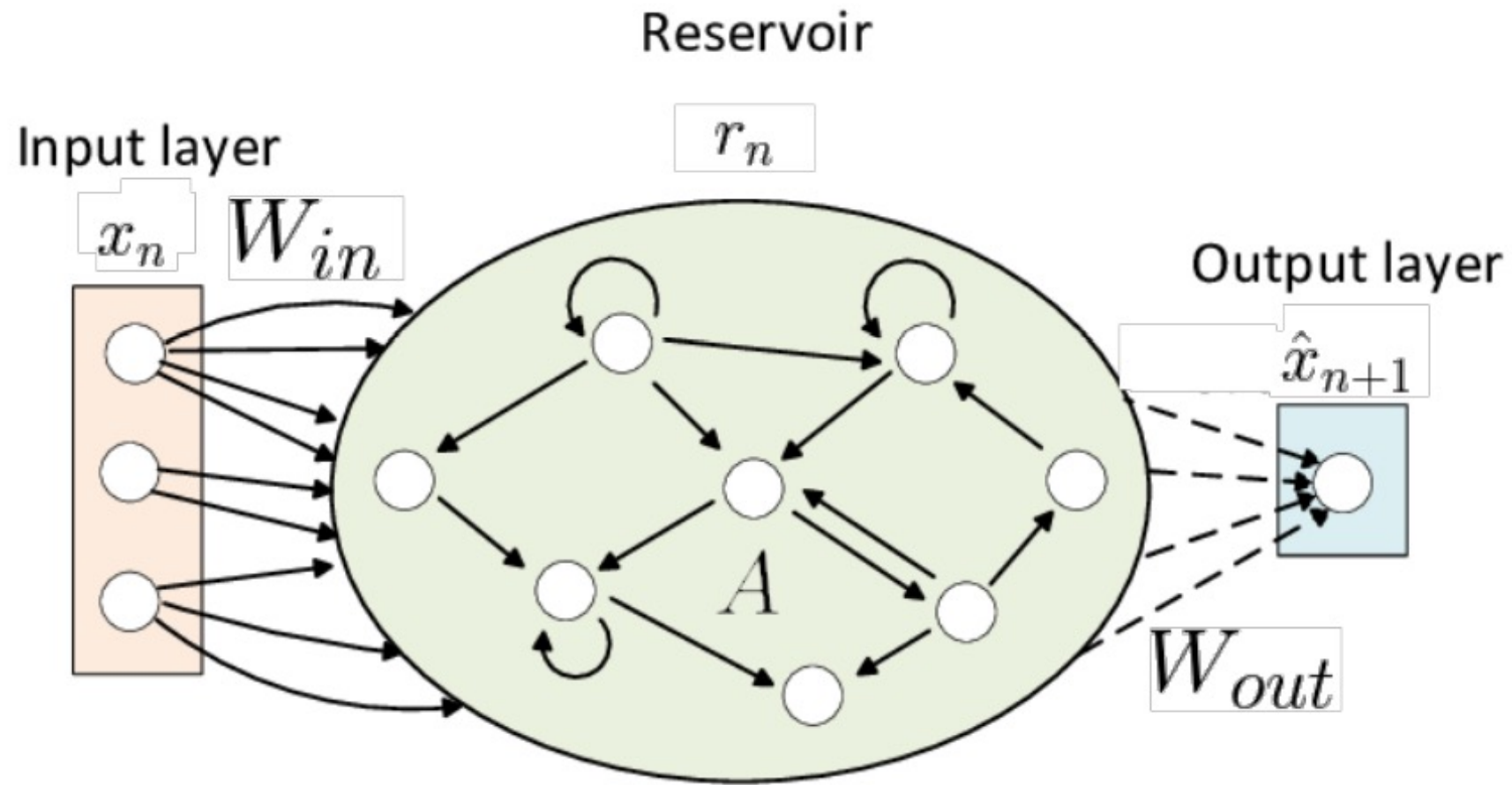
# Architecture sketches: Recurrent Neural Networks, LSTMs, GRU



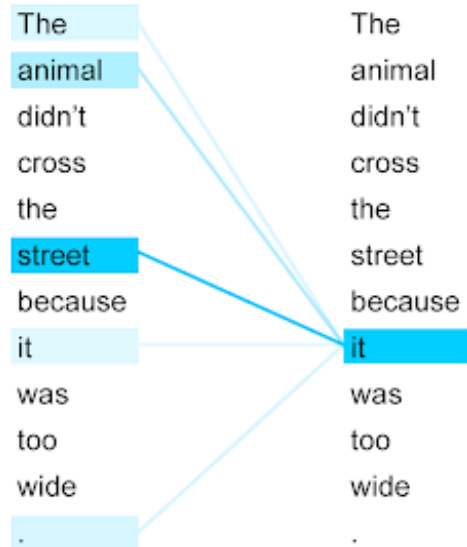
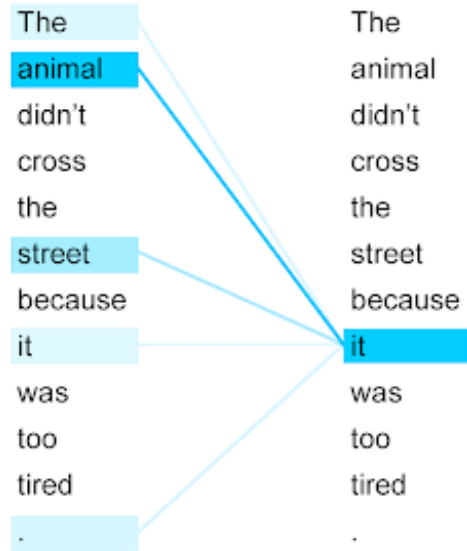
**LSTM:** Long short-term memory

**GRU:** Gated Recurrent Unit

# Architecture sketches: Reservoir and Echo State Networks



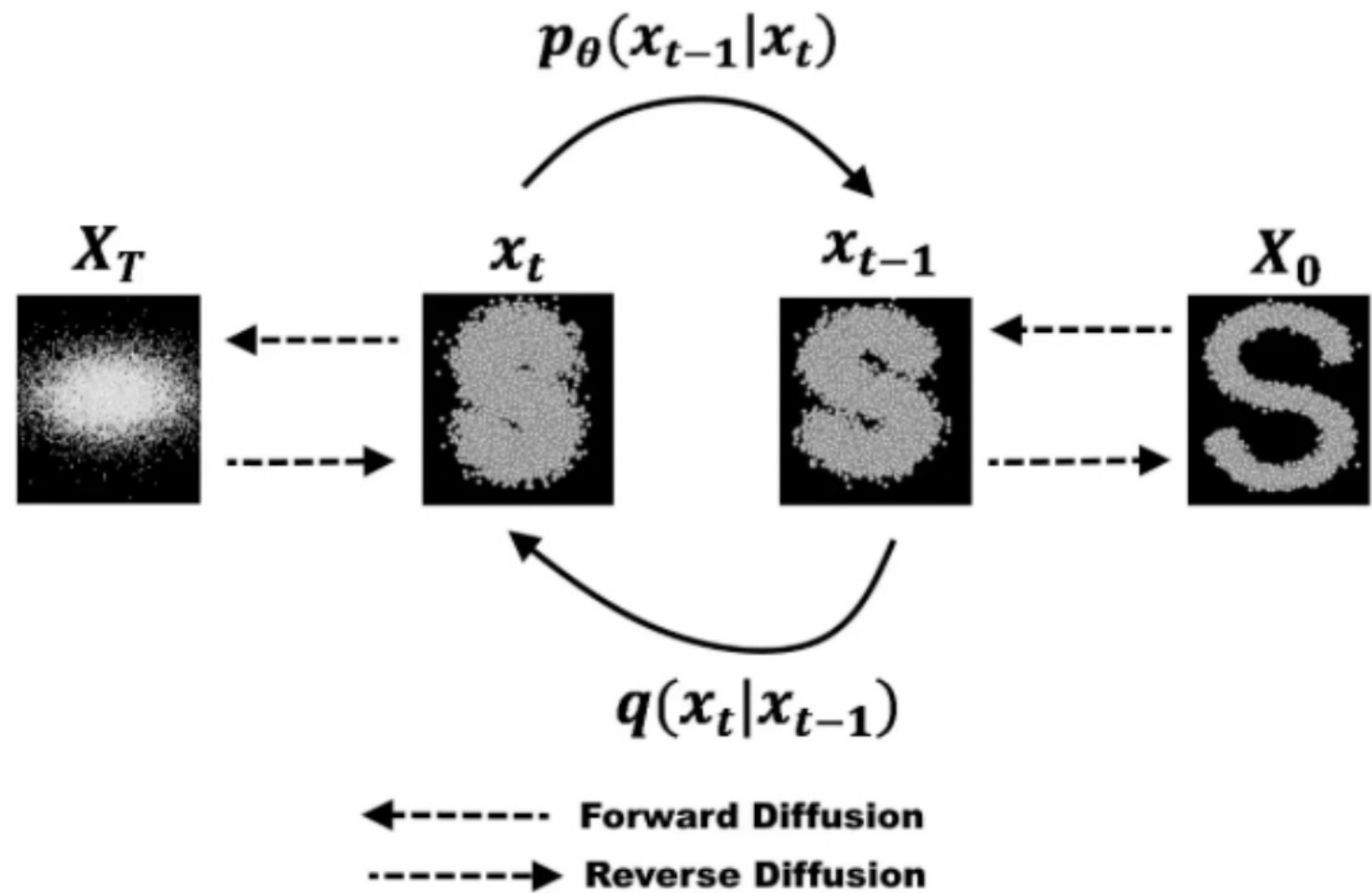
# Transformers and self attention layers



*Attention layers* are used to understand the dependence and importance of words in natural language processing, even if the words are separated in sentences.

*Transformer networks* are using self-attention in a powerful way and are key components in natural language processing and computer vision.

# Diffusion networks



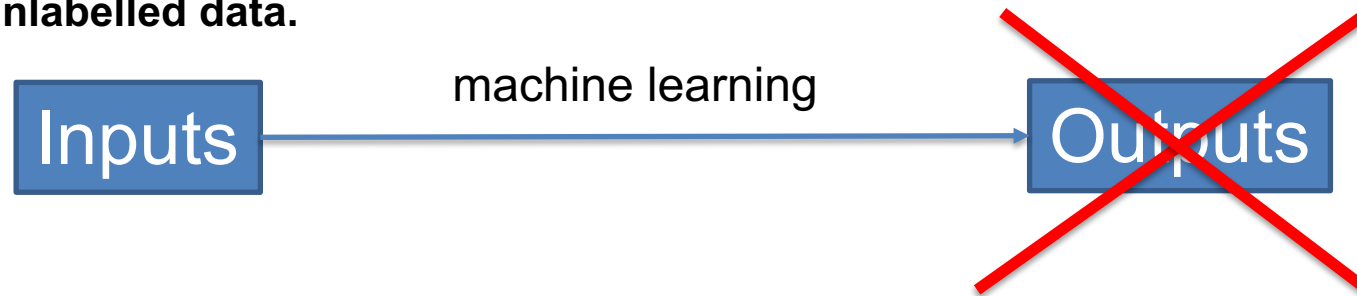
In the same way we can turn a nice signal into noise using diffusion processes and Markov Chains (e.g. by adding Gaussian noise a number of times), we revert this process in diffusion networks.

The networks are learning to re-establish gradients and re-construct shapes.

See for example: <https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da>

# Unsupervised learning

Unsupervised learning is used to learn patterns and to extract information from unlabelled data.

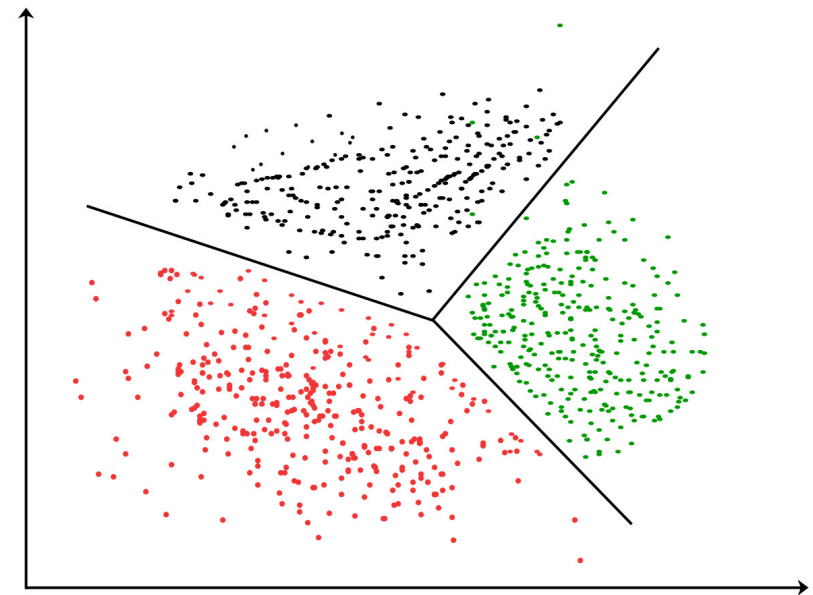


## Examples?

**Dimensionality reduction:** Reduce the number of degrees of freedoms in your dataset while keeping as much information as possible (e.g. EOF/Principle Component analysis)

**Clustering:** Task to divide data-points into a number of groups.

**Association:** Task to discover rules that describe large portions of your data (if you eat cereals, it is probably morning)



# Clustering – Content borrowed from Devis Tuia

Clustering is an unsupervised learning technique to identify groups in datasets

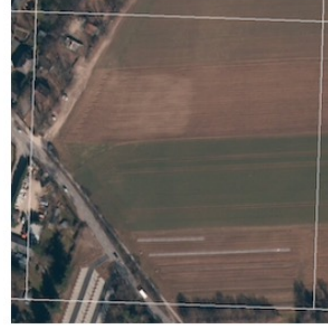
City center



R. mid dense



Agriculture



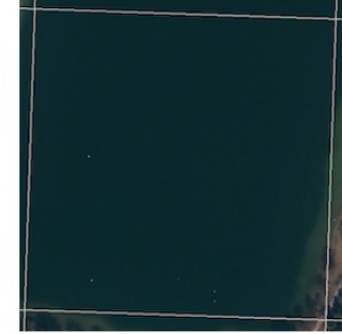
R. dense



R. low dense



Water plans



CBD



Commercial



Forests



Industry



Leisure



Land use classes from the Local Climate Zones categorisation. From: Poncet-Montanges et al. Classification of urban structural types with multisource data and contextual priors. *Joint Urban Remote Sensing Event*, Lausanne, Switzerland, 2015.

# K-means clustering

**Step 1: Choose the number of clusters  $K$ . E.g.  $K = 3$**

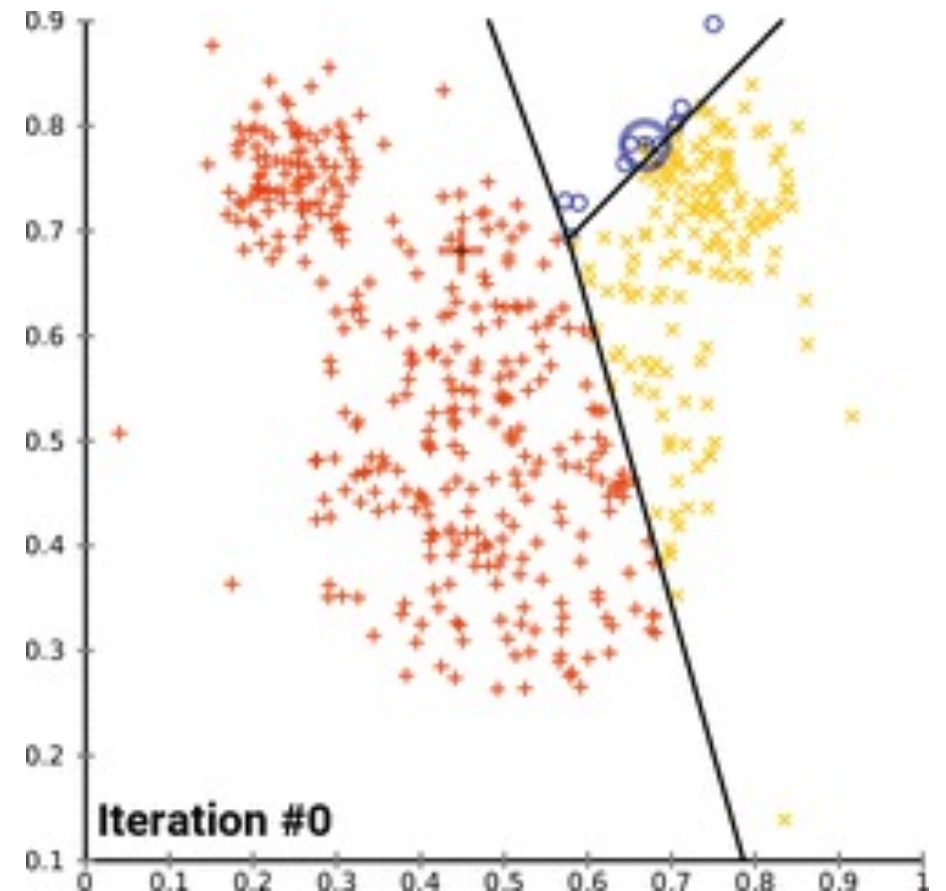
**Step 2: Pick centroids – these are centres of the cluster**

**Step 3: Assign data points to the nearest cluster (by calculating the distance to each centroid for each point)**

**Step 4: Re-initialize centroids by calculating the average of all data points of that cluster**

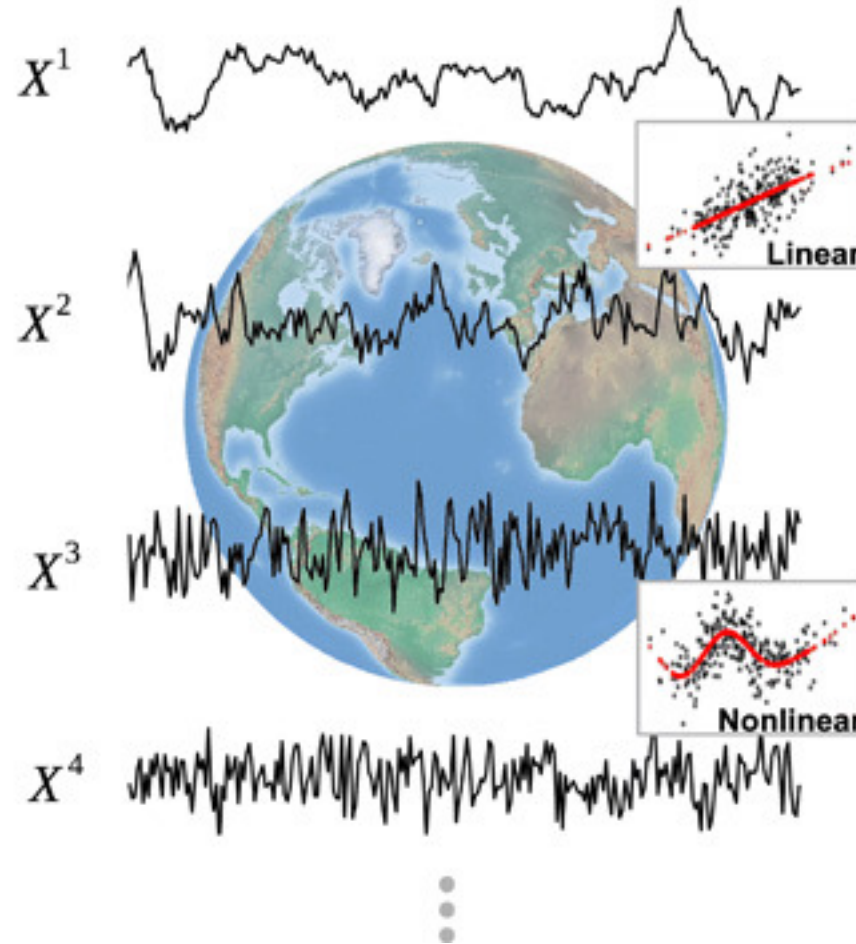
**Repeat Step 3 and 4 until convergence**

**There are a number of different clustering techniques available.**

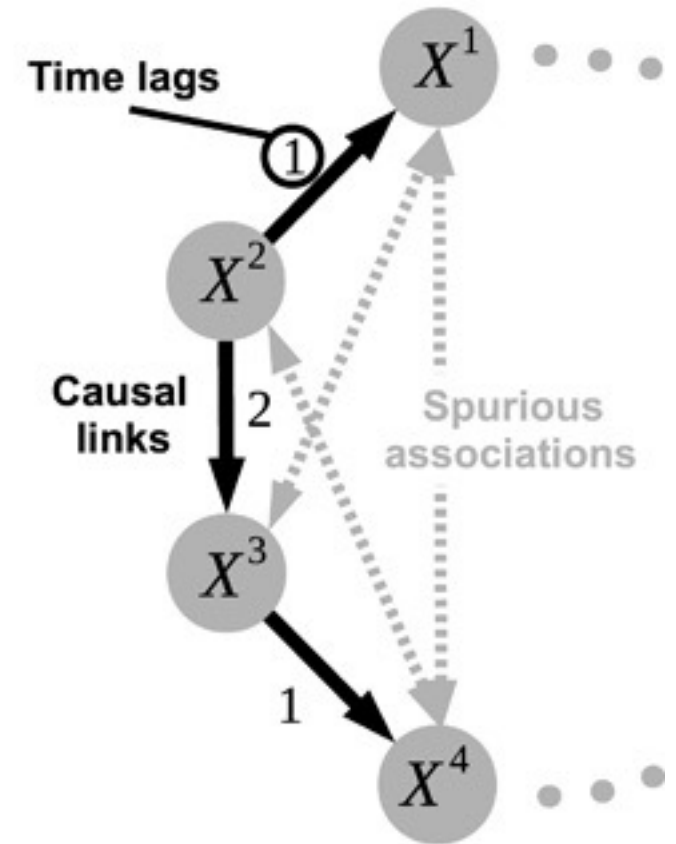


# Causal discovery

## A Large-scale time series dataset



## B Causal discovery

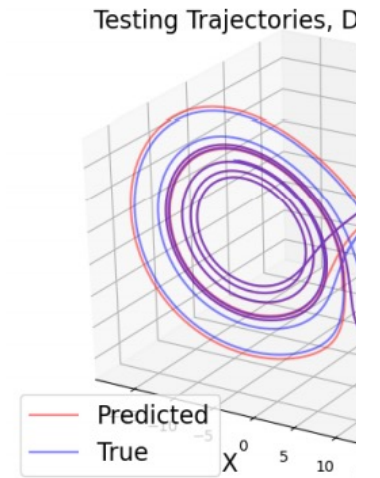
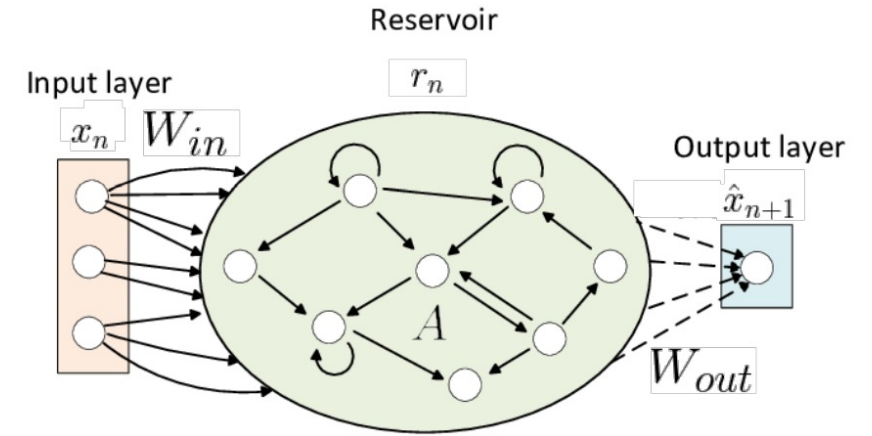
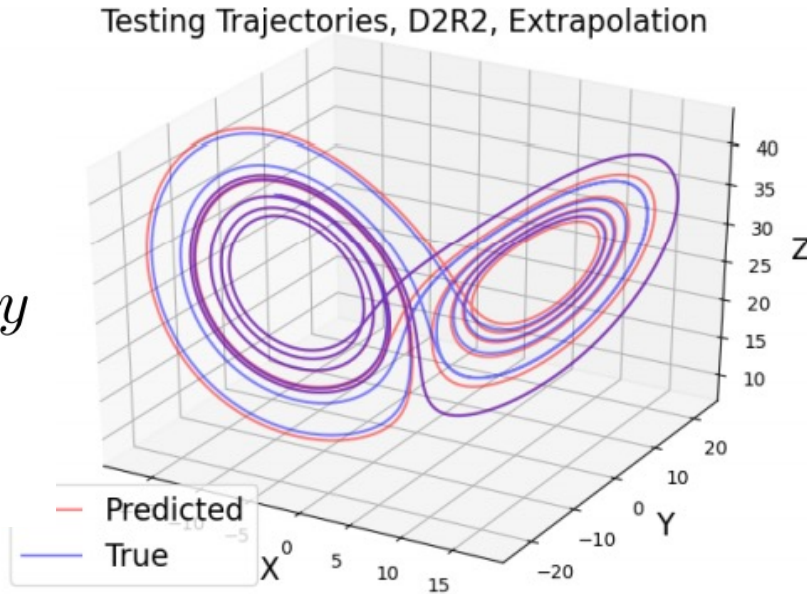


Consider a large-scale time series dataset (**A**) from a complex system such as the Earth system of which we try to estimate the underlying causal dependencies (**B**), accounting for linear and nonlinear dependencies and including their time lags (link labels). Pairwise correlations yield spurious dependencies due to common drivers (e.g.,  $X^1 \leftarrow X^2 \rightarrow X^3$ ) or transitive indirect paths (e.g.,  $X^2 \rightarrow X^3 \rightarrow X^4$ ). Causal discovery aims to unveil such spurious dependencies, leading to estimated causal networks that are, therefore, much sparser than correlation networks.

[Runge, Nowack, Kretschmer, Flaxman, Sejdinovic Science Advances 2019](#)

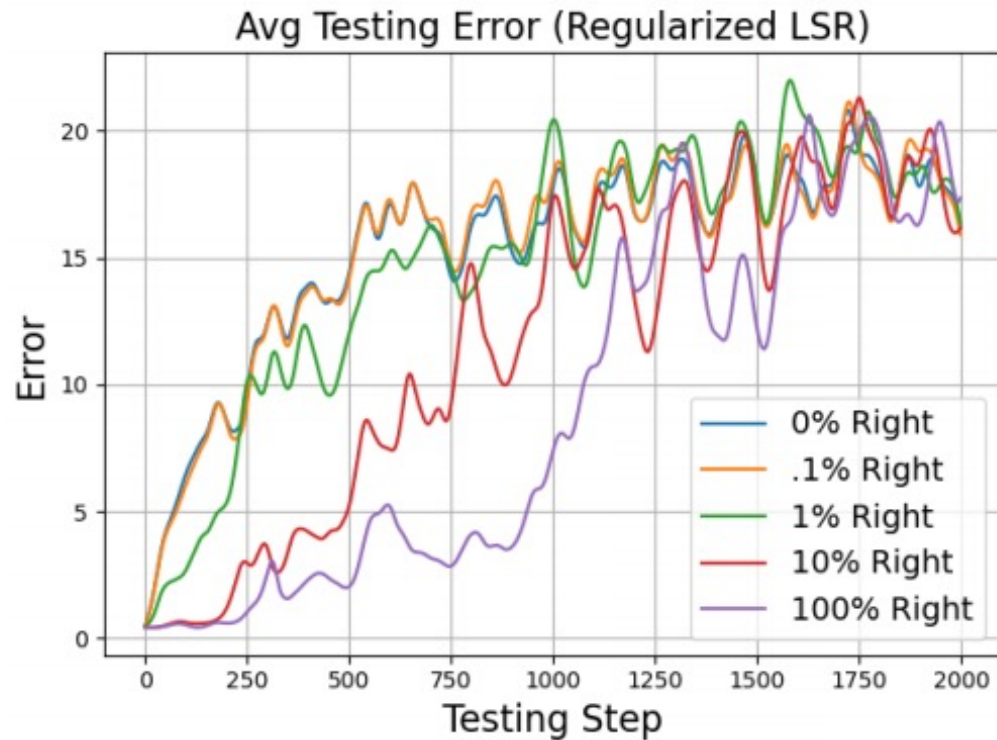
# Challenge #1: Extrapolation into unknown states – Science example

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z\end{aligned}$$



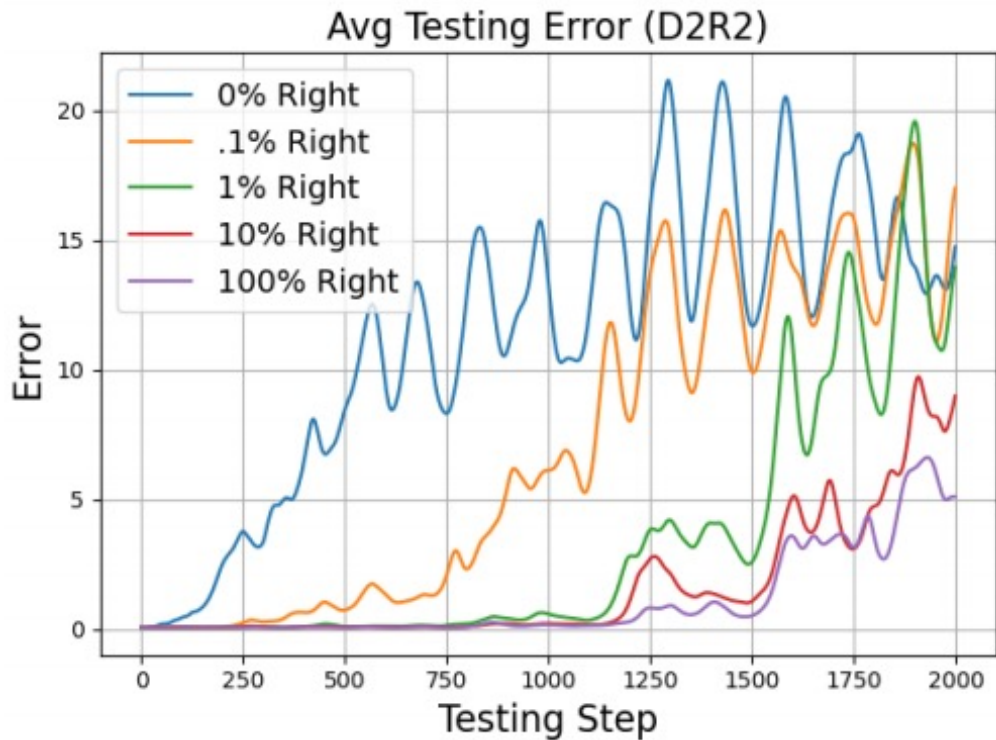
- Let's train a machine learning tool in a changing "climate"
- Let's start simple to be able to make clear statements → The Lorenz'63 model
- Let's take two different approaches to learn the model from a truth trajectory:
  1. Echo State Networks (Vlachas et al. 2020 and Chattopadhyay et al. 2020)
  2. Domain-Driven Regularized Regression (D2R2; Pyle et al. 2021)
- Let's assume that today's climate is the "left-lobe regime" and that climate change is kicking us into the "two-lobe regime".
- What if we only train from 1%, 2%, 5%... of the training data from the right lobe?

# Science and tool developments



## Echo State Network

- The Echo State Network performs horrible unless you provide at least 10% of the data of the right lobe.
- The regression technique needs a very small amount of the right lobe to perform well.



## Regression Technique (D2R2)

## Challenge #2: Interpretation



The clever Hans effect

Horse-picture from Pascal VOC data set



Source tag present



Classified as horse



No source tag present



Not classified as horse

# How can you build trust in machine learning tools and make them reliable?

Trustworthy AI, explainable AI and physics informed machine learning

There are several ways to incorporate physical knowledge into machine learning tools:

- Formulate the machine learning problem in a way that makes it physical
- Change the architecture of the neural network
- Close the budget for the output variables or correct the outputs to fulfil the constraint
- Incorporate physical constraints into the loss function that is used for training

There are also ways to evaluate whether the machine learning solution is reproducing the right physics

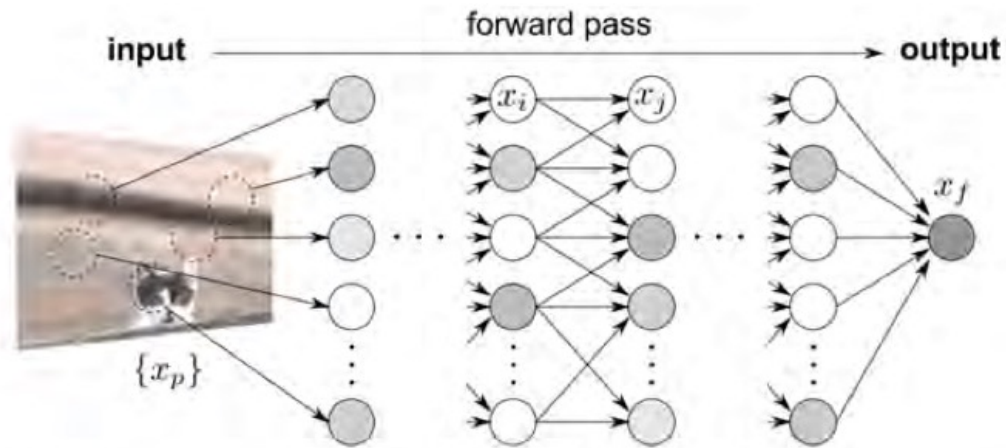
- Consider specific use cases and weather regimes
- Perform sensitivity tests on the inputs or outputs
- Test for physical reasoning

*Reichstein, M. et al. Deep learning and process understanding for data-driven Earth system science. Nature 566, 195–204 (2019).*

*McGovern, et al. Making the Black Box More Transparent: Understanding the Physical Implications of Machine Learning, Bulletin of the American Meteorological Society, 100(11), 2175-2199 (2019)*

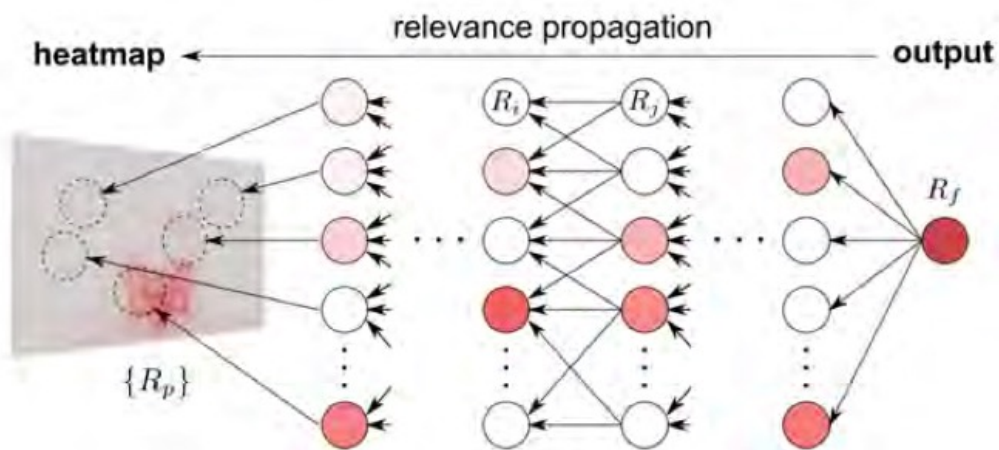
# Layerwise Relevance Propagation

Prediction  
of 1 sample



$\text{Pr}(\text{cat})$

LRP  
of 1 sample



$\text{Pr}(\text{cat})$



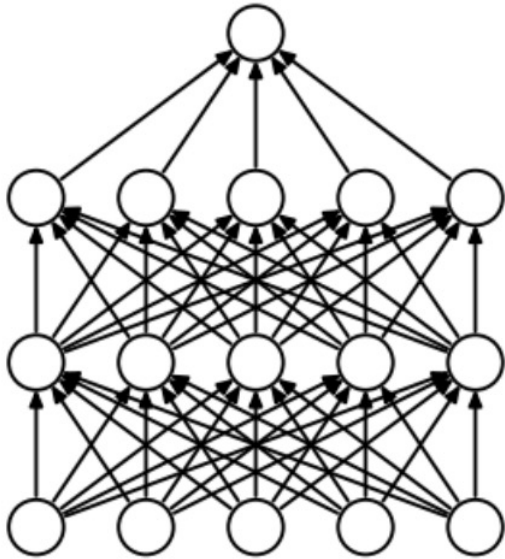
where the network looked to  
determine it was a "cat"

# Challenge #3: How can we represent uncertainties?

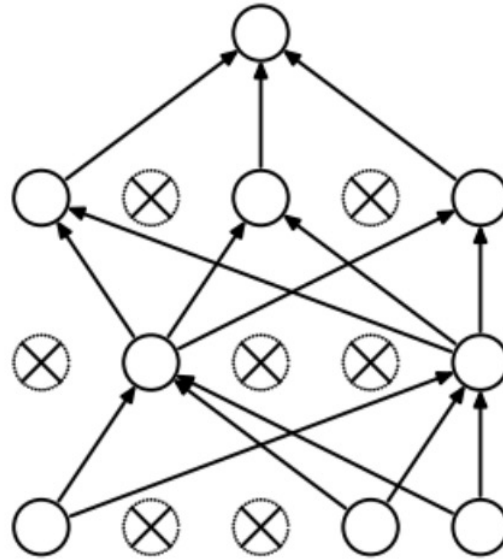
## How can you treat model uncertainty with machine learning?

- You can learn the error
- You can learn distributions
- You can train ensembles
- You can use Bayesian machine learning techniques

## Simple example: Dropout techniques



(a) Standard Neural Net



(b) After applying dropout.

## Challenge #4: Explore a new scientific field and merge with existing infrastructure

Different philosophy for domain and machine learning scientists

**Approach:** *Support close collaborations // study explainable AI, trustworthy AI and physics informed machine learning*

For many applications off-the-shelf machine learning tools will not be sufficient

**Approach:** *Foster cross-disciplinary collaborations // develop customised machine learning tools // Benchmark Datasets*

Difficult to learn from noisy observations and to *improve* models

**Approach:** *Learn from and exploit data assimilation // learn boundary conditions from observations*

Data avalanche

**Approach:** *Anticipate data access and channelise requests // efficient use of heterogeneous hardware*

Different set of tools (e.g. CPUs vs. GPUs)

**Approach:** *Training // Software // Hardware*

Integrate machine learning tools into the scientific workflow

**Approach:** *Develop tools that can be re-used many times*

# Challenge #5: Getting greedy – not everything can be learned...

**When you have a hammer, everything looks like a nail...**

**Stephan Rasp's five questions to ask before you start a machine learning problem:**

- What exactly is the problem I am trying to solve?
- How exactly will I get my training data?
- How exactly will I verify my results?
- Am I still solving a real problem?
- Is the project feasible within my constraints?

**Bonus questions:**

- (Hybrid modeling) Am I testing the skill that matters?
- (Explainable/interpretable AI) How will I use the output?

<https://raspstephan.github.io/blog/hammer-and-nail/#>

<https://twitter.com/raspstephan/status/1374695383931191298>



# Challenge #6: Make developments comparable – Benchmark datasets

## **Benchmark datasets include:**

- A problem statement
- Data that is available online
- Python code or Jupyter notebooks
- Quantitative evaluation metrics
- A reference machine learning solution
- Visualisation, diagnostics and robustness tests
- Computational benchmarks

## **Benchmark datasets are useful because:**

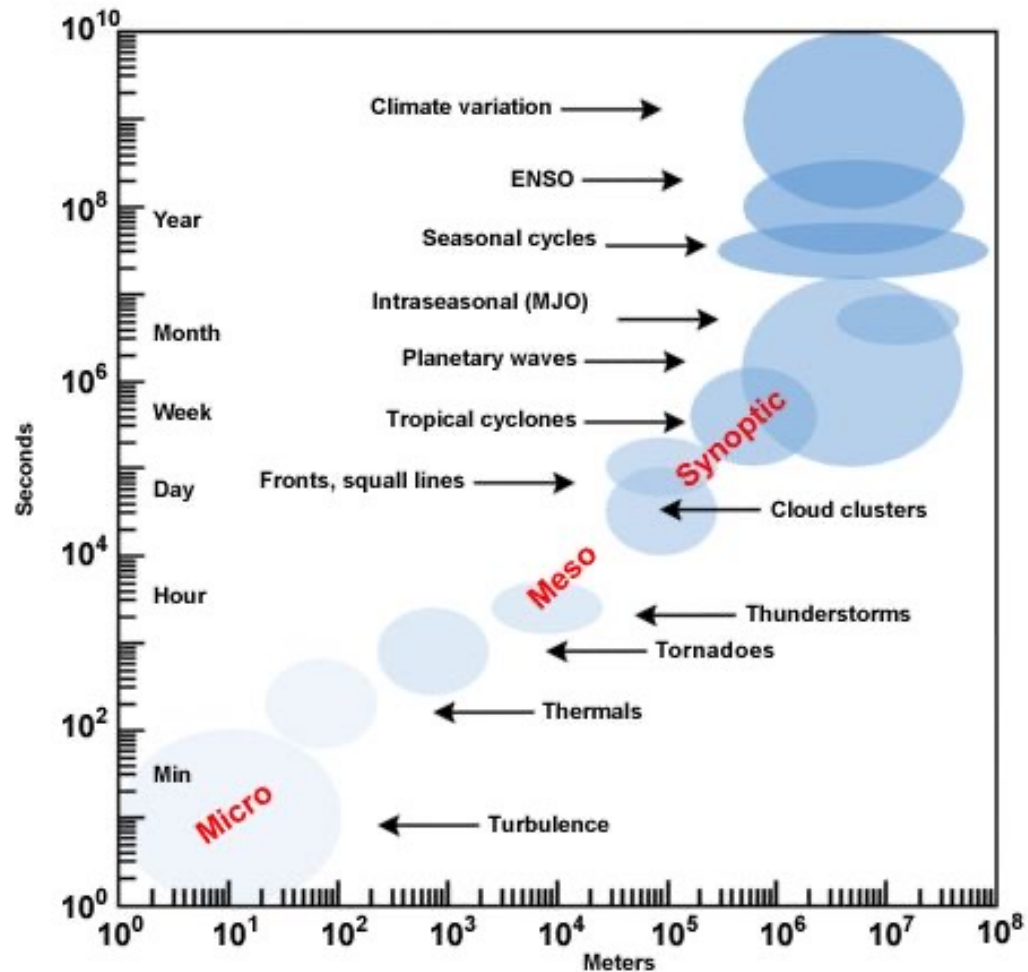
- They allow a quantitative evaluation of machine learning approaches (including efficiency tests with e.g. reduced precision or sparse solutions)
- They reduce data access and help scientists to get access to relevant data
- They allow for a separation of concerns between domain sciences and machine learning experts
- They allow for a separation of concerns between domain sciences and HPC experts

# Challenge #7: Customised architectures?

Can we represent scale interactions with machine learning tools?

## Weather and climate modelling:

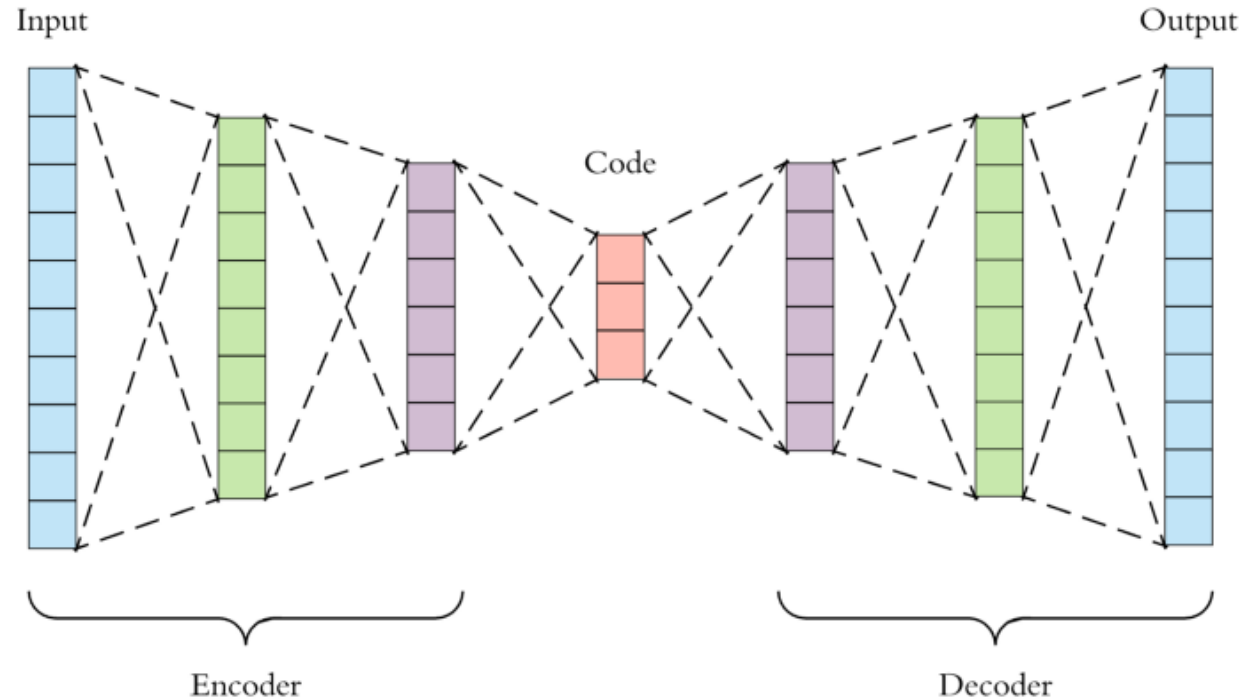
Tools need to allow for scale interactions



Source: UCAR

## Machine learning:

Neural network tools allow for encoding/decoding structures



Source: <https://towardsdatascience.com>

Challenge #8:

**THE DATA!**

# Recent examples for machine learning

## ChatGPT:



If Hamlet were a musical comedy, rather than a tragedy, how would it have ended? Describe the plot.



If Hamlet were a musical comedy, rather than a tragedy, the ending would likely be happier and more lighthearted. Instead of ending in a violent and tragic show with the deaths of several major characters, the play would likely have a more conventional satisfying resolution.

For example, the play could end with Hamlet successfully avenging his father, then choosing to forgive and reconcile with his uncle, the new king. This would allow Hamlet to put aside his grudges and bitterness, and to find peace and happiness.

As part of this resolution, there could be a big, energetic finale song, where all the characters sing and dance together, expressing their newfound sense of joy and hope. This could be a lively and upbeat number, with a catchy and infectious melody that reflects the themes and events of the play, while also expressing a sense of optimism for the future.

Overall, a musical comedy version of Hamlet would likely have a more uplifting and optimistic ending, with the main character achieving a sense of closure and reconciliation and moving on with his life in a positive and fulfilling way.

### THE SHIFT

## *An A.I.-Generated Picture Won an Art Prize. Artists Aren't Happy.*

"I won, and I didn't break any rules," the artwork's creator says.

Give this article

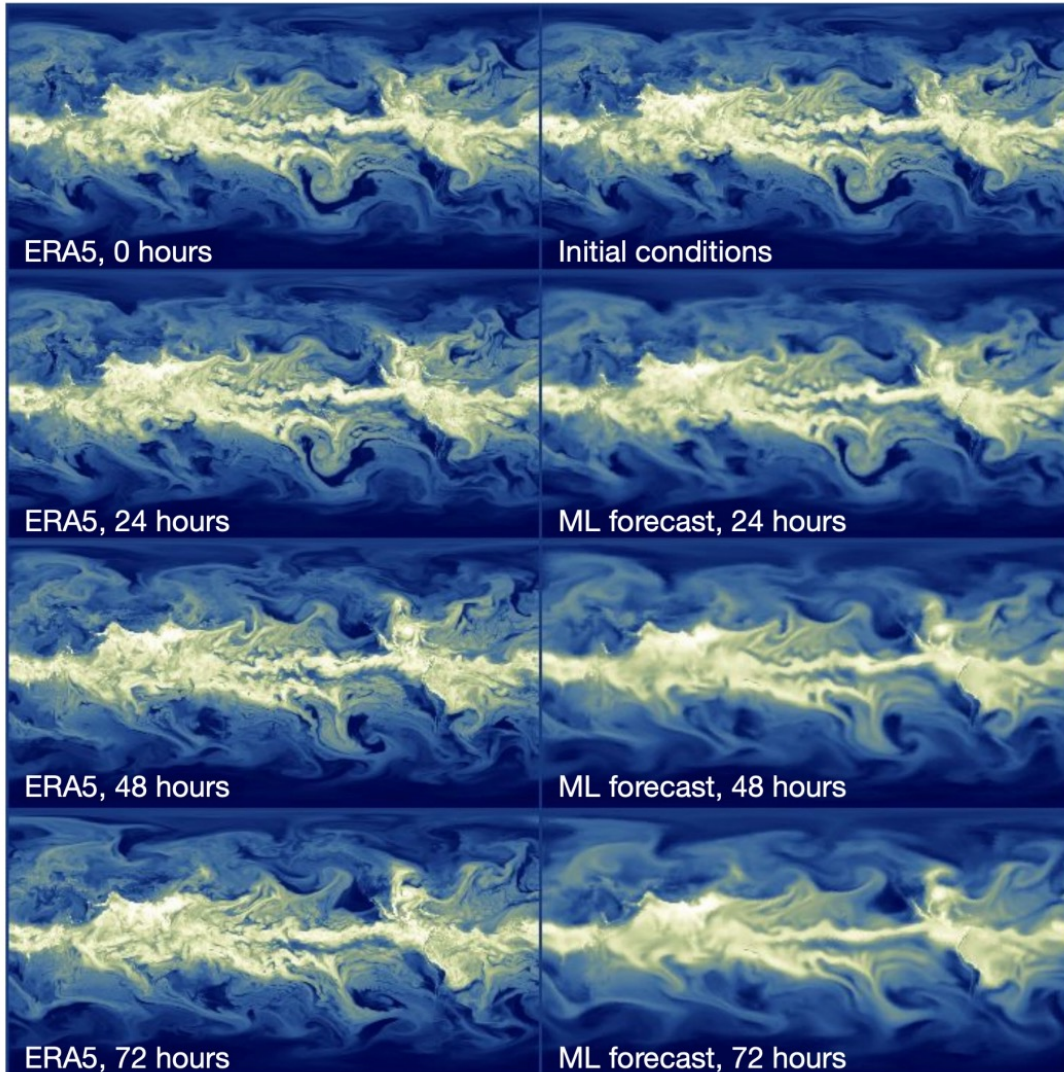
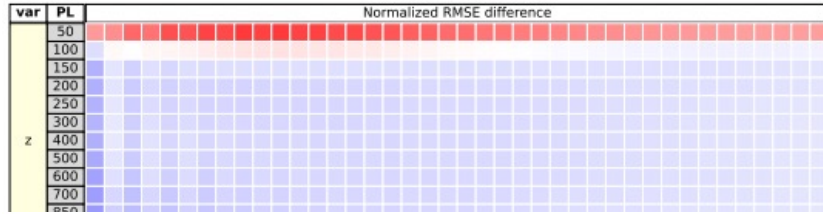


1.5K



Jason Allen's A.I.-generated work, "Théâtre D'opéra Spatial," took first place in the digital category at the Colorado State Fair. via Jason Allen

# 2022-today: The machine learning revolution



GraphCast from Google/Deepmind and Fourcastnet from NVIDIA are beating conventional weather forecast model in deterministic scores and are orders of magnitudes faster.

But how do these models actually work?

They get the best results when using very large timesteps.

They are trained for a small Root Mean Square Error.  
→ They smear out for large lead times.

Many questions remain:

Can the models extrapolate?

Can they represent extreme events?

Can they learn uncertainty?

Can they be trained from observations?

Can they represent physical consistency?

Images from Keisler (2022)

# 2022-today: The machine learning revolution

arXiv > physics > arXiv:2307.10128

Physics > Atmospheric and Oceanic Physics

[Submitted on 19 Jul 2023]

## The rise of data-driven weather forecasting

Zied Ben-Bouallegue, Mariana C A Clare,  
Dramschi, Simon T K Lang, Baudouin Raoult

Data-driven modeling based on machine learning is revolutionizing some applications. The uptake of ML methods in the 'data-driven revolution' of weather forecasting. The combination of increasing model resolution and ensemble forecasts that require much lower computational cost than standard NWP-based forecasts in an operational context. Verification tools to assess to what extent a forecast from one of the leading global models is reliable when verified against both the operational and the ML-based forecasts. A new NWP initialization and model training.


Subjects: Atmospheric and Oceanic Physics (physics)

Cite as: arXiv:2307.10128 [physics.ao-ph]  
(or arXiv:2307.10128v1 [physics.ao-ph] for this version)  
<https://doi.org/10.48550/arXiv.2307.10128>

### Submission history

From: Zied Ben Bouallegue [view email]

[v1] Wed, 19 Jul 2023 16:51:08 UTC (18,531 KB)



Search site...

Help Log in

HomeAboutForecastsComputingResearchLearningPublications

Who we areWhat we doJobsMedia centreSuppliersLocation

NewsIn focusScience blogKey facts and figuresMedia resourcesVideos

## ECMWF unveils alpha version of new ML model

13 October 2023  
The AIFS team

ECMWF is today launching a newborn companion to the IFS (Integrated Forecasting System), the AIFS, our Artificial Intelligence/Integrated Forecasting System (one "I" covering both Intelligence and Integrated).

The AIFS is barely a few months old and proudly entering its alpha version. Its arrival signals the strengthening of ECMWF's efforts in the field of machine learning (ML), which we have been navigating for a few years now. The AIFS forms one of three components of our new ML project, which began in summer 2023 and aims to expand our applications of machine learning to Earth system modelling.

### Recent posts

ECMWF unveils alpha version of new ML model

<https://www.ecmwf.int/en/about/media-centre/news/2023/how-ai-models-are-transforming-weather-forecasting-showcase-data>  
<https://www.ecmwf.int/en/about/media-centre/aifs-blog/2023/ECMWF-unveils-alpha-version-of-new-ML-model>

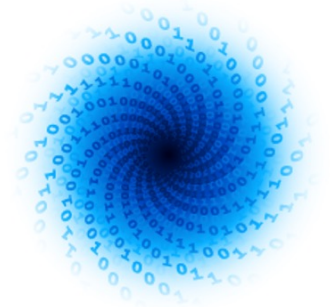
# 2022-today: The machine learning revolution

## **Why was the approach of global machine learning models so successful?**

- Because there was a very large unified training dataset available with ERA5 from Copernicus.
- Open benchmark datasets are needed to allow for quantitative comparisons and to bridge communities.
- Km-scale models will make a difference for the generation of training datasets.

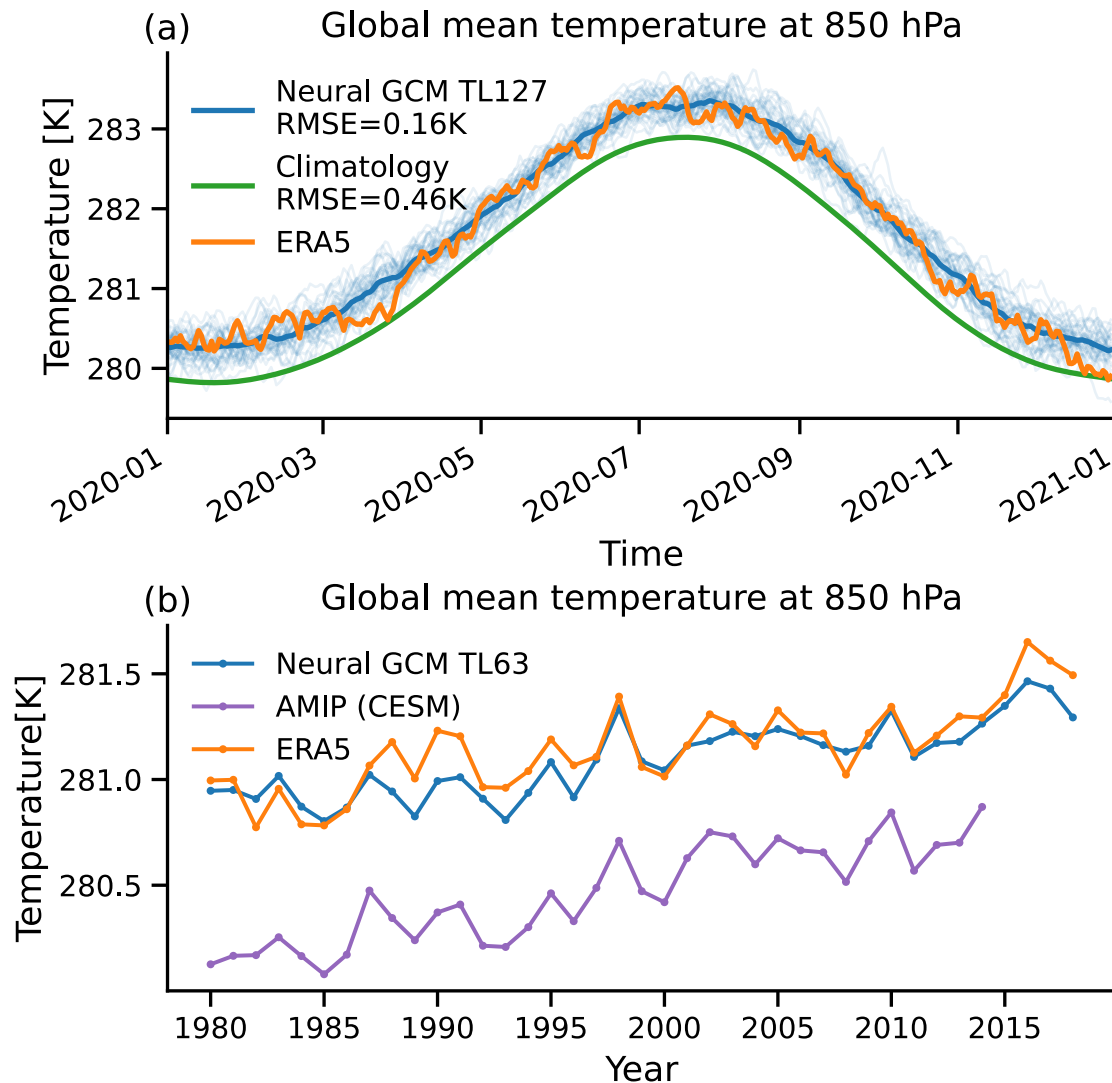
## **What machine learned models can and cannot do?**

- Conventional models will not be replaced by machine learning models entirely.
- Within the next couple of years most weather predictions will come from machine learning models.
- Machine learning will be the perfect glue between models and observations.
- Machine learning models have also potential for climate projections despite the extrapolation problem.

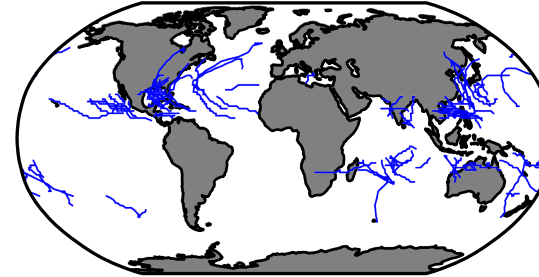


MAELSTROM

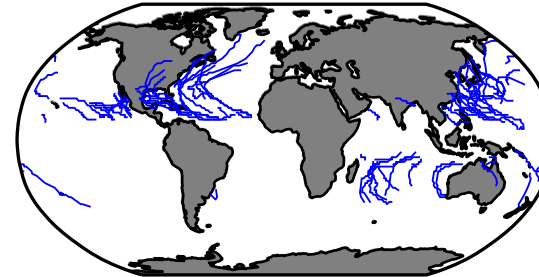
# 2022-today: The machine learning revolution



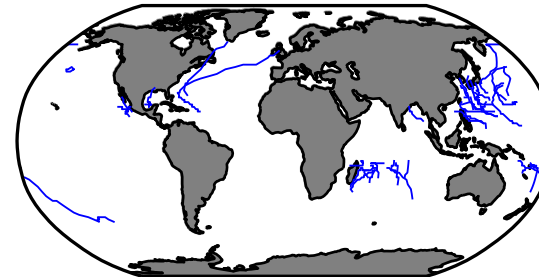
(c) ERA5, 80 Tropical Cyclones



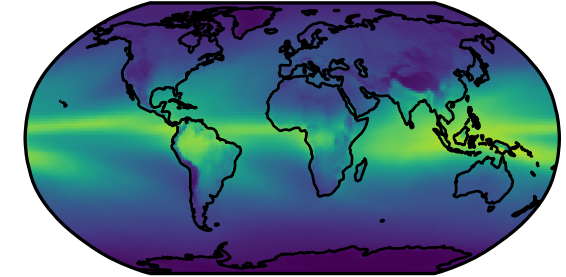
(e) Neural-GCM, 79 TCs



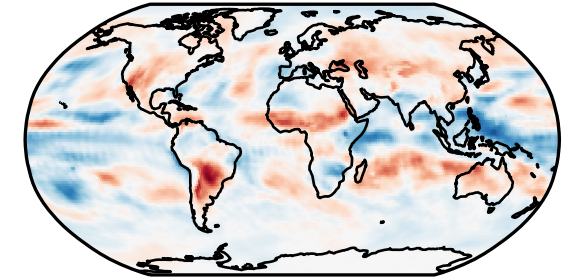
(g) X-SHiELD, 35 TCs



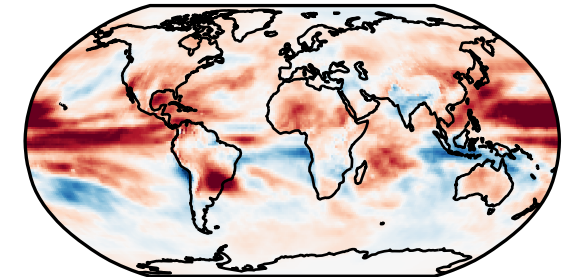
(d) ERA5 Precipitable Water



(f) Neural-GCM, RMSE=1.07mm



(h) X-SHiELD, RMSE=1.74mm



**Machine learned models can now also do AMIP simulations.**  
Kochokov et al. @Google in preparation

**And Foundation Models will enter the domain...**

# A shallow entry into deep learning

**What is a Neural Network?**

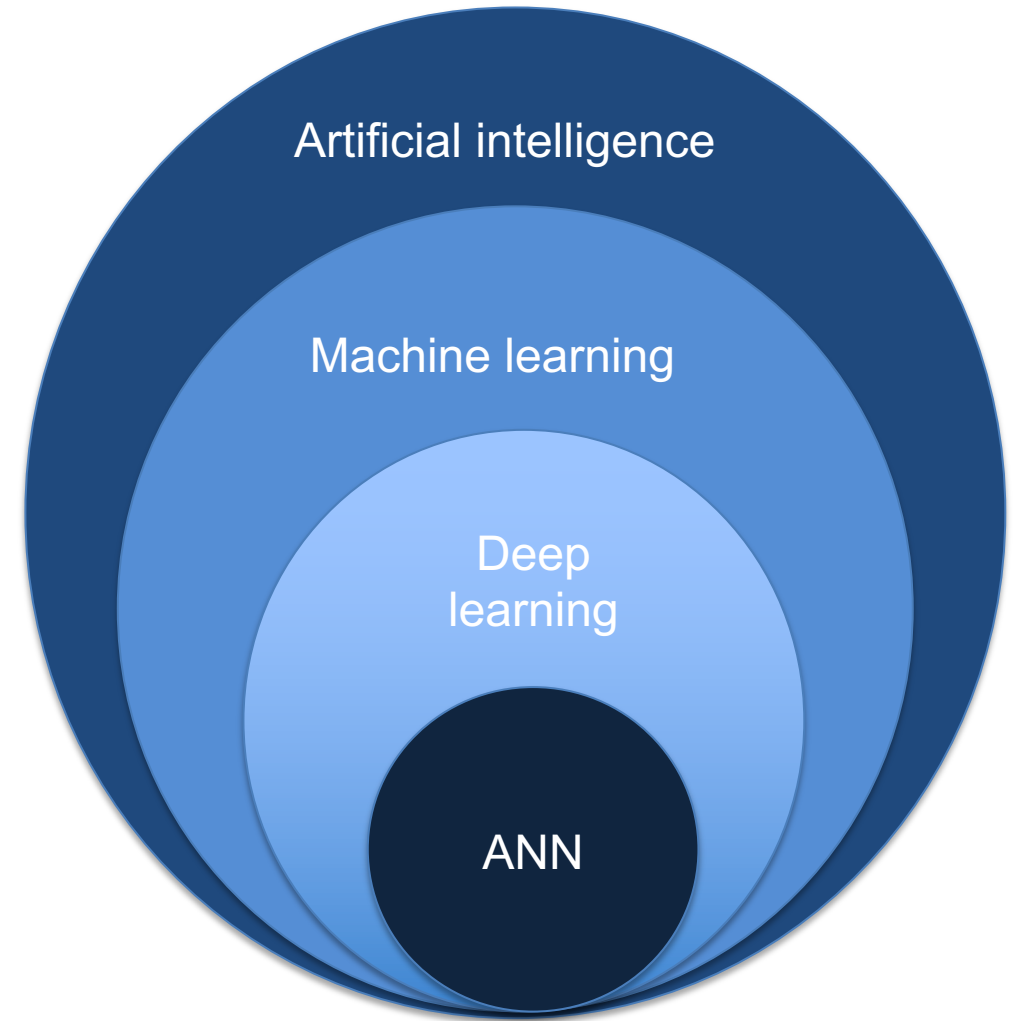
**Components of a Neural Network**

**Math at a neuron level**

**Training and optimising a Neural Networks**

**How to deal with data**

**Final remarks**



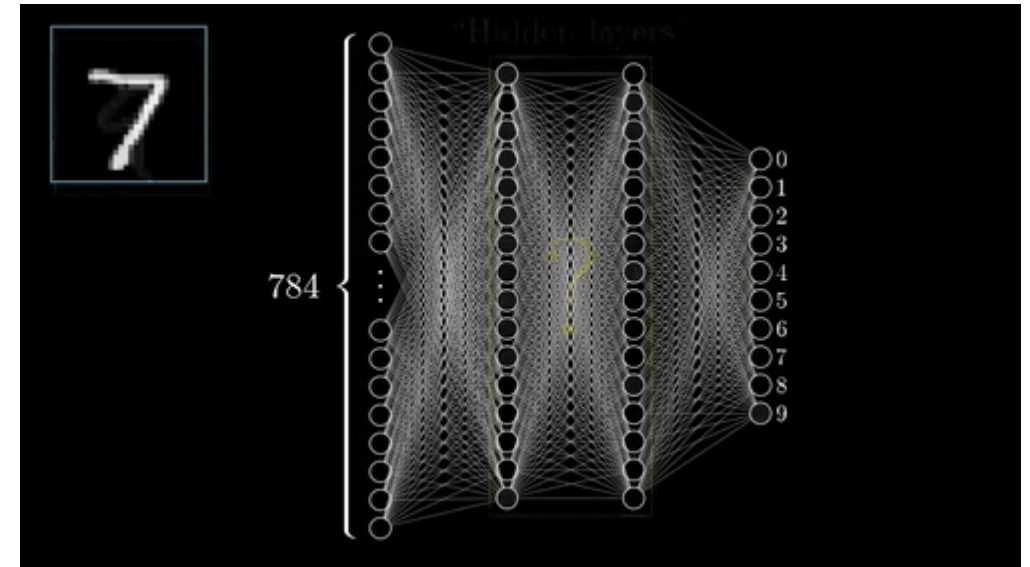
# What is a Neural Network?

## The concept:

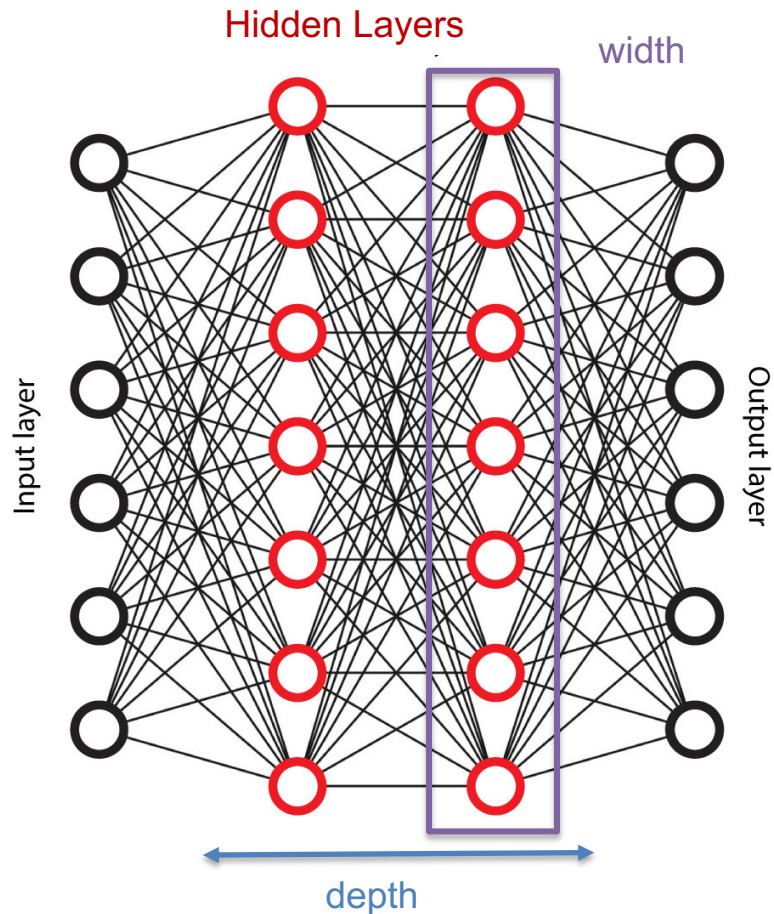
- Take input and output samples from a large data set
- Learn to predict outputs from inputs
- Predict the output for unseen inputs
- There are many types of NNs – focus on MLPs (FFNN)

## The key:

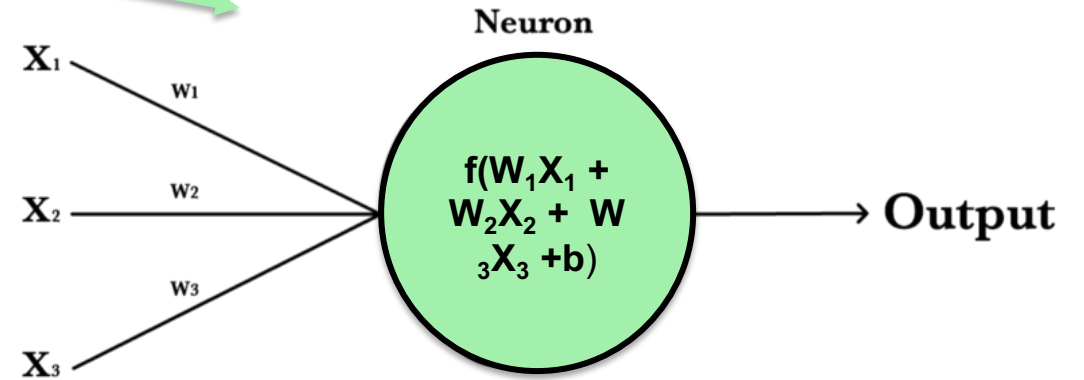
- Neural networks can learn a complex task as a “black box”
- No previous knowledge about the system is required
- More data will allow for better networks
- Neural network rely on data to learn and improve their performance over time



# Neural networks: Components



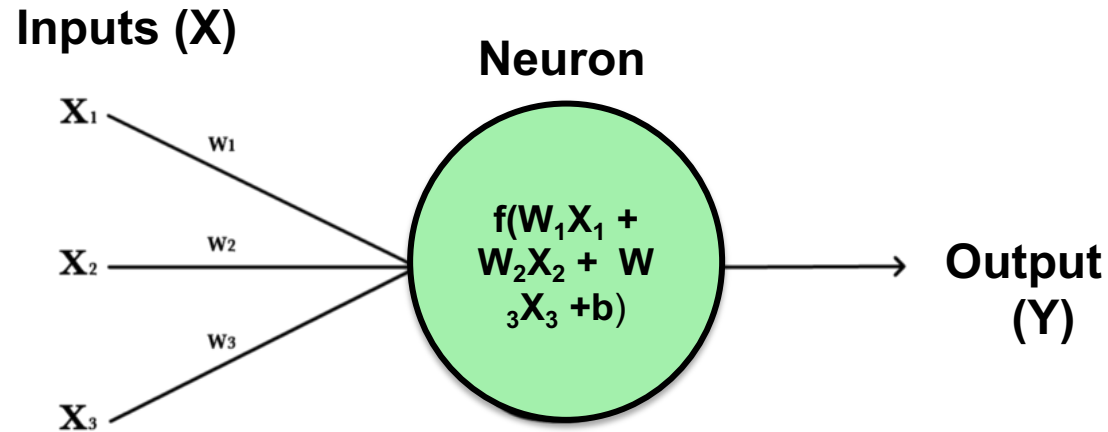
**A neuron:** Basic component of the network that is connected to other neurons of different layers of the network.



Source: [https://ml-cheatsheet.readthedocs.io/en/latest/nn\\_concepts.html](https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html)

- **Input layer:** First layer of the network. Each entry is an independent variable of the input dataset.
- **Output layer:** The final layer of the network. It returns the model prediction for a certain variable of the output dataset.
- **A hidden layer:** A layer of neurons that is located between the input and the output layer.
- **Depth:** Number of layers of neurons within the neural network.
- **Width:** The number of neurons in that particular layer

# Neural networks: Math at a neuron level



The value of a neuron will be calculated as:

$$\text{Output} = f ( W_1X_1 + W_2X_2 + \dots + W_nX_n + b )$$

with **activation function**  $f$ , **weights**  $W_i$  and inputs  $X_i$  and **bias**  $b$ .

The weights and biases will be different for each neuron.






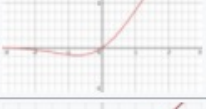



**Weights (W)** : Strength of the connectivity between the different neurons

**Bias (b)**: Additional constant attached to each neuron which is added to the weighted inputs before the activation function is applied

**Model Parameters (W,b)**: term used to refer to the set of values for the weight and biases

**Activation function**: A nonlinear function that is applied to the inputs of each neuron.

# Activation functions

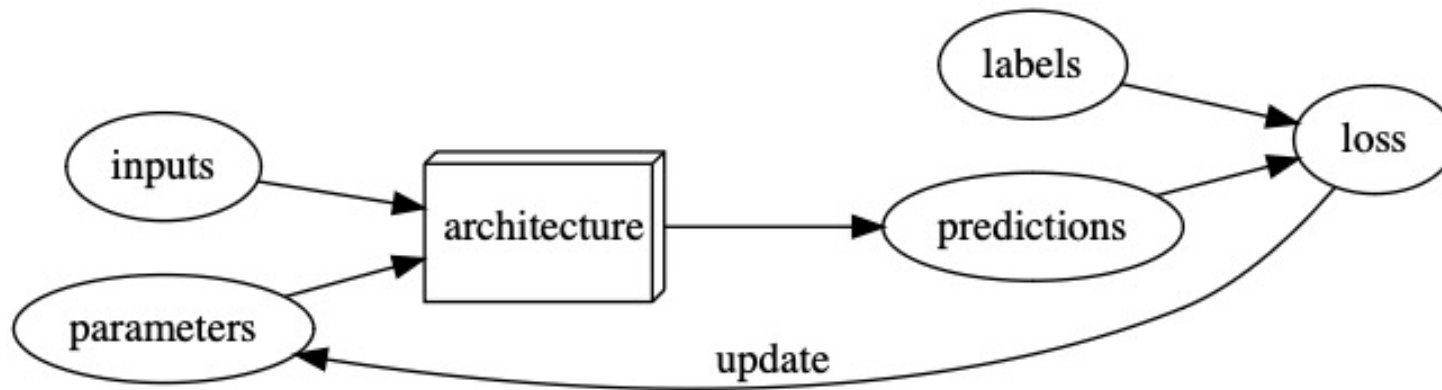
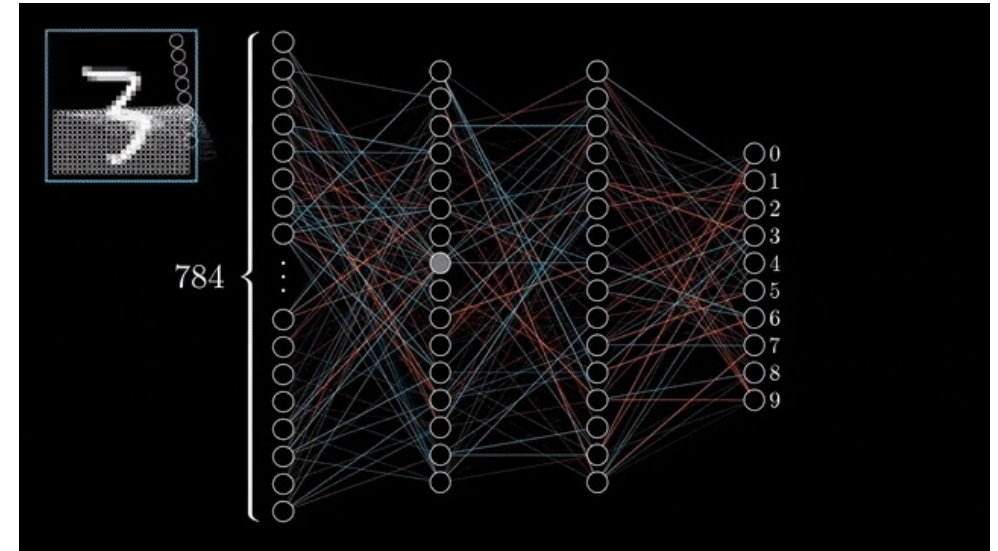
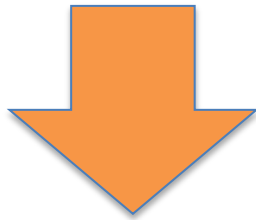
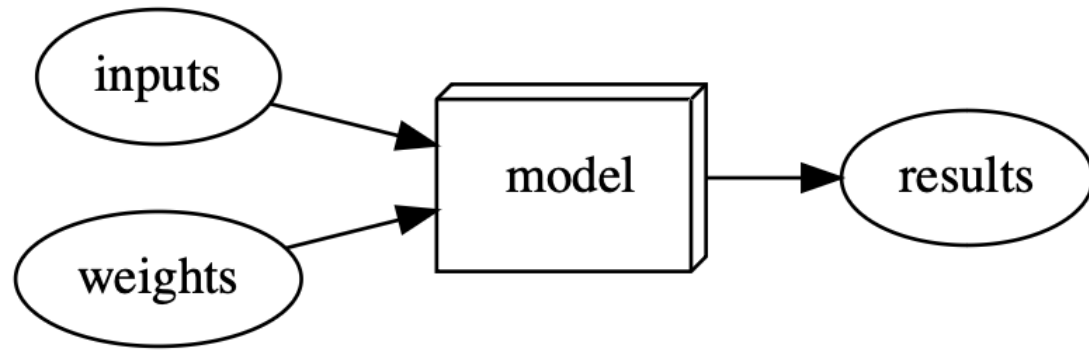
Name	Plot	Function, $f(x)$
Identity		$x$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Rectified linear unit (ReLU) <sup>[7]</sup>		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x \mathbf{1}_{x>0}$
Gaussian Error Linear Unit (GELU) <sup>[4]</sup>		$\frac{1}{2}x \left( 1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) \right)$ $= x\Phi(x)$
Softplus <sup>[8]</sup>		$\ln(1 + e^x)$
Exponential linear unit (ELU) <sup>[9]</sup>		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter $\alpha$
Scaled exponential linear unit (SELU) <sup>[10]</sup>		$\lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameters $\lambda = 1.0507$ and $\alpha = 1.67326$
Leaky rectified linear unit (Leaky ReLU) <sup>[11]</sup>		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$

## Normalisation:

The different activation functions like to work in specific number ranges and normally close to zero.

Input and output data is therefore often normalised via the subtraction of the mean and the division by the standard deviation or difference between the minimal and maximal value.

# How do you train a neural network: Training and Backpropagation

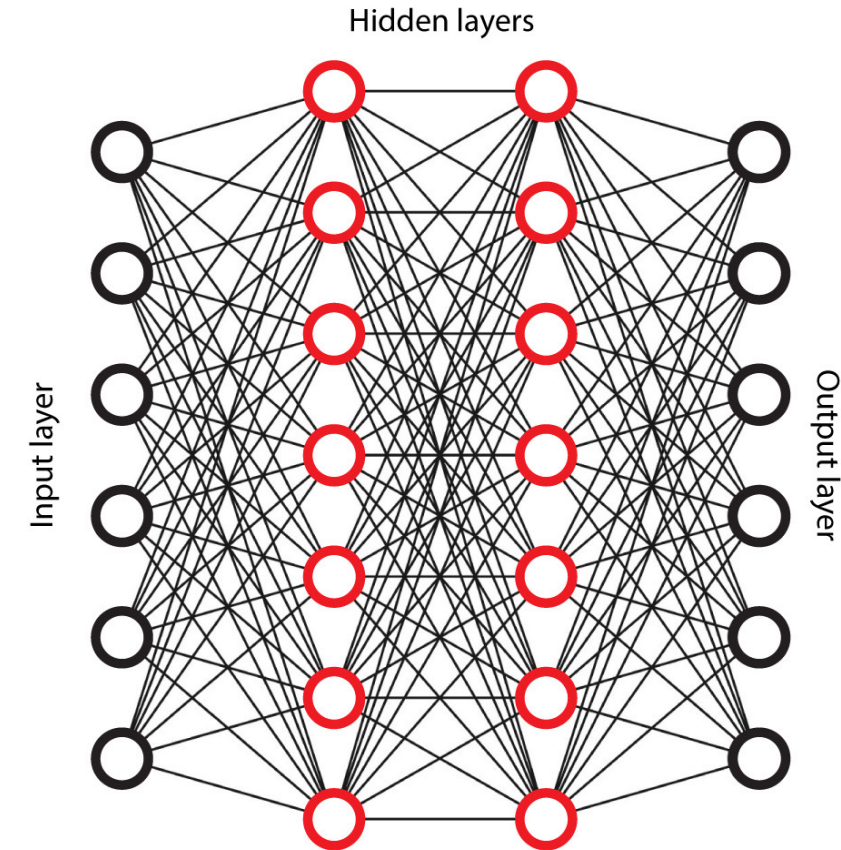
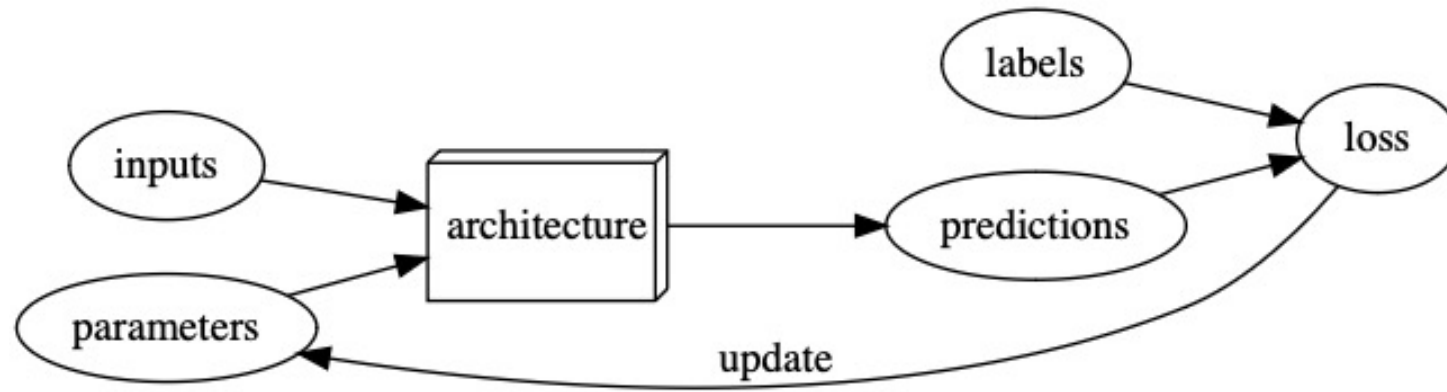


**A loss/cost function:** A quantitative measure how good the model is. For example, the mean squared error. The loss function is reduced during training

# How do you train a neural network: Training and Backpropagation

## What is training?

- Network “learning” just means changing the weights and biases to minimize a cost function, which improves the network's performance on the training data
- During the training of a neural network, we find the optimal value of the weights.
- To train, we need to find a way to change the values of weights to reduce the value of the loss function.



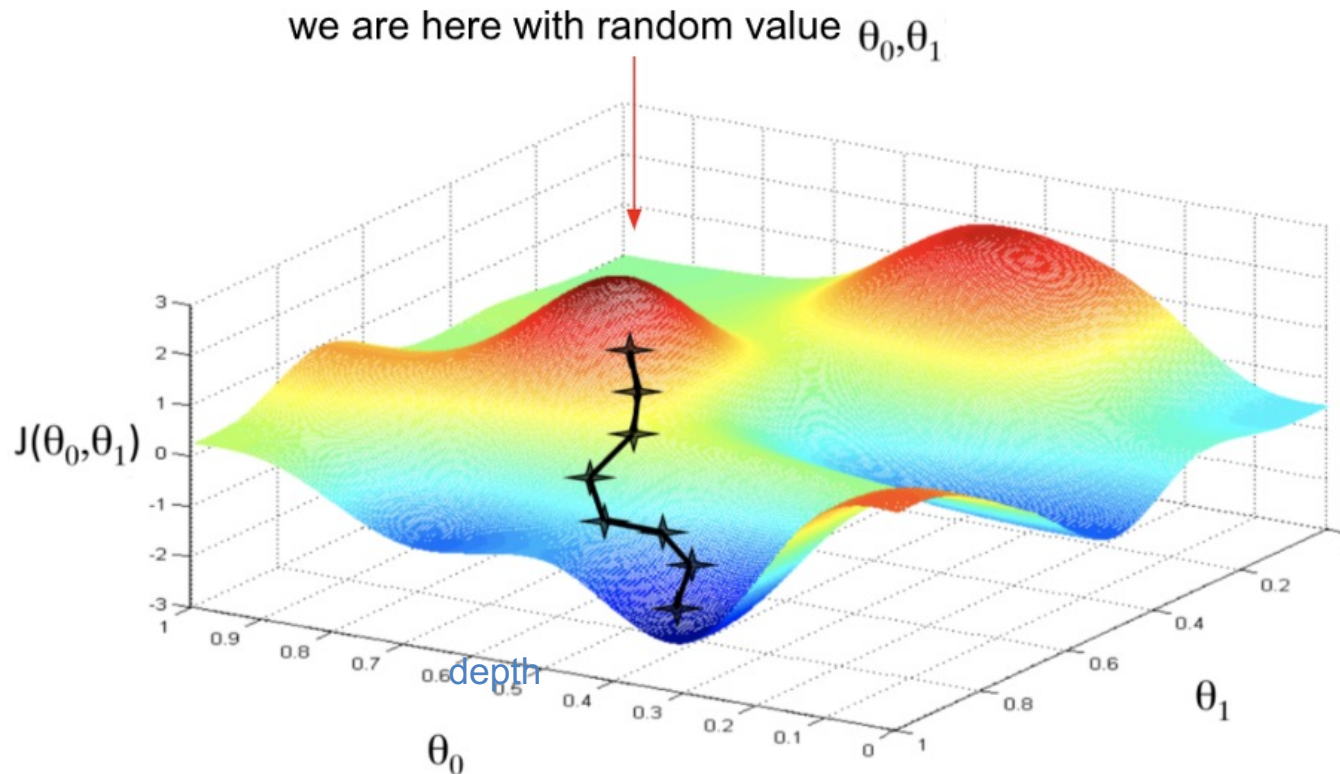
→ We need a method that can calculate the gradient of a loss function with respect to all the weights in the network.

The **Backpropagation algorithm** computes the gradient of the loss function for a single weight using the chain rule.

# Neural networks: The optimiser

**Task:** We know the gradient of the loss function for all weights via Backpropagation, the **optimiser** is now adjusting the weights and biases of the network to find the minimal value of the loss function.

**Example:** Gradient Descent



- Start with some  $\theta_0, \theta_1$
  - Keep changing  $\theta_0, \theta_1$  to reduce  $J(\theta_0, \theta_1)$
- until we hopefully end up at a minimum

**There are many different optimisers:**

1. Gradient Descent
2. Stochastic Gradient Descent
3. Adagrad
4. Adadelata
5. RMSprop
6. Adam
7. ...

Source: <https://www.kdnuggets.com/2020/05/5-concepts-gradient-descent-cost-function.html>

**What can go wrong?**

We may need to adjust the “**learning rate**”. We also may not find the global minimum. We may also find a saddle point.

# Neural networks: The training

## Learning rate:

A parameter that controls by how much the weights are changed during a training step. The learning rate can be adjusted during training (mostly reduced) either automatically or when the training stalls.

## A batch:

The number of training samples that are evaluated before the model weights are updated during training.

## An epoch:

The number of times that the training will go through the entire dataset.

## Hyper-parameter:

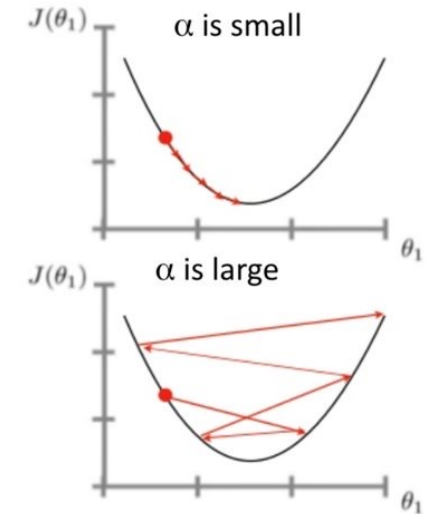
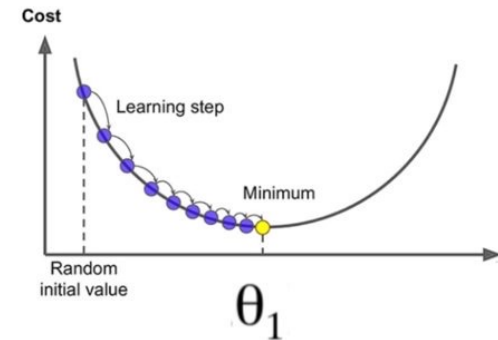
The number of neurons, number of layers, learning rate, batch size, number of epochs, kind of optimizer, kind of activation function, kind of loss function... are all hyper-parameters.

## Hyper-parameter optimization:

The task to optimise the value of the hyper-parameters which is typically based on a large number of trainings.

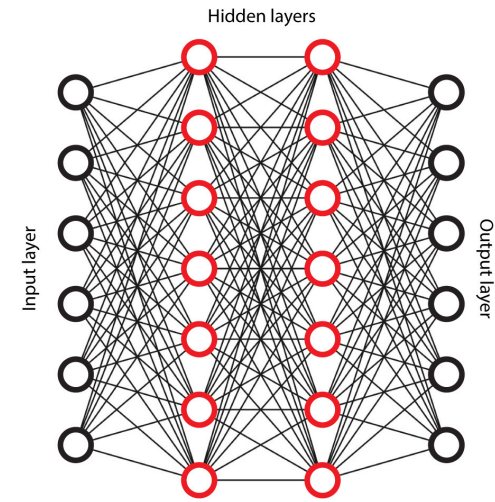
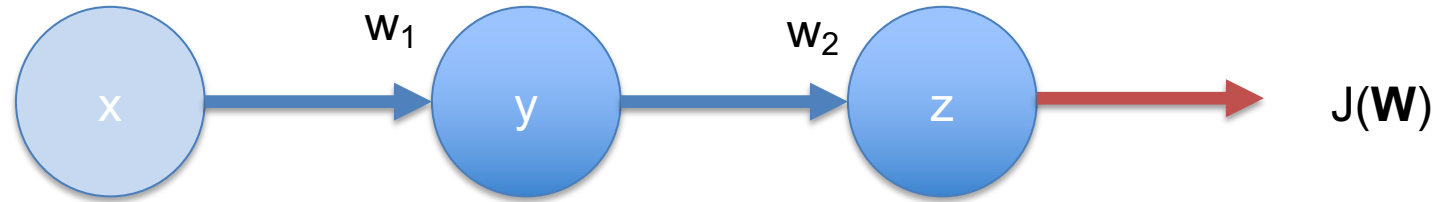
### Learning rate ( $\alpha$ )

$$\begin{array}{l} \text{repeat until convergence } \{ \\ \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\ \quad \text{(for } j = 1 \text{ and } j = 0) \\ \} \end{array}$$



Note that we used ' $:=$ ' to denote an assignment or an **update**.

# Backpropagation



We want to know how to change the  $w_i$  to reduce the cost function  $J(\mathbf{W})$  that depends on all weights  $\mathbf{W}$

To update  $w_2$ , we need to find an expression for:  $\frac{\partial J(\mathbf{W})}{\partial w_2}$

We can use the chain rule:  $\frac{\partial J(\mathbf{W})}{\partial w_2} = \frac{\partial J(\mathbf{W})}{\partial z} \cdot \frac{\partial z}{\partial w_2}$

$$W_{new} = W_{old} - \alpha \underbrace{\frac{dJ}{dW}}_{\text{gradient}}$$

How do we update  $w_1$ ?

We can use the chain rule:  $\frac{\partial J(\mathbf{W})}{\partial w_1} = \frac{\partial J(\mathbf{W})}{\partial z} \cdot \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial w_1}$

# How do you treat your data?

## Avoid overfitting!

Overfitting occurs when a machine learning model fits exactly against its training data. This will degrade performance when used for unseen data.

## How do you avoid overfitting?

- Split data into a training (~80%) and a test dataset (~20%).
- Reduce the complexity of the machine learning solution.

## Be aware!

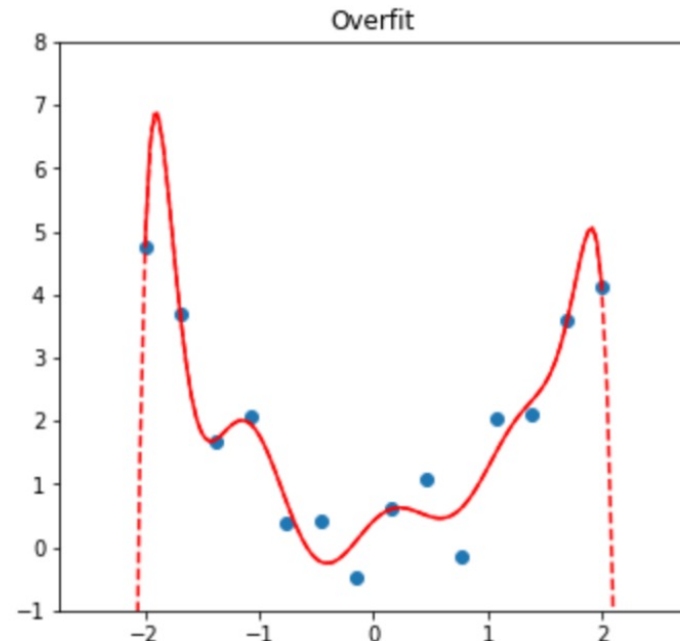
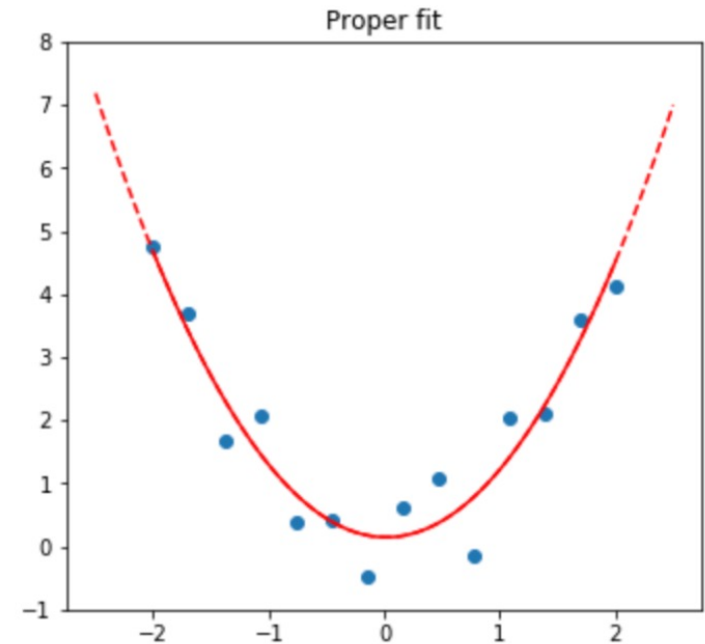
The training and test dataset needs to be uncorrelated.

**For neural networks, the hyper-parameter tuning can take so many options that you often need three datasets:**

**Training (~60%):** Part of the dataset to fit the parameters

**Validation (~20%):** Part of the dataset to estimate the quality of the machine learning solution during hyper-parameter tuning

**Test (~20%):** Part of the dataset to estimate quality of the machine learning solution of the final model



# How do you treat your data?

## Avoid overfitting!

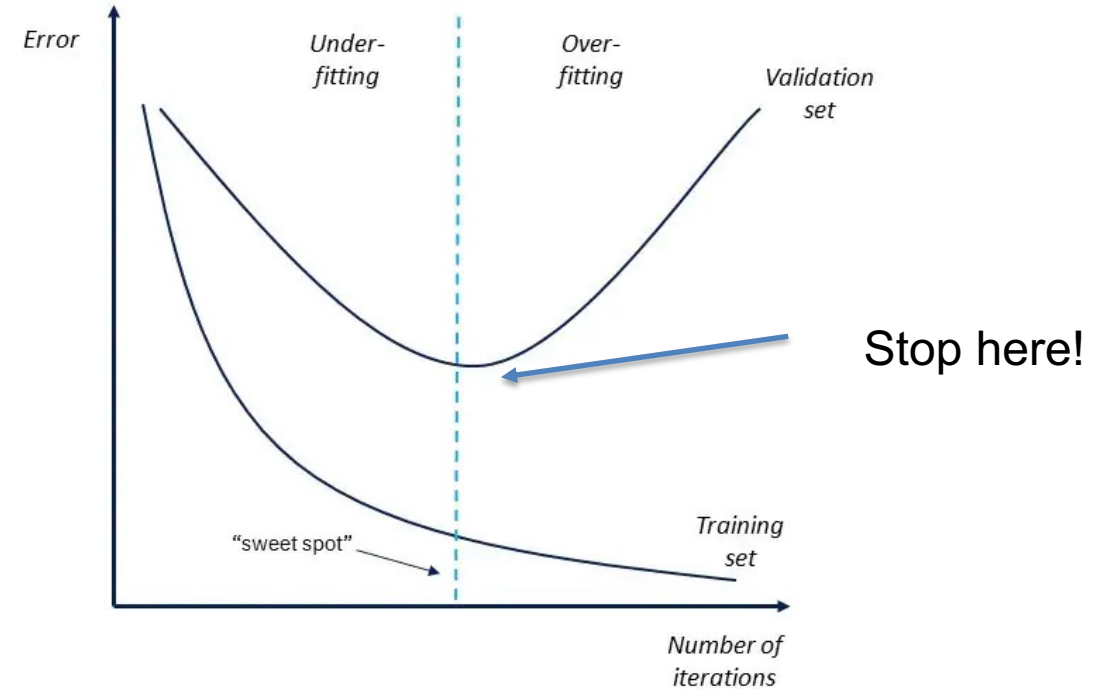
Overfitting occurs when a machine learning model fits exactly against its training data. This will degrade performance when used for unseen data.

## How do you avoid overfitting?

- Split data into a training (~80%) and a test dataset (~20%).
- Reduce the complexity of the machine learning solution.

## Be aware!

The training and test dataset needs to be uncorrelated.



Source: <https://www.ibm.com/cloud/learn/overfitting>

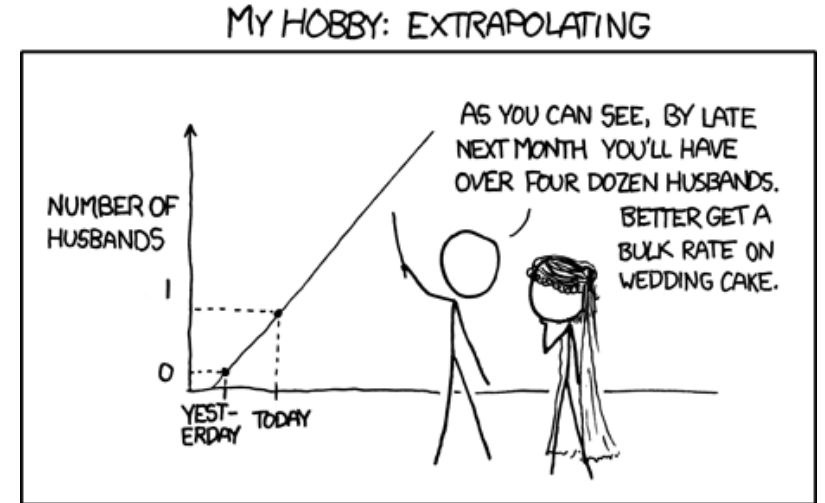
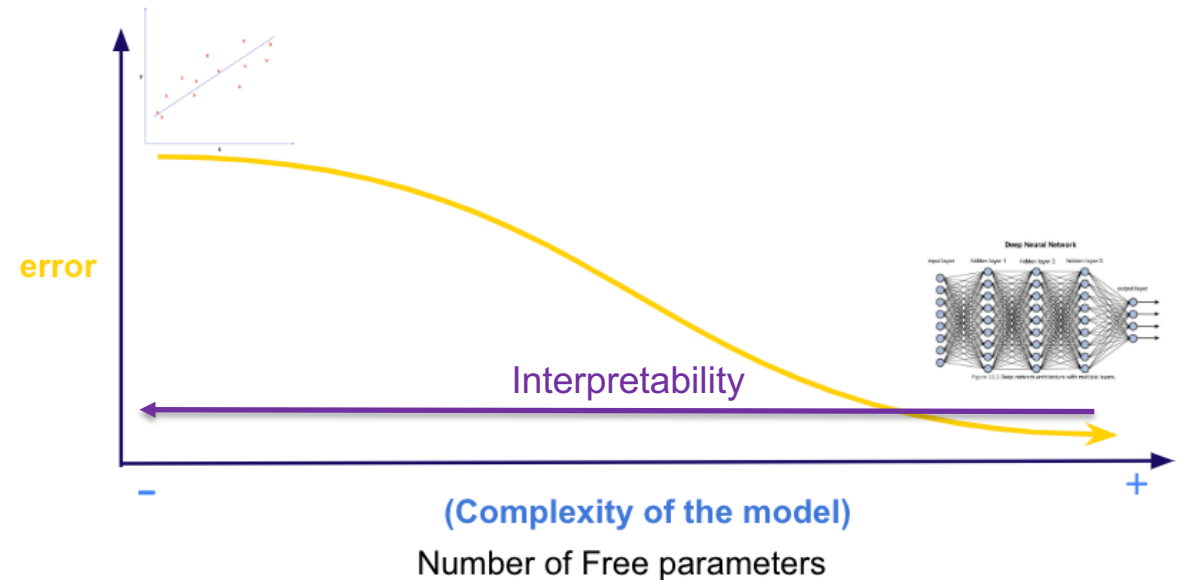
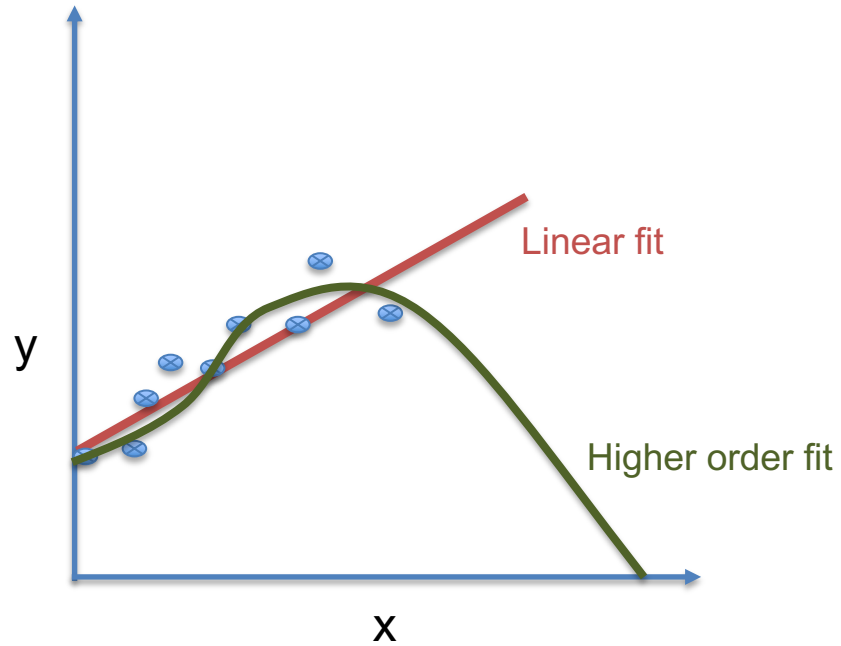
# Neural networks – the good, the bad, and the ugly

**The good news:** Neural networks are universal. No matter what function we want to learn, there is a neural network that can do the job if enough data is available and if the neural network is complex enough. See the “universal approximation theorem”.

## **The bad news:**

However, neural networks are often not the best tool for the job. They are also very difficult to interpret.

**The ugly news:** Neural networks are horrible when they are asked to extrapolate.



Source: <https://xkcd.com/605/>

# Exercises – Apply machine learning trained on historical forecast errors

## AIM – Improve ECMWF IFS forecast of temperature using postprocessing

Specifically we are trying to predict the different between station observations 2m-temperature and the IFS forecast prediction at the nearest gridpoint to the observation location

### Inputs

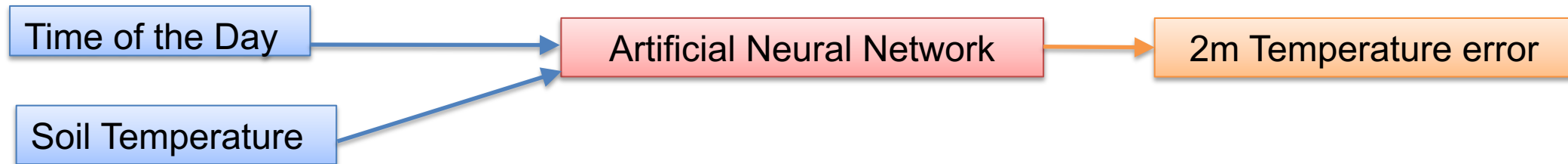
Time of the Day

Soil Temperature

Artificial Neural Network

### Outputs

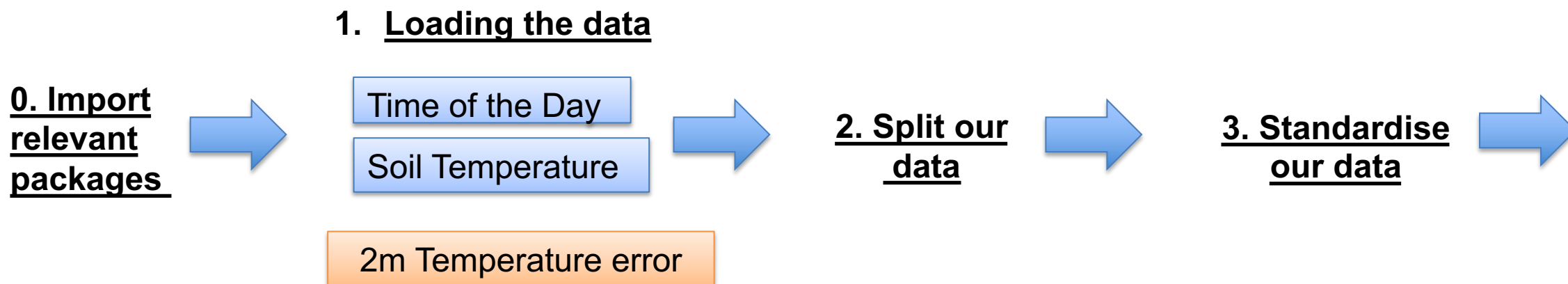
2m Temperature error



# Exercises – Apply machine learning trained on historical forecast errors

## AIM – Improve ECMWF IFS forecast of temperature using postprocessing

Specifically we are trying to predict the different between station observations 2m-temperature and the IFS forecast prediction at the nearest gridpoint to the observation location



# Exercises – Apply machine learning trained on historical forecast errors

## AIM – Improve ECMWF IFS forecast of temperature using postprocessing

Specifically we are trying to predict the different between station observations 2m-temperature and the IFS forecast prediction at the nearest gridpoint to the observation location

### 4. Define our Neural Network

```
model = fullyconnected_functional(  
    time_of_day_train_norm.shape[1],  
    depth=3,width=32,  
    activation='tanh',learning_rate=10**(-3))
```

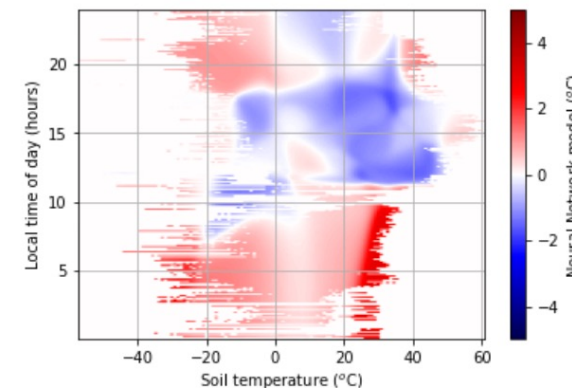


### 5. Train the NN

```
# Train the model using the data  
model.fit(time_of_day_train_norm,  
    forecast_error_train,  
    validation_data=(time_of_day_test_norm, forecast_error_test),  
    batch_size = 128,  
    epochs = 3  
)
```



### 6. Generate new predictions and asses the model performance



# Time for exercises

**Thanks to Matthew Chantry!**

**How to access the exercises:**

- Log in into Jupyter JSC
- Clone the following repo - [Maelstrom\\_Bootcamp](#)
- Go to the folder [Intro Exercise/Tensorflow101.ipynb](#)
- Choose the kernel 'venv\_bootcamp'

*There is no need to follow these steps right now. You will have time to go over this notebook later today, and if you have any issues or further questions, you are very welcome to send me a message with them!*

# Transfer learning

Let's assume you have build a deep learning tool to detect cats in pictures based on 1,000,000 labelled pictures.

Now you are asked to build a similar tool to detect foxes, but you only have 1,000 labelled pictures. How do you get the best results?

You can use the network that was trained to detect cats as a starting point and retrain the network on the pictures with foxes. The results will be better than the use of only 1,000 labelled pictures.

This approach is often used and you can download and use pre-trained networks for specific tasks to refine them on your own dataset.

Example: Twitter data, BERT and MAELSTROM.

# Regression analysis

*“In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome' or 'response' variable) and one or more independent variables (often called 'predictors', 'covariates', 'explanatory variables' or 'features').” (Wikipedia)*

## Concept:

- You assume that a mapping will take a certain shape, for example:  $f(x) = \alpha x + \beta y + \xi x*y + \dots$
- You learn the coefficients ( $\alpha$ ,  $\beta$ ,  $\xi$ ) of the different terms of your solution by minimizing a cost function

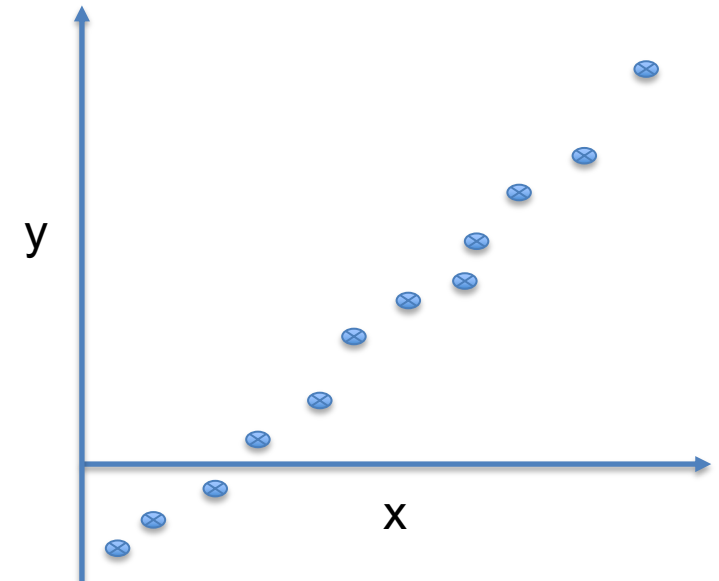
A typical cost function is the minimization of the “least squares”

→ Minimize the sum of the squares of the residuals

**Example:  $x \rightarrow y$**

Assumption:  $f(x) = \alpha x + \beta$

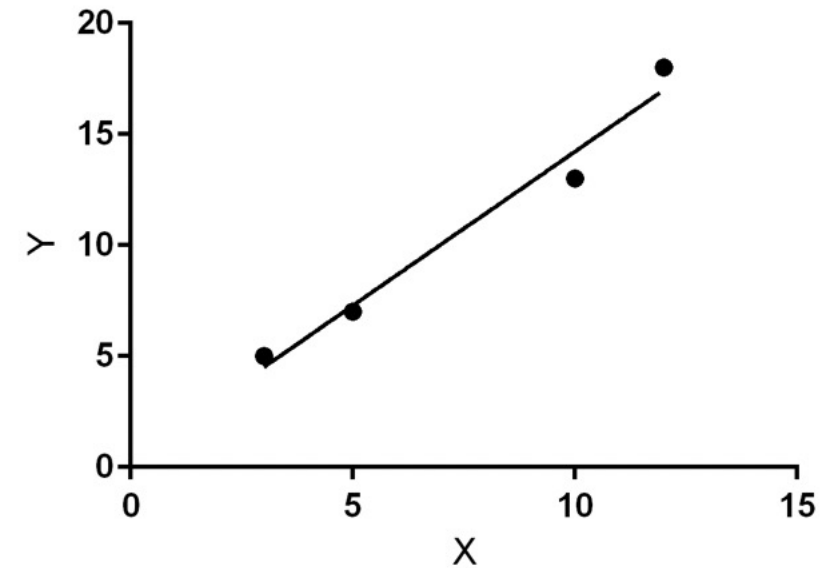
with  $\alpha$  and  $\beta$  constants that need to be learned



# Linear Regression Example

Assume the following data:

	X	Y
1	3	5
2	5	7
3	10	13
4	12	18



Assume that you can approximate this by a linear regression with:  $y_{lr} = ax + b$ .

**How do you calculate  $a$  and  $b$ ?**

**Let's assume a loss function:**

$$e = (y_1 - y_{lr1})^2 + (y_2 - y_{lr2})^2 + (y_3 - y_{lr3})^2 = (y_1 - (ax_1 + b))^2 + (y_2 - (ax_2 + b))^2 + (y_3 - (ax_3 + b))^2$$

**Now be smart and calculate the regression equation:**

$$b = \frac{N\sum xy - (\sum x)(\sum y)}{N\sum x^2 - (\sum x)^2} \text{ and } a = \frac{\sum y - b\sum x}{N} \text{ with } N \text{ being the number of samples.}$$

Then you never need to be smart again, and can use your equation even if you have 10,000,000 samples.

# Regression analysis

**There are different types of regression that can be used:**

- Simple linear regression – one variable is predicted with one variable
- Multiple linear regression – one variable is predicted with more than one variable
- Multivariate linear regression – multiple variables are predicted
- Nonlinear regression – including exp, log, sin, cos...

...

## **Advantage:**

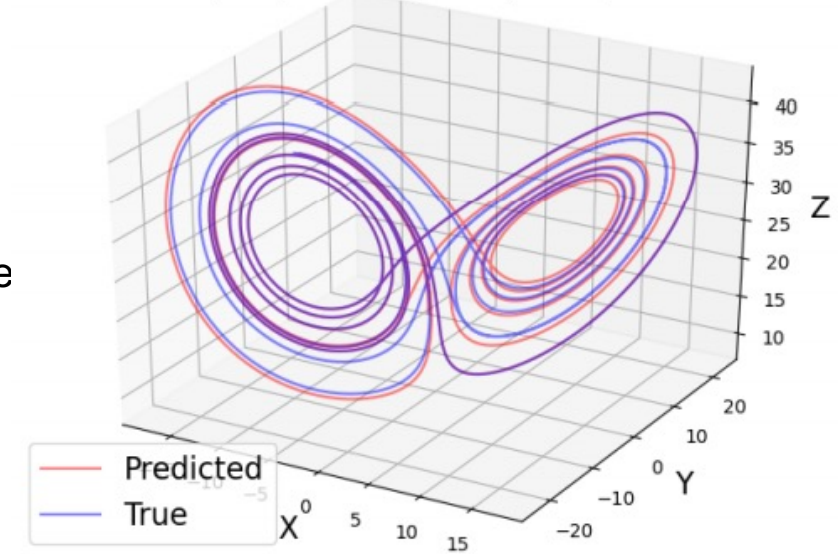
The result is a formula that can be interpreted. Terms that are not important can be dropped.

## **Disadvantage:**

The quality of the result depends on the initial assumptions

**Imagine that you are learning the equations of the Lorenz system from model output data.**

Testing Trajectories, D2R2, Extrapolation



$$\frac{dx}{dt} = \sigma(y - x)$$

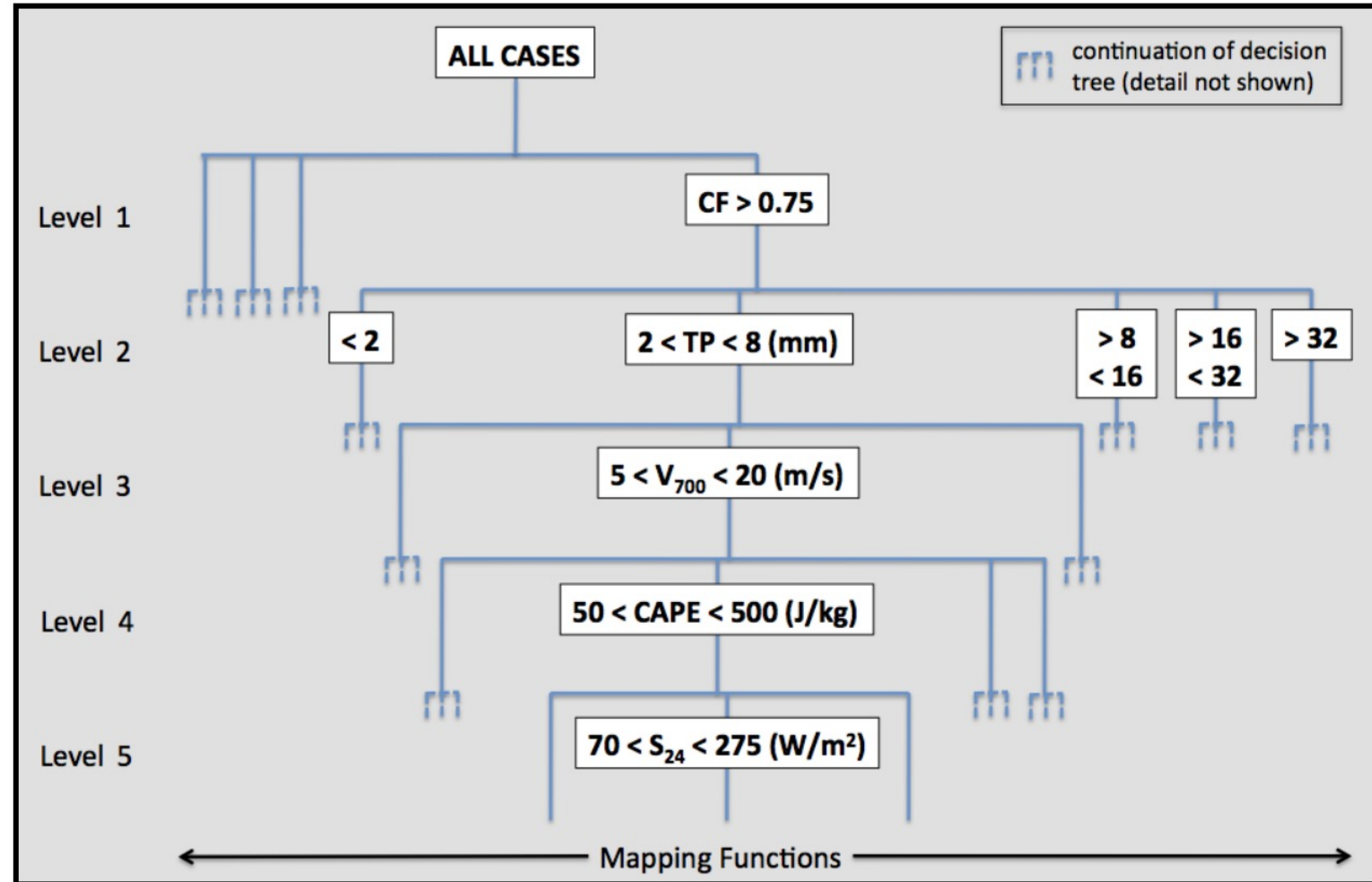
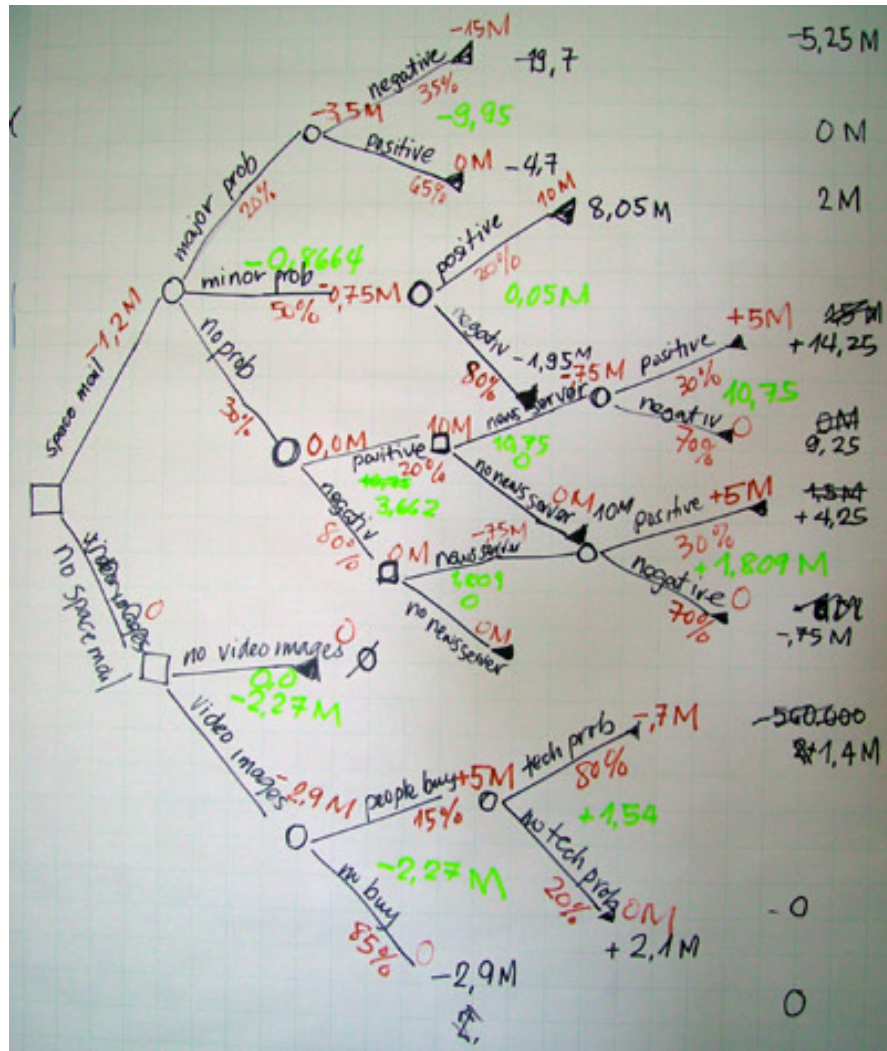
$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

# Decision trees

Next to neural networks, there are plenty of other ways to perform “machine learning”. A very prominent class of methods are decision trees.

What needs to be learned? What are advantages or disadvantages?



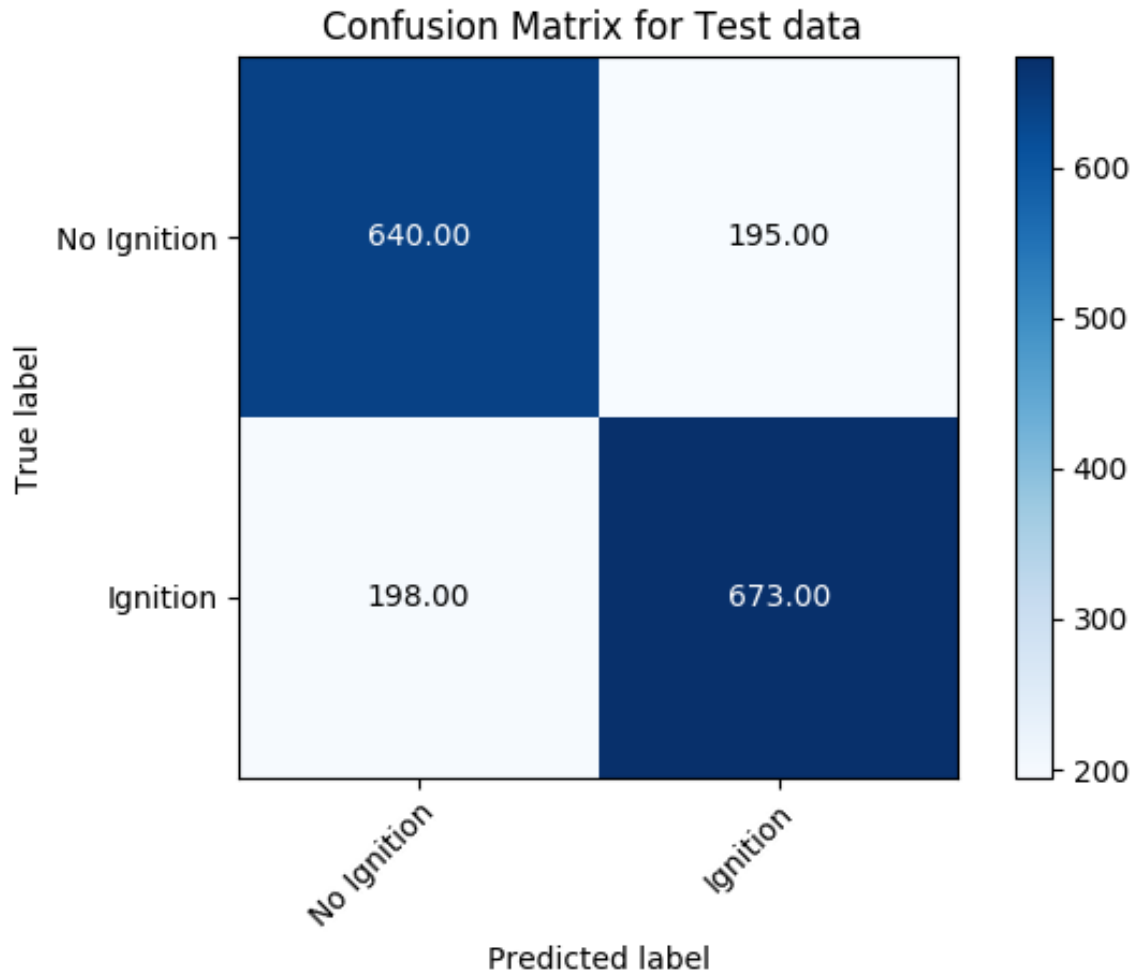
# Decision trees

- The number of layers and the choice of the node splitting function can be adjusted.
- Good splits lead to “pure” results and show trunks of similar strength.
- Again, the decision tree can overfit.

## There can be more than one tree!

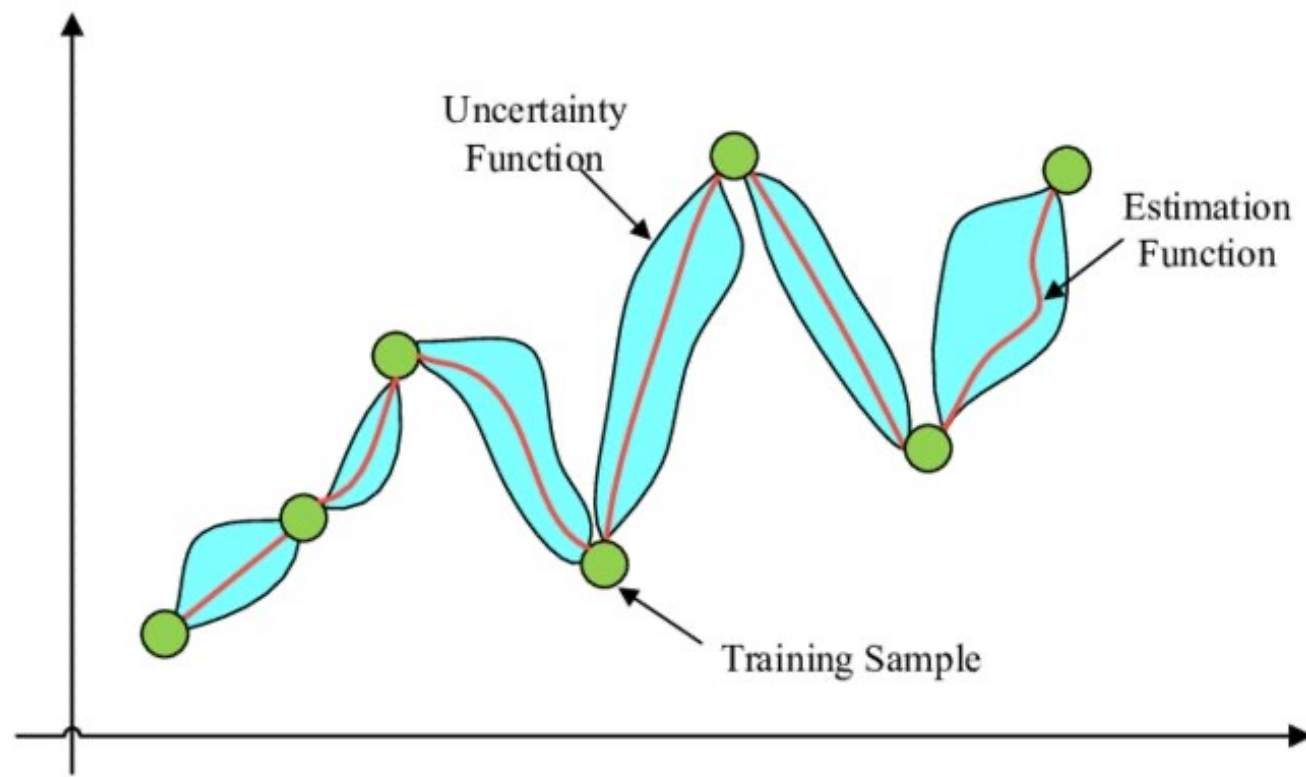
- **Random forest** methods are training a multitude of decision trees eventually using the mean prediction or the value with the most hits as a result.
- **Boosting methods** train decision trees as a sequence with each new tree trained to perform well on the samples the previous tree got wrong.

# Science example: Detect the risk for the ignition of wild fires by lightnings



- Observations for 15 variables are used as inputs including soil moisture, 2m temperature, soil type, vegetation cover, relative humidity, and precipitation
- The rate of radiant heat output from the Global Fire Assimilation System (GFAS) of CAMS (monitored by the MODIS satellite) was used to generate a “truth”
- 12,000 data points were used for training
- Different machine learning tools (decision trees, random forest and Ada Boost) are used to classify the cases into “ignition” and “no-ignition”
- The best classifier has an accuracy of 77 %

# Gaussian Processes – in a nutshell



Source: Dong, Qin, Mo electronics 2019

**Bayes' rule** is used to gradually update the probability of an event as the evidence or data is gathered sequentially.

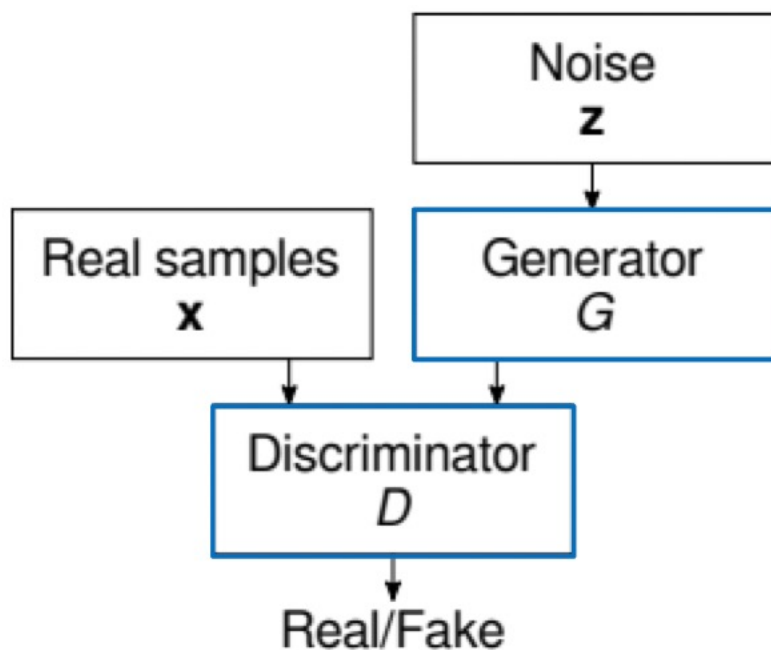
A **Gaussian process** is a probability distribution that is updated using Bayes' rule and observing training data.

**Main Advantage:** Gaussian Processes know when they are uncertain.

# Can we replace conventional weather forecast systems by deep learning?

- Data?
- Complexity?
- Network architecture?
- Software?
- Hardware?
- Transfer learning?

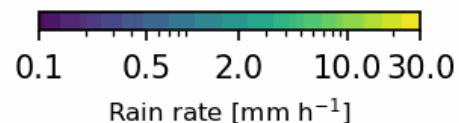
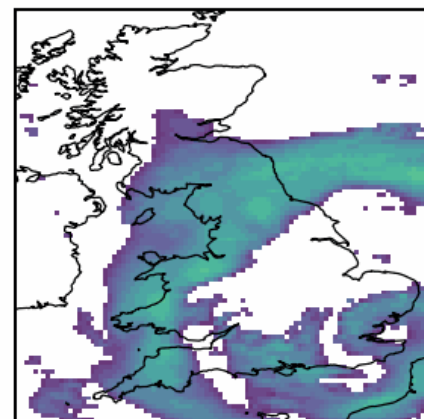
# Downscaling with Generative Adversarial Networks



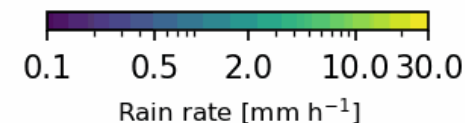
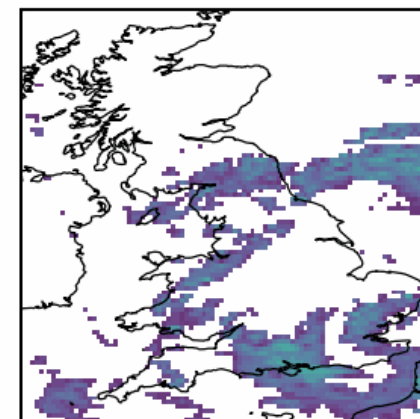
**Input:** IFS Model Simulation fields on coarse (9 km) grid

**Output:** Precipitation observation on fine (1 km) grid

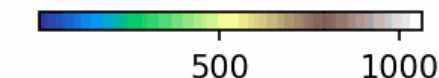
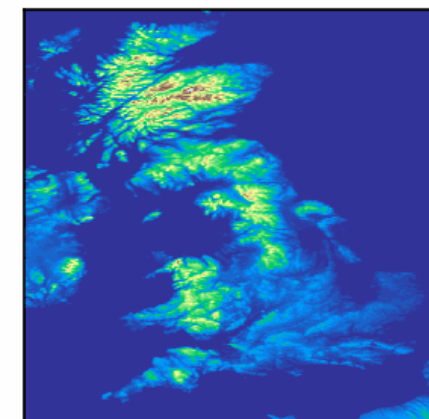
IFS - total precip



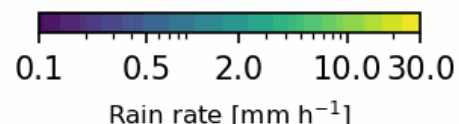
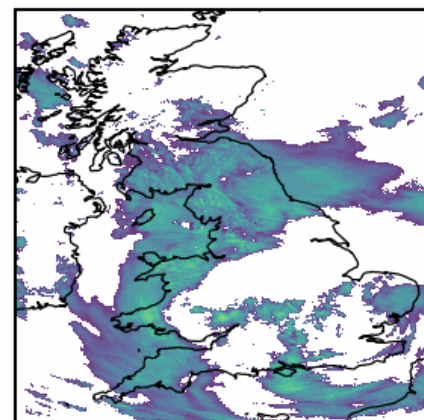
IFS - convective precip



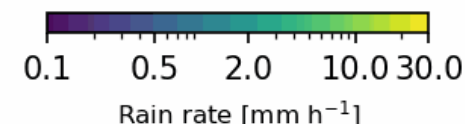
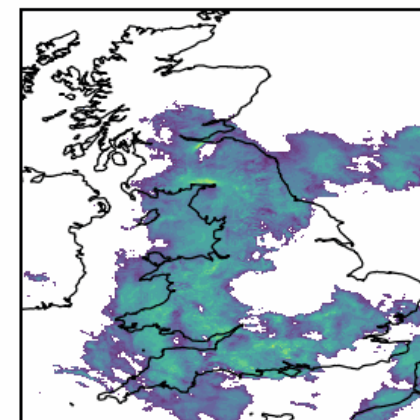
Orography



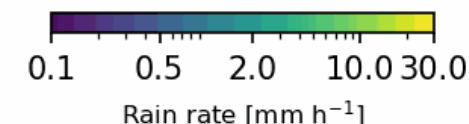
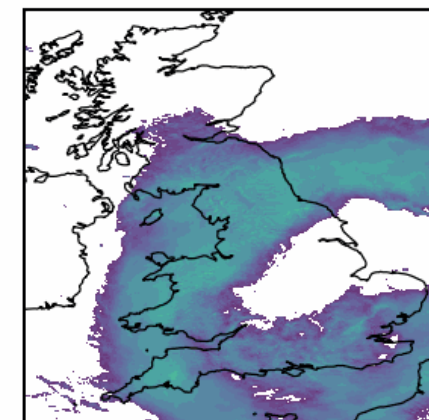
NIMROD - ground truth



GAN prediction



GAN - mean prediction





The strength of a common goal

# Exercises – Apply machine learning trained on historical forecast errors

Neural network

Random forest

Linear regression

Forecast temperature

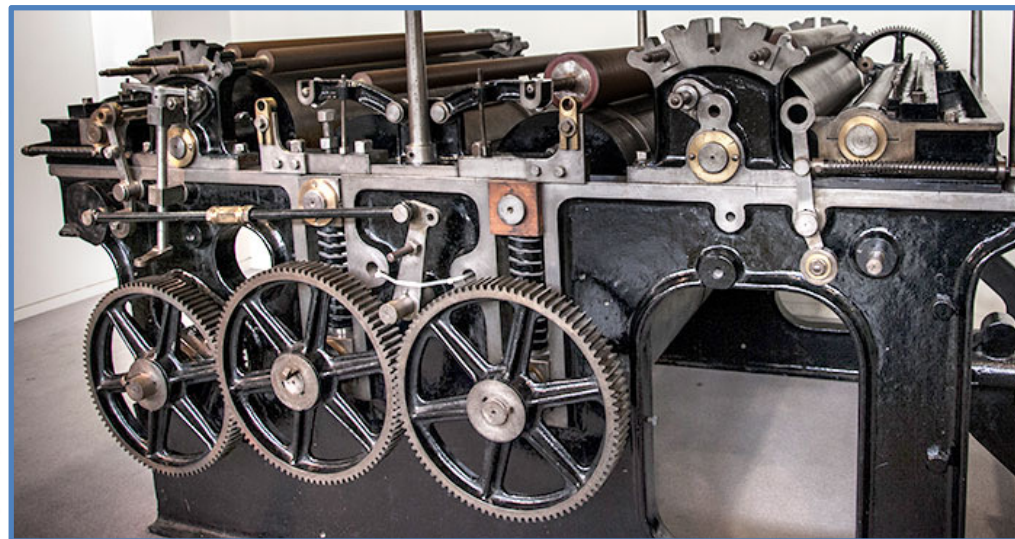
Forecast wind

Forecast sunshine

Measured temperature

Snow cover

... more ...



(credit: Science museum, London)

After a forecast has been made:

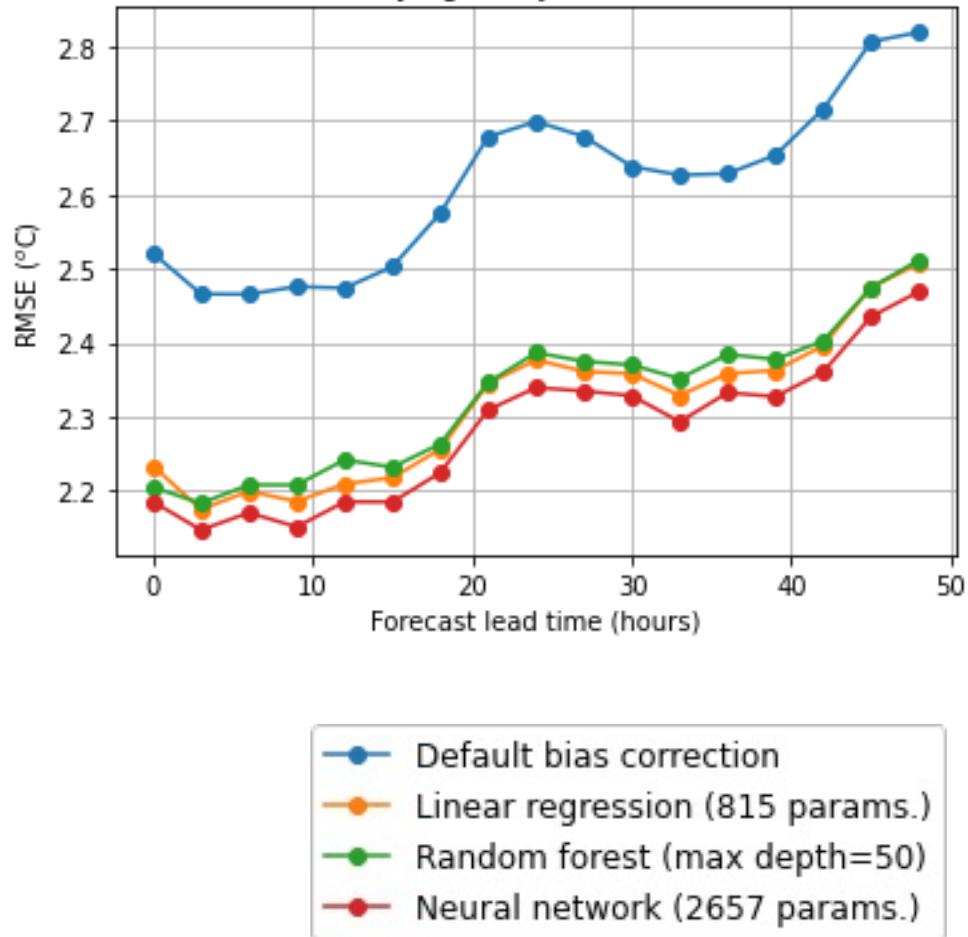
Forecast correction

Forecast uncertainty

# Machine learning applied to forecast 2m temperature and 10m wind

Fenwick Cooper, Zied Ben Bouallegue, Matthew Chantry, Peter Düben, Peter Bechtold, Irina Sandu

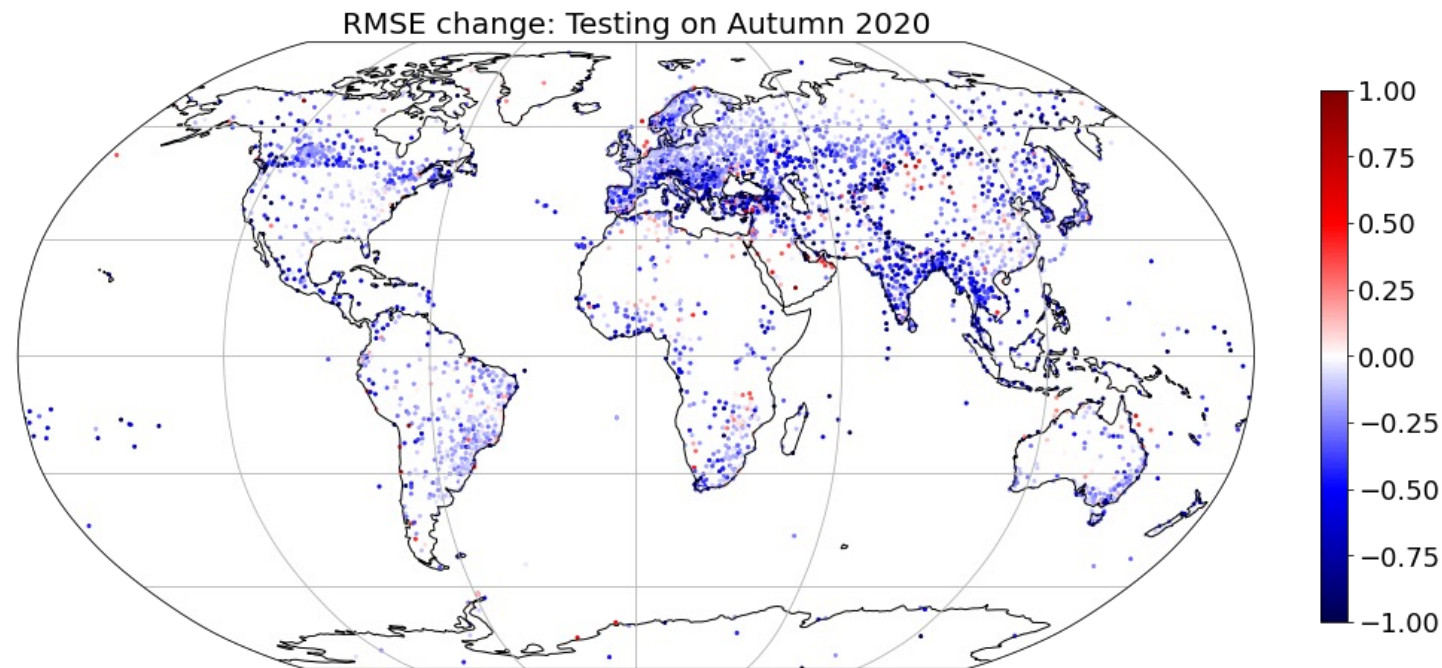
Verifying on DJF (dataset 1)



Example: 2m temperature, Winter 2020, 1 year of training data

Root-Mean-Squared Error (RMSE) with respect to station measurements (SYNOP).

All stations (left) – Individual stations change (below)



# Time for excersises

**Thanks to Fenwick Cooper and Matthew Chantry!**

## **How to access the exercises:**

**Desired:** (If you do not have it yet, download - look e.g. here for more information: <https://jupyter.org/> - and) run Jupyter notebooks locally on your laptop/computer, and install the few python libraries that are needed. You can find a tar file that contains the exercises here:  
<https://drive.google.com/file/d/1PCkX-c1HQCjvmEmGJLyhPqhEGp6oVPiM/view?usp=sharing>

We will be looking into the two notebooks:

`./ML_Crash_Course/Example_regression.ipynb`

`./ML_Crash_Course/Example_NNs.ipynb`