ENGO 697

Remote Sensing Systems and Advanced Analytics

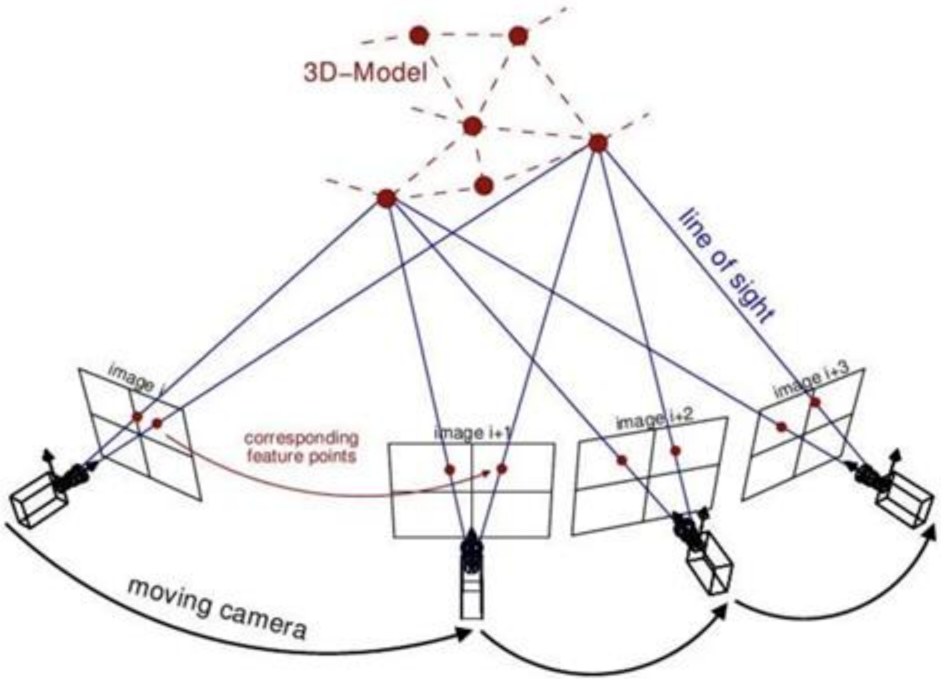
Session 11: How to Integrate Knowledge (e.g., Physical and Geometric models) into Deep Learning Models

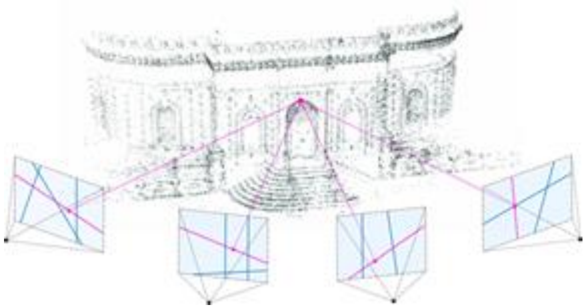
Dr. Linlin (Lincoln) Xu

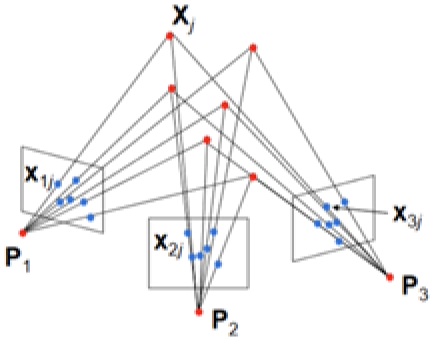
Linlin.xu@ucalgary.ca

Office: ENE 221

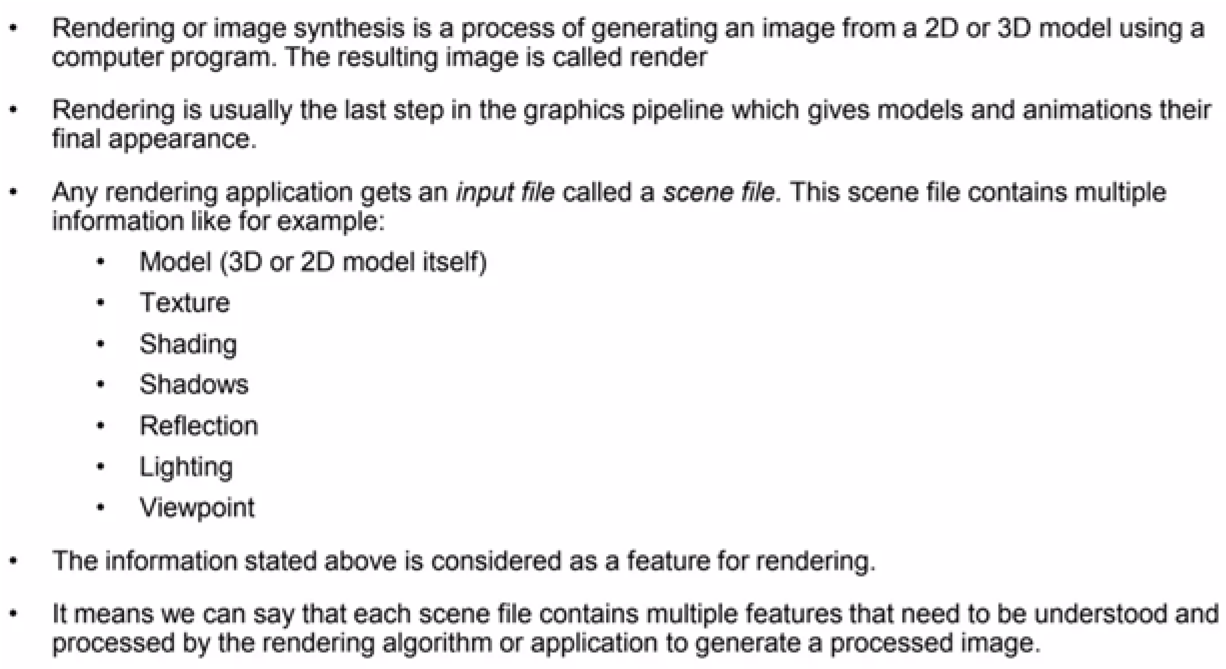
Structure from Motion: 2D -> 3D





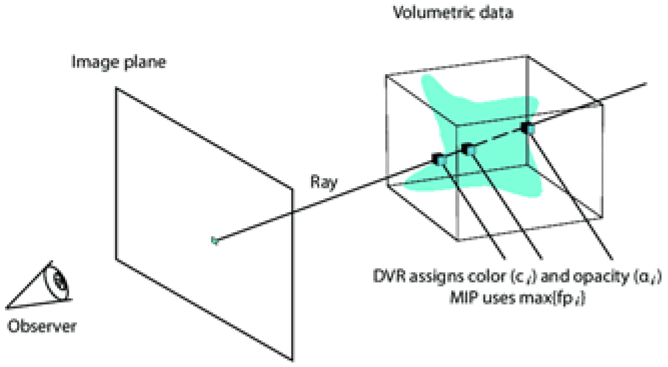


Rendering: 3D -> 2D



- Rendering or image synthesis is a process of generating an image from a 2D or 3D model using a computer program. The resulting image is called render
- Rendering is usually the last step in the graphics pipeline which gives models and animations their final appearance.
- Any rendering application gets an *input file* called a *scene file*. This scene file contains multiple information like for example:
  - Model (3D or 2D model itself)
  - Texture
  - Shading
  - Shadows
  - Reflection
  - Lighting
  - Viewpoint
- The information stated above is considered as a feature for rendering.
- It means we can say that each scene file contains multiple features that need to be understood and processed by the rendering algorithm or application to generate a processed image.

Rendering: 3D -> 2D



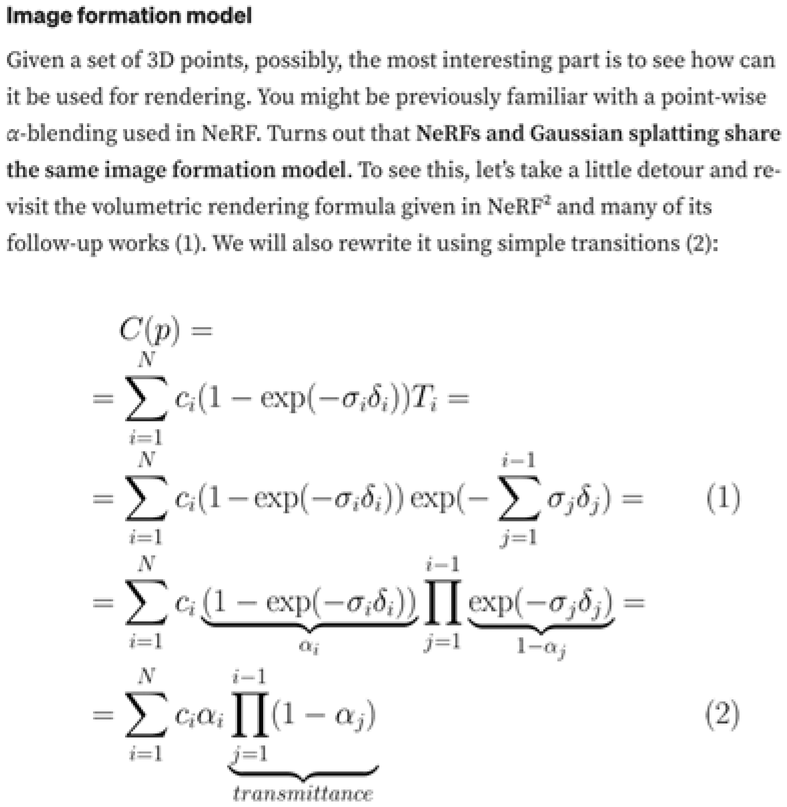


Image formation model

Given a set of 3D points, possibly, the most interesting part is to see how can it be used for rendering. You might be previously familiar with a point-wise α-blending used in NeRF. Turns out that **NeRFs and Gaussian splatting share the same image formation model.** To see this, let's take a little detour and revisit the volumetric rendering formula given in NeRF[2] and many of its follow-up works (1). We will also rewrite it using simple transitions (2):

$$
\begin{aligned}
C(p) &= \\
&= \sum_{i=1}^{N} c_i(1-\exp(-\sigma_i\delta_i))T_i = \\
&= \sum_{i=1}^{N} c_i(1-\exp(-\sigma_i\delta_i))\exp(-\sum_{j=1}^{i-1}\sigma_j\delta_j) = \qquad (1) \\
&= \sum_{i=1}^{N} c_i \underbrace{(1-\exp(-\sigma_i\delta_i))}_{\alpha_i} \prod_{j=1}^{i-1} \underbrace{\exp(-\sigma_j\delta_j)}_{1-\alpha_j} = \\
&= \sum_{i=1}^{N} c_i\alpha_i \underbrace{\prod_{j=1}^{i-1}(1-\alpha_j)}_{transmittance} \qquad (2)
\end{aligned}
$$

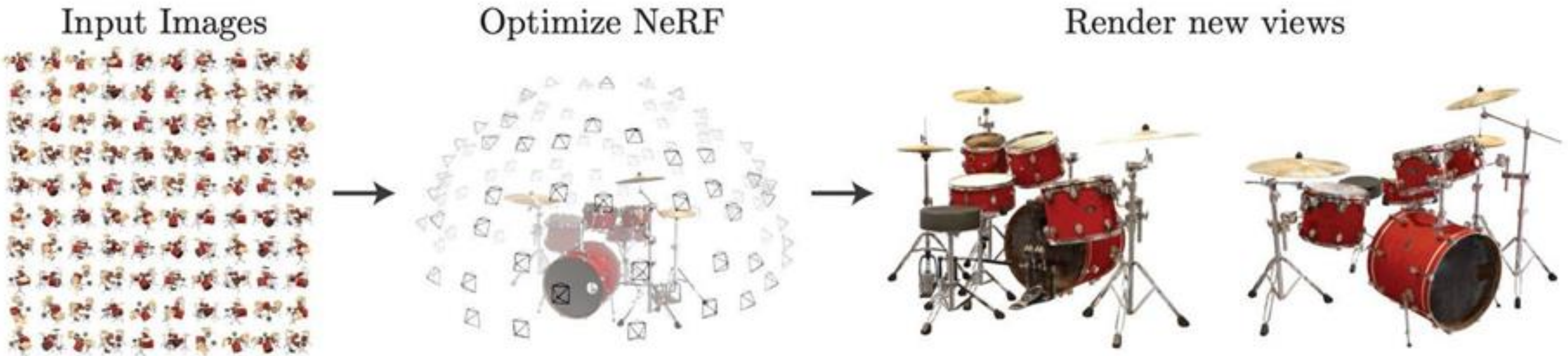
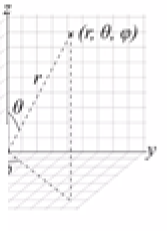


Fig. 1: We present a method that optimizes a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. We use techniques from volume rendering to accumulate samples of this scene representation along rays to render the scene from any viewpoint. Here, we visualize the set of 100 input views of the synthetic *Drums* scene randomly captured on a surrounding hemisphere, and we show two novel views rendered from our optimized NeRF representation.

Neural radiative field



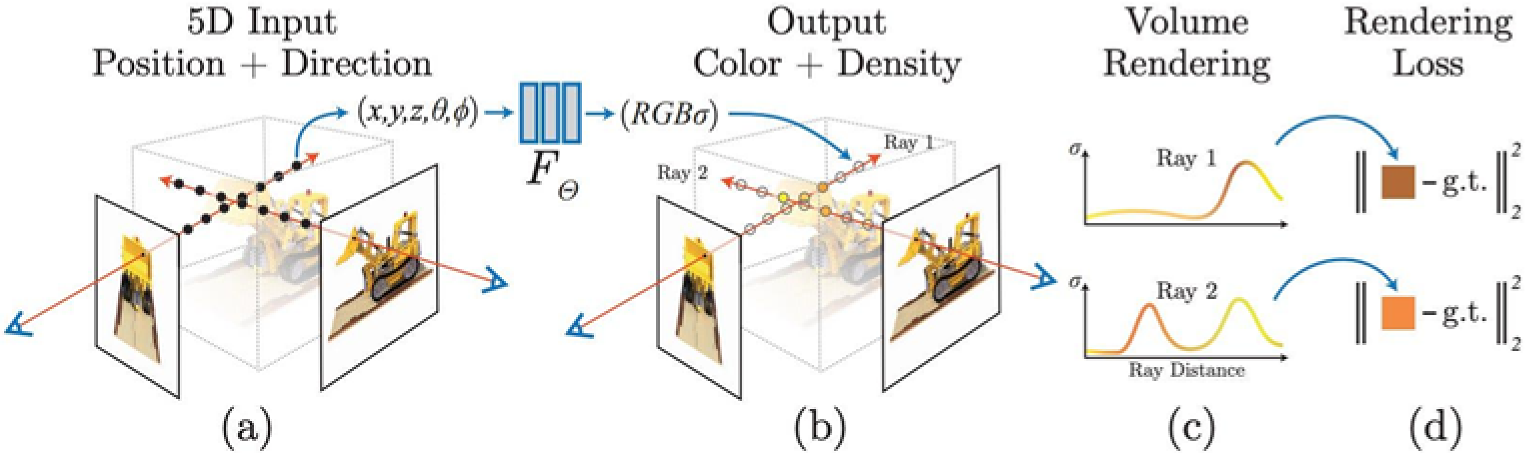
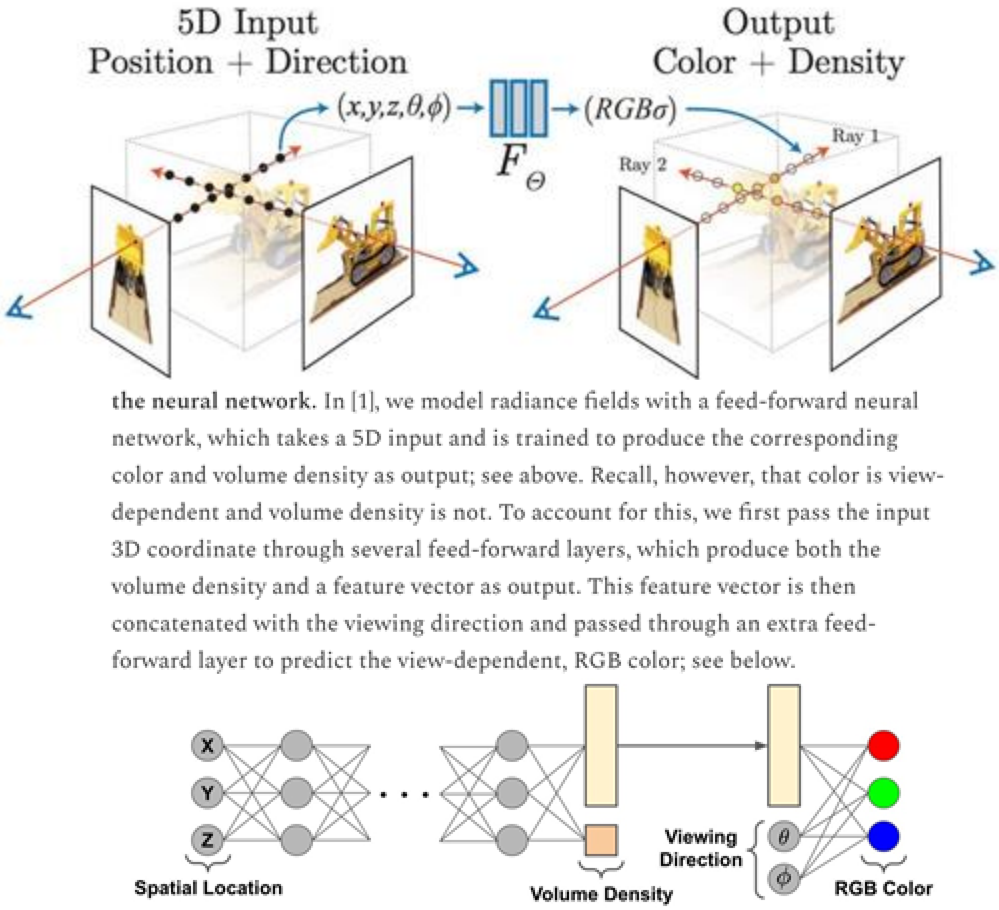
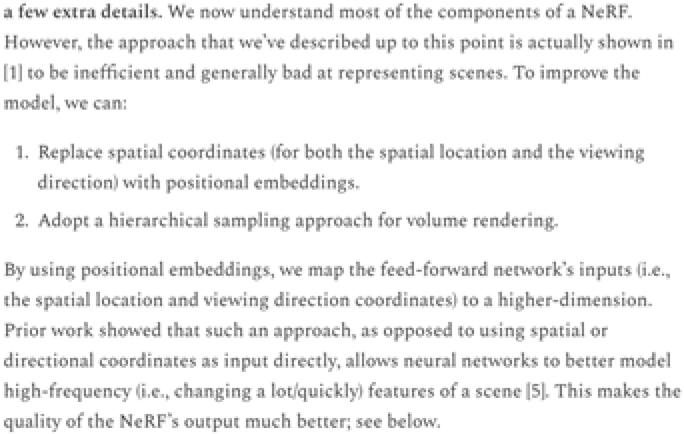


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).



the neural network. In [1], we model radiance fields with a feed-forward neural network, which takes a 5D input and is trained to produce the corresponding color and volume density as output; see above. Recall, however, that color is view-dependent and volume density is not. To account for this, we first pass the input 3D coordinate through several feed-forward layers, which produce both the volume density and a feature vector as output. This feature vector is then concatenated with the viewing direction and passed through an extra feed-forward layer to predict the view-dependent, RGB color; see below.

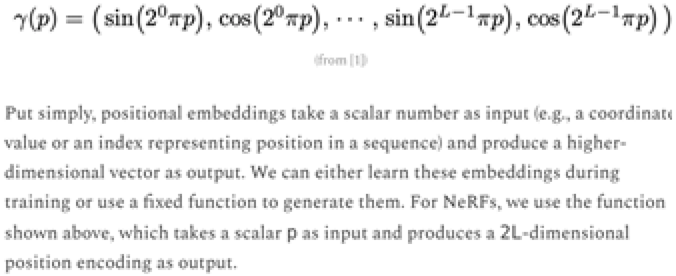
View dependent RGB output



a few extra details. We now understand most of the components of a NeRF. However, the approach that we've described up to this point is actually shown in [1] to be inefficient and generally bad at representing scenes. To improve the model, we can:

1. Replace spatial coordinates (for both the spatial location and the viewing direction) with positional embeddings.
2. Adopt a hierarchical sampling approach for volume rendering.

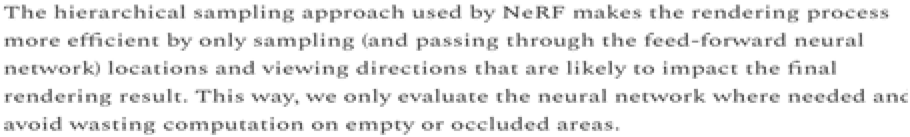
By using positional embeddings, we map the feed-forward network's inputs (i.e., the spatial location and viewing direction coordinates) to a higher-dimension. Prior work showed that such an approach, as opposed to using spatial or directional coordinates as input directly, allows neural networks to better model high-frequency (i.e., changing a lot/quickly) features of a scene [5]. This makes the quality of the NeRF's output much better; see below.



$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right)$$

(from [1])

Put simply, positional embeddings take a scalar number as input (e.g., a coordinat value or an index representing position in a sequence) and produce a higher-dimensional vector as output. We can either learn these embeddings during training or use a fixed function to generate them. For NeRFs, we use the function shown above, which takes a scalar p as input and produces a 2L-dimensional position encoding as output.



The hierarchical sampling approach used by NeRF makes the rendering process more efficient by only sampling (and passing through the feed-forward neural network) locations and viewing directions that are likely to impact the final rendering result. This way, we only evaluate the neural network where needed anc avoid wasting computation on empty or occluded areas.

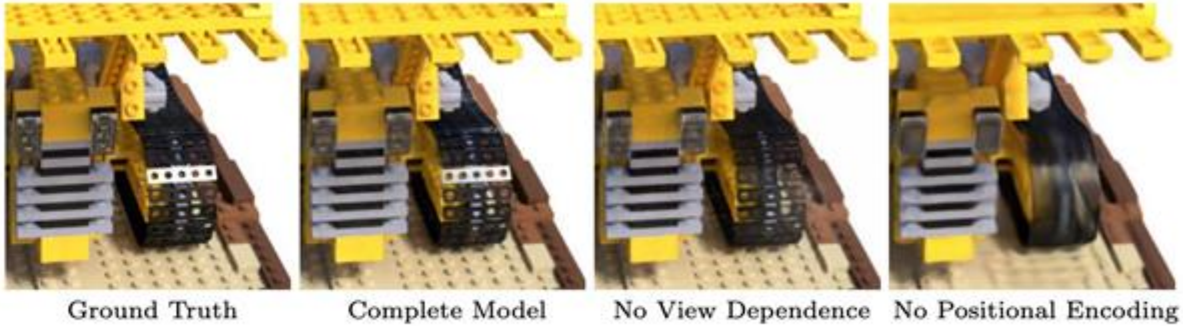
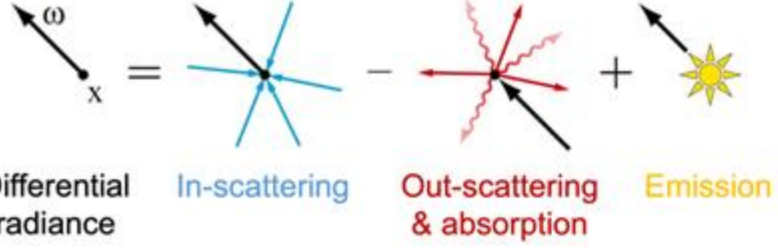
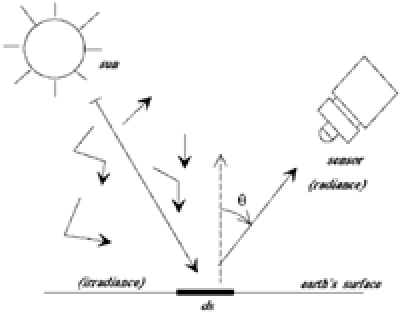


Fig. 4: Here we visualize how our full model benefits from representing view-dependent emitted radiance and from passing our input coordinates through a high-frequency positional encoding. Removing view dependence prevents the model from recreating the specular reflection on the bulldozer tread. Removing the positional encoding drastically decreases the model's ability to represent high frequency geometry and texture, resulting in an oversmoothed appearance.

Radiative Transfer Equations - how to describe ***<u>the variation of the radiance L per unit distance along ω</u>***?

The equation of radiative transfer simply says that ***<u>as a beam of radiation travels</u>***, it loses energy to absorption, gains energy by





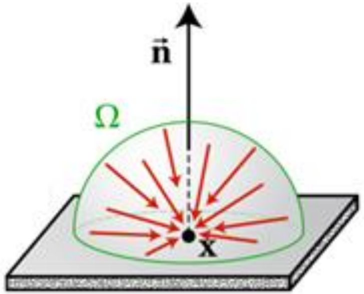
$$(\omega\cdot\nabla)L(x,\omega)=\underbrace{\sigma_s(x)\int_{\mathbb{S}^2} f_p(x,\omega_i\to\omega)L(x,\omega_i)\,\mathrm{d}\omega_i}_{\text{In-scattering}}\underbrace{-\sigma_t(x)L(x,\omega)}_{\text{Out-scattering \& absorption}}\underbrace{+Q(x,\omega)}_{\text{Emission}}$$

Scattering coefficient: $\sigma_s(x)\in\mathbb{R}_{\geq 0}$,
Phase function: $f_p(x,\omega_i\to\omega)$, a probability density over S^2 given x and ω_i

The ratio between σ_s and σ_t controls the fraction of radiant energy *not* being absorbed at each scattering and is also known as the *single-scattering albedo*

Extinction coefficient: $\sigma_t(x)\in\mathbb{R}_{\geq\sigma_s(x)}$
σ_t controls how frequently light scatters and is also known as the *optical density*

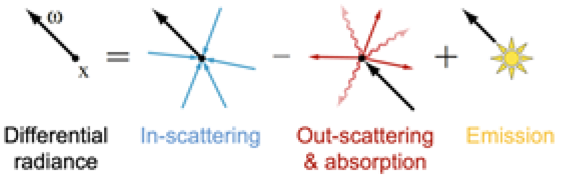
Source term: $Q(x,\omega)\in\mathbb{R}_{\geq 0}$

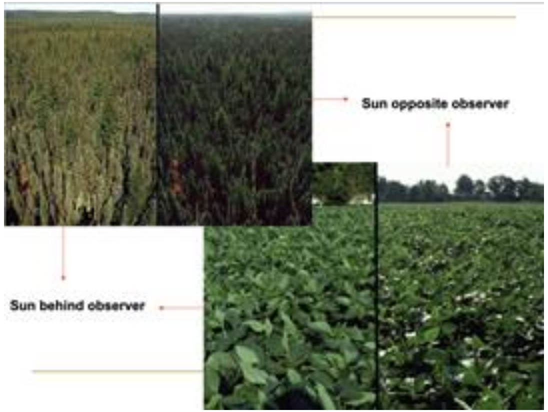


- Differential radiance

$$(\omega\cdot\nabla)L(x,\omega)=\left.\frac{\mathrm{d}L(x+\tau\omega,\omega)}{\mathrm{d}\tau}\right|_{\tau=0}=\lim_{\tau\to 0}\frac{L(x+\tau\omega,\omega)-L(x,\omega)}{\tau}$$

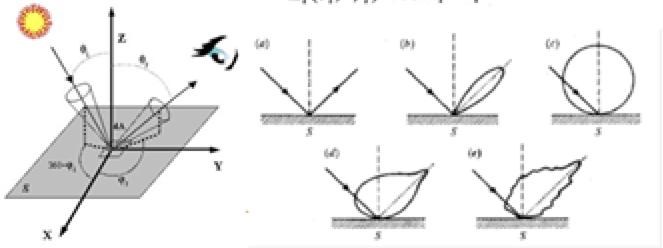
In-scattering - How to describe radiation directional properties? BRDF

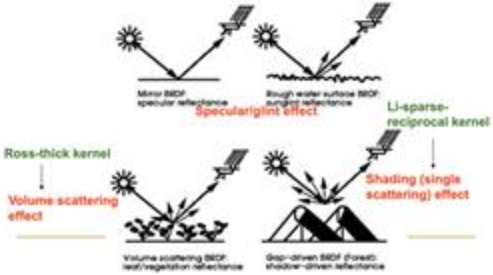




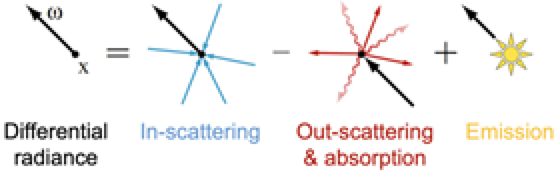
The Bi-directional Reflectance Distribution Function (BRDF) is used to describe the dependence of reflected radiation on the incident (i) and outgoing (v) directions (Nicodemus, 1977).

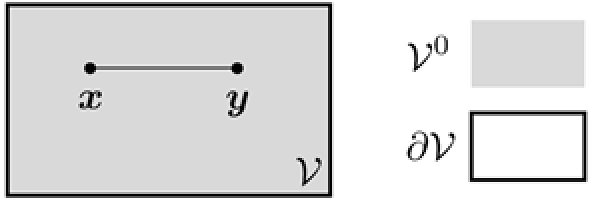
$$B(\theta_i, \phi_i, \theta_v, \phi_v) = \frac{dL_v(\theta_i, \phi_i, \theta_v, \phi_v)}{L_i(\theta_i, \phi_i) \cos \theta_i d\Omega_i} \text{ sr}^{-1}$$





Out-scattering & absorption - How to quantify ***attenuation***? Beer's Law





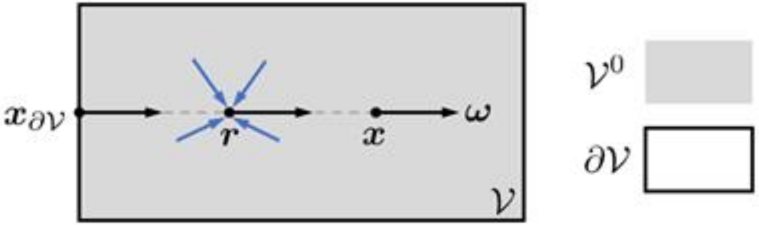
- For any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{V}$, the attenuation between ***x*** and ***y*** is

$$T(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) := \exp\left(-\int_{(\boldsymbol{x},\boldsymbol{y})} \sigma_t(\boldsymbol{r})\,\mathrm{d}\boldsymbol{r}\right)$$

  - A line integral between ***x*** and ***y***
  - $0 \leq T(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) \leq 1$ for all ***x*** and ***y***
  - For homogeneous media with $\sigma_t(\boldsymbol{x}) \equiv \sigma_t$,

$$T(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|\sigma_t)$$

Solving Radiative Transfer Equations - Derive Integral form of RTEs



$$L(\boldsymbol{x},\boldsymbol{\omega}) = \int_0^{h(\boldsymbol{x},\boldsymbol{\omega})} \overbrace{T(\boldsymbol{r}\leftrightarrow\boldsymbol{x})}^{\text{Attenuation}} \left[\overbrace{\sigma_s(\boldsymbol{r}) \int_{\mathbb{S}^2} f_p(\boldsymbol{r},\boldsymbol{\omega}_i \to \boldsymbol{\omega}) L(\boldsymbol{r},\boldsymbol{\omega}_i)\,\mathrm{d}\boldsymbol{\omega}_i}^{\text{In-scattering}} + \overbrace{Q(\boldsymbol{r},\boldsymbol{\omega})}^{\text{Emission}} \right] \mathrm{d}\tau$$

$$+ \underbrace{T(\boldsymbol{x}_{\partial\mathcal{V}}\leftrightarrow\boldsymbol{x})}_{\text{Attenuation}} \underbrace{L(\boldsymbol{x}_{\partial\mathcal{V}},\boldsymbol{\omega})}_{\text{Boundary cond.}} \quad \text{where } \boldsymbol{r} := \boldsymbol{x} - \tau\boldsymbol{\omega}$$

(The second term vanishes when $h(\boldsymbol{x},\boldsymbol{\omega}) = +\infty$)

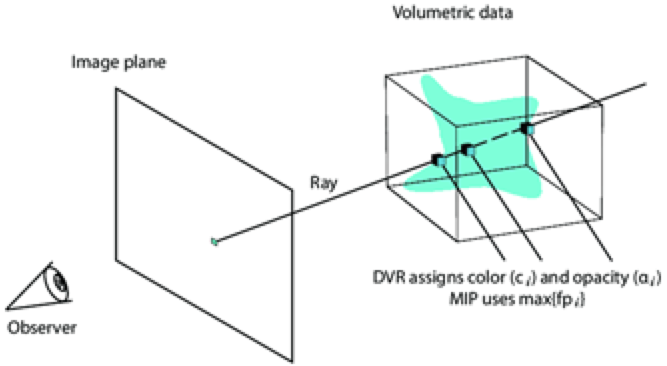
All RTMs follow this general form.

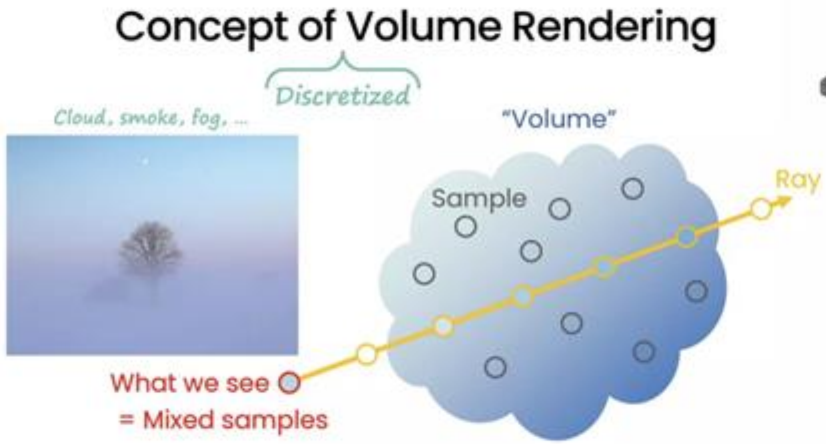
The differences however, are essentially due to the various forms for the emission and absorption coefficients.

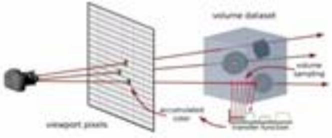
Rendering: 3D -> 2D

It does not consider the following factors:

1. The light source and its geometry
2. The in-scattering?
3. The bidirectional reflectance distribution function
4. The influence of the properties of the volume, e.g., biochemical properties





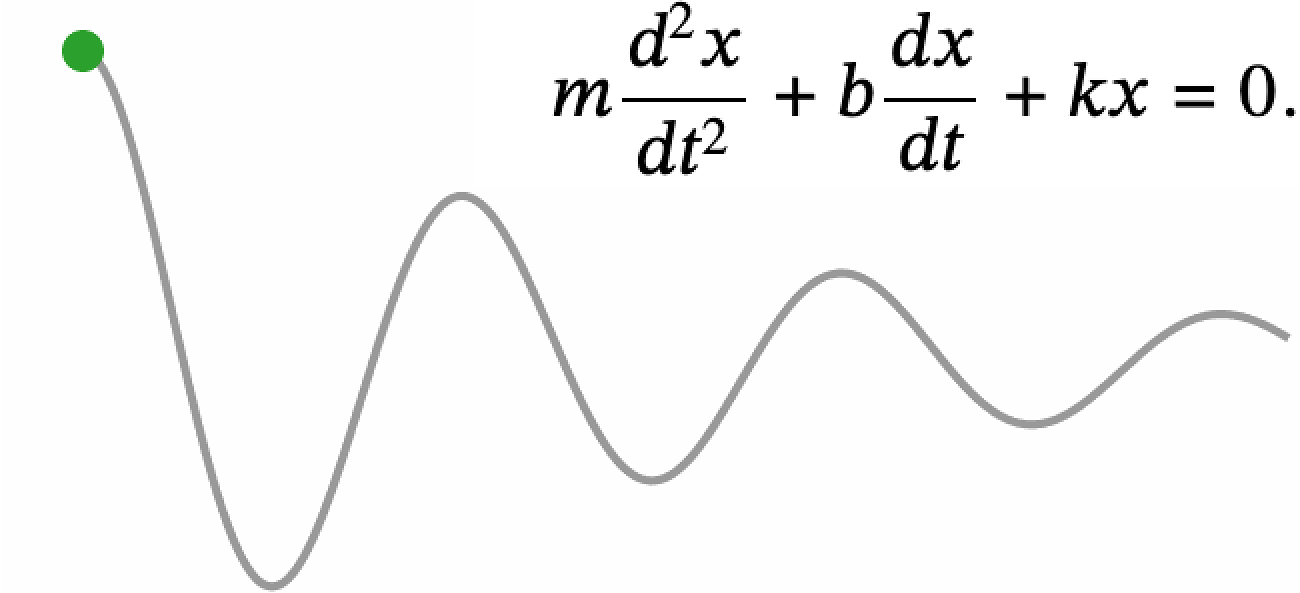


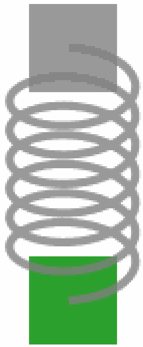
Improve Neural radiative field for Remote Sensing

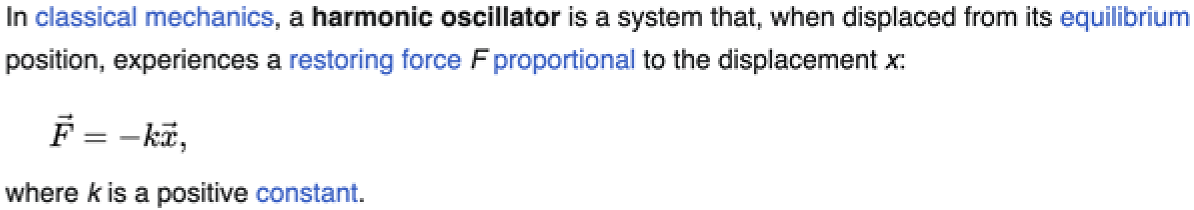
- Encoder should output properties of the media, not just density and color.
- Encoder & Decoder should address not only the camera geometry, but also light source geometry
- Decoder should use bidirectional reflectance distribution function to model geometry dependance
- Decoder should integrate the properties of the media, e.g., biochemical properties, into radiative transfer, via RTMs, such as Prosail?

DampedHarmonic oscillator - how to solve partial differential equation?

$$m\frac{d^2x}{dt^2}+b\frac{dx}{dt}+kx=0.$$





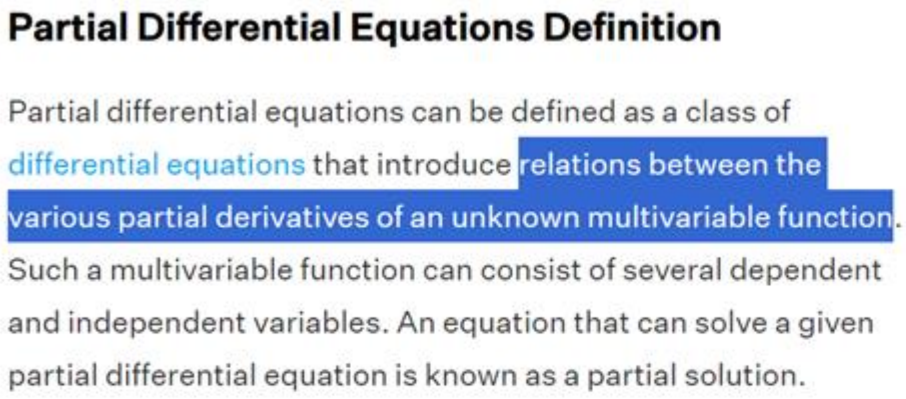


In classical mechanics, a **harmonic oscillator** is a system that, when displaced from its equilibrium position, experiences a restoring force *F* proportional to the displacement *x*:

$$\vec{F}=-k\vec{x},$$

where *k* is a positive constant.

For a mass on a spring oscillating in a viscous fluid, the period remains constant, but the amplitudes of the oscillations decrease due to the **damping caused by the fluid**.

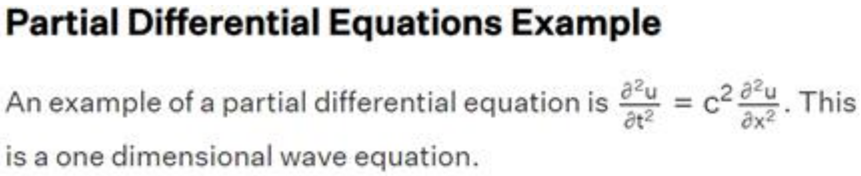


Partial Differential Equations Definition

Partial differential equations can be defined as a class of differential equations that introduce relations between the various partial derivatives of an unknown multivariable function. Such a multivariable function can consist of several dependent and independent variables. An equation that can solve a given partial differential equation is known as a partial solution.

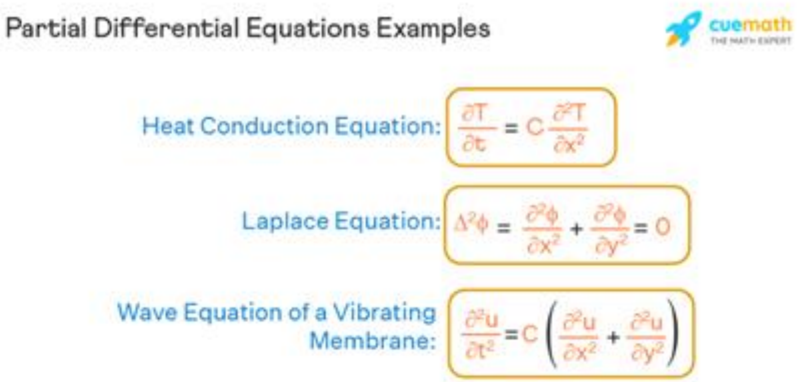
It is usually impossible to write down explicit formulae for solutions of partial differential equations. There is correspondingly a vast amount of modern mathematical and scientific research on methods to numerically approximate solutions of certain partial differential equations using computers.

PDEs are foundational in the modern scientific understanding of sound, heat, diffusion, electrostatics, electrodynamics, thermodynamics, fluid dynamics, elasticity, general relativity, and quantum mechanics (Schrödinger equation, Pauli equation etc.)



Partial Differential Equations Example

An example of a partial differential equation is $\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$. This is a one dimensional wave equation.



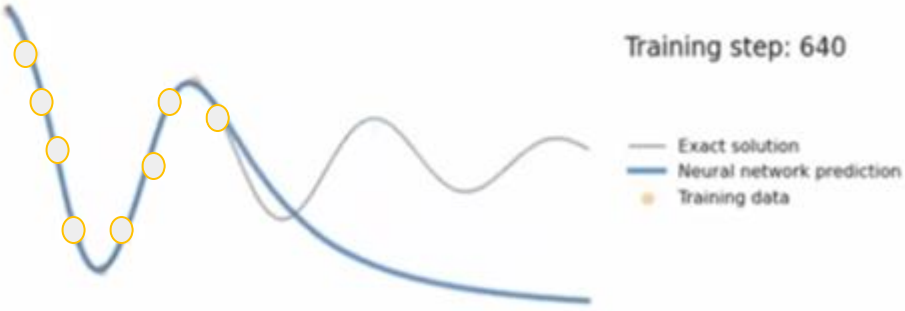
Partial Differential Equations Examples

Heat Conduction Equation: $\frac{\partial T}{\partial t} = c \frac{\partial^2 T}{\partial x^2}$

Laplace Equation: $\Delta^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$

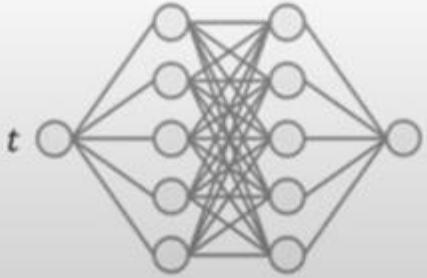
Wave Equation of a Vibrating Membrane: $\frac{\partial^2 u}{\partial t^2} = c\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$

Naïve neural network



Data-driven NN? Disadvantages?

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}\left(NN(t_i;\theta) - \underline{u_i}\right)^2$$

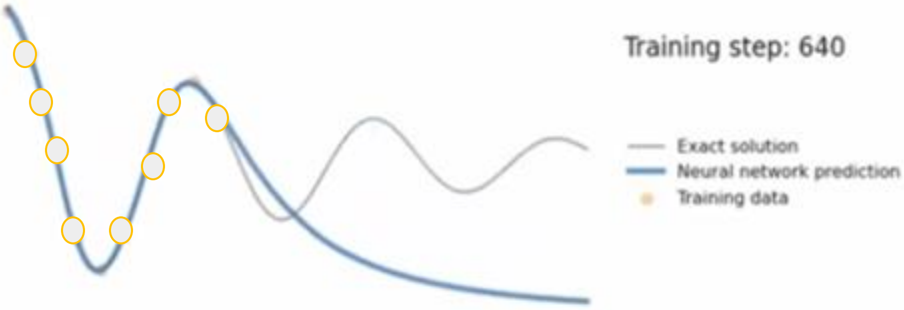


$NN(t_i;\theta) \approx u(t)$

Damped harmonic oscillator

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

Naïve neural network



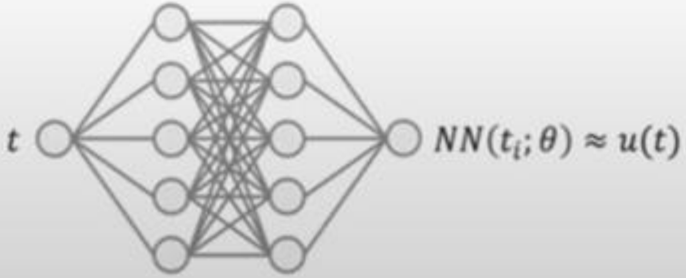
$$L(\theta) = \frac{1}{N}\sum_{i}^{N}(NN(t_i;\theta) - \underline{u_i})^2$$

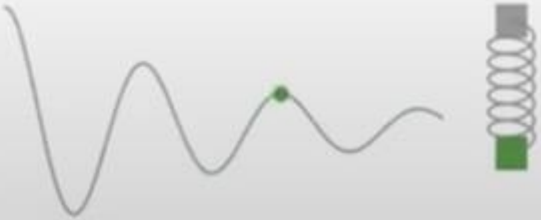
Data-driven NN? Disadvantages?

(1) Relies on training data, cannot extroloplate

(2) Sentitive to data quantify and quality, overfitting

(3) Not physics informated

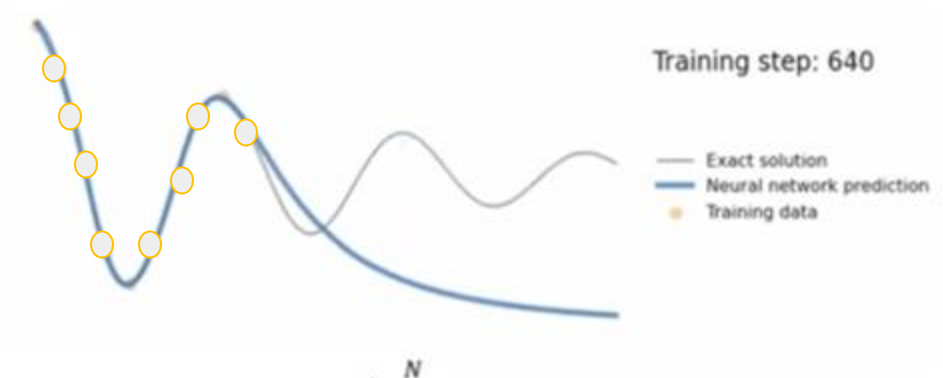




Damped harmonic oscillator

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

Naïve neural network



$$L(\theta) = \frac{1}{N}\sum_{i}^{N}\left(NN(t_i;\theta) - \underline{u_i}\right)^2$$

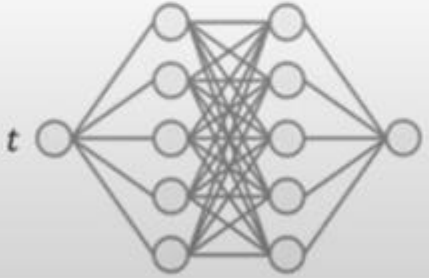
Data-driven NN? Disadvantages?

(1) Relies on training data, cannot extroloplate

(2) Sentitive to data quantify and quality, overfitting

(3) Not physics informated

How to incorporate partial differential equation into the neural network?

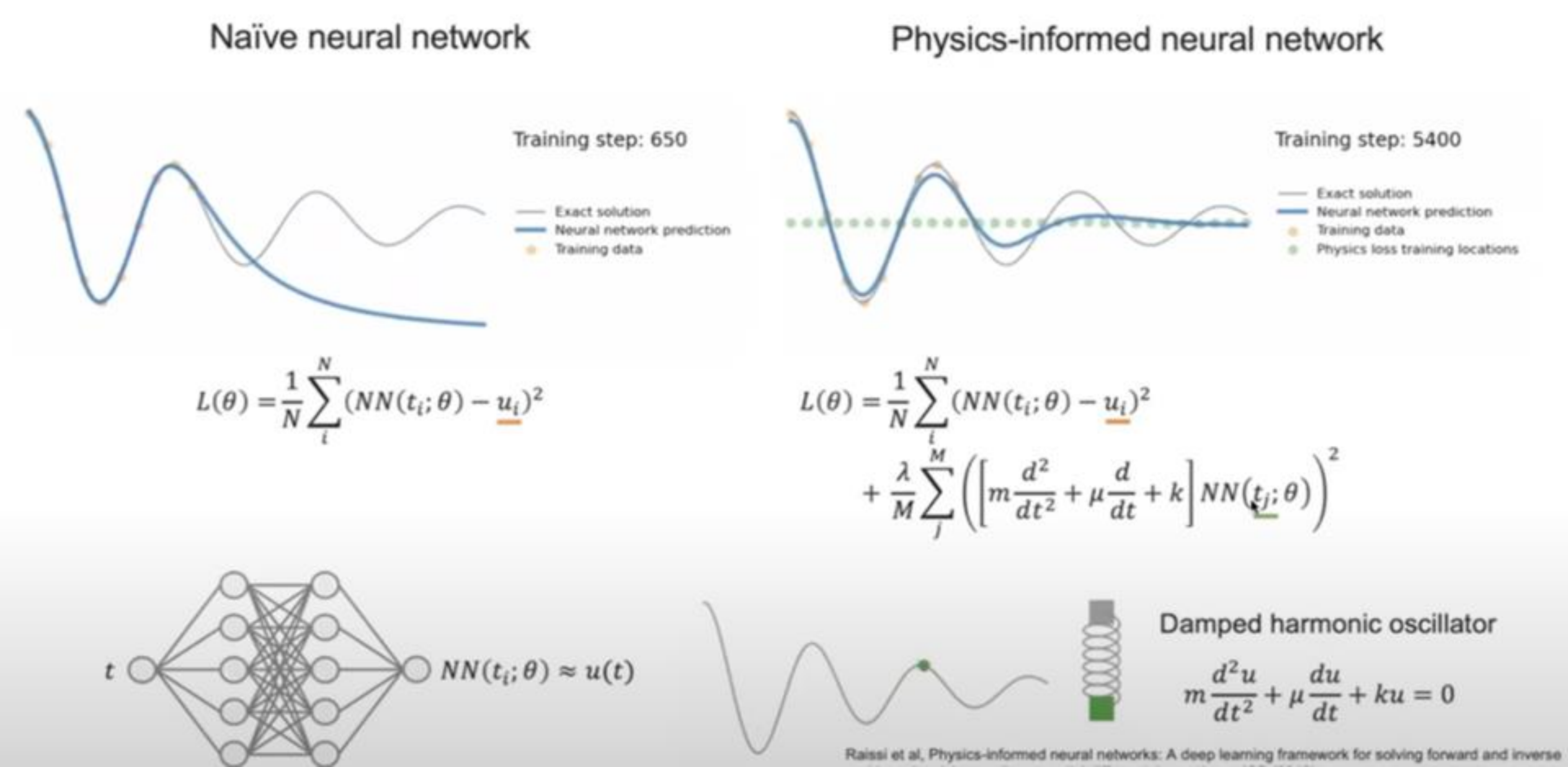


$$NN(t_i;\theta) \approx u(t)$$

Damped harmonic oscillator

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

How to train physics-informed NN? Benefits?



Naïve neural network

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}(NN(t_i;\theta) - \underline{u_i})^2$$

$$NN(t_i;\theta) \approx u(t)$$

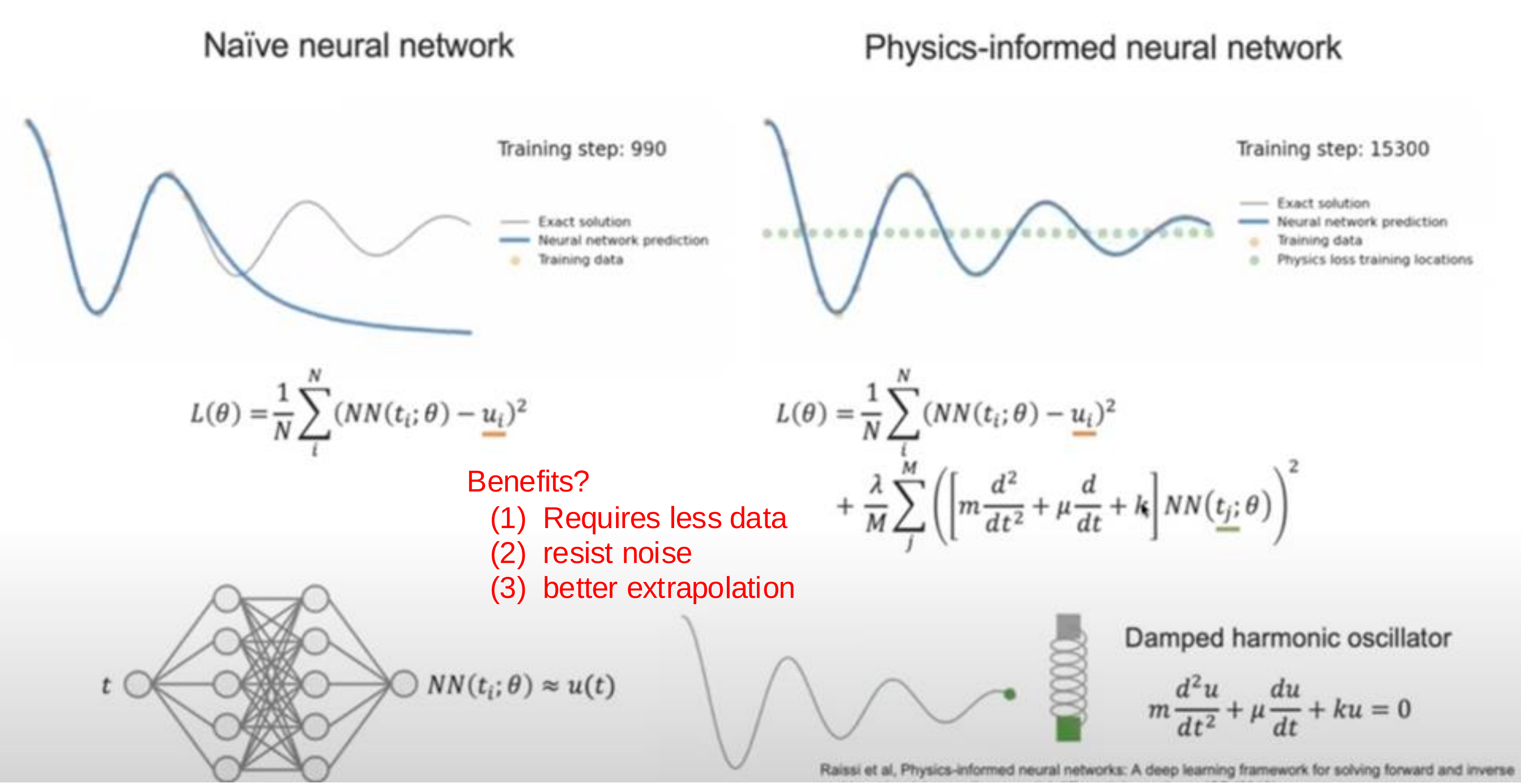
Physics-informed neural network

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}(NN(t_i;\theta) - \underline{u_i})^2 + \frac{\lambda}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]NN(\underline{t_j};\theta)\right)^2$$

Damped harmonic oscillator

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

Raissi et al, Physics-informed neural networks: A deep learning framework for solving forward and inverse



Naïve neural network

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}(NN(t_i;\theta) - \underline{u_i})^2$$

Physics-informed neural network

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}(NN(t_i;\theta) - \underline{u_i})^2$$

$$+ \frac{\lambda}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]NN(\underline{t_j};\theta)\right)^2$$

Benefits?

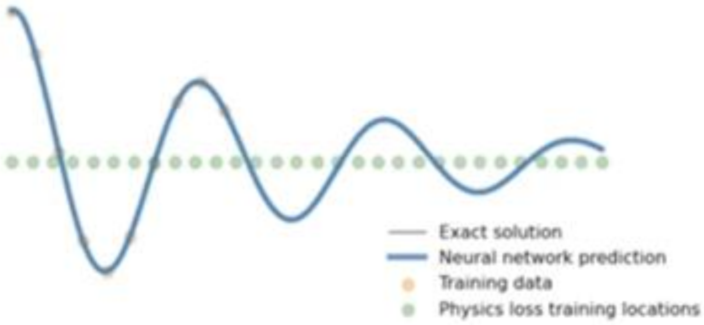
1. Requires less data
2. resist noise
3. better extrapolation

Damped harmonic oscillator

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

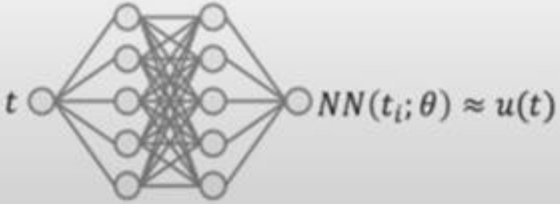
Raissi et al, Physics-informed neural networks: A deep learning framework for solving forward and inverse

Physics-informed neural network (PINN)



$$L(\theta) = \frac{1}{N}\sum_{i}^{N}\left(NN(t_i;\theta) - \underline{u_i}\right)^2$$ Supervised loss

$$+ \frac{\lambda}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]NN(\underline{t_j};\theta)\right)^2$$ Physics loss
aka PDE residual



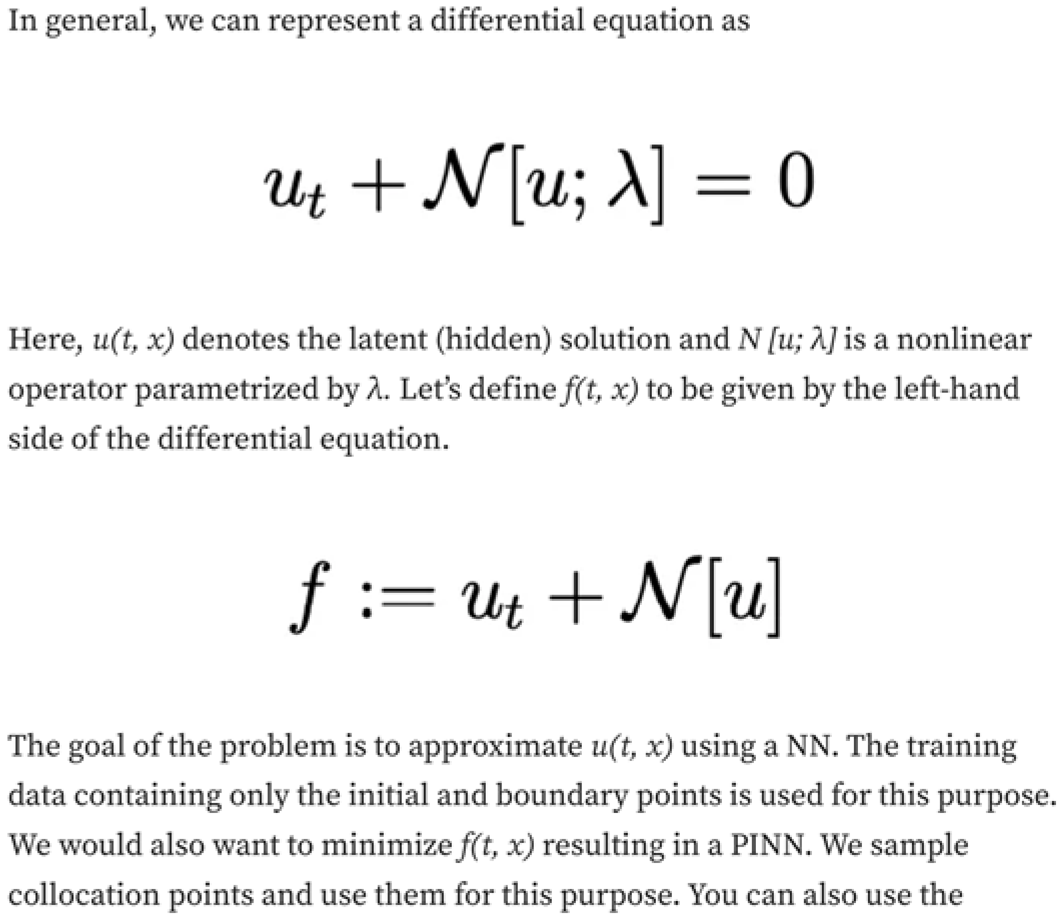
From a ML perspective:

- Physics loss is an **unsupervised** regulariser, which adds prior knowledge

From a mathematical perspective:

- PINNs provide a way to solve PDEs:
  - Neural network is a mesh-free, **functional** approximation of PDE solution
  - Physics loss is used to assert solution is **consistent** with PDE
  - Supervised loss is used to assert boundary/initial conditions, to ensure solution is **unique**

Using NN to incorporate any PD equations?



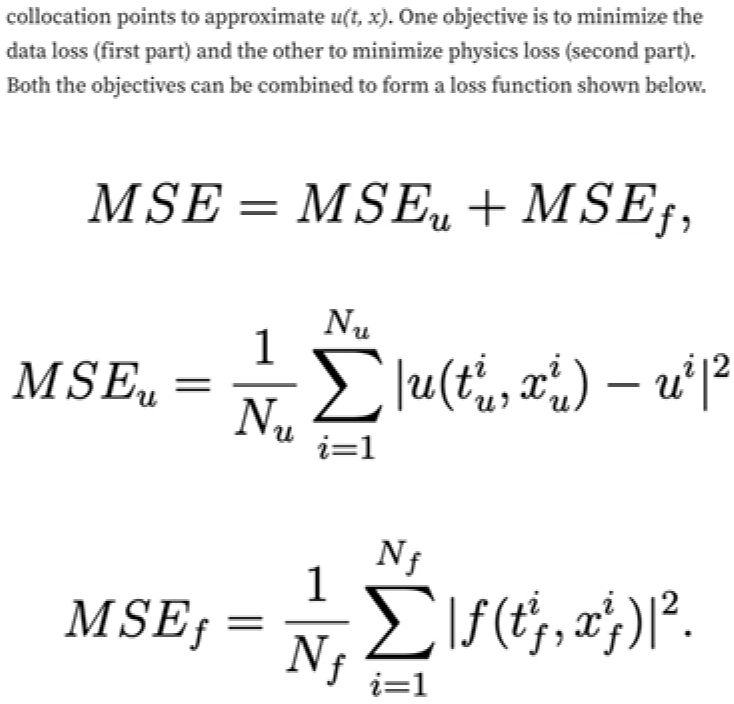
In general, we can represent a differential equation as

$$u_t + \mathcal{N}[u; \lambda] = 0$$

Here, *u(t, x)* denotes the latent (hidden) solution and *N [u; λ]* is a nonlinear operator parametrized by λ. Let's define *f(t, x)* to be given by the left-hand side of the differential equation.

$$f := u_t + \mathcal{N}[u]$$

The goal of the problem is to approximate *u(t, x)* using a NN. The training data containing only the initial and boundary points is used for this purpose. We would also want to minimize *f(t, x)* resulting in a PINN. We sample collocation points and use them for this purpose. You can also use the



collocation points to approximate *u(t, x)*. One objective is to minimize the data loss (first part) and the other to minimize physics loss (second part). Both the objectives can be combined to form a loss function shown below.

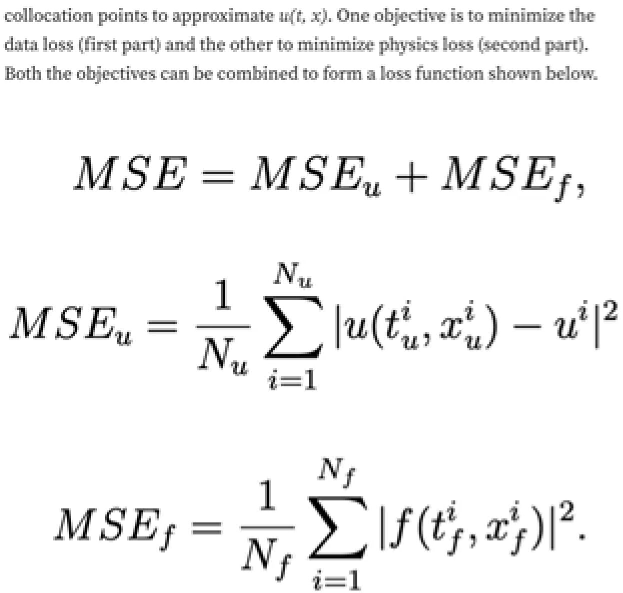
$$MSE = MSE_u + MSE_f,$$

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations." *arXiv preprint arXiv:1711.10561* (2017).

Using NN to solve Burgers' equation?

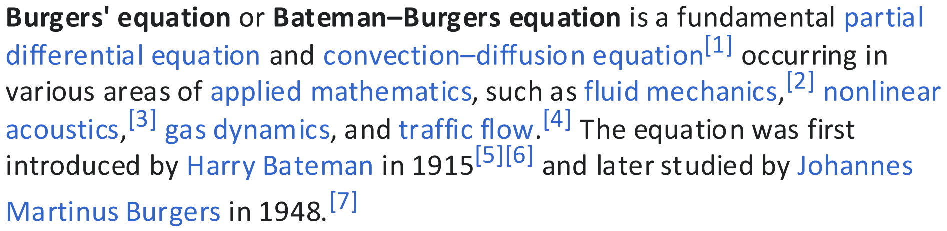


collocation points to approximate *u(t, x)*. One objective is to minimize the data loss (first part) and the other to minimize physics loss (second part). Both the objectives can be combined to form a loss function shown below.

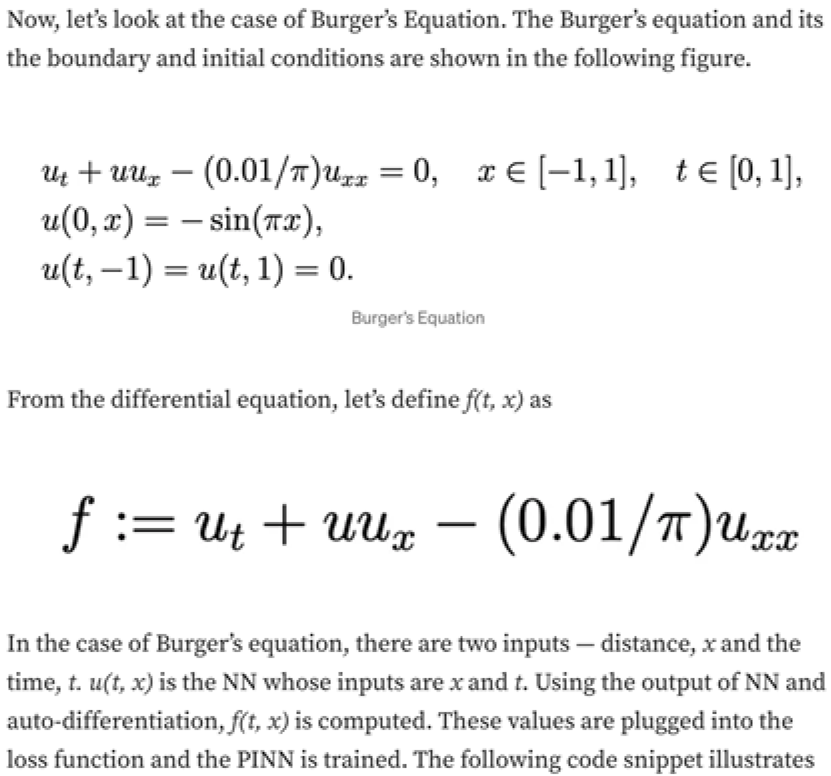
$$MSE = MSE_u + MSE_f,$$

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$



Burgers' equation or **Bateman–Burgers equation** is a fundamental partial differential equation and convection–diffusion equation[1] occurring in various areas of applied mathematics, such as fluid mechanics,[2] nonlinear acoustics,[3] gas dynamics, and traffic flow.[4] The equation was first introduced by Harry Bateman in 1915[5][6] and later studied by Johannes Martinus Burgers in 1948.[7]



Now, let's look at the case of Burger's Equation. The Burger's equation and its the boundary and initial conditions are shown in the following figure.

$$\begin{aligned} &u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1,1], \quad t \in [0,1],\\ &u(0,x) = -\sin(\pi x),\\ &u(t,-1) = u(t,1) = 0. \end{aligned}$$

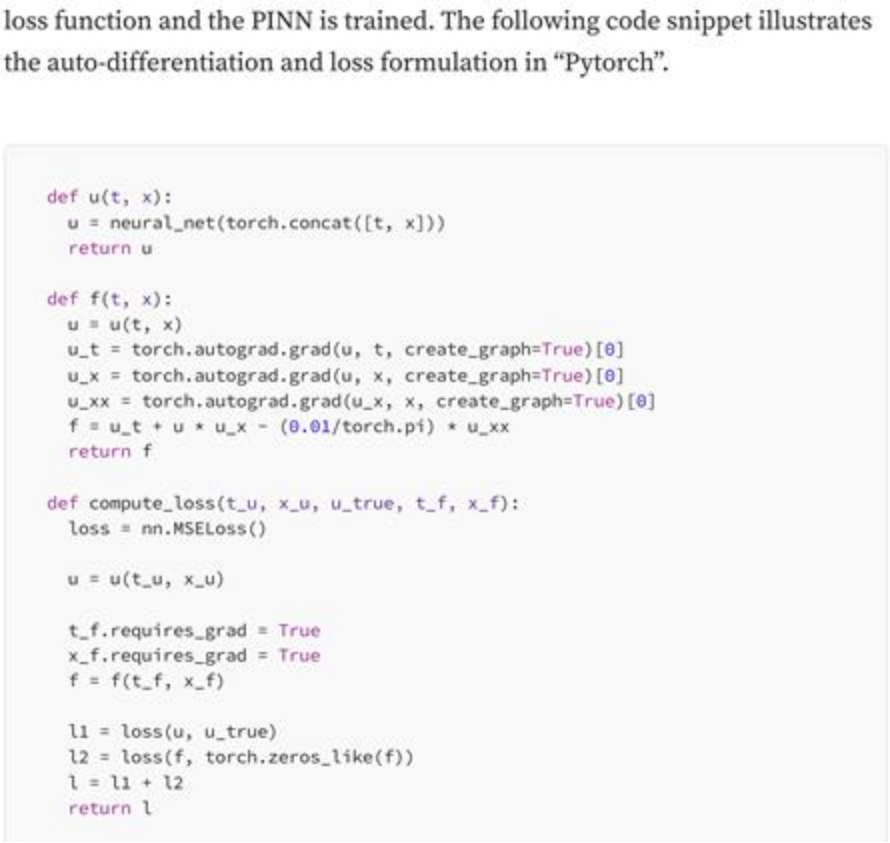
Burger's Equation

From the differential equation, let's define *f(t, x)* as

$$f := u_t + uu_x - (0.01/\pi)u_{xx}$$

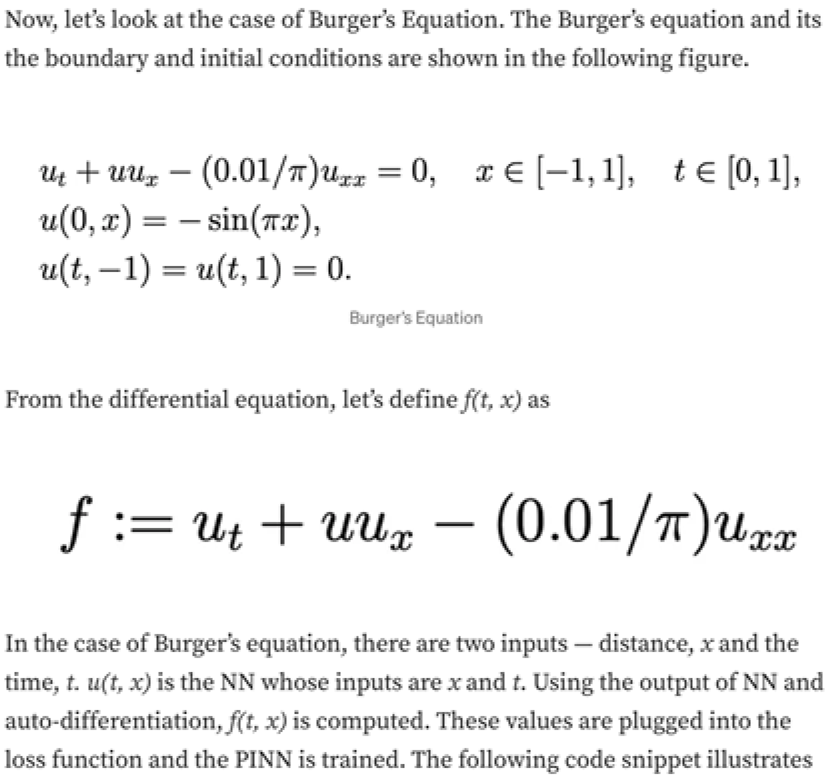
In the case of Burger's equation, there are two inputs — distance, *x* and the time, *t*. *u(t, x)* is the NN whose inputs are *x* and *t*. Using the output of NN and auto-differentiation, *f(t, x)* is computed. These values are plugged into the loss function and the PINN is trained. The following code snippet illustrates

How to implement it in pytorch? Auto-differentiation make things easier



loss function and the PINN is trained. The following code snippet illustrates the auto-differentiation and loss formulation in “Pytorch”.

```
def u(t, x):
  u = neural_net(torch.concat([t, x]))
  return u

def f(t, x):
  u = u(t, x)
  u_t = torch.autograd.grad(u, t, create_graph=True)[0]
  u_x = torch.autograd.grad(u, x, create_graph=True)[0]
  u_xx = torch.autograd.grad(u_x, x, create_graph=True)[0]
  f = u_t + u * u_x - (0.01/torch.pi) * u_xx
  return f

def compute_loss(t_u, x_u, u_true, t_f, x_f):
  loss = nn.MSELoss()

  u = u(t_u, x_u)

  t_f.requires_grad = True
  x_f.requires_grad = True
  f = f(t_f, x_f)

  l1 = loss(u, u_true)
  l2 = loss(f, torch.zeros_like(f))
  l = l1 + l2
  return l
```



Now, let’s look at the case of Burger’s Equation. The Burger’s equation and its the boundary and initial conditions are shown in the following figure.

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

Burger’s Equation

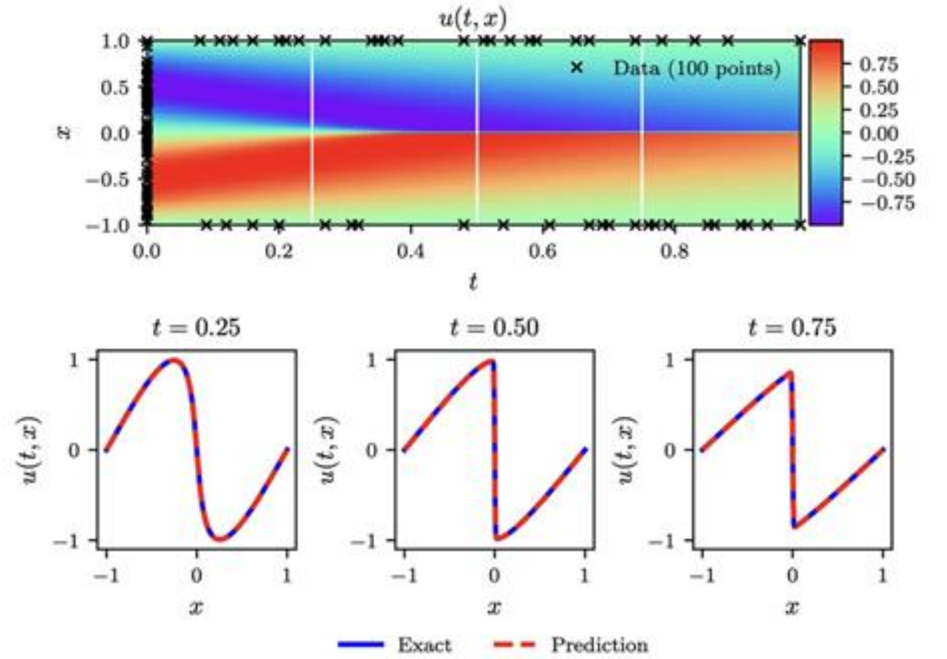
From the differential equation, let’s define *f(t, x)* as

$$f := u_t + uu_x - (0.01/\pi)u_{xx}$$

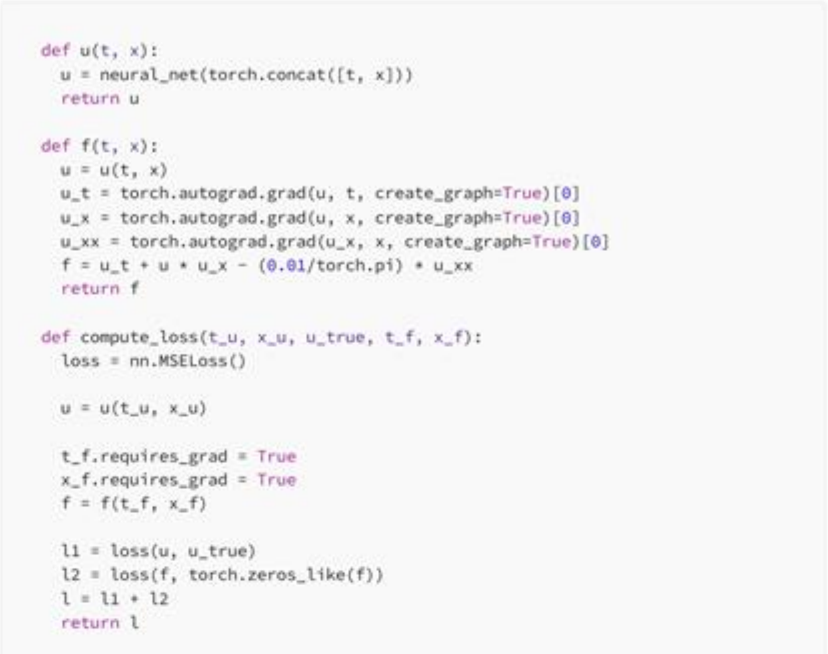
In the case of Burger’s equation, there are two inputs — distance, *x* and the time, *t*. *u(t, x)* is the NN whose inputs are *x* and *t*. Using the output of NN and auto-differentiation, *f(t, x)* is computed. These values are plugged into the loss function and the PINN is trained. The following code snippet illustrates

The authors in the paper have used a nine-layered NN containing 20 neurons each with a hyperbolic tangent (tanh) activation function. They used 10,000 collocation points generated using a Latin Hypercube Sampling strategy and 100 initial/boundary points. The image below illustrates the results. On the top, the output u(t, x) from the trained NN is shown. On the bottom, comparison of the predicted and exact solutions corresponding to the three temporal snapshots are depicted. The model obtained a L2-error of 6.7e-4 on the test set.

How many training samples, where? Why sparse samples lead to good exptropolations?

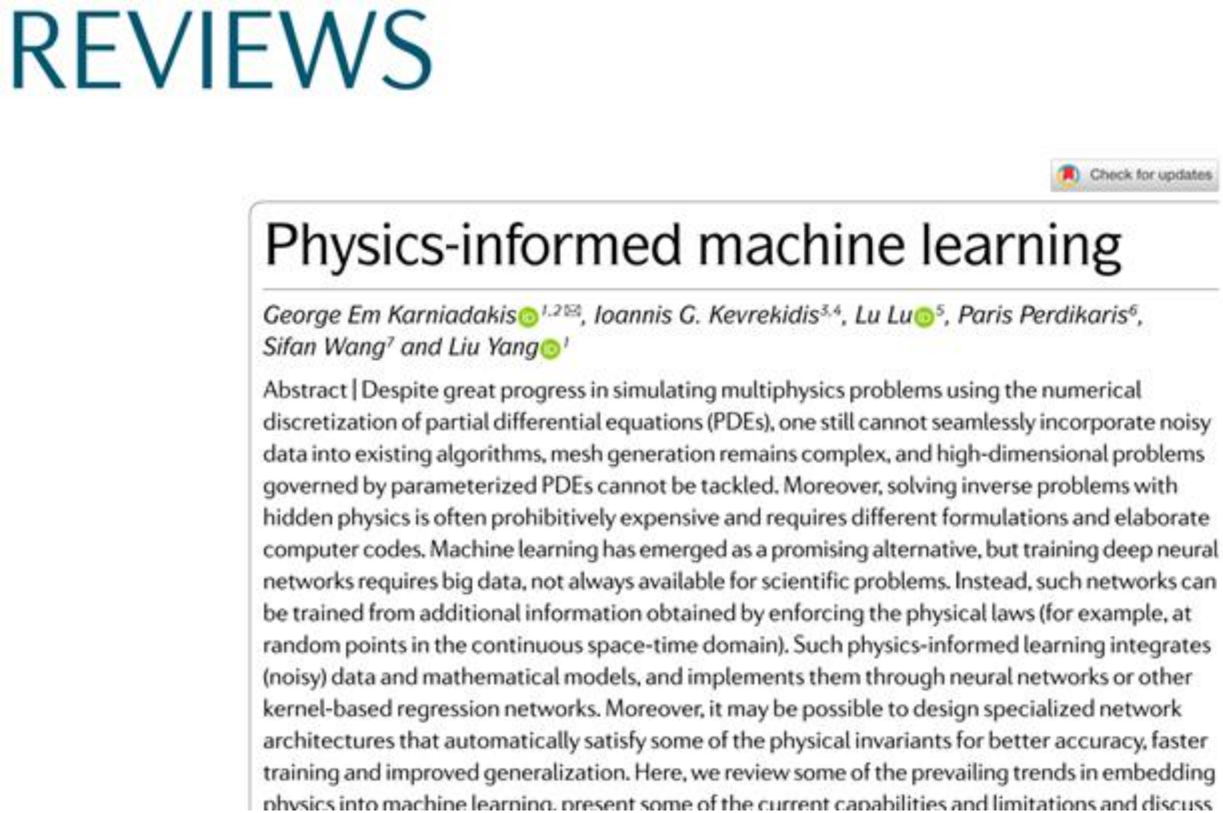


loss function and the PINN is trained. The following code snippet illustrates the auto-differentiation and loss formulation in "Pytorch".



```
def u(t, x):
  u = neural_net(torch.concat([t, x]))
  return u

def f(t, x):
  u = u(t, x)
  u_t = torch.autograd.grad(u, t, create_graph=True)[0]
  u_x = torch.autograd.grad(u, x, create_graph=True)[0]
  u_xx = torch.autograd.grad(u_x, x, create_graph=True)[0]
  f = u_t + u * u_x - (0.01/torch.pi) * u_xx
  return f

def compute_loss(t_u, x_u, u_true, t_f, x_f):
  loss = nn.MSELoss()

  u = u(t_u, x_u)

  t_f.requires_grad = True
  x_f.requires_grad = True
  f = f(t_f, x_f)

  l1 = loss(u, u_true)
  l2 = loss(f, torch.zeros_like(f))
  l = l1 + l2
  return l
```

Using only a handful of initial and boundary data, the physics informed neural network can accurately capture the intricate non-linear behaviour of the Burgers' equation that leads to the development of a sharp internal layer around t = 0.4. The latter is notoriously hard to accurately resolve with classical numerical methods and requires a laborious spatio-temporal discretization of equation.



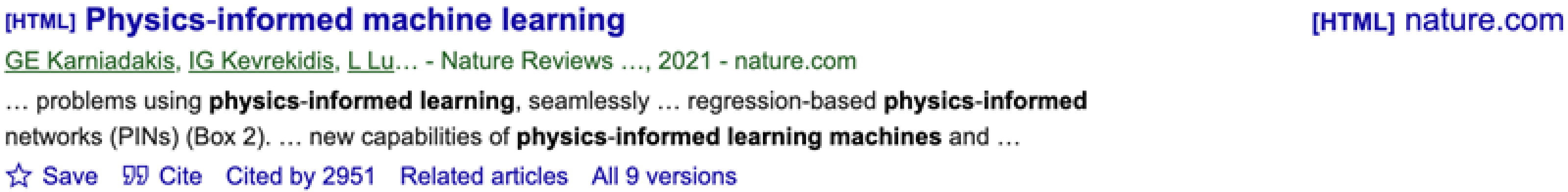
REVIEWS

Check for updates

Physics-informed machine learning

George Em Karniadakis[1,2], Ioannis G. Kevrekidis[3,4], Lu Lu[5], Paris Perdikaris[6], Sifan Wang[7] and Liu Yang[1]

Abstract | Despite great progress in simulating multiphysics problems using the numerical discretization of partial differential equations (PDEs), one still cannot seamlessly incorporate noisy data into existing algorithms, mesh generation remains complex, and high-dimensional problems governed by parameterized PDEs cannot be tackled. Moreover, solving inverse problems with hidden physics is often prohibitively expensive and requires different formulations and elaborate computer codes. Machine learning has emerged as a promising alternative, but training deep neural networks requires big data, not always available for scientific problems. Instead, such networks can be trained from additional information obtained by enforcing the physical laws (for example, at random points in the continuous space-time domain). Such physics-informed learning integrates (noisy) data and mathematical models, and implements them through neural networks or other kernel-based regression networks. Moreover, it may be possible to design specialized network architectures that automatically satisfy some of the physical invariants for better accuracy, faster training and improved generalization. Here, we review some of the prevailing trends in embedding physics into machine learning, present some of the current capabilities and limitations and discuss



[HTML] Physics-informed machine learning

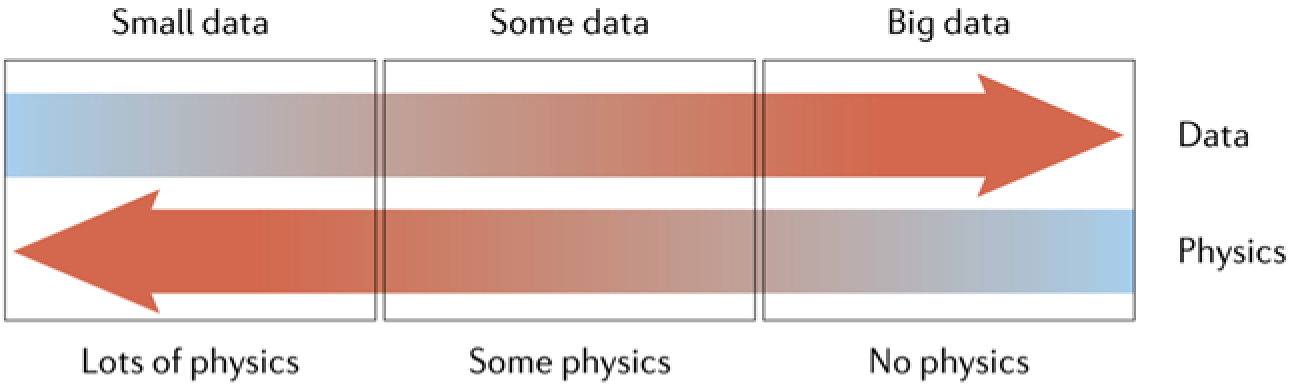
GE Karniadakis, IG Kevrekidis, L Lu… - Nature Reviews …, 2021 - nature.com

… problems using **physics-informed learning**, seamlessly … regression-based **physics-informed** networks (PINs) (Box 2). … new capabilities of **physics-informed learning machines** and …

☆ Save 99 Cite Cited by 2951 Related articles All 9 versions

[HTML] nature.com

Three possible categories of physical problems and associated available data



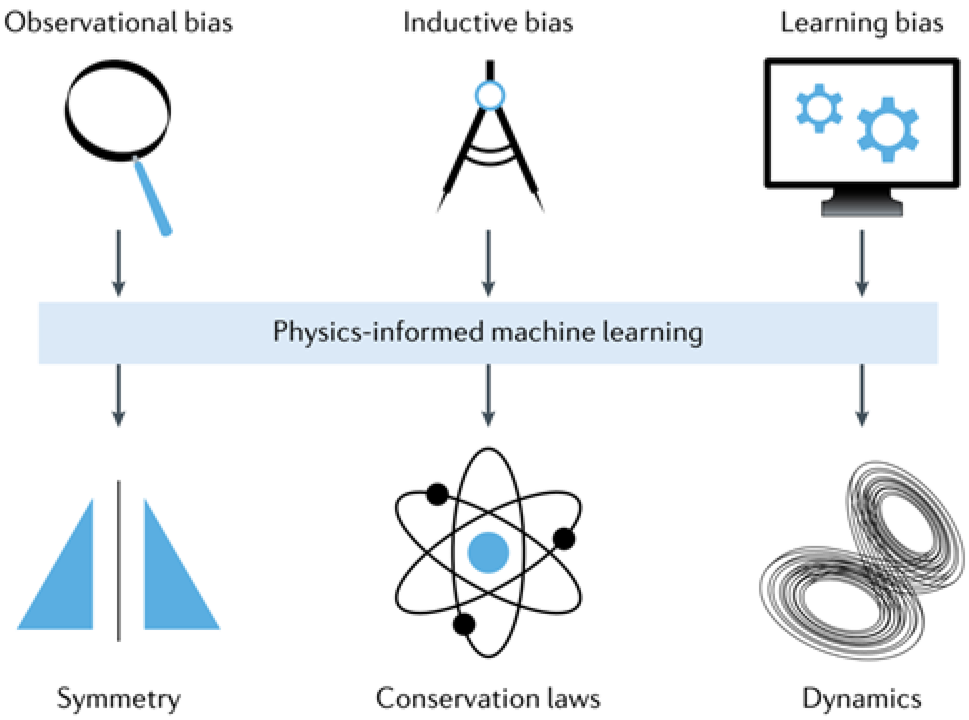
In the small data regime, it is assumed that **one knows all the physics**, and data are provided for the *initial and boundary conditions as well as the coefficients of a partial differential equation*.

The ubiquitous regime in applications is the middle one, where **one knows some data and some physics**, possibly *missing some parameter values or even an entire term in the partial differential equation*, for example, reactions in an advection–diffusion–reaction system.

Finally, there is the regime with **big data, where one may not know any of the physics,** and where a data-driven approach may be most effective, for example, using operator regression methods to *discover new physics*.

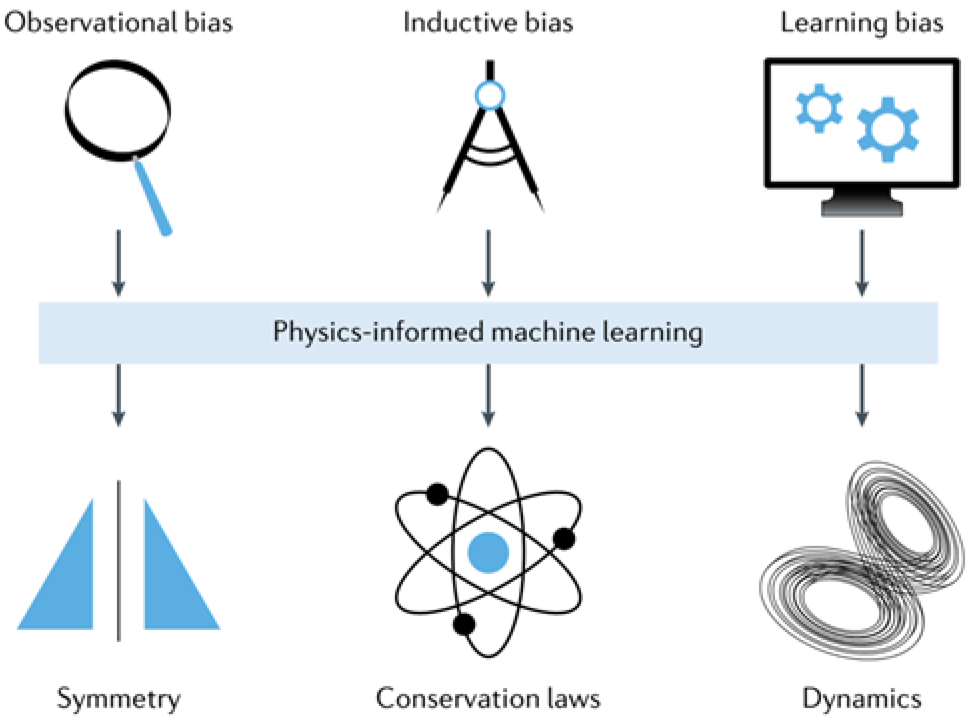
Physics-informed machine learning can seamlessly integrate data and the governing physical laws, including models with partially missing physics, in a unified way. This can be expressed compactly *using automatic differentiation and neural networks* that are designed to produce predictions that respect the underlying physical principles.

What is a physics-informed solution?



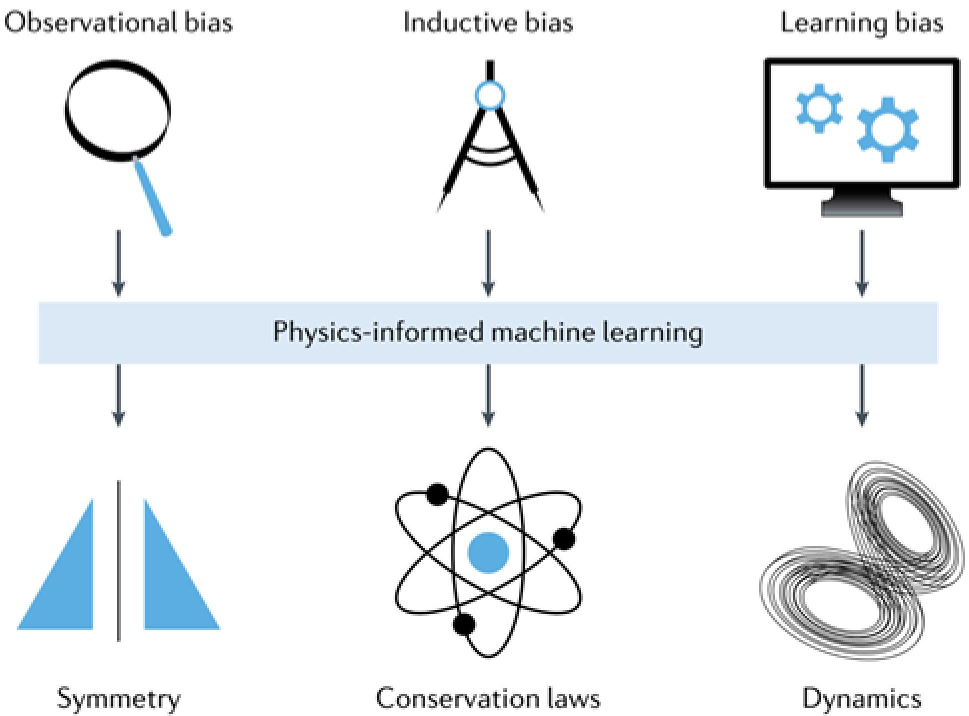
Making a learning algorithm physics-informed amounts to introducing appropriate observational, inductive or learning biases that can steer the learning process towards identifying physically consistent solutions

Observational bias?



Observational biases can be introduced directly through data that embody the underlying physics or carefully crafted data augmentation procedures. Training a machine learning (ML) system on such data allows it to learn functions, vector fields and operators that reflect the physical structure of the data.

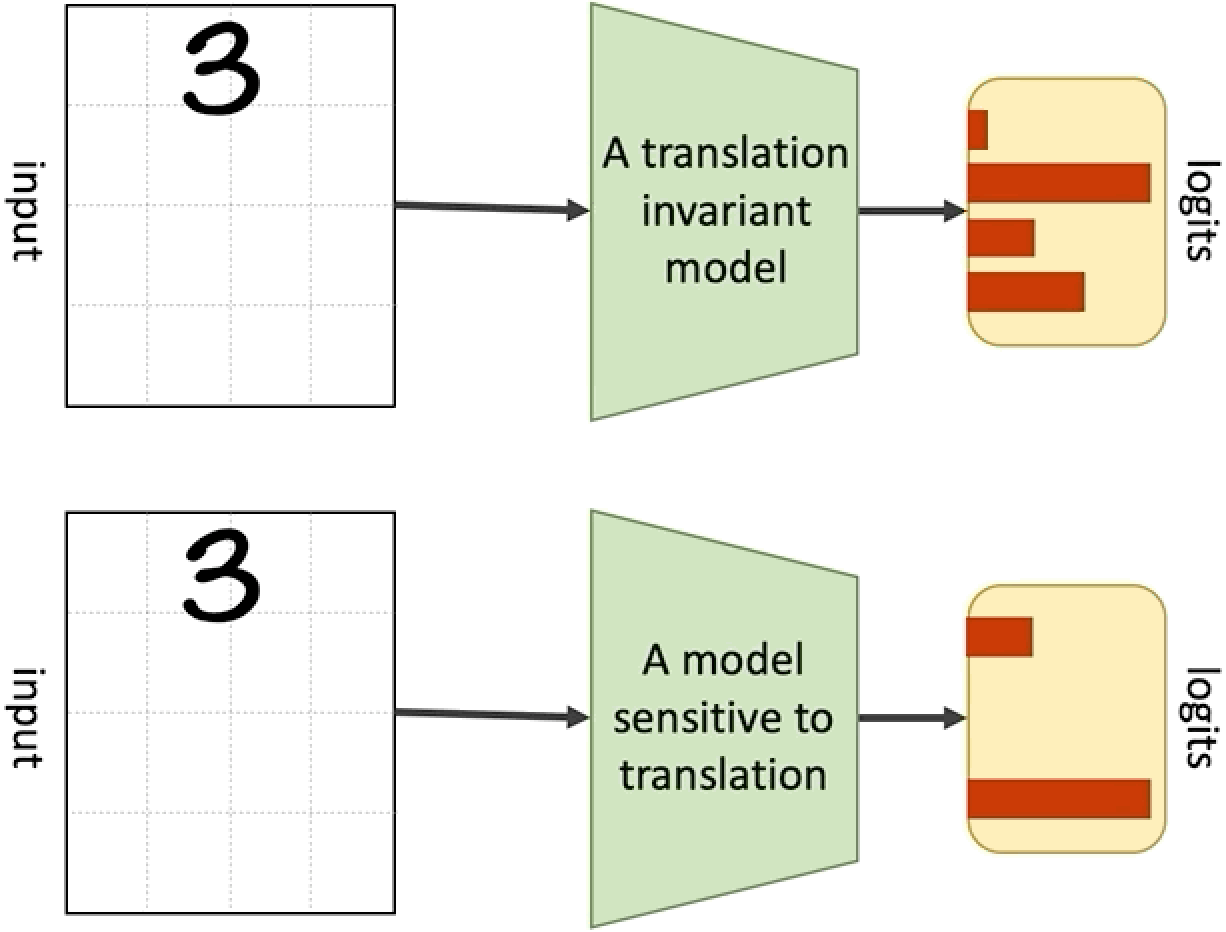
Inductive bias?



Conservation law, in physics, a principle that states that a certain physical property (i.e., a measurable quantity) does not change in the course of time within an isolated physical system.

Inductive biases correspond to prior assumptions that can be incorporated by tailored interventions to an ML model architecture, such that the predictions sought are guaranteed to implicitly satisfy a set of given physical laws, typically expressed in the form of certain mathematical constraints. One would argue that this is the most principled way of making a learning algorithm physics-informed, as it allows for the underlying physical constraints to be strictly satisfied. However, such approaches can be limited to accounting for relatively simple symmetry groups (CNN?) (such as translations, permutations, reflections, rotations and so on) that are known a priori, and may often lead to complex implementations that are difficult to scale.

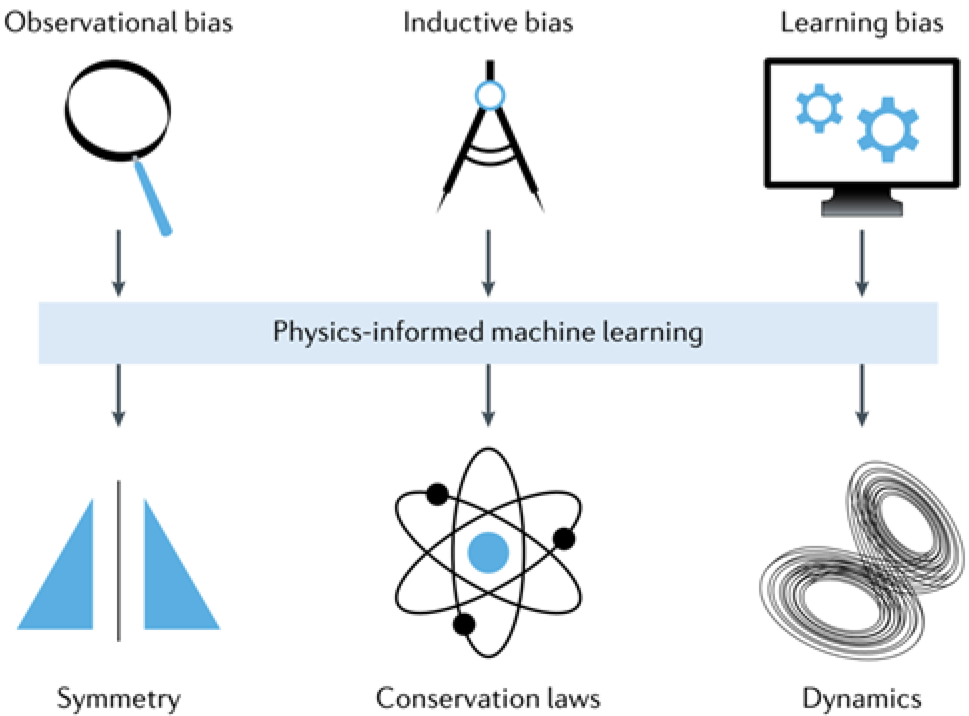
Does Neural radiative field fall in this category?


input
3
A translation invariant model
logits
input
3
A model sensitive to translation
logits

A convolutional neural network (CNN) has a structural bias for **translational invariance**, meaning it is designed to recognize patterns that are invariant to shifts in position

Another CNN inductive bias is locality i.e the features are generated using local pixels and then combined heirarchically.

Learning bias?



Learning biases can be introduced by appropriate choice of loss functions, constraints and inference algorithms that can modulate the training phase of an ML model to explicitly favour convergence towards solutions that adhere to the underlying physics.

By using and tuning such soft penalty constraints, the underlying physical laws can only be approximately satisfied; however, this provides a very flexible platform for introducing a broad class of physics-based biases that can be expressed in the form of integral, differential or even fractional equations.

Does the example with the Damped Harmonic oscillator falls into this category?

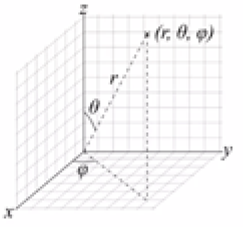
Learning bias vs. Inductive bias

(1) Soft constraint vs. Hard constraint

(2) The weight of physics-constraint is determined by a weight parameter.

(3) Multitask-learning, supervised data-driven loss + unsupervised physics-driven loss

Neural radiative field: Which category?



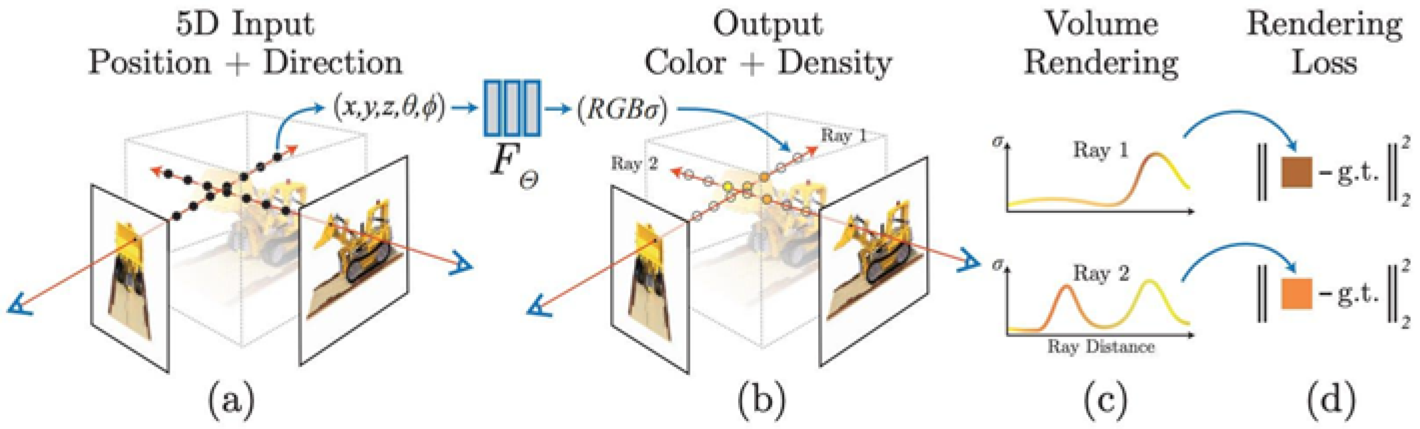
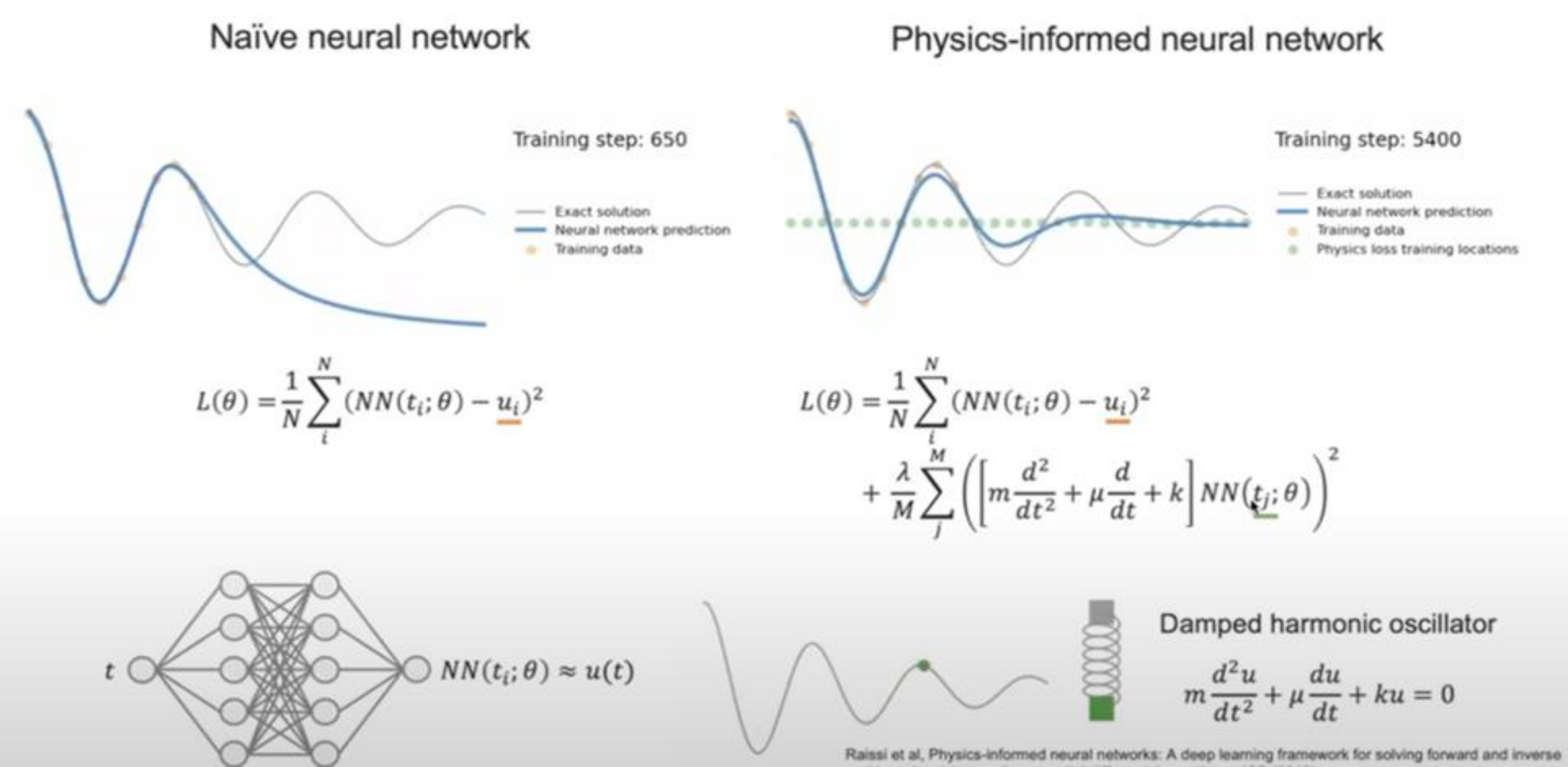


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

Physics-informed NN: Which category?



Naïve neural network

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}\left(NN(t_i;\theta) - \underline{u_i}\right)^2$$

$$NN(t_i;\theta) \approx u(t)$$

Physics-informed neural network

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}\left(NN(t_i;\theta) - \underline{u_i}\right)^2 + \frac{\lambda}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]NN(\underline{t_j};\theta)\right)^2$$

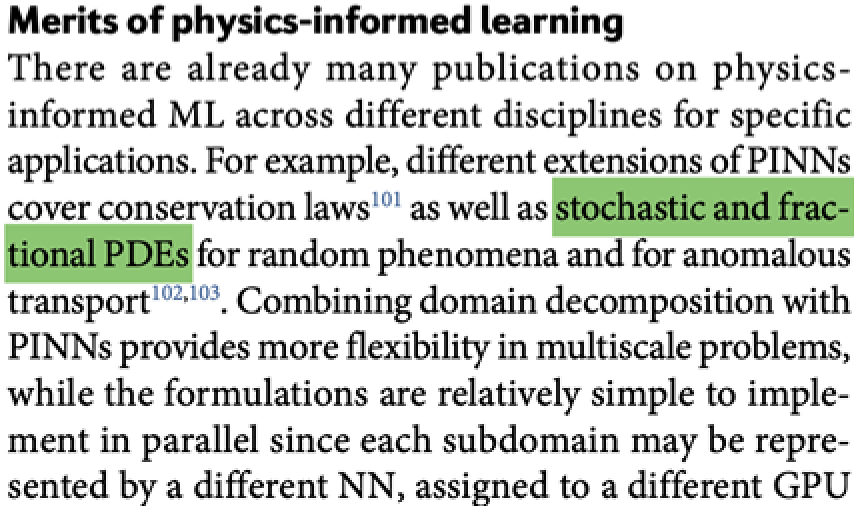
Damped harmonic oscillator

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

Raissi et al, Physics-informed neural networks: A deep learning framework for solving forward and inverse

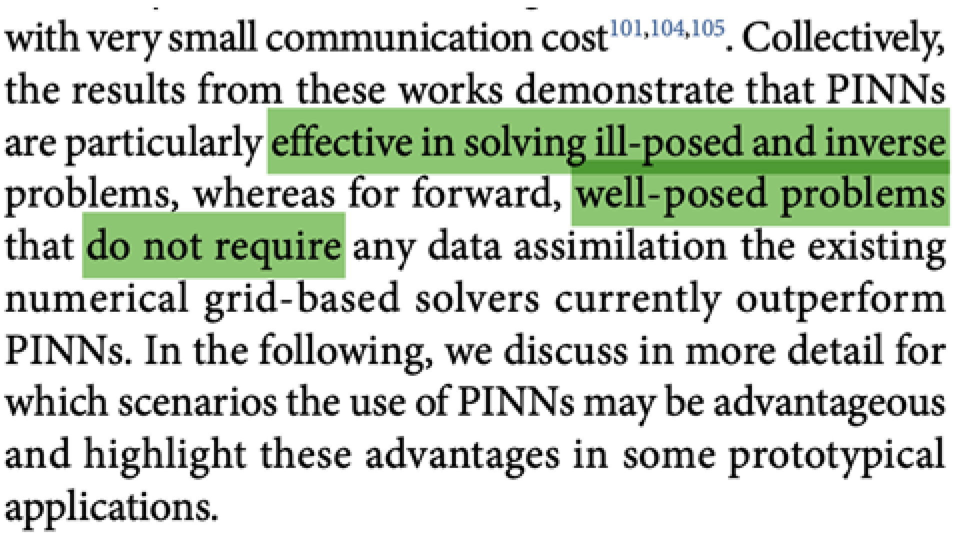
	(1) Direct inversion	(2) LUT approach	(3) Numerical Approach	(4) Simulation & ML : Which category?	(5) ML	**(6) DL**
f(.) is known	yes	yes	yes	yes	yes	**yes**
f(.) is partially known, i.e., form known, but with some unknown parameters U	no	no	Yes, estimate X and U together	no	no	**no**
f(.) unknown, (X,Y) known	no	no	no	no	yes	**yes**
f(.) unknown, (X,Y) unknown	no	no	no	no	no	**no**
If both f(.) and (X,Y) known, can accommodate both?	no	yes?	Yes? Use (X,Y) to estimate parameters in f(.)	yes?	Yes, use both simulated and observed data	**Yes, use both simulated and observed data**
Can use prior information? e.g., spatial prior and value prior	no	Yes? Use value prior for sampling	Yes? Use value prior of X in Bayesian estimation	Yes, Use value prior in sampling and spatial prior in Random fields	Yes, spatial prior in Random field approaches	**Yes, similar to ML**
Advantages	Knowledge-driven; Simple, easy	Knowledge-driven; Intuitive, easy, discrete fitting;	Knowledge-driven; estimate U; Efficient for simple f(.) in convex problems	Knowledge-driven; flexible; continuous fitting; good inter/extrapolation; faster than LUT	Data-driven; flexible; Classic;	**Strong modeling capability; automatic feature learning;**
Disadvantages	Unrealistic; rely on simple f(.)	Sensitive to accuracy of f(.), similarity metrics, sampling density and range; slow if LUT is large; bad for extrapolation;	Rely on efficiency of nonlinear solver; Slow; Local optimum;	Overfitting and underfitting risk to simulated data; difficult model selection; Sensitive to accuracy of f(.), similarity metrics, sampling density and range;	Weak modeling capability; Rely on "good" engineered features; Black-box; Overfitting, underfitting; Feature and model selection is difficult and slow	**Overfitting and underfitting; Black-box;**

Advantages of physics-informated machine learning



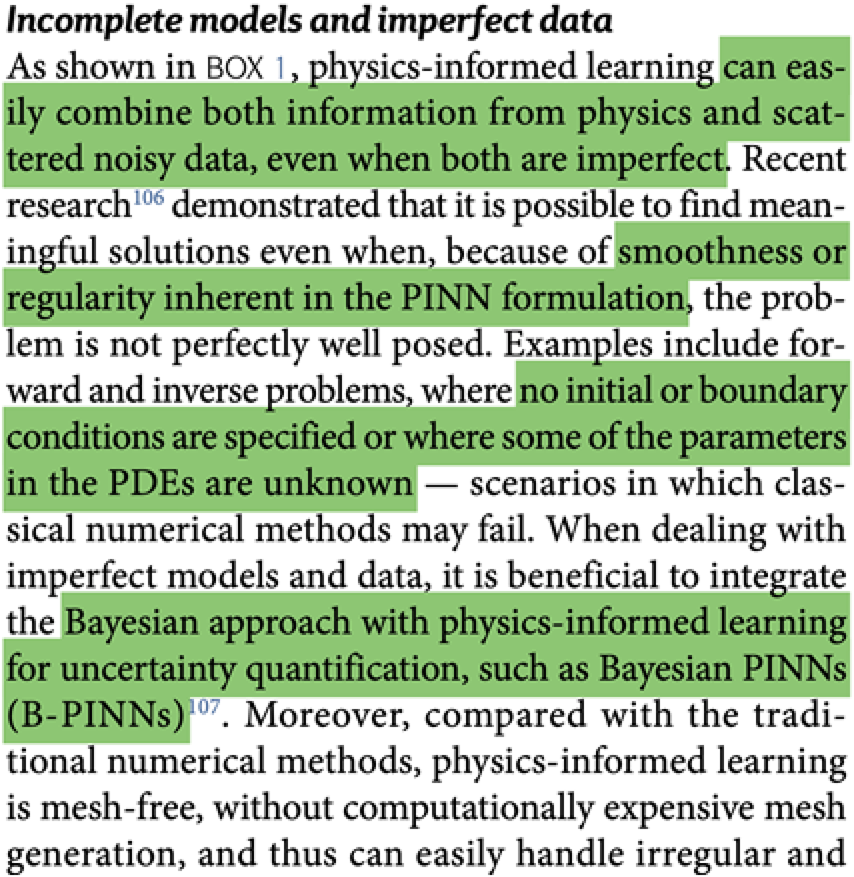
Merits of physics-informed learning

There are already many publications on physics-informed ML across different disciplines for specific applications. For example, different extensions of PINNs cover conservation laws[101] as well as stochastic and fractional PDEs for random phenomena and for anomalous transport[102,103]. Combining domain decomposition with PINNs provides more flexibility in multiscale problems, while the formulations are relatively simple to implement in parallel since each subdomain may be represented by a different NN, assigned to a different GPU



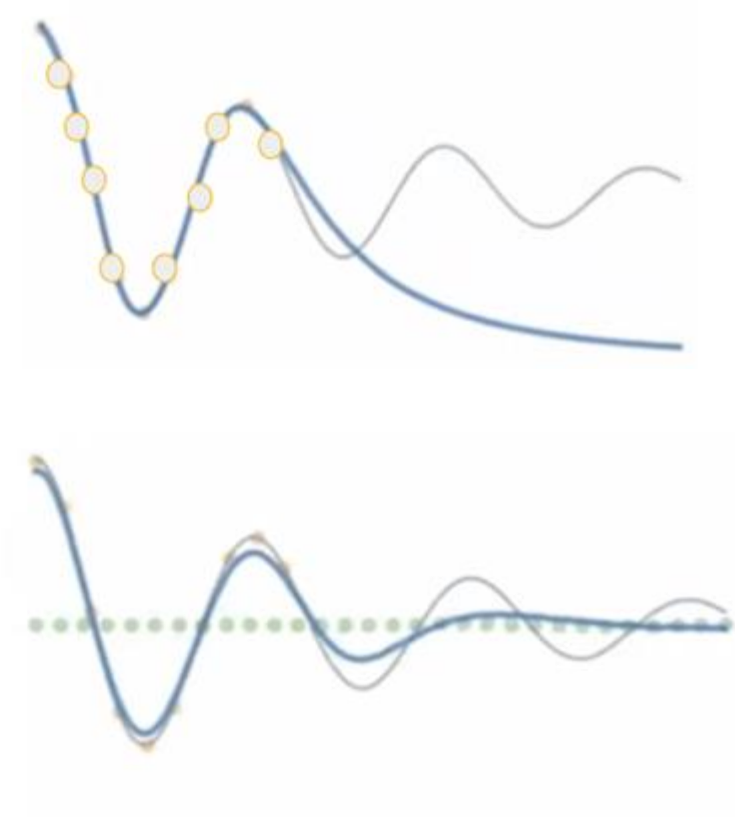
with very small communication cost[101,104,105]. Collectively, the results from these works demonstrate that PINNs are particularly effective in solving ill-posed and inverse problems, whereas for forward, well-posed problems that do not require any data assimilation the existing numerical grid-based solvers currently outperform PINNs. In the following, we discuss in more detail for which scenarios the use of PINNs may be advantageous and highlight these advantages in some prototypical applications.

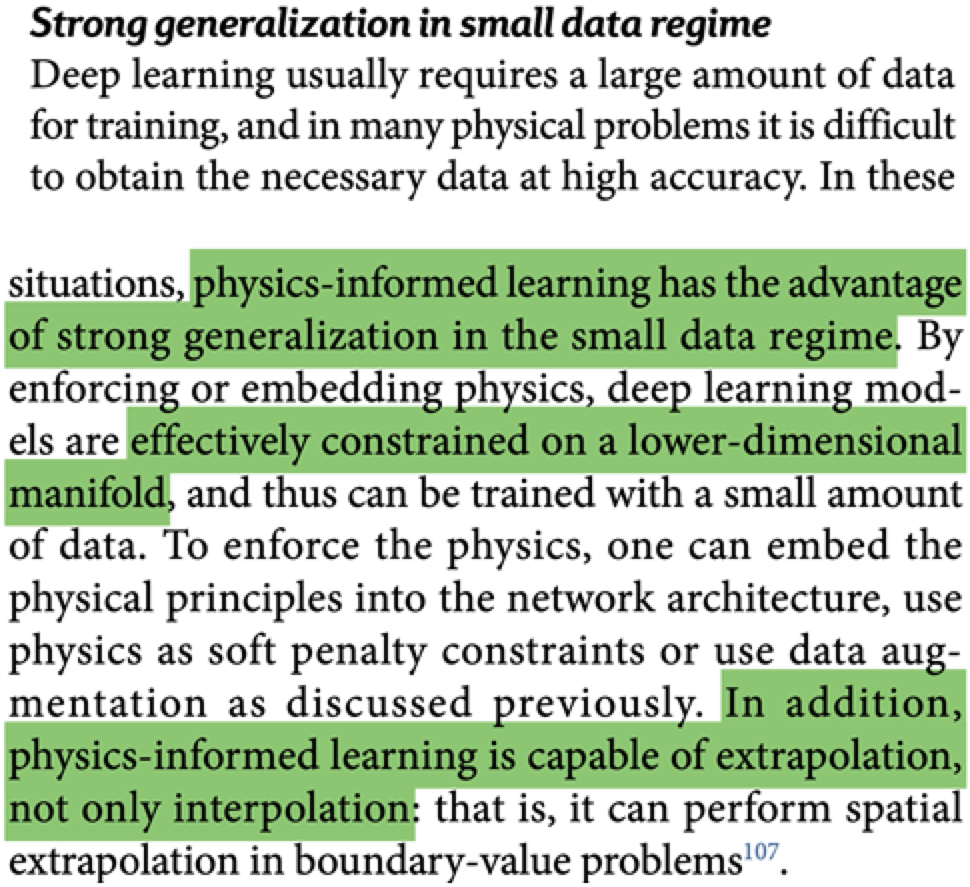
Better at addressing imperfect model and data



Incomplete models and imperfect data

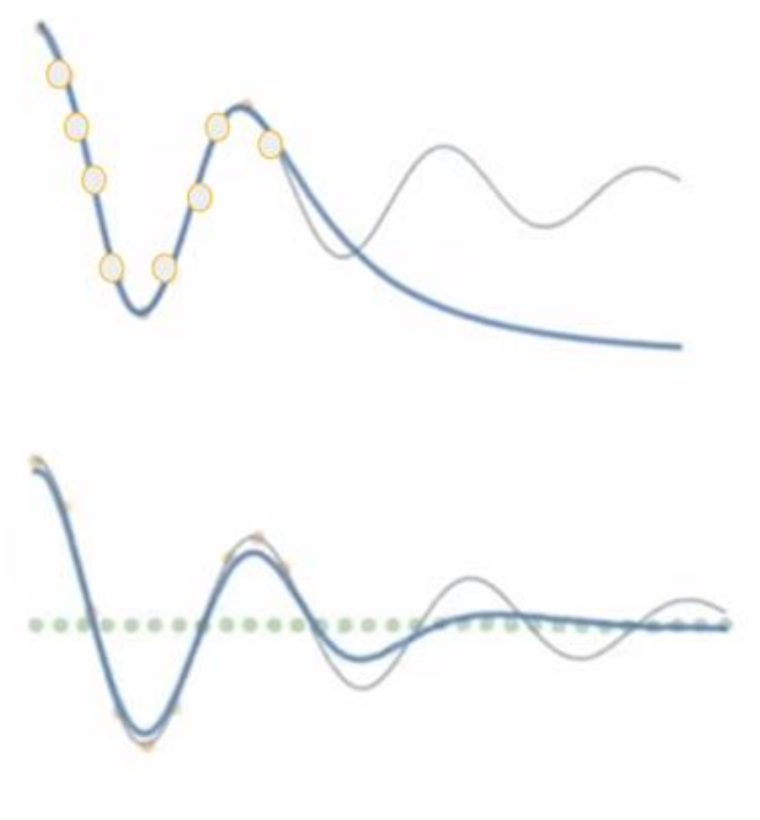
As shown in BOX 1, physics-informed learning can easily combine both information from physics and scattered noisy data, even when both are imperfect. Recent research[106] demonstrated that it is possible to find meaningful solutions even when, because of smoothness or regularity inherent in the PINN formulation, the problem is not perfectly well posed. Examples include forward and inverse problems, where no initial or boundary conditions are specified or where some of the parameters in the PDEs are unknown — scenarios in which classical numerical methods may fail. When dealing with imperfect models and data, it is beneficial to integrate the Bayesian approach with physics-informed learning for uncertainty quantification, such as Bayesian PINNs (B-PINNs)[107]. Moreover, compared with the traditional numerical methods, physics-informed learning is mesh-free, without computationally expensive mesh generation, and thus can easily handle irregular and

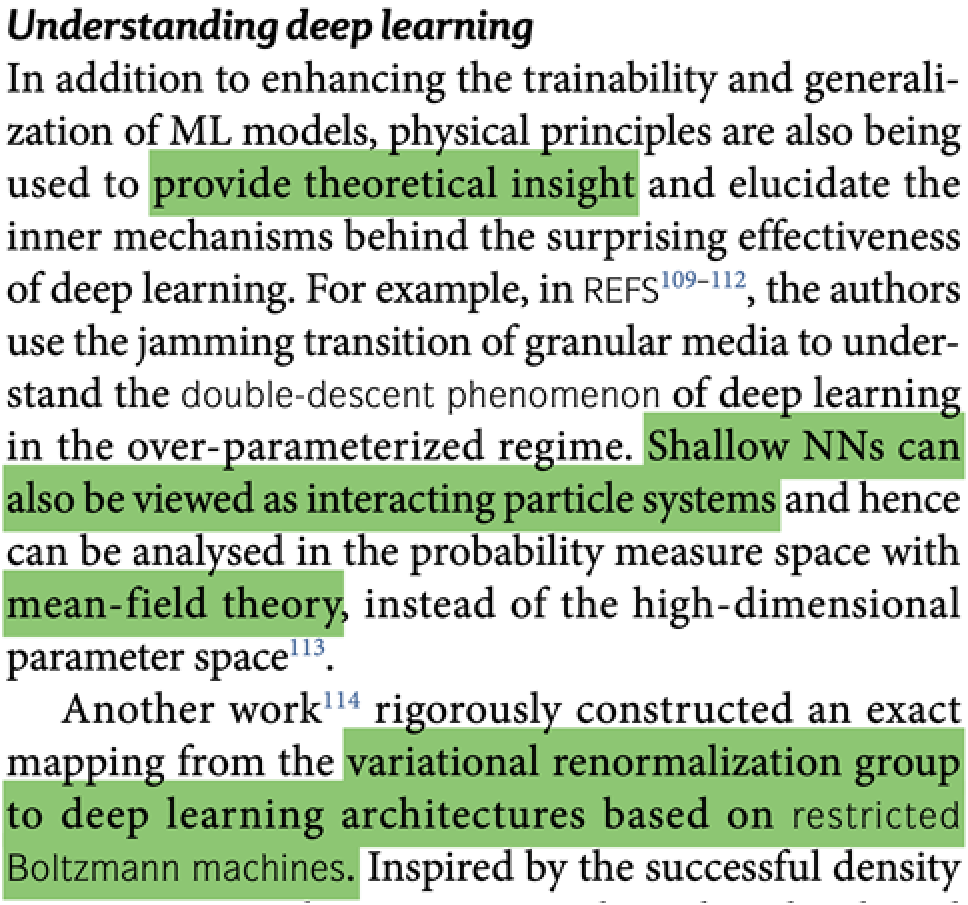




Strong generalization in small data regime

Deep learning usually requires a large amount of data for training, and in many physical problems it is difficult to obtain the necessary data at high accuracy. In these situations, physics-informed learning has the advantage of strong generalization in the small data regime. By enforcing or embedding physics, deep learning models are effectively constrained on a lower-dimensional manifold, and thus can be trained with a small amount of data. To enforce the physics, one can embed the physical principles into the network architecture, use physics as soft penalty constraints or use data augmentation as discussed previously. In addition, physics-informed learning is capable of extrapolation, not only interpolation: that is, it can perform spatial extrapolation in boundary-value problems[107].

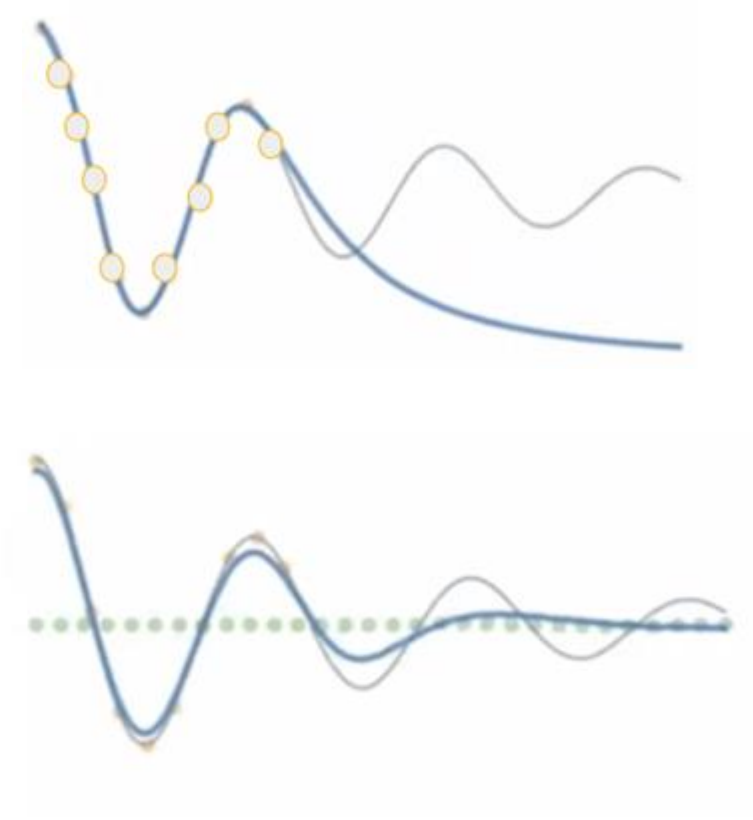


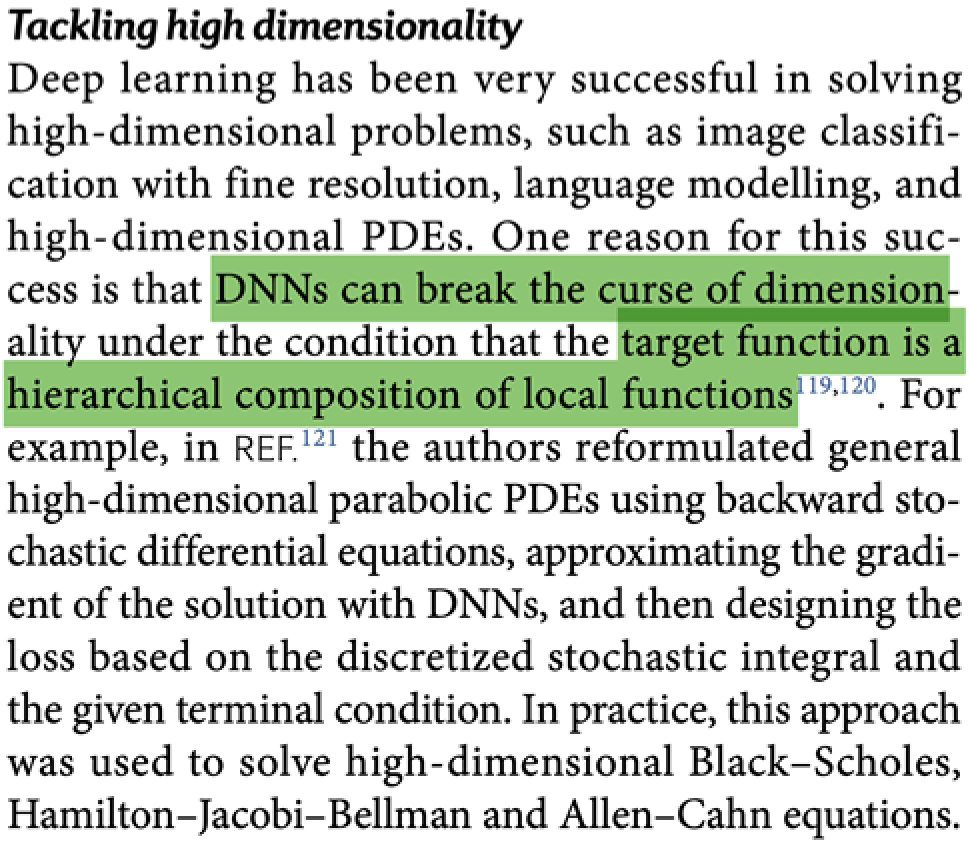


Understanding deep learning

In addition to enhancing the trainability and generalization of ML models, physical principles are also being used to provide theoretical insight and elucidate the inner mechanisms behind the surprising effectiveness of deep learning. For example, in REFS[109–112], the authors use the jamming transition of granular media to understand the double-descent phenomenon of deep learning in the over-parameterized regime. Shallow NNs can also be viewed as interacting particle systems and hence can be analysed in the probability measure space with mean-field theory, instead of the high-dimensional parameter space[113].

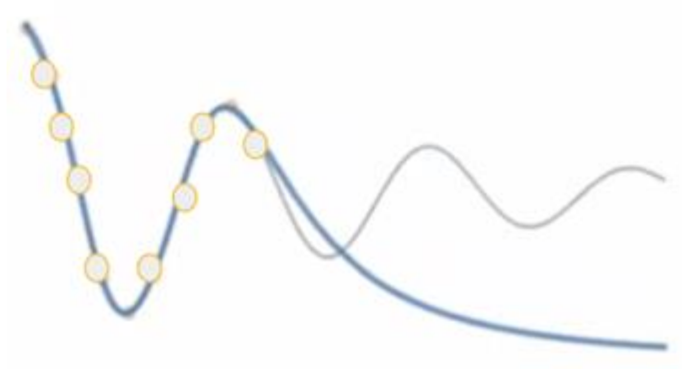
Another work[114] rigorously constructed an exact mapping from the variational renormalization group to deep learning architectures based on restricted Boltzmann machines. Inspired by the successful density

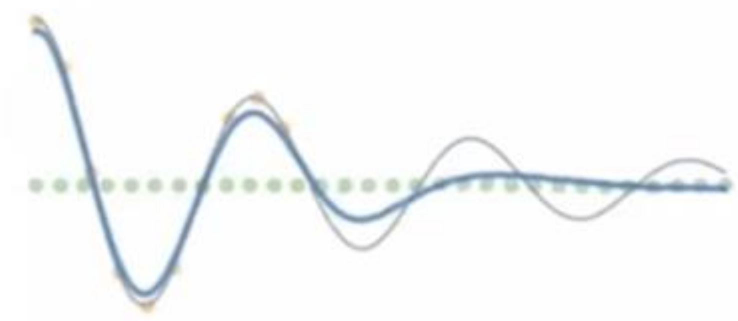


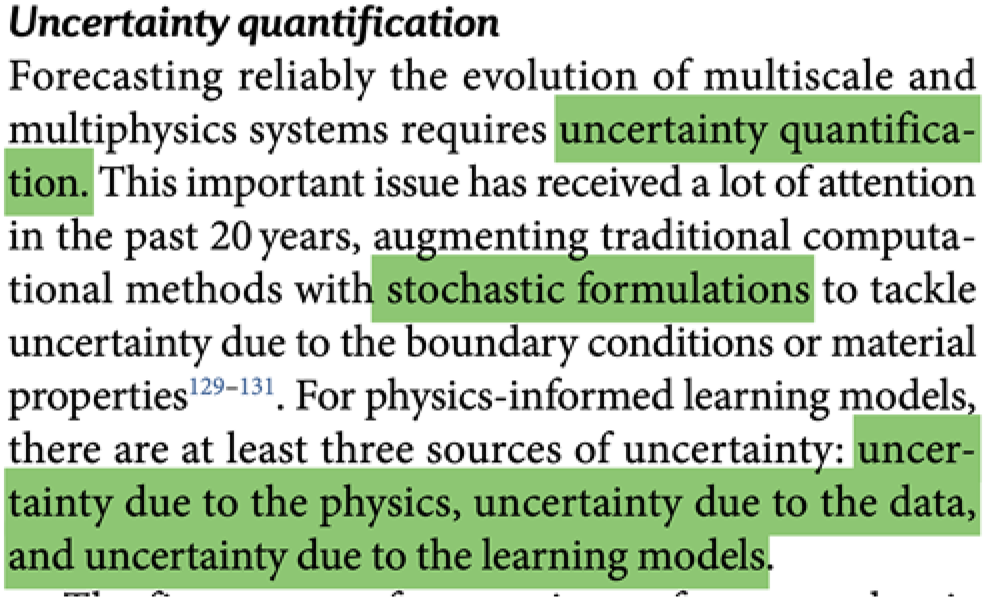


Tackling high dimensionality

Deep learning has been very successful in solving high-dimensional problems, such as image classification with fine resolution, language modelling, and high-dimensional PDEs. One reason for this success is that DNNs can break the curse of dimensionality under the condition that the target function is a hierarchical composition of local functions[119,120]. For example, in REF.[121] the authors reformulated general high-dimensional parabolic PDEs using backward stochastic differential equations, approximating the gradient of the solution with DNNs, and then designing the loss based on the discretized stochastic integral and the given terminal condition. In practice, this approach was used to solve high-dimensional Black–Scholes, Hamilton–Jacobi–Bellman and Allen–Cahn equations.

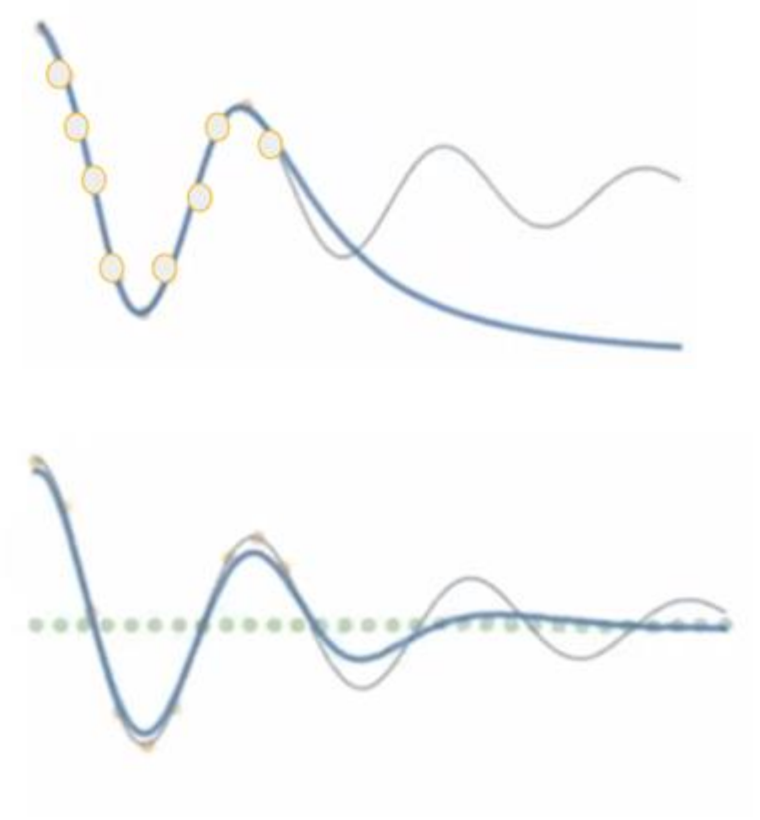






Uncertainty quantification

Forecasting reliably the evolution of multiscale and multiphysics systems requires uncertainty quantification. This important issue has received a lot of attention in the past 20 years, augmenting traditional computational methods with stochastic formulations to tackle uncertainty due to the boundary conditions or material properties[129–131]. For physics-informed learning models, there are at least three sources of uncertainty: uncertainty due to the physics, uncertainty due to the data, and uncertainty due to the learning models.



Solving Prosail model via physics-informed approaches?

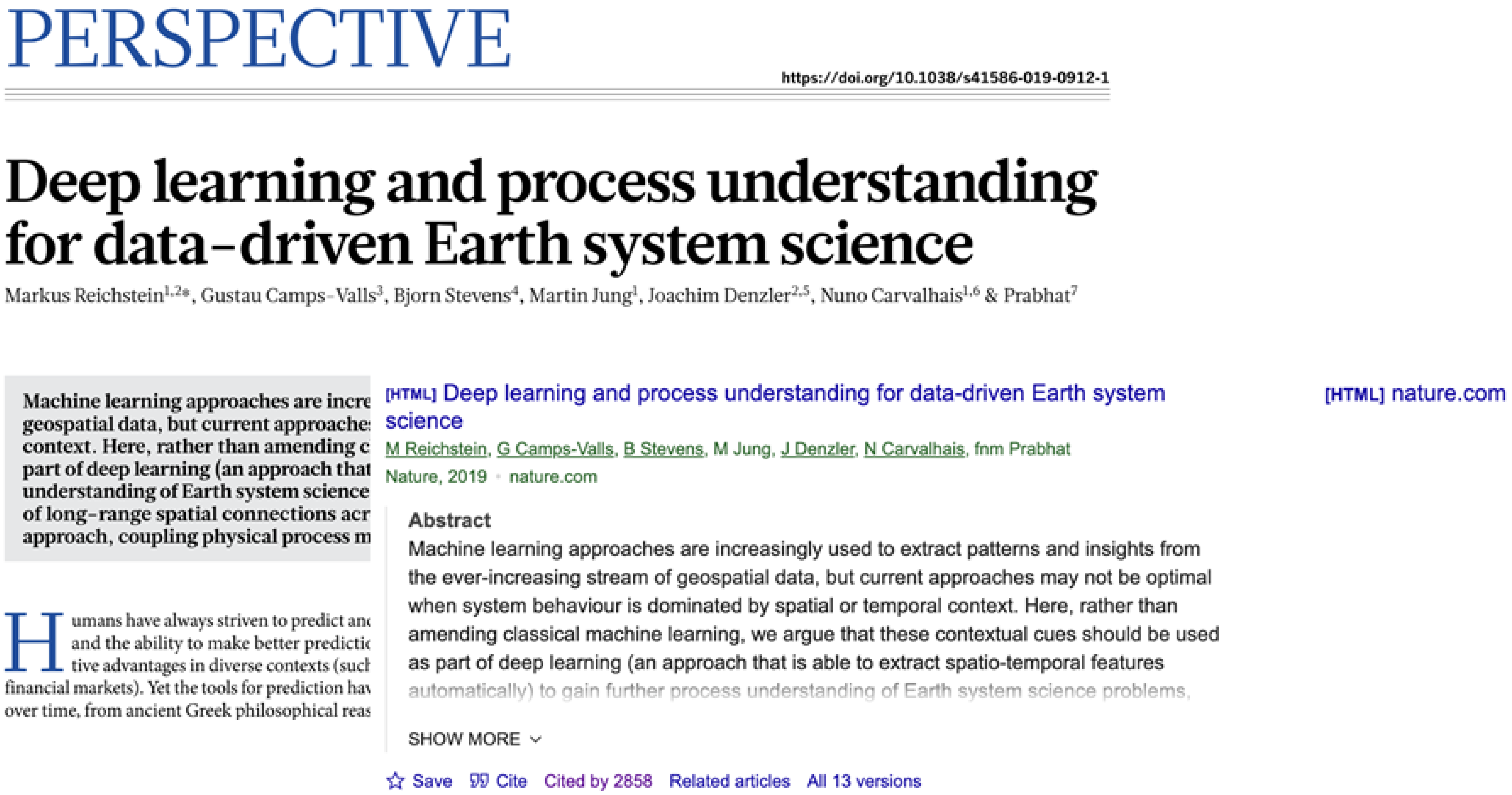
(1) Observation bias -> simulate data using physical models

(2) Inductive bias -> integrate physics into model architecture

(3) Learning bias -> use unsupervise PDE loss

Spectra observations -> Encoder (NN) -> bioparameters -> Prosail -> Simulated spectra

Loss = (Spectra observations - Simulated spectra)



PERSPECTIVE

https://doi.org/10.1038/s41586-019-0912-1

Deep learning and process understanding for data-driven Earth system science

Markus Reichstein[1,2]*, Gustau Camps-Valls[3], Bjorn Stevens[4], Martin Jung[1], Joachim Denzler[2,5], Nuno Carvalhais[1,6] & Prabhat[7]

Machine learning approaches are incre
geospatial data, but current approache
context. Here, rather than amending c
part of deep learning (an approach that
understanding of Earth system science
of long-range spatial connections acr
approach, coupling physical process m

Humans have always striven to predict anc
and the ability to make better predictic
tive advantages in diverse contexts (such
financial markets). Yet the tools for prediction hav
over time, from ancient Greek philosophical reas

[HTML] Deep learning and process understanding for data-driven Earth system science

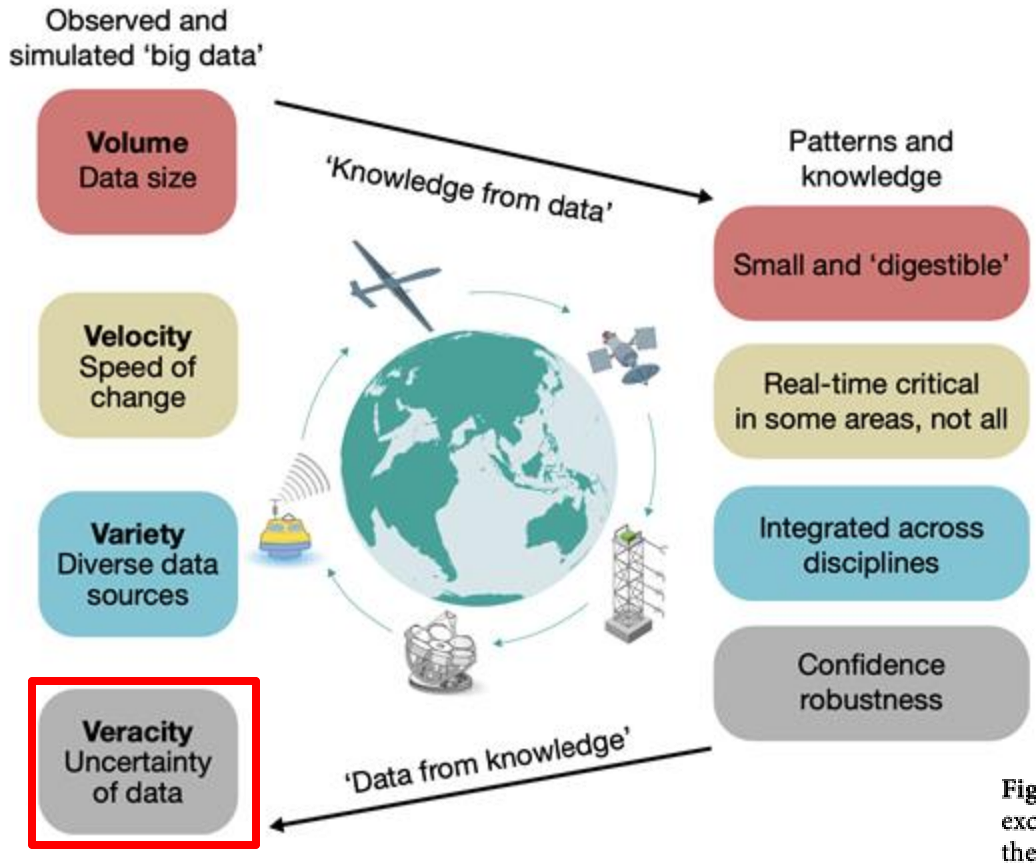
[HTML] nature.com

M Reichstein, G Camps-Valls, B Stevens, M Jung, J Denzler, N Carvalhais, fnm Prabhat

Nature, 2019 · nature.com

Abstract

Machine learning approaches are increasingly used to extract patterns and insights from the ever-increasing stream of geospatial data, but current approaches may not be optimal when system behaviour is dominated by spatial or temporal context. Here, rather than amending classical machine learning, we argue that these contextual cues should be used as part of deep learning (an approach that is able to extract spatio-temporal features automatically) to gain further process understanding of Earth system science problems,



Observed and simulated ‘big data’

Volume Data size

Velocity Speed of change

Variety Diverse data sources

Veracity Uncertainty of data

‘Knowledge from data’

‘Data from knowledge’

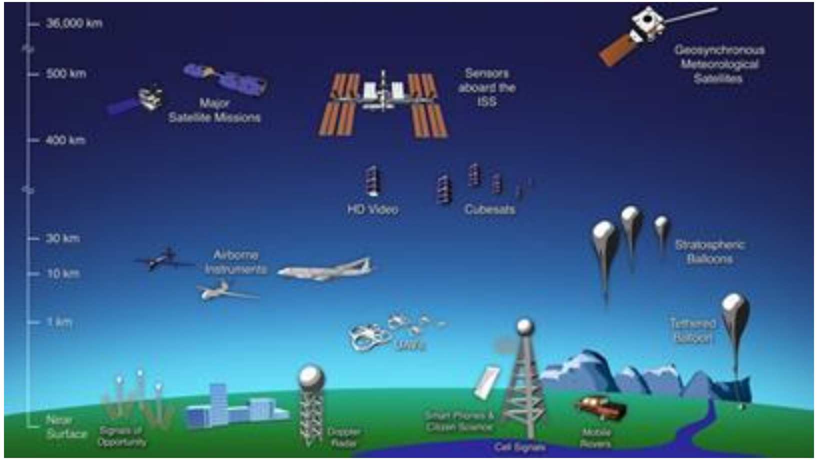
Patterns and knowledge

Small and ‘digestible’

Real-time critical in some areas, not all

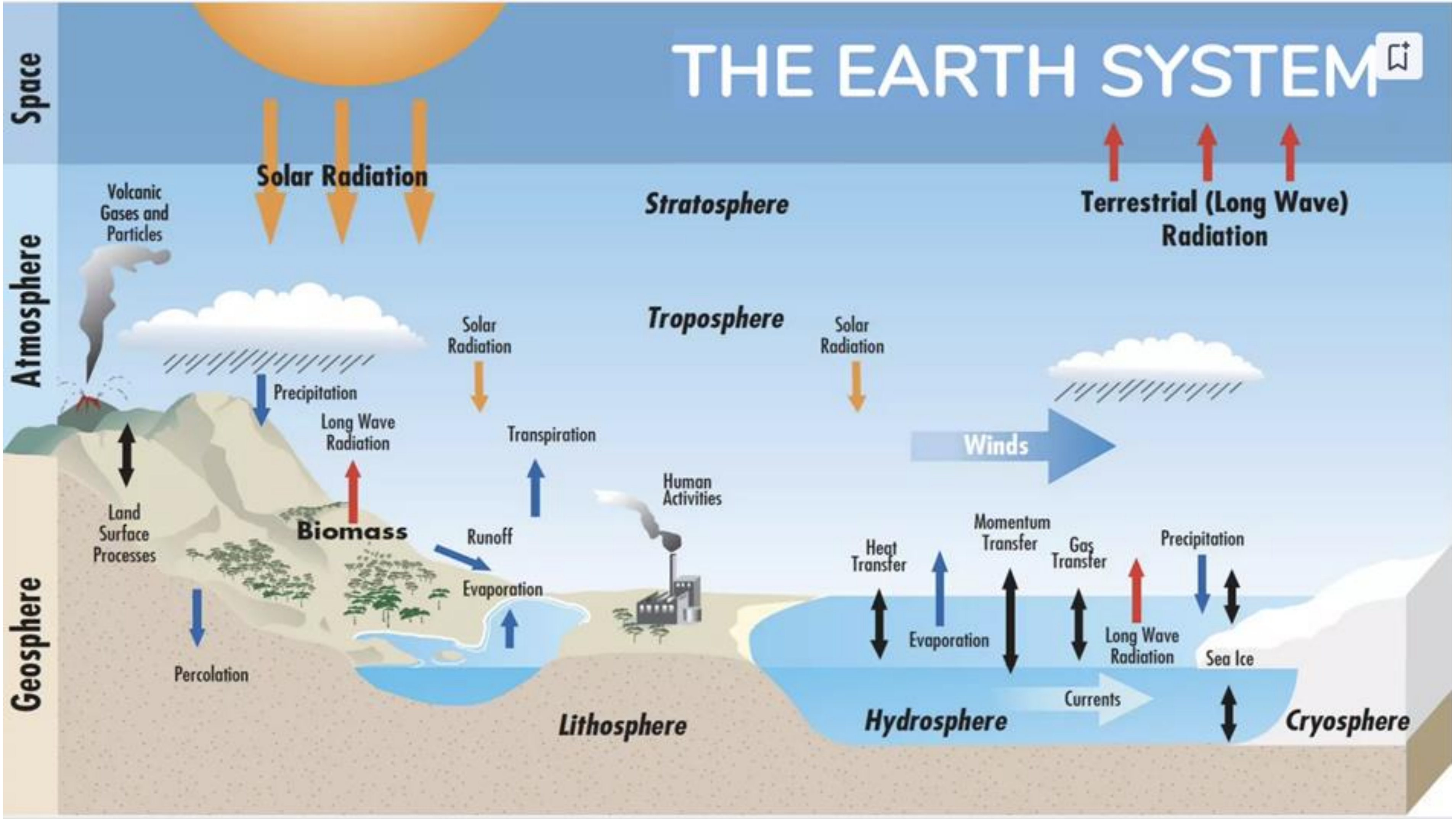
Integrated across disciplines

Confidence robustness

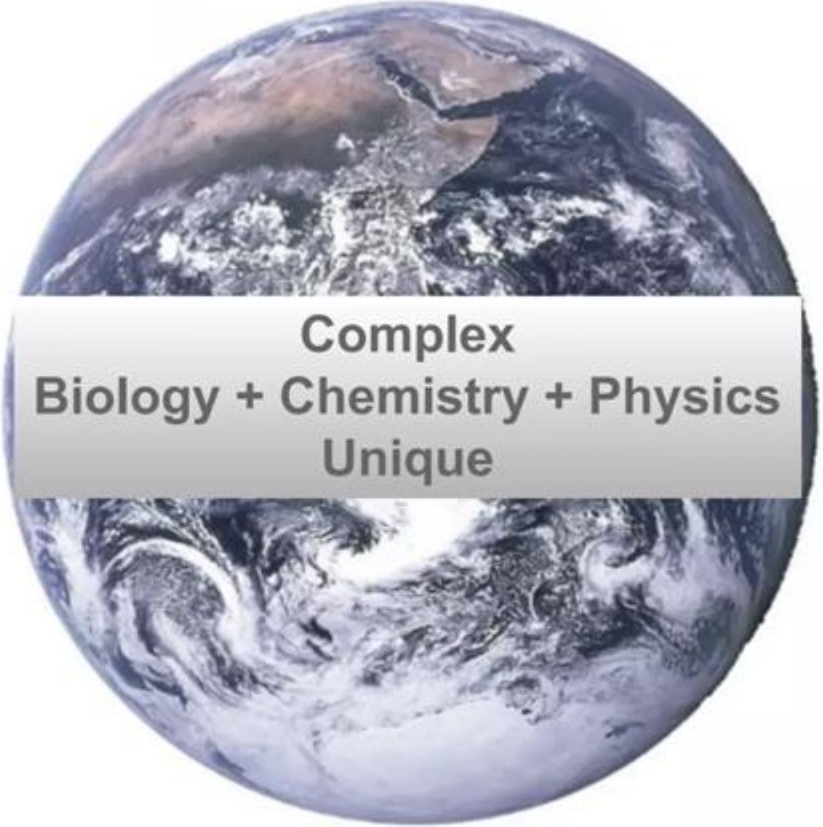


Veracity: the quality of being true, honest, or accurate.

Fig. 1 | Big data challenges in the geoscientific context. Data size now exceeds 100 petabytes, and is growing quasi-exponentially (tapering of the figure to the right indicates decreasing data size.) The speed of change exceeds 5 petabytes a year; data are taken at frequencies of up to 10 Hz or more; reprocessing and versioning are common challenges. Data sources can be one- to four-dimensional, spatially integrated, from the organ level (such as leaves) to the global level. Earth has diverse observational systems, from remote sensing to in situ observation. The uncertainty of data can stem from observational errors or conceptual inconsistencies.


THE EARTH SYSTEM
Space
Atmosphere
Geosphere
Solar Radiation
Volcanic Gases and Particles
Stratosphere
Terrestrial (Long Wave) Radiation
Troposphere
Solar Radiation
Solar Radiation
Precipitation
Long Wave Radiation
Transpiration
Winds
Human Activities
Land Surface Processes
Biomass
Runoff
Evaporation
Percolation
Heat Transfer
Momentum Transfer
Gas Transfer
Precipitation
Evaporation
Long Wave Radiation
Sea Ice
Currents
Lithosphere
Hydrosphere
Cryosphere

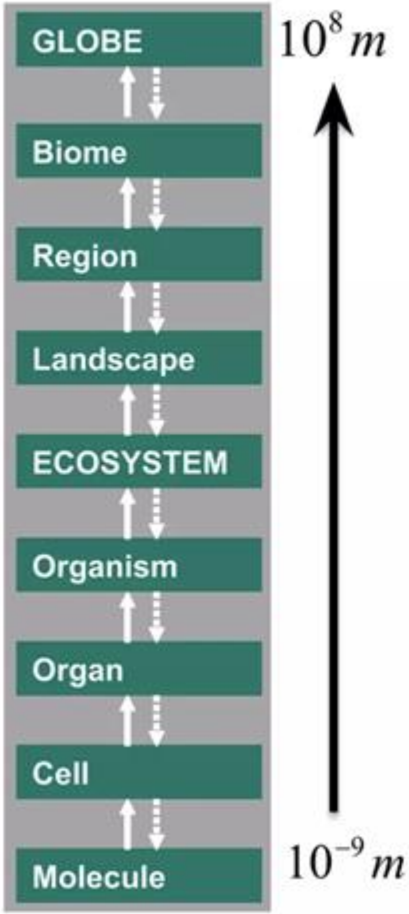
The Earth System



Complex

Biology + Chemistry + Physics

Unique



$10^8 m$

GLOBE

Biome

Region

Landscape

ECOSYSTEM

Organism

Organ

Cell

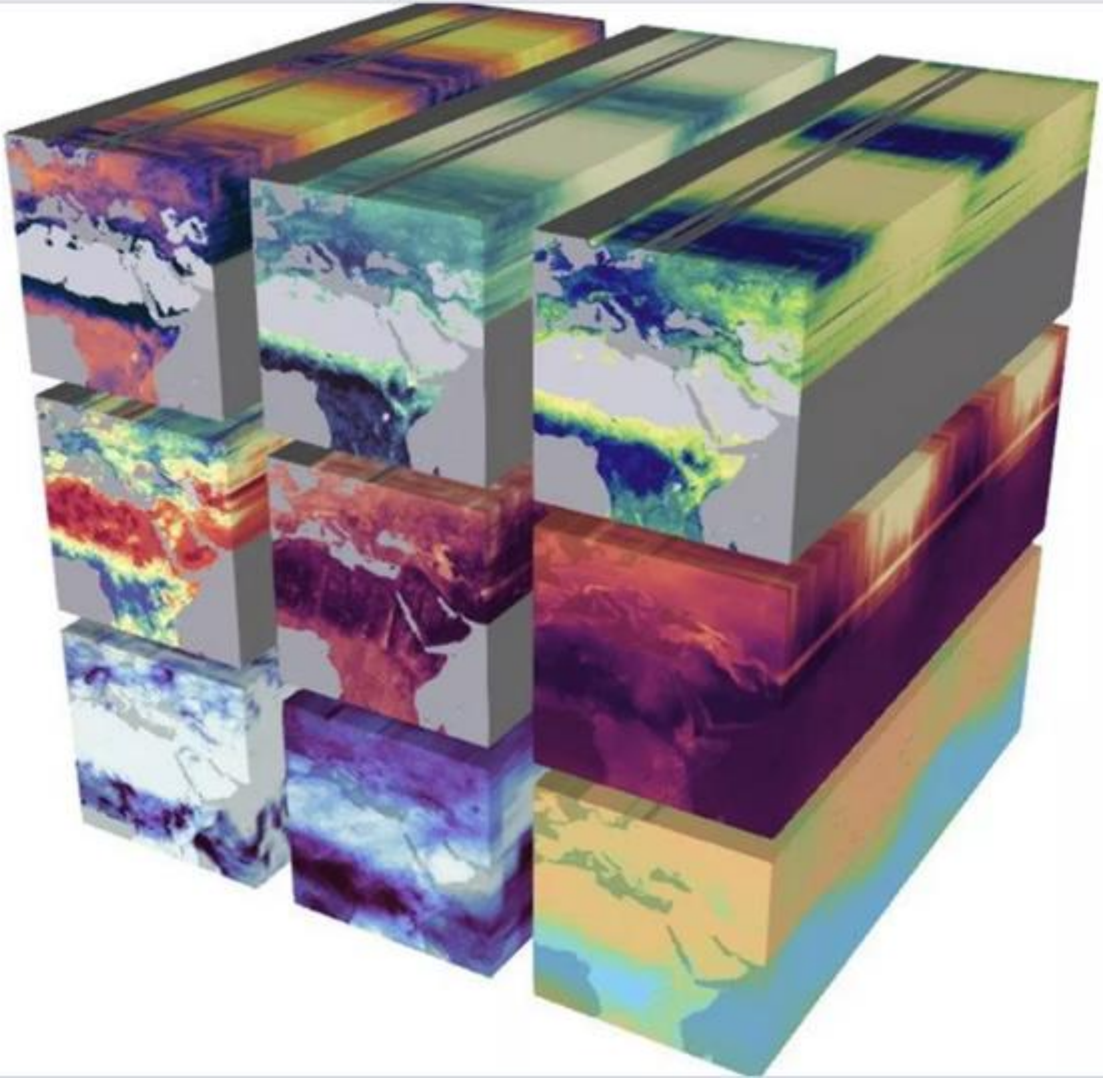
Molecule

$10^{-9} m$

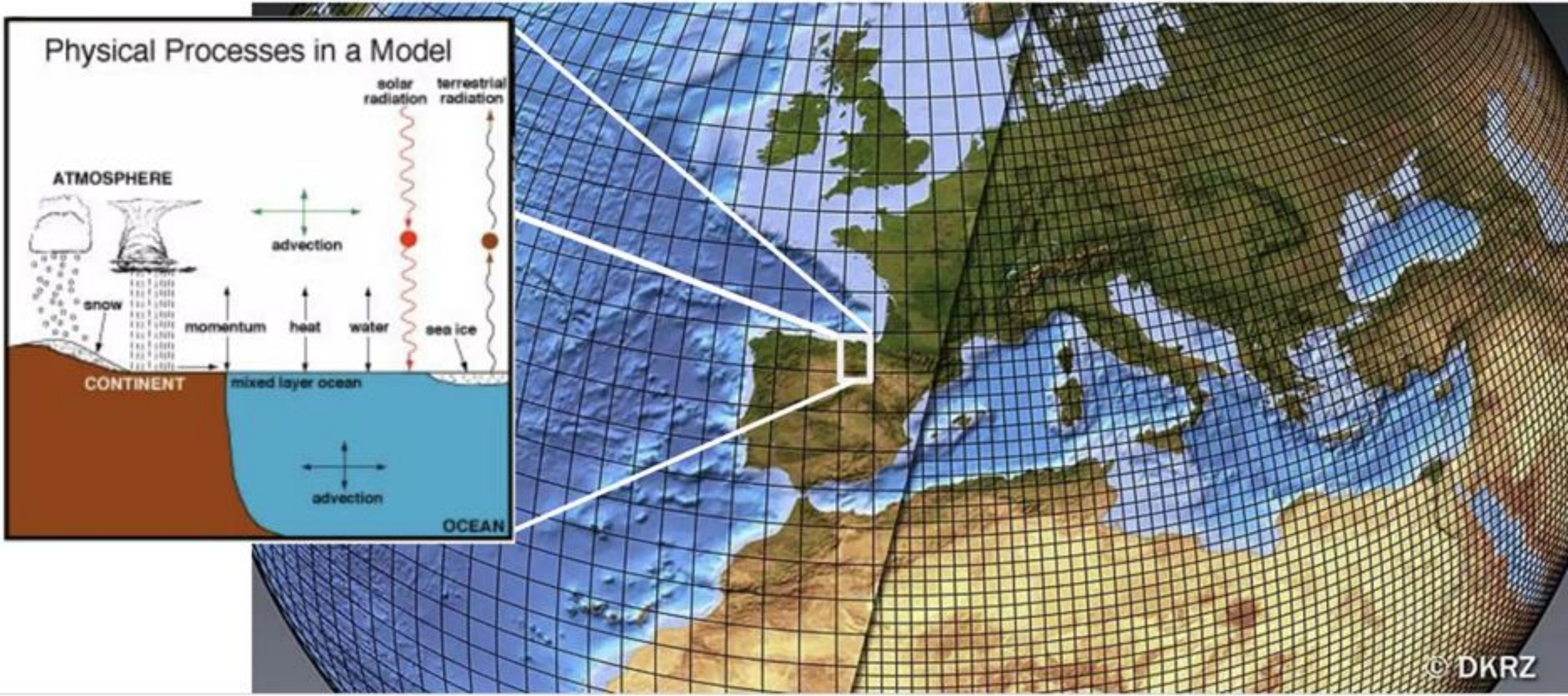
Slide courtesy from the author Markus Reichstein

It’s not like we haven’t got enough data at our hands…

There’s observational data…



Model data are the result of simulations generated by numerically solved differential equations derived from physical models by discretizing the Earth and representing key processes with parameterizations



© DKRZ

Multimodel analysis

WCRP CMIP6 World Climate Research Programme

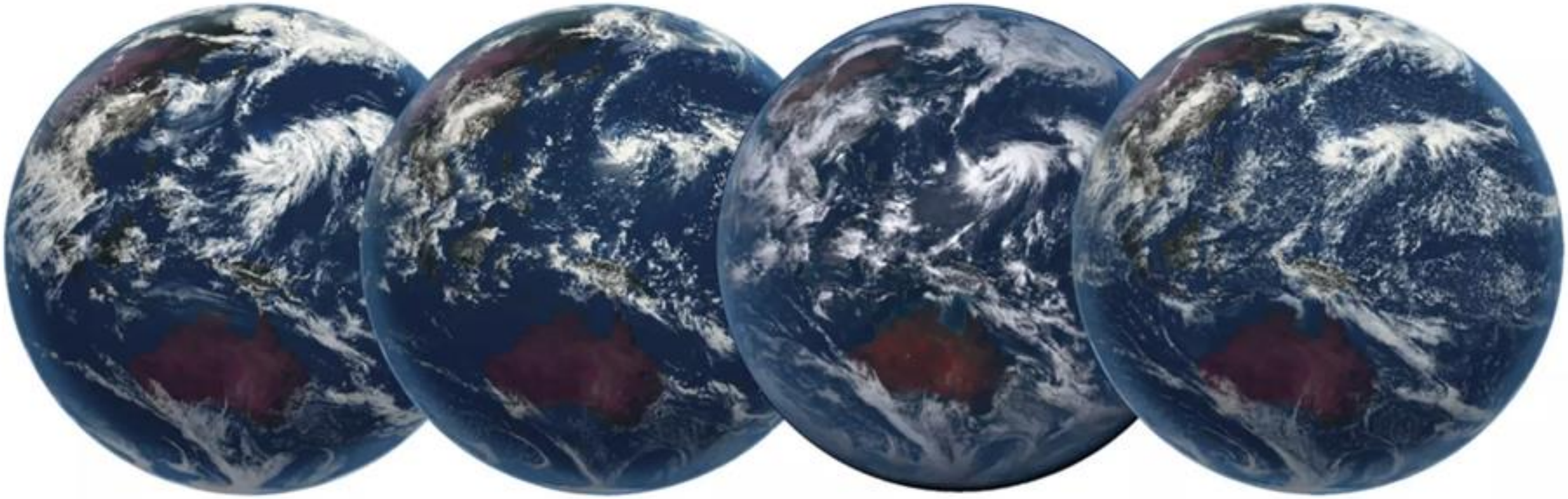
ESGF Earth System Grid Federation

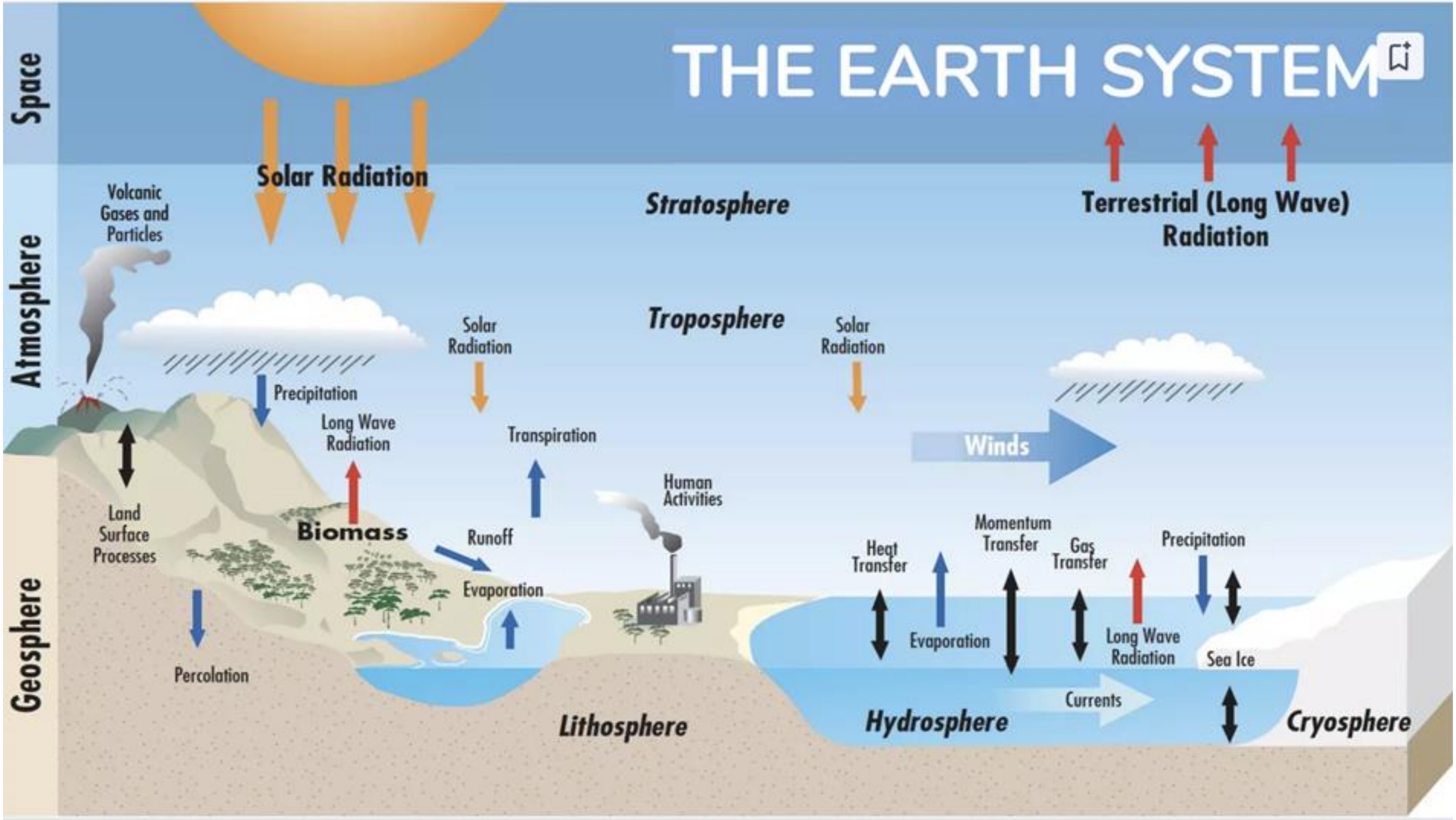
ipcc INTERGOVERNMENTAL PANEL ON climate change WMO UNEP

Last report:
3.5PBytes
Next report:
~ 30PBytes

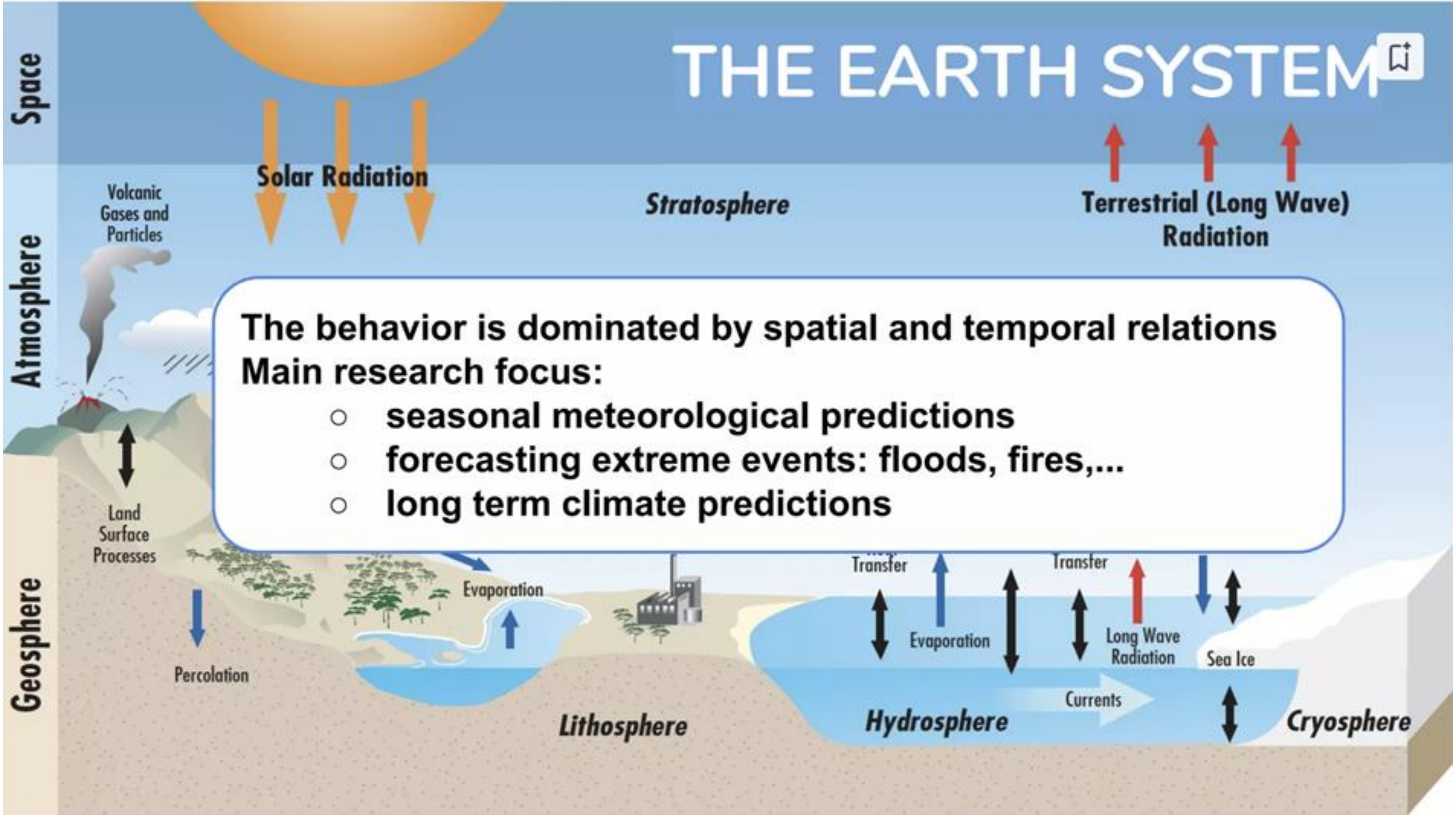
©DKRZ/MPI-M

Climate Models at the 1km scale are coming up
~650 GB of data per output time step




THE EARTH SYSTEM
Space
Atmosphere
Geosphere
Solar Radiation
Volcanic Gases and Particles
Stratosphere
Terrestrial (Long Wave) Radiation
Troposphere
Solar Radiation
Solar Radiation
Precipitation
Long Wave Radiation
Transpiration
Winds
Land Surface Processes
Biomass
Human Activities
Runoff
Evaporation
Percolation
Heat Transfer
Momentum Transfer
Gas Transfer
Precipitation
Evaporation
Long Wave Radiation
Sea Ice
Currents
Lithosphere
Hydrosphere
Cryosphere

THE EARTH SYSTEM



The behavior is dominated by spatial and temporal relations

Main research focus:

- **seasonal meteorological predictions**
- **forecasting extreme events: floods, fires,...**
- **long term climate predictions**

example of **spatio-temporal** relations:

the prediction of fire occurrence, the stimation of burnt area, and the trace of gas emissions depend on:

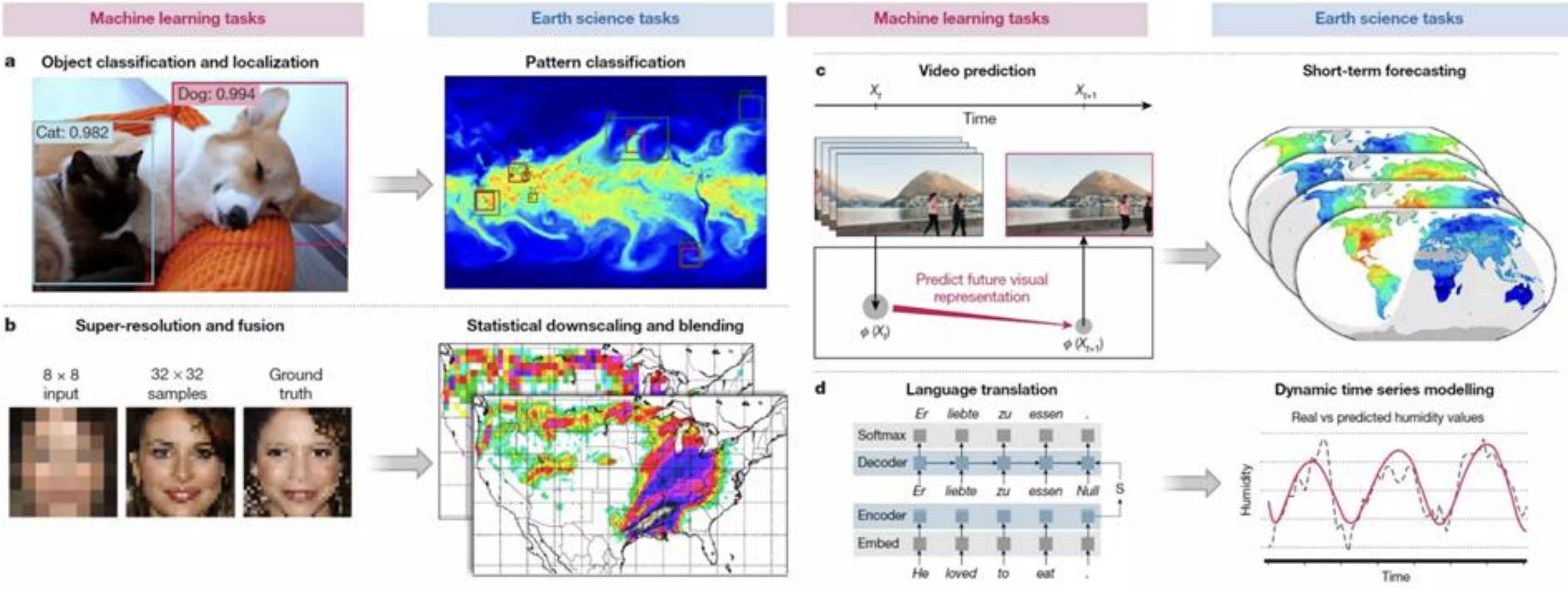
- instantaneous climatic drivers: temperature, humidity,...
- sources of ignition: humans, lightning,...
- state variables: available fuel,..
- moisture, terrain, wind speed and direction,..

Machine learning applications often do not directly and exhaustively account for spatio-temporal correlations

Deep learning is a promising approach

Example: convolutional networks (spatial) + recurrent networks (memory, sequence learning)

Examples of Deep Learning applications in Earth System Science



Slide courtesy from the author Markus Reichstein

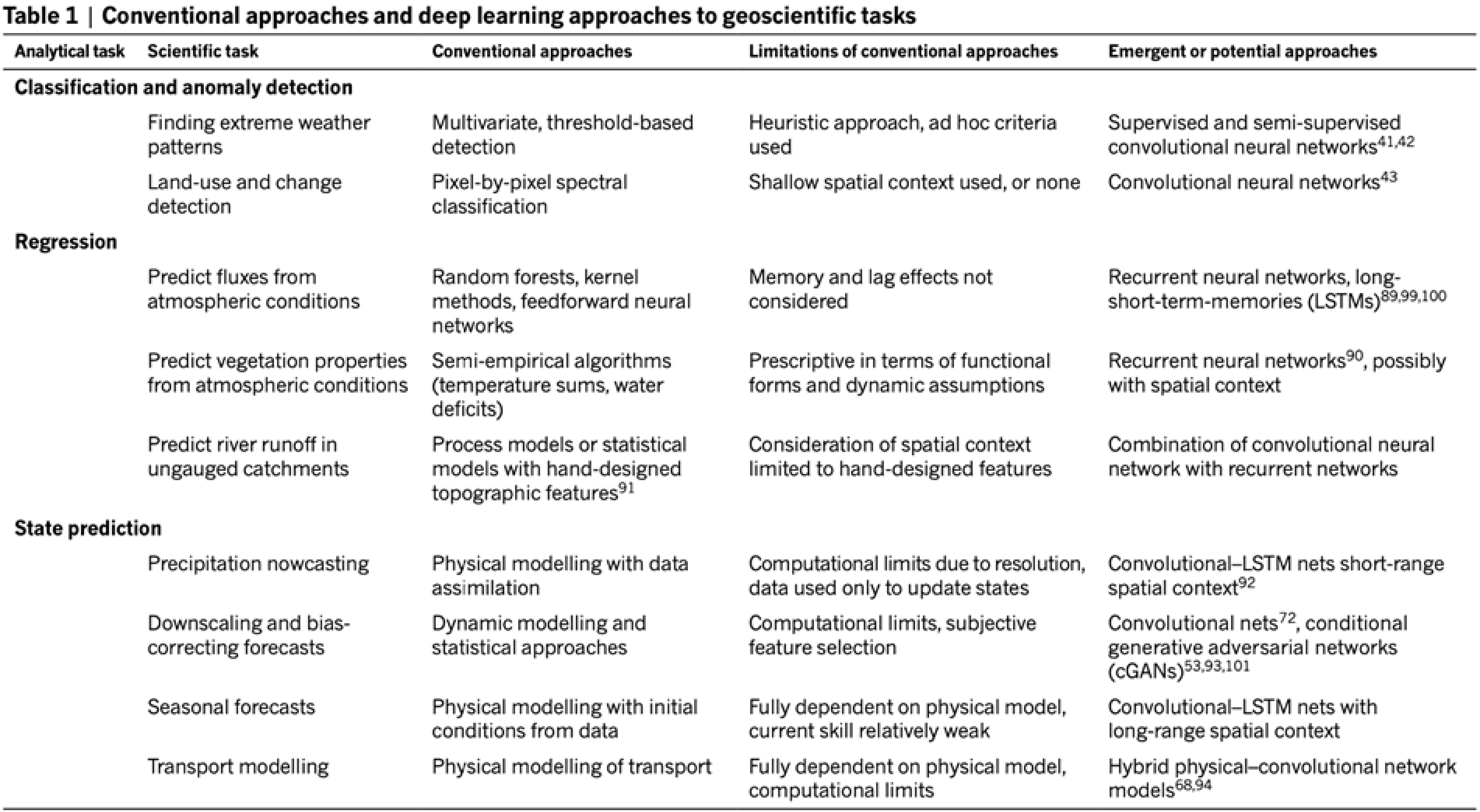


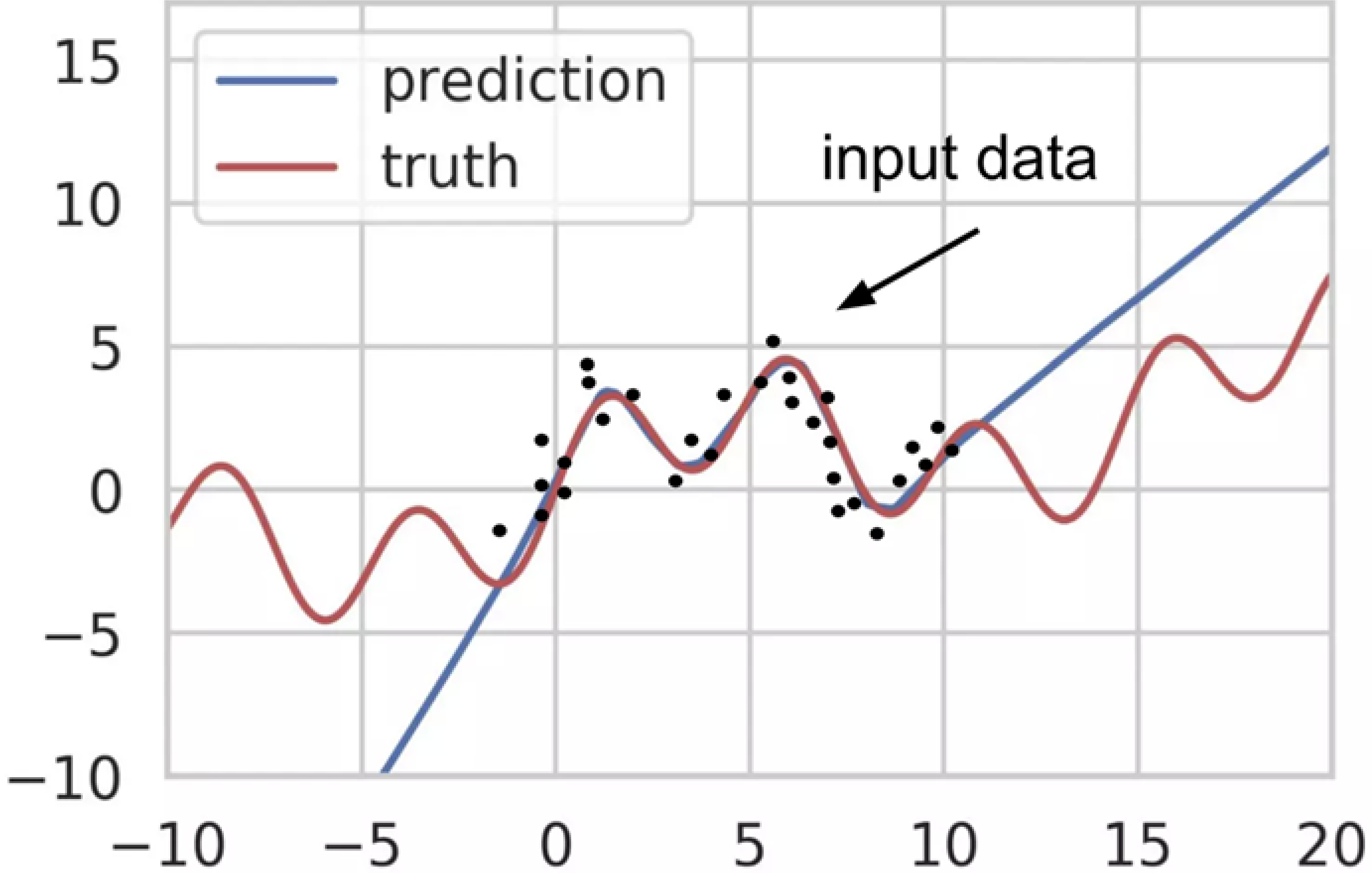
Table 1 | Conventional approaches and deep learning approaches to geoscientific tasks

Analytical task	Scientific task	Conventional approaches	Limitations of conventional approaches	Emergent or potential approaches
Classification and anomaly detection				
	Finding extreme weather patterns	Multivariate, threshold-based detection	Heuristic approach, ad hoc criteria used	Supervised and semi-supervised convolutional neural networks[41,42]
	Land-use and change detection	Pixel-by-pixel spectral classification	Shallow spatial context used, or none	Convolutional neural networks[43]
Regression				
	Predict fluxes from atmospheric conditions	Random forests, kernel methods, feedforward neural networks	Memory and lag effects not considered	Recurrent neural networks, long-short-term-memories (LSTMs)[89,99,100]
	Predict vegetation properties from atmospheric conditions	Semi-empirical algorithms (temperature sums, water deficits)	Prescriptive in terms of functional forms and dynamic assumptions	Recurrent neural networks[90], possibly with spatial context
	Predict river runoff in ungauged catchments	Process models or statistical models with hand-designed topographic features[91]	Consideration of spatial context limited to hand-designed features	Combination of convolutional neural network with recurrent networks
State prediction				
	Precipitation nowcasting	Physical modelling with data assimilation	Computational limits due to resolution, data used only to update states	Convolutional–LSTM nets short-range spatial context[92]
	Downscaling and bias-correcting forecasts	Dynamic modelling and statistical approaches	Computational limits, subjective feature selection	Convolutional nets[72], conditional generative adversarial networks (cGANs)[53,93,101]
	Seasonal forecasts	Physical modelling with initial conditions from data	Fully dependent on physical model, current skill relatively weak	Convolutional–LSTM nets with long-range spatial context
	Transport modelling	Physical modelling of transport	Fully dependent on physical model, computational limits	Hybrid physical–convolutional network models[68,94]

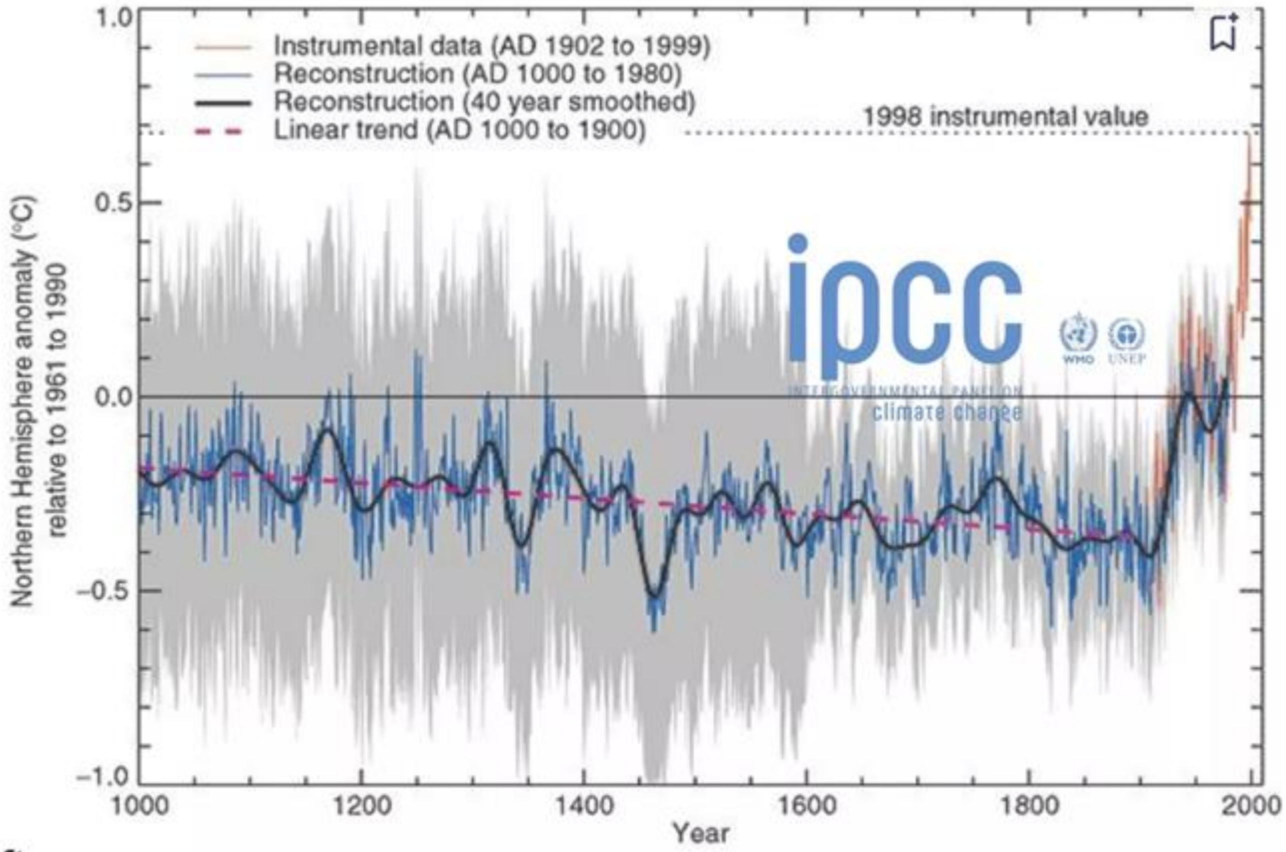
Deep learning challenges in Earth System science

- Diverse sources of noise→ poor signal-to-noise ratio
- Inconsistencies → energy and mass conservations, density must be positive,...
- Extrapolation problem → system changes in time: data shift or concept drift

Extrapolation problem: regression



Non-stationary system



Data shift or concept drift

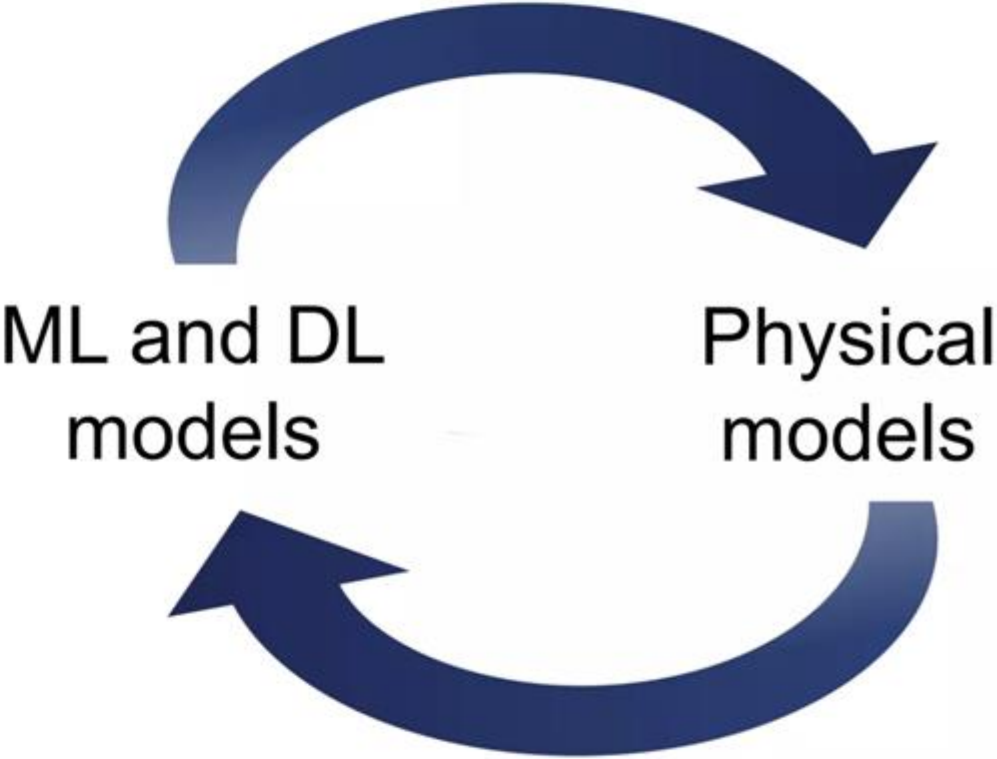
- training data are not longer representative if the system has changed
- the accuracy of the trained model definitely decreased under data shift/concept drift

Deep learning challenges in Earth System science

Images

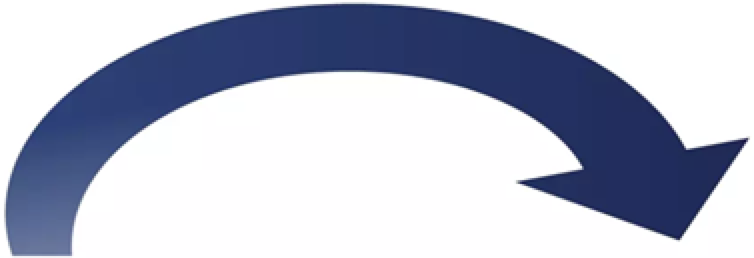
- Beyond visible spectrum → different statistical properties, no i.i.d. sets
- 40 000 x 20 000 pixels for a regular 1 km global resolution
- Multiple scales
- Scale invariant features
- No ImageNet → and difficult to have, example: labelling clouds
- Missing data → a solution Christopher Kadow, the leader of DKRZ machine learning research group, adapted the Nvidia Technology for image inpainting

Hybrid models



ML and DL models

Physical models



ML and DL models

Physical models

Lightweighting/simplifying/speeding up physical models

- improve parametrizations
- analysis of model-observations mismatch
- emulation

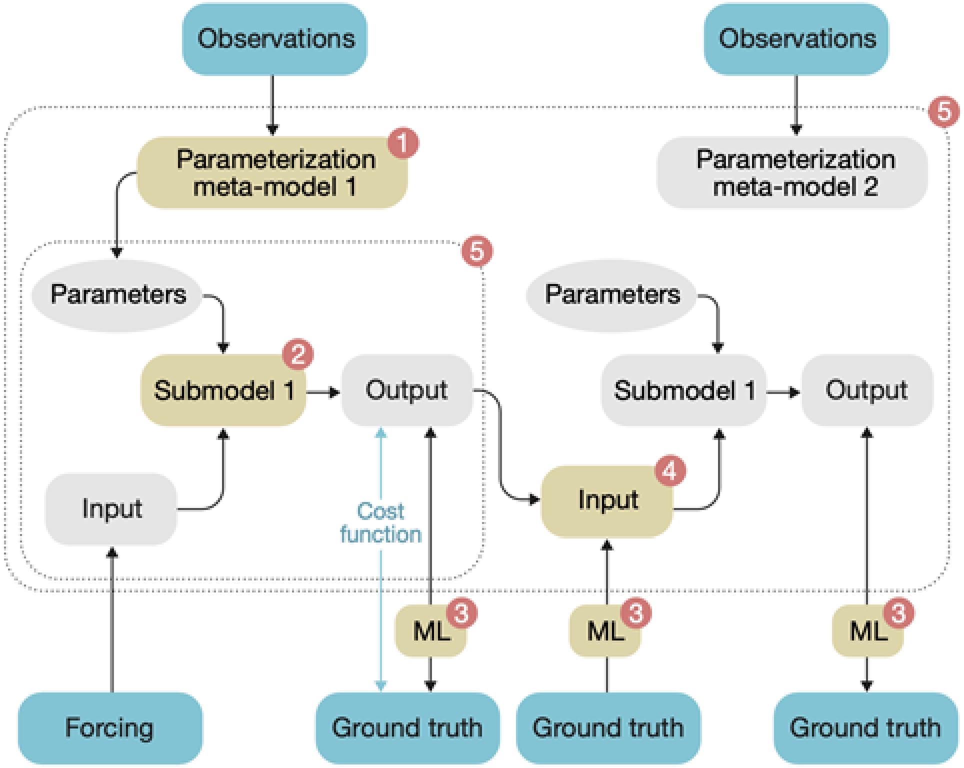


Fig. 3 | Linkages between physical models and machine learning. An abstraction of a part of a physical system—for example, an Earth system model—is depicted here. The model consists of submodels; each submodel has parameters and forcing variables as inputs and produces output, which can be input (forcing) to another sub-model. Data-driven learning approaches can be helpful in various instances, as indicated by the circled numbers. For example, the circle labelled 2 represents hybrid modelling. See the text for more detail. ML, machine learning.

(1) Improving parameterizations

See Fig. 3 (circle 1). Physical models require parameters, but many of those cannot be easily derived from first principles. Machine learning can learn parameterizations to optimally describe the ground truth that can be observed or generated from detailed and high-resolution models through first principles. For example, instead of assigning parameters of the vegetation in an Earth system model to plant functional types (a common ad hoc decision in most global land surface models), one can allow these parameterizations to be learned from appropriate sets of statistical covariates, allowing them to be more dynamic, interdependent and contextual. A prototypical approach has been taken already in hydrology where the mapping of environmental variables (for example, precipitation and surface slope) to catchment parameters (such as mean, minimum and maximum streamflow) has been learned from a few thousand catchments and applied globally to feed hydrological models[63]. Another example from global atmospheric modelling is learning the effective coarse-scale physical parameters of precipitating convection (for example, the fraction of water that is precipitating out of a cloud during convection) from data or high-resolution models[64,65] (the high-resolution models are too expensive to run, which is why coarse-scale parametrizations are needed).

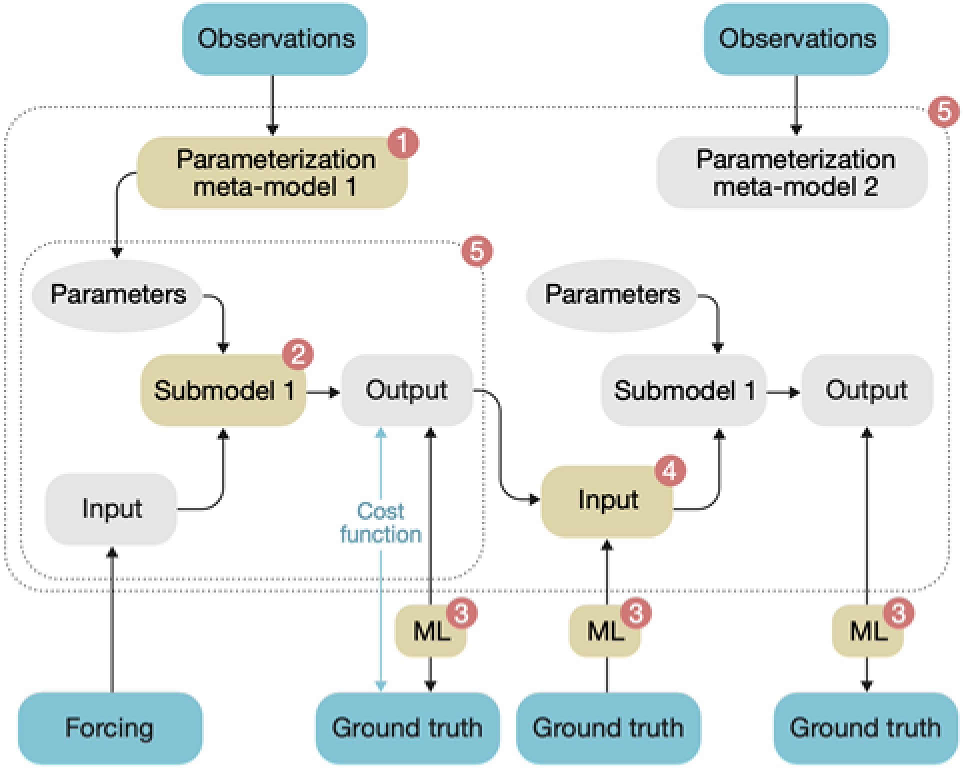


Fig. 3 | Linkages between physical models and machine learning. An abstraction of a part of a physical system—for example, an Earth system model—is depicted here. The model consists of submodels; each submodel has parameters and forcing variables as inputs and produces output, which can be input (forcing) to another sub-model. Data-driven learning approaches can be helpful in various instances, as indicated by the circled numbers. For example, the circle labelled 2 represents hybrid modelling. See the text for more detail. ML, machine learning.

(2) Replacing a 'physical' sub-model with a machine learning model

See Fig. 3 (circle 2). If formulations of a submodel are of semi-empirical nature, where the functional form has little theoretical basis (for example, biological processes), this submodel can be replaced by a machine learning model if a sufficient number of observations are available. This leads to a hybrid model, which combines the strengths of physical modelling (theoretical foundations, interpretable compartments) and machine learning (data-adaptiveness). For example, we could couple well established physical (differential) equations of diffusion for transport of water in plants with machine learning for the poorly understood biological regulation of water transport conductance. This results in a more 'physical' model that obeys accepted conservation of mass and energy laws, but its regulation (biological) is flexible and learned from data. Such principles have recently been taken to efficiently model motion of water in the ocean and specifically predict sea surface temperatures. Here, the motion field was learned via a deep neural network, and then used to update the heat content and temperatures via physically modelling the movement implied by the motion field[68]. Also, a number of atmospheric scientists have begun experimenting with related approaches to circumvent long-standing biases in physically based parameterizations of atmospheric convection[65,69].

The problem may become more complicated if physical model and machine learning parameters are to be estimated simultaneously while maintaining interpretability, especially when several sub-models are replaced with machine learning approaches. In the field of chemistry this approach has been used in calibration exercises and to describe changes in unknown kinetic rates while maintaining mass balance in biochemical reactor modelling[70], which, although less complex, bears many similarities to hydrological and biogeochemical modelling.

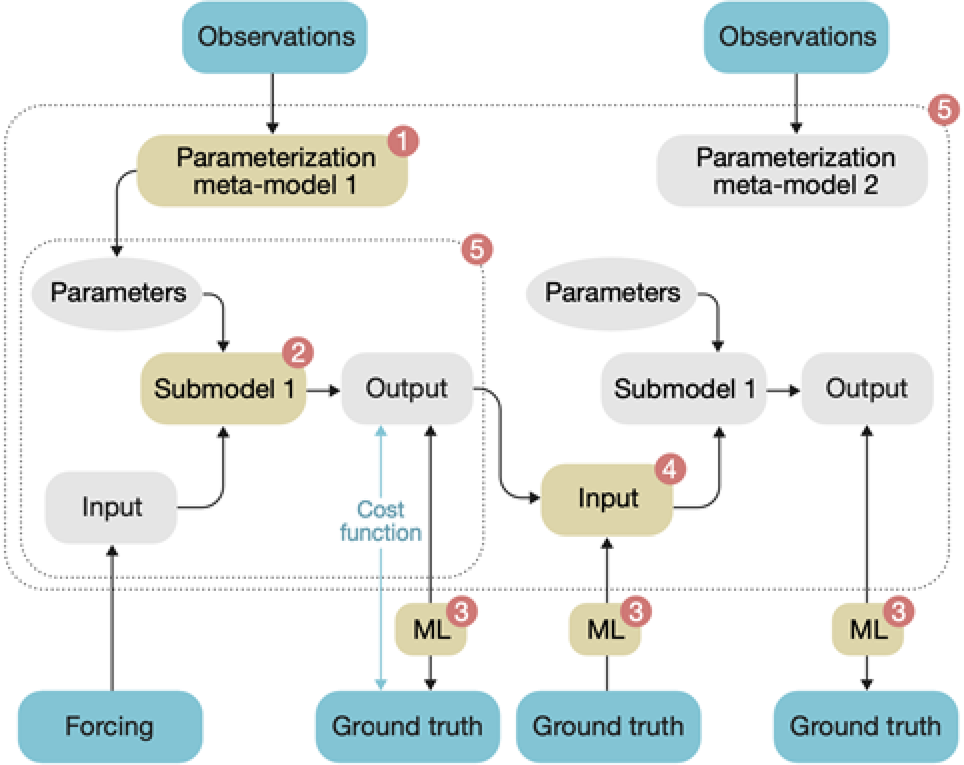


Fig. 3 | Linkages between physical models and machine learning. An abstraction of a part of a physical system—for example, an Earth system model—is depicted here. The model consists of submodels; each submodel has parameters and forcing variables as inputs and produces output, which can be input (forcing) to another sub-model. Data-driven learning approaches can be helpful in various instances, as indicated by the circled numbers. For example, the circle labelled 2 represents hybrid modelling. See the text for more detail. ML, machine learning.

(3) Analysis of model–observation mismatch

See Fig. 3 (circle 3). Deviations of a physical model from observations can be perceived as imperfect knowledge causing model error, assuming no observational biases. Machine learning can help to identify, visualize and understand the patterns of model error, which allows us also to correct model outputs accordingly. For example, machine learning can extract patterns from data automatically and identify those which are not explicitly represented in the physical model. This approach helps to improve the physical model and theory. In practice, it can also serve to correct the model bias of dynamic variables, or it can facilitate improved downscaling to finer spatial scales compared to tedious and ad hoc hand-designed approaches[71,72].

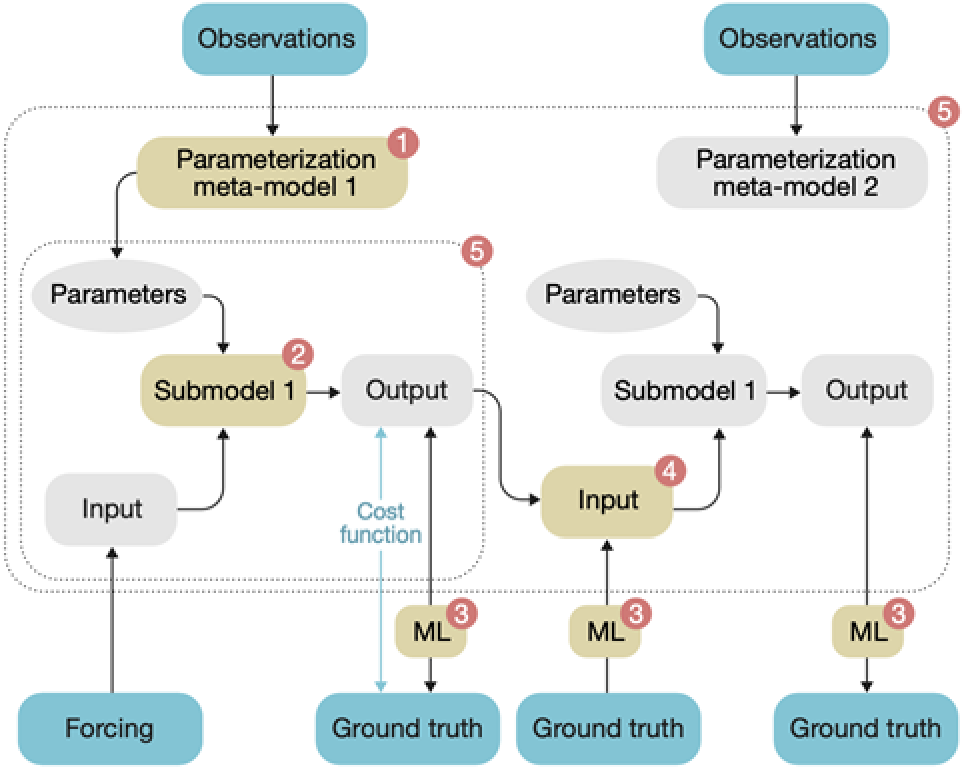


Fig. 3 | Linkages between physical models and machine learning. An abstraction of a part of a physical system—for example, an Earth system model—is depicted here. The model consists of submodels; each submodel has parameters and forcing variables as inputs and produces output, which can be input (forcing) to another sub-model. Data-driven learning approaches can be helpful in various instances, as indicated by the circled numbers. For example, the circle labelled 2 represents hybrid modelling. See the text for more detail. ML, machine learning.

(4) Constraining submodels

See Fig. 3 (circle 4). One can drive a submodel with the output from a machine learning algorithm, instead of another (potentially biased) submodel in an offline simulation. This helps to disentangle model error originating from the submodule of interest from errors of coupled submodules. As a consequence, this simplifies and reduces biases and uncertainties in model parameter calibration or the assimilation of observed system state variables.

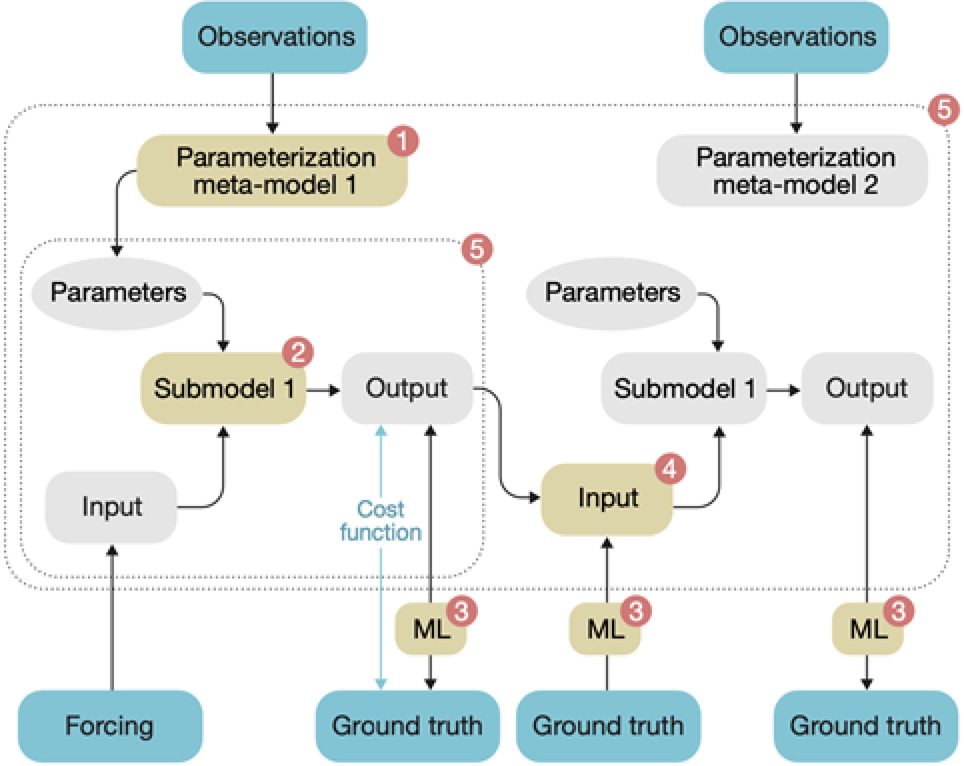
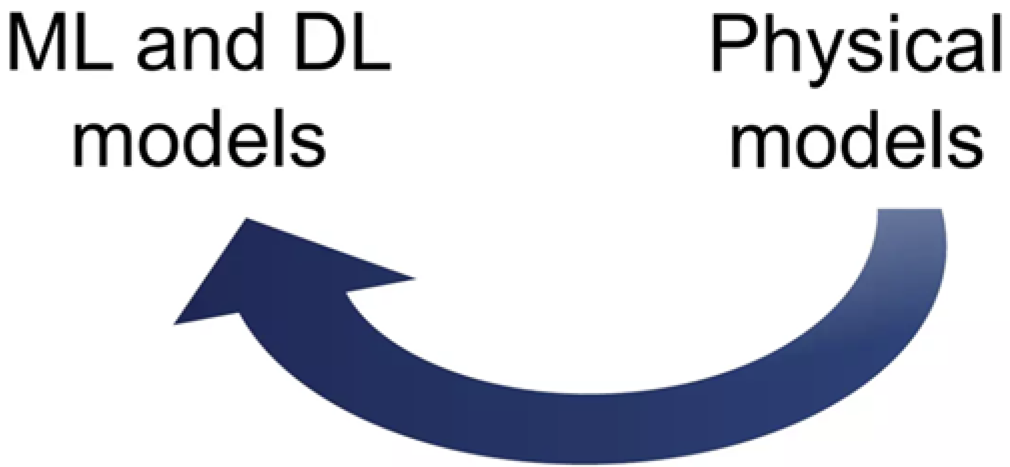


Fig. 3 | Linkages between physical models and machine learning. An abstraction of a part of a physical system—for example, an Earth system model—is depicted here. The model consists of submodels; each submodel has parameters and forcing variables as inputs and produces output, which can be input (forcing) to another sub-model. Data-driven learning approaches can be helpful in various instances, as indicated by the circled numbers. For example, the circle labelled 2 represents hybrid modelling. See the text for more detail. ML, machine learning.

(5) Surrogate modelling or emulation

See Fig. 3 (circle 5). Emulation of the full (or specific parts of) a physical model can be useful for computational efficiency and tractability reasons. Machine learning emulators, once trained, can achieve simulations orders of magnitude faster than the original physical model without sacrificing much accuracy. This allows for fast sensitivity analysis, model parameter calibration, and derivation of confidence intervals for the estimates. For example, machine learning emulators are used to replace computationally expensive, physics-based radiative-transfer models of the interactions between radiation, vegetation and atmosphere[57,73,74], which are critical for the interpretation and assimilation



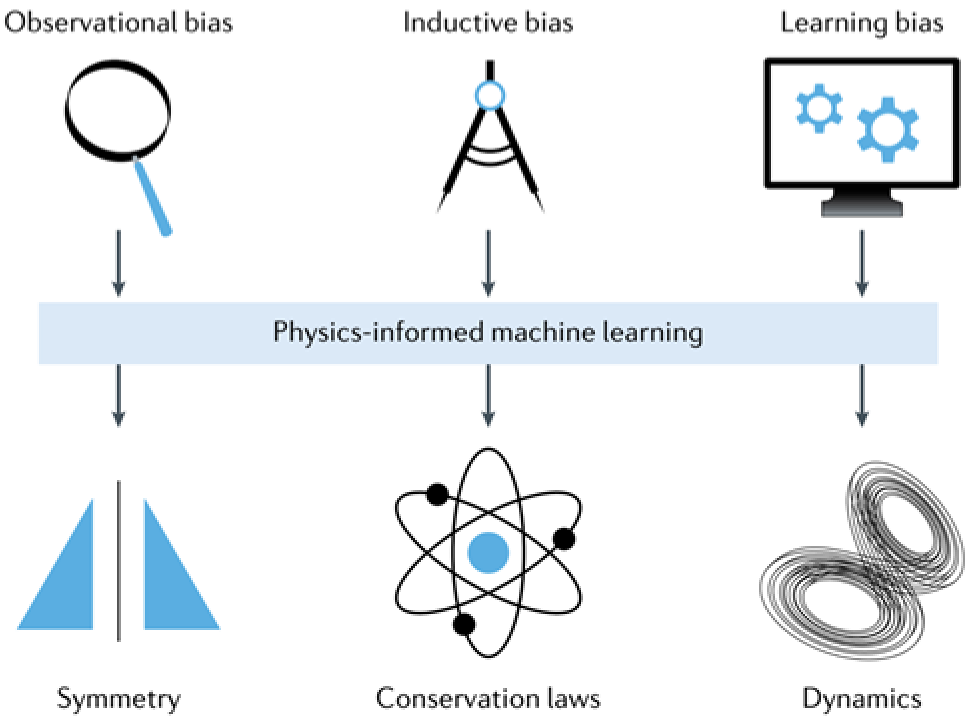
ML and DL models

Physical models

Domain knowledge can guide/optimize the pure data-driven methods

- design the architecture
- constrain the cost (or reward) function
- physically based data augmentation: expansion of the data set for undersampled regions

What is a physics-informed solution?



Making a learning algorithm physics-informed amounts to introducing appropriate observational, inductive or learning biases that can steer the learning process towards identifying physically consistent solutions

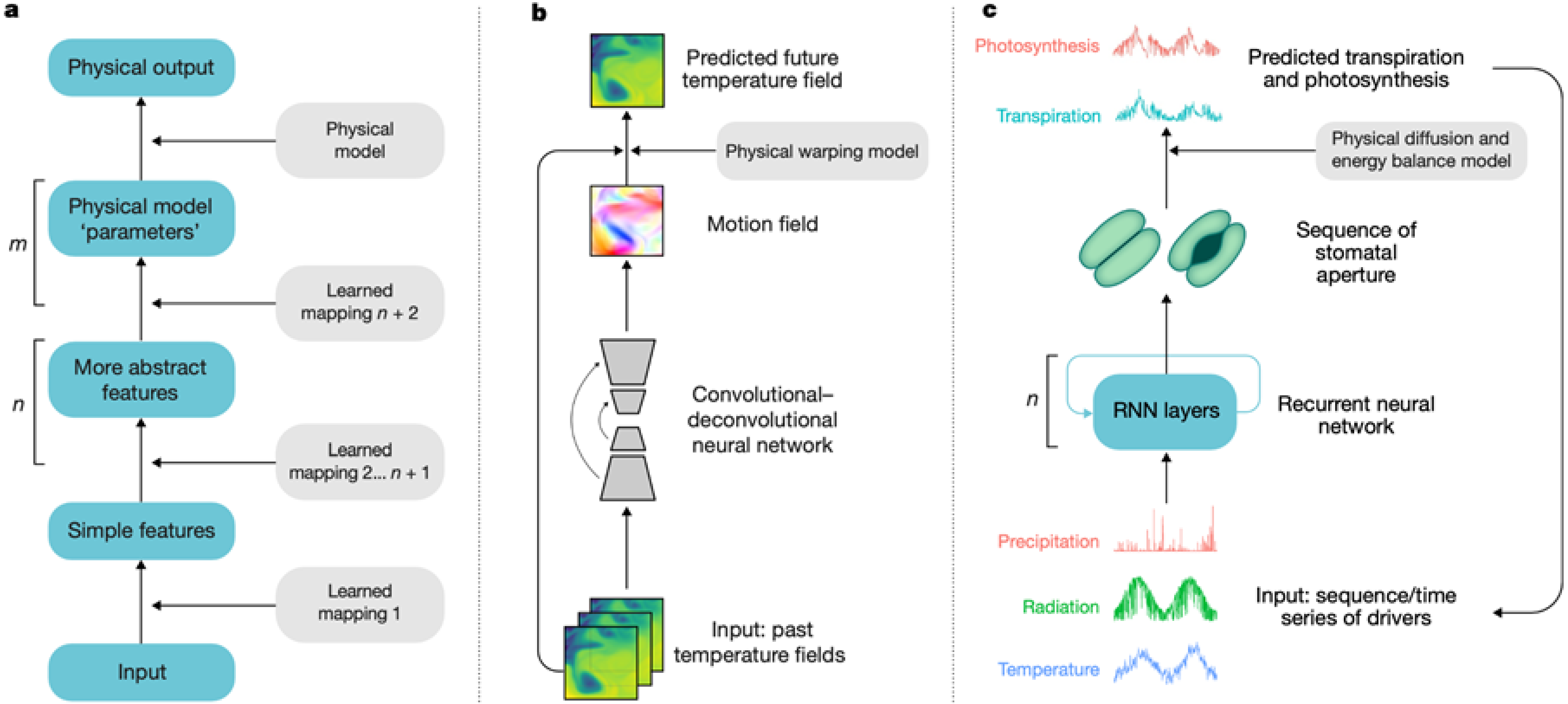

a
Physical output
Physical model
m
Physical model 'parameters'
Learned mapping n + 2
n
More abstract features
Learned mapping 2... n + 1
Simple features
Learned mapping 1
Input
b
Predicted future temperature field
Physical warping model
Motion field
Convolutional–deconvolutional neural network
Input: past temperature fields
c
Photosynthesis
Predicted transpiration and photosynthesis
Transpiration
Physical diffusion and energy balance model
Sequence of stomatal aperture
n
RNN layers
Recurrent neural network
Precipitation
Radiation
Input: sequence/time series of drivers
Temperature

Fig. 4 | Interpretation of hybrid modelling as deepening a deep learning architecture by adding one or several physical layers after the multilayer neural network to make the model more physically realistic. a, The multilayer neural network, with *n* the number of neural layers and *m* the number of physical layers. **b** and **c** are concrete examples of hybrid modelling (circle 2 in Fig. 3). **b**, Prediction of sea-surface temperatures, where a motion field of the water is learned with a convolutional–deconvolutional neural network, and the motion field is further processed with a physical model to predict future states. Adapted from figure 1 of de Bezenac et al.[68]. **c**, A biological regulation process (opening of the stomatal 'valves' controlling water vapour flux from the leaves) is modelled with a recurrent neural network. Then a physical diffusion model is used to estimate transpiration, which in turn influences some of the drivers, such as soil moisture. The basic scheme in **a** is inspired by figure 1.5 in Goodfellow et al.[98] and redrawn.