MACHINE LEARNING ON FUNCTION SPACES

NEURAL OPERATORS

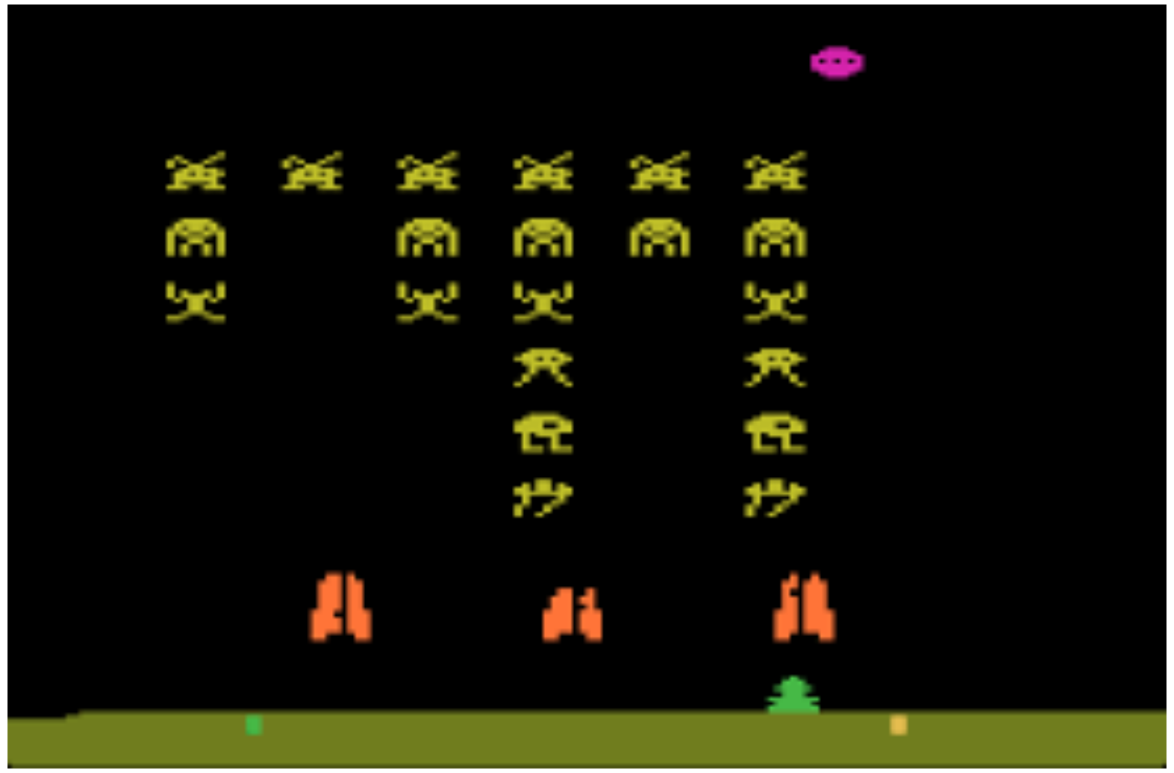
Kamyar Azizzadenesheli
Research Staff, NVIDIA

ICML
International Conference
On Machine Learning

TRADITIONAL DEEP LEARNING

Text, speech, image, etc.

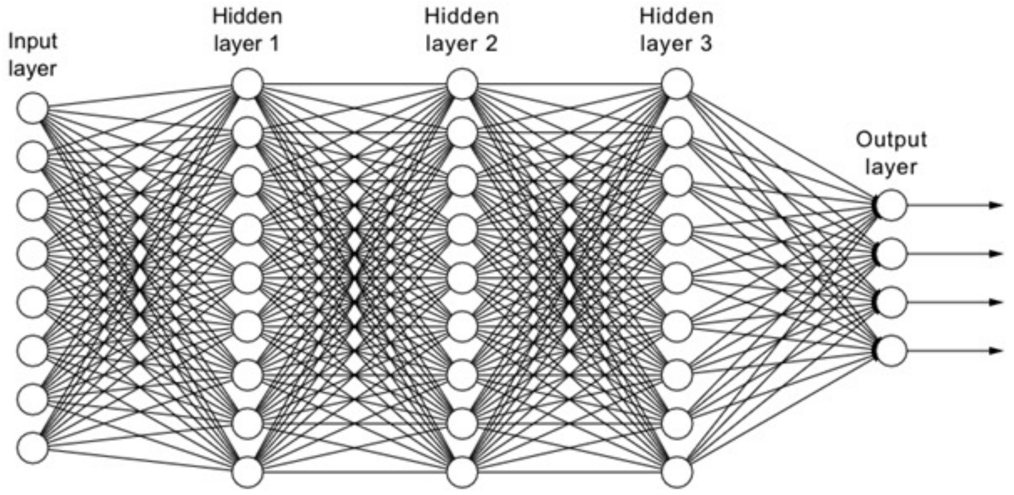
Data: Finite dimensional objects



Despite the rapid advances in AI, computer vision (CV) is still challenged in matching the precision of human perception. The training data here is as important as algorithms. The more accurate the input data annotation, the more effective the model prediction.

How do we annotate data, though? There are multiple ways to go with this one, but it all depends on your use case. For the purposes of this article, we'll take a deeper dive into bounding boxes as one of the most extensively used annotation techniques. Moving forward, we'll walk you through the following:

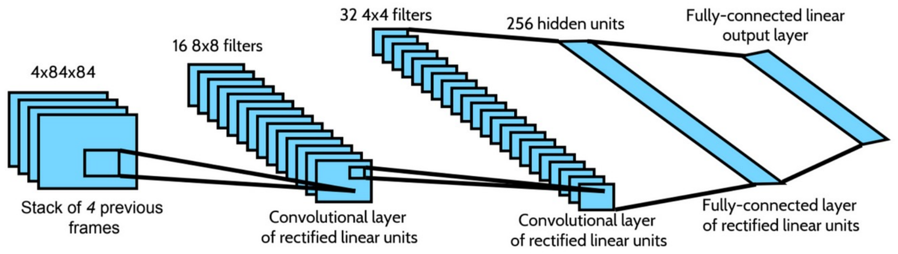
TRADITIONAL DEEP LEARNING

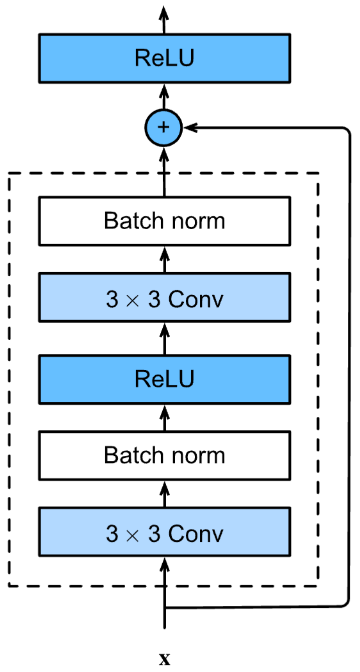


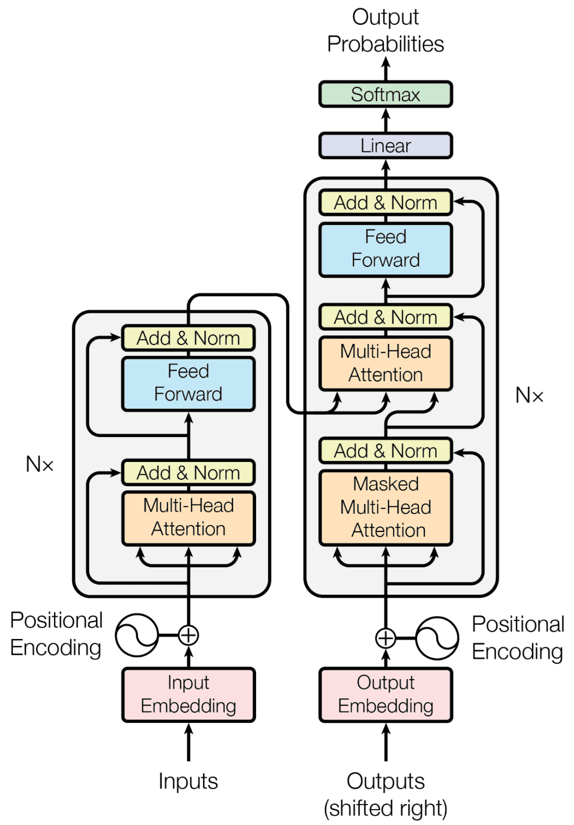
Text, speech, image, etc.

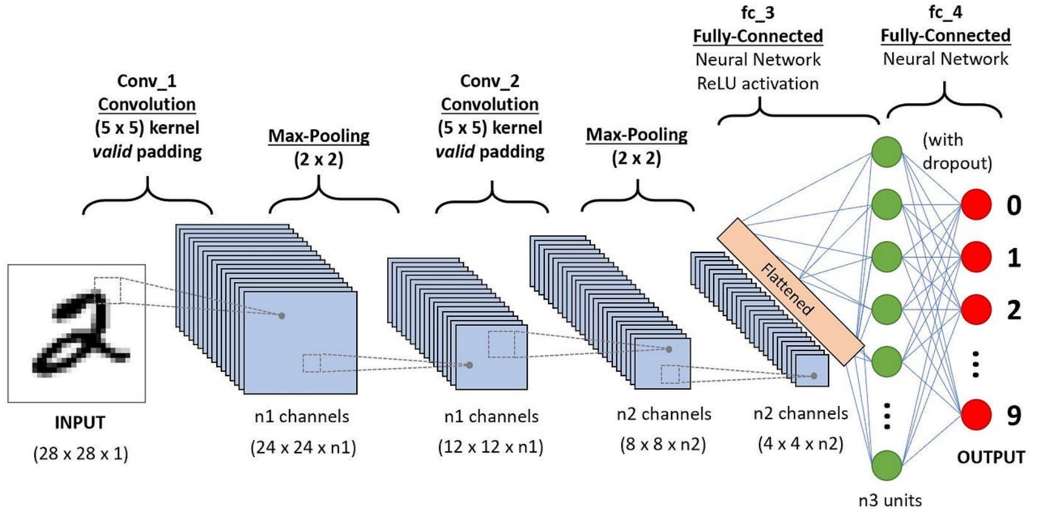
Paradigm: Neural networks

Architectures: CNN, AlexNet, LSTM, ResNet, UNet, EfficientNet, MobileNet, Transformer, ViT.



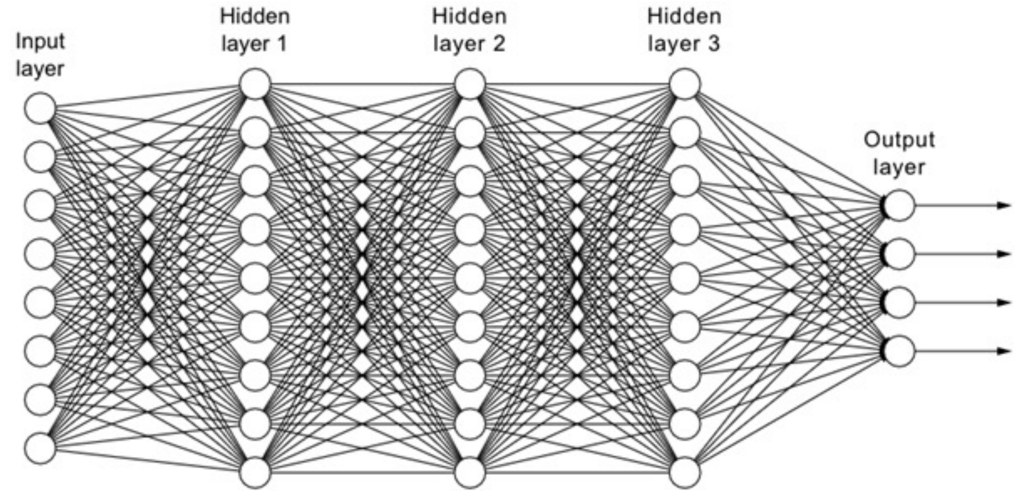






TRADITIONAL DEEP LEARNING

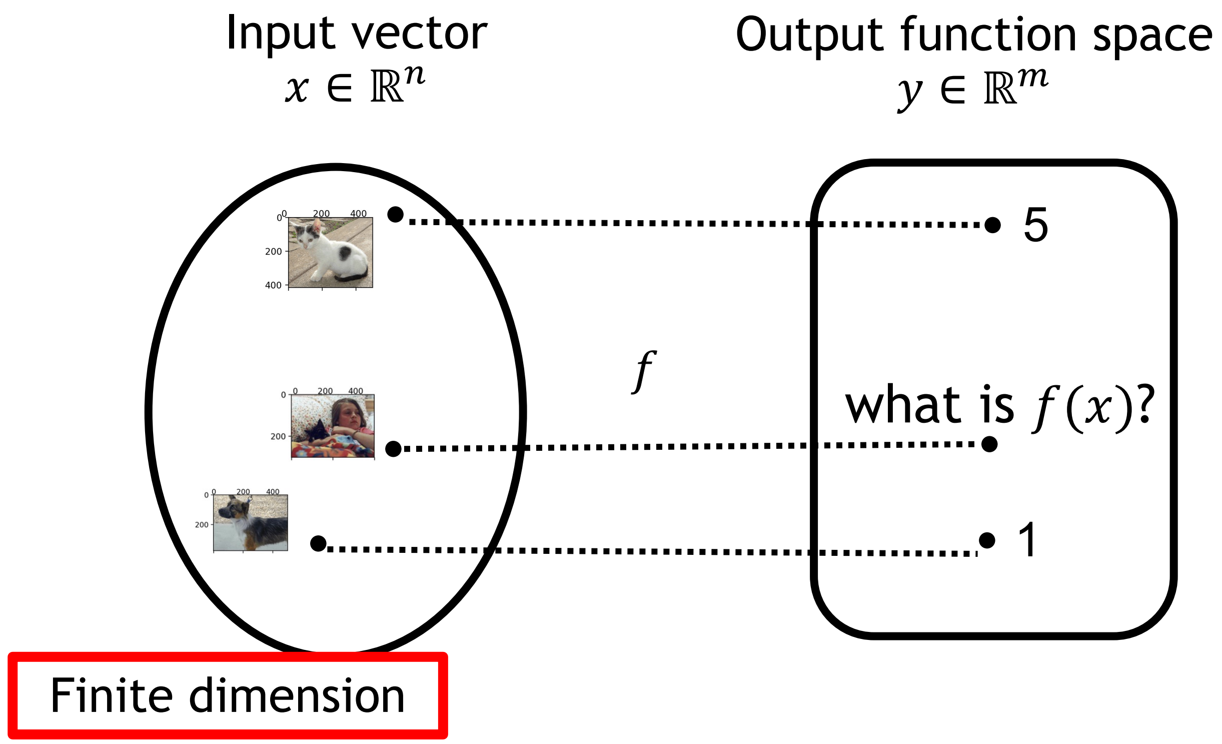
Text, speech, image, etc.



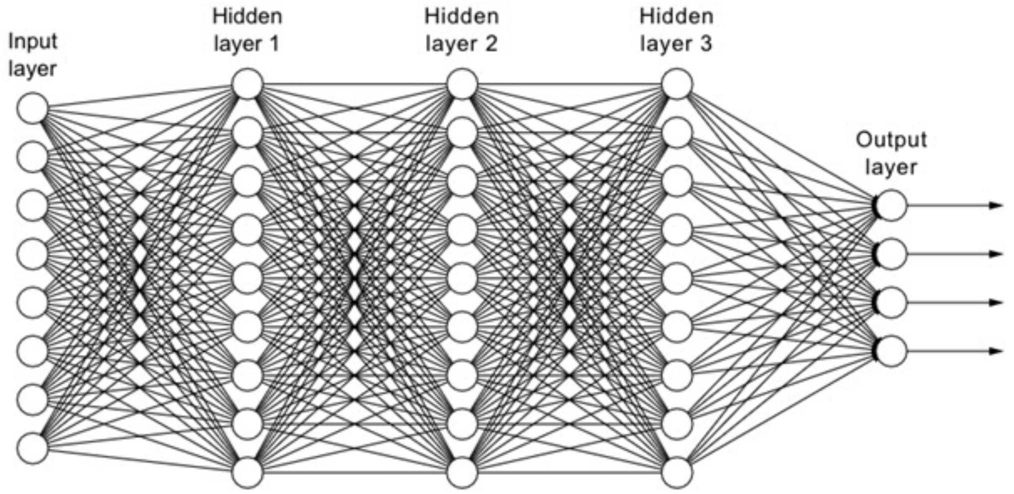
Paradigm: Neural networks

Conventional machine learning practice,

Learning a function between finite dimensional spaces, $f_\theta: \mathbb{R}^n \to \mathbb{R}^m$



TRADITIONAL DEEP LEARNING



Text, speech, image, etc.

Paradigm: Neural networks

Architectures: CNN, AlexNet, LSTM, ResNet, UNet, EfficientNet, MobileNet, Transformer, ViT.

Datasets: UCI-dataset, MNITS, ImageNet, Common Crawl.

Problem setup: input/output, supervision, Image/label, laws, metric, loss,

- Inception score,
- FID
- RMSE
- L1
- CLIP
- …

With the main focus in CV and language

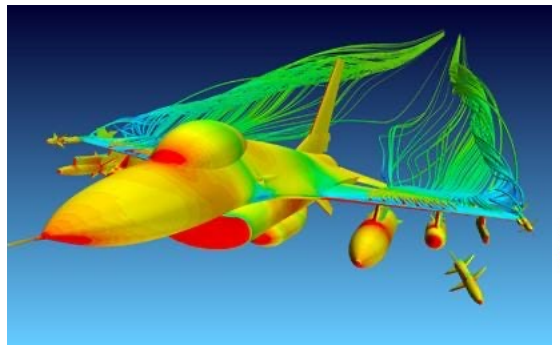
Applications: Recommendation systems, content generation, self driving cars, knowledge, search, etc.

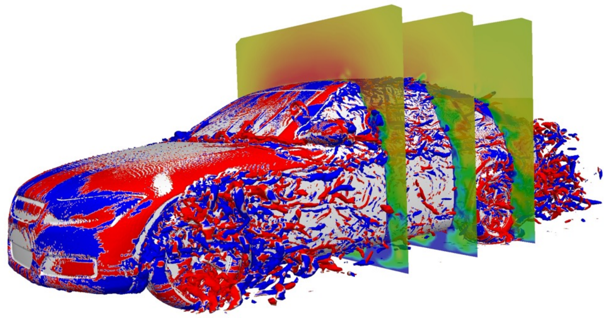
A series of great developments for domains such as images, languages, …

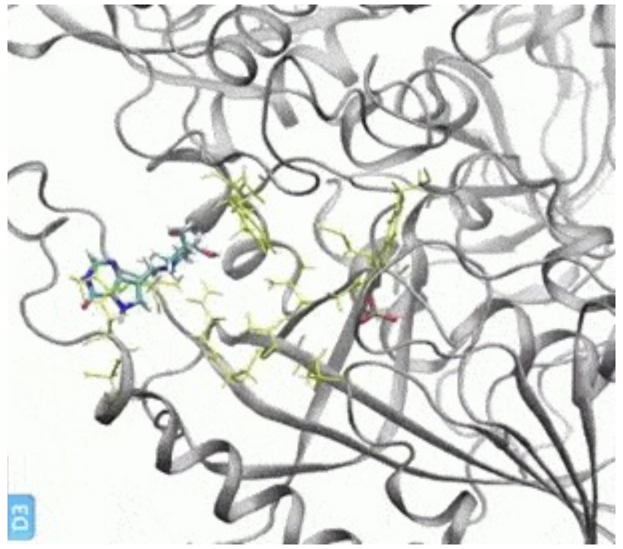
COMMUTATING

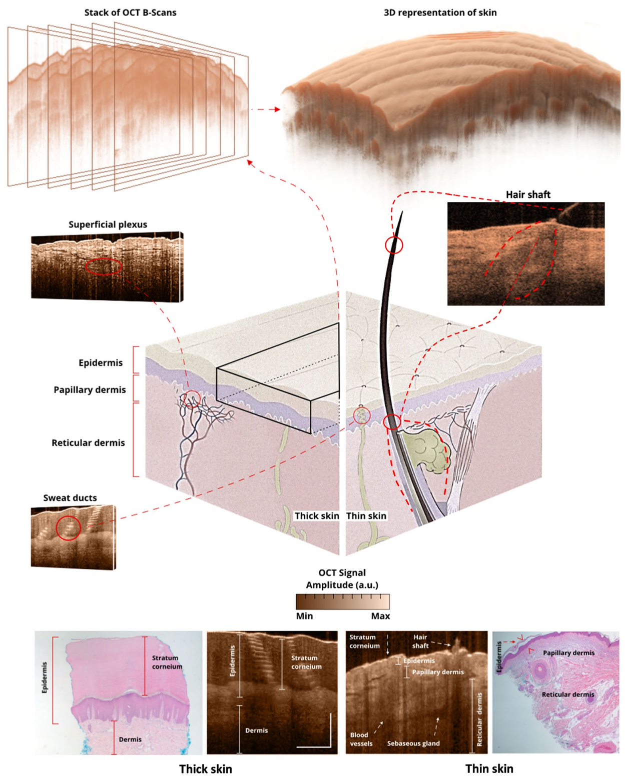
Aeronautics & Astronautics Department
Anthropology
Applied Physics Department
Biochemistry Department
Bioengineering Department
Biology Department
Biology, Developmental
Biomedical Informatics
Business,
Chemical and Systems Biology
Chemical Engineering Department
Chemistry Department
Civil & Environmental Engineering Department
Computer Science Department
Developmental Biology Department
Dermatology Department
Earth and Planetary Sciences
Earth System Science
Economics Department
Electrical Engineering Department
Energy Science & Engineering
Genetics Department
Geophysics
Management Science & Engineering Department
Materials Science & Engineering Department
Mechanical Engineering Department
Mathematics Department
Medicine Department
Microbiology & Immunology Department
Molecular & Cellular Physiology Department
Neurobiology Department
Neurology & Neurological Sciences Department
Neurosurgery Department
Obstetrics and Gynecology Department
Oceans Department
Ophthalmology Department
Physics Department
Radiation Oncology Department
Radiology Department
Stanford Doerr School of Sustainability
Structural Biology Department

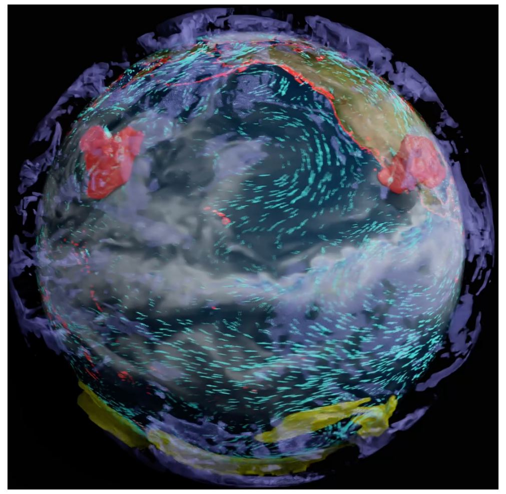
Text, speech, and image what else?

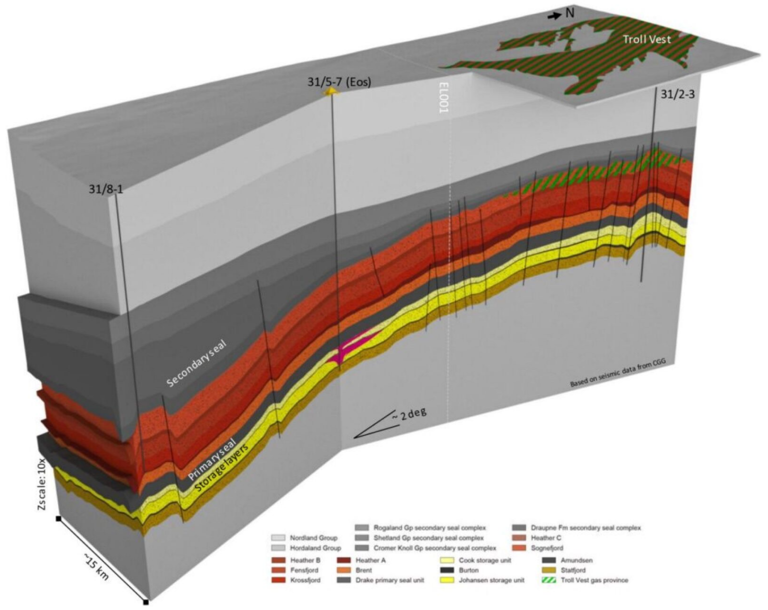


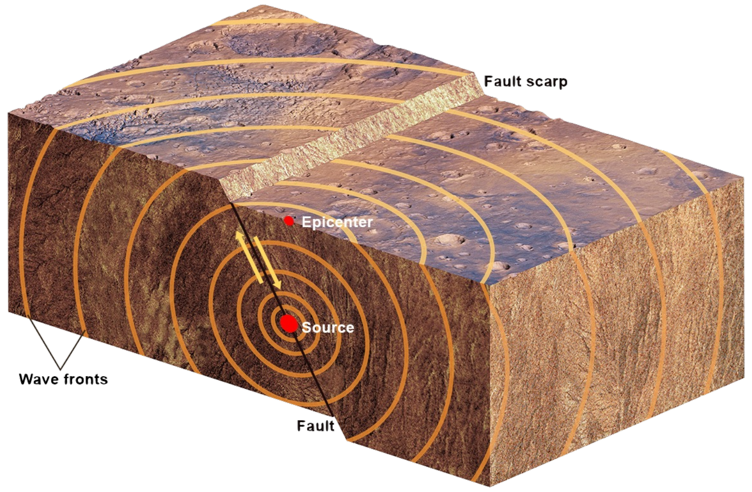












Now project it to the broader industry

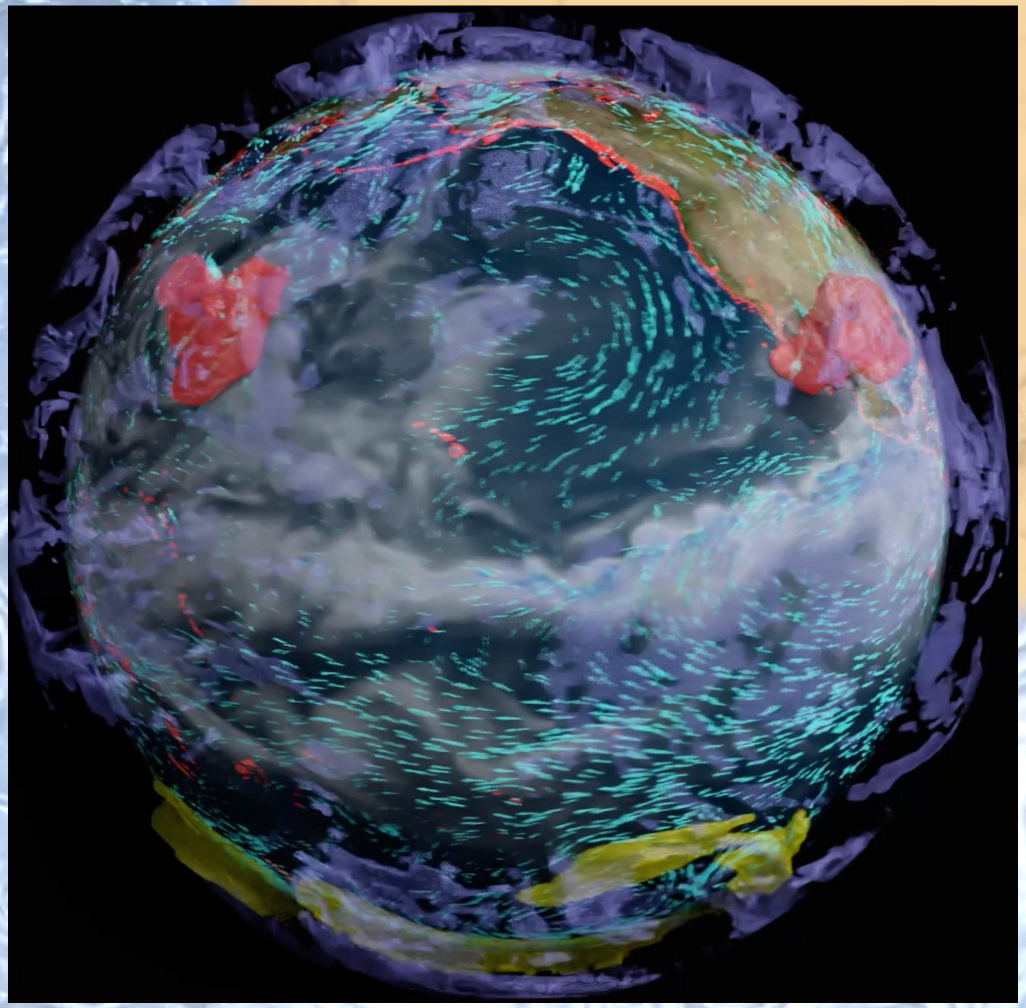
REAL WORLD DATA

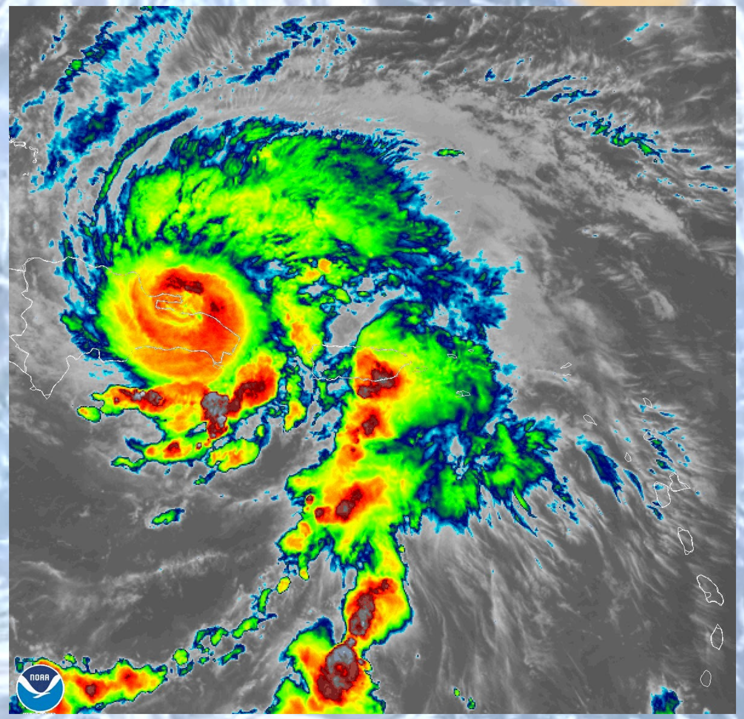
Jun 2006

Weather, ocean, climate

Domain is function → data is function, e.g.,

- Given temperature and wind today, forecast temperature and wind tomorrow





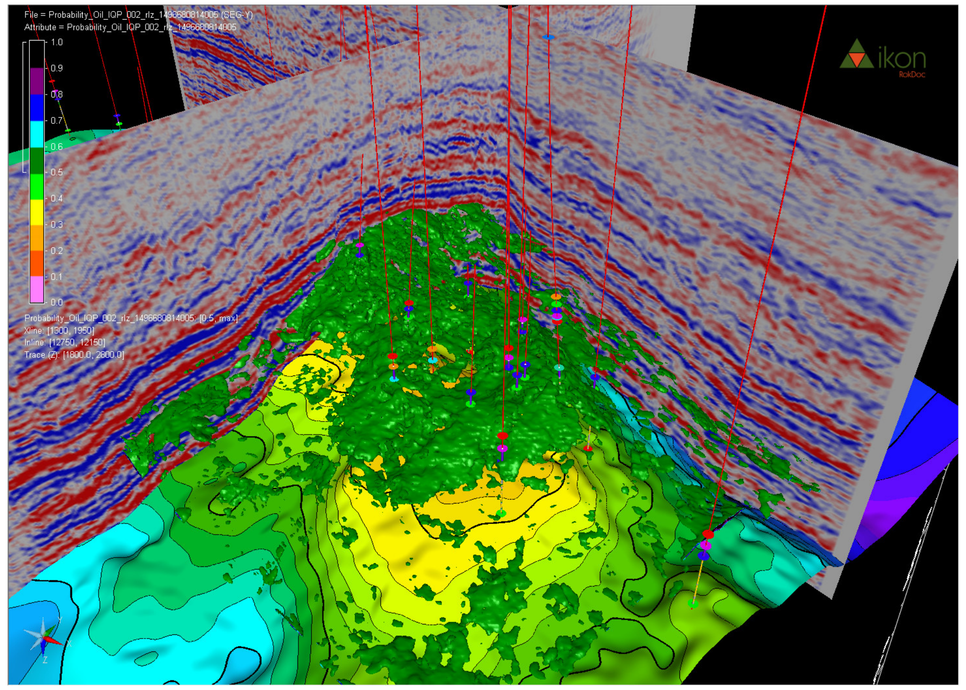
NASA NVIDIA Omniverse

REAL WORLD DATA

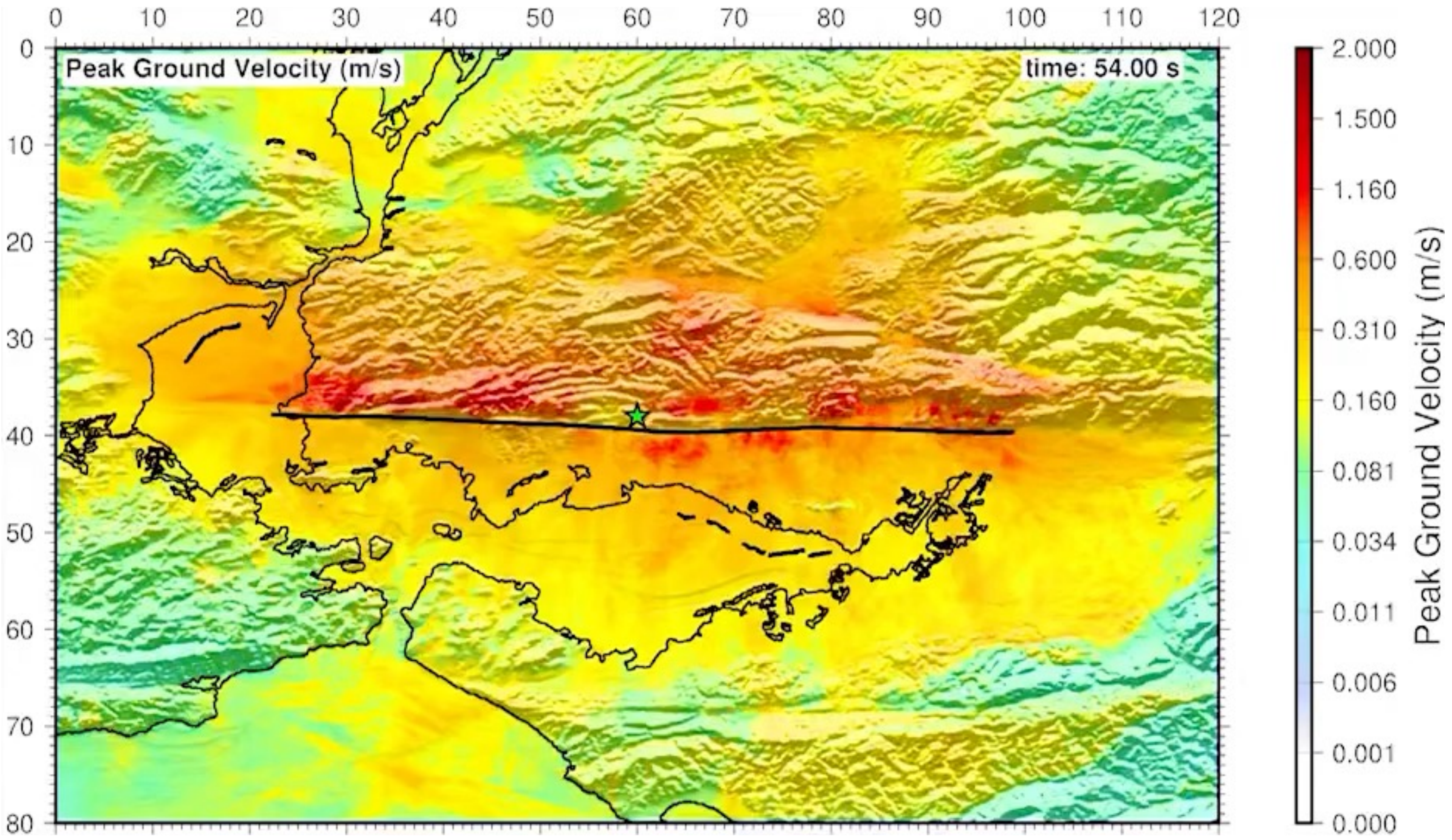
Geophysics, seismology, earth systems, ocean, climate

Domain is function → data is function, e.g.,

- Given the velocity field in subsurface, how the wave propagates and cause earthquakes



Los Angeles basin



San Francisco

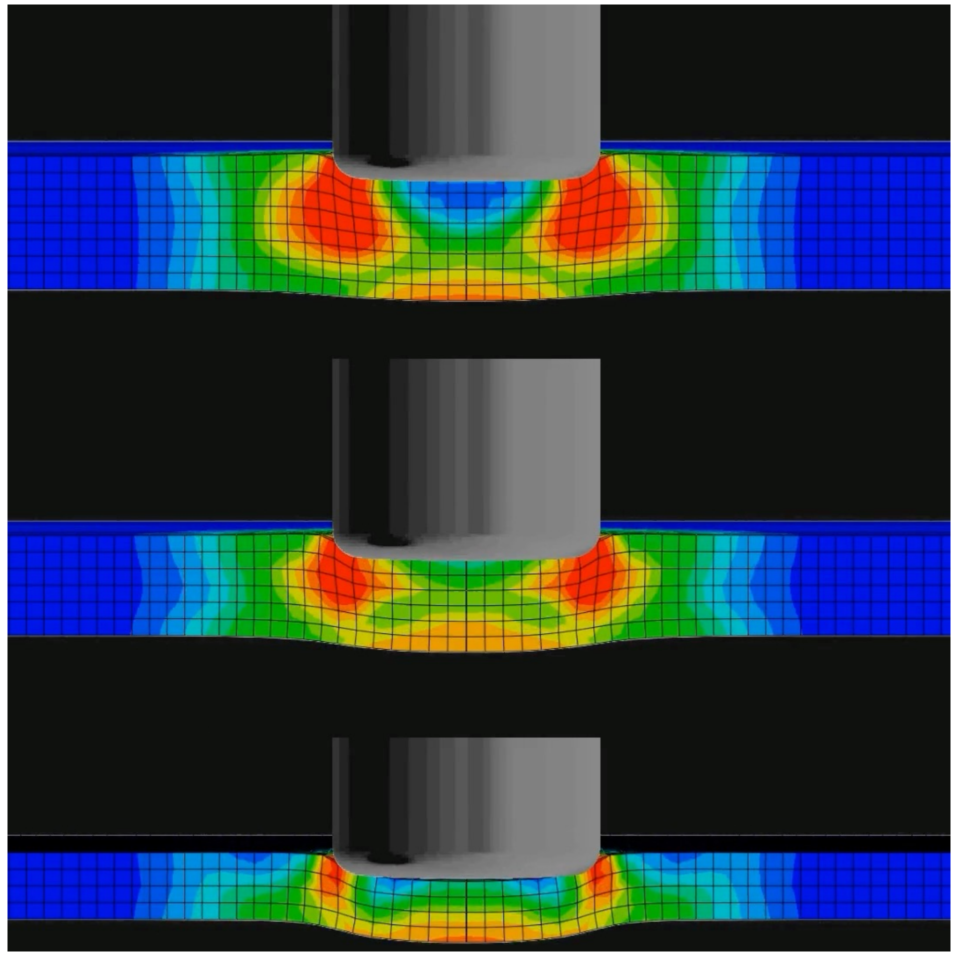
Broadband (0--5 Hz) fully deterministic 3D ground-motion simulations of a magnitude 7.0 Hayward fault earthquake: Comparison with empirical ground-motion models and 3D path and site effects from source normalized intensities

REAL WORLD DATA

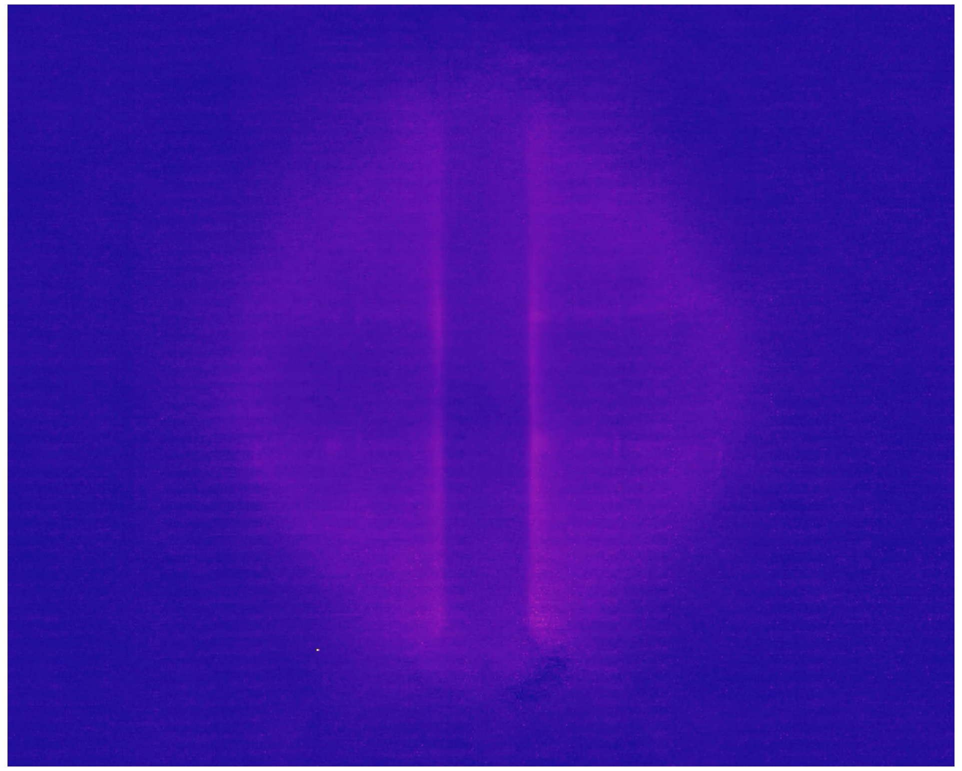
Geophysics, seismology, earth systems, ocean, climate

Domain is function → data is function, e.g.,

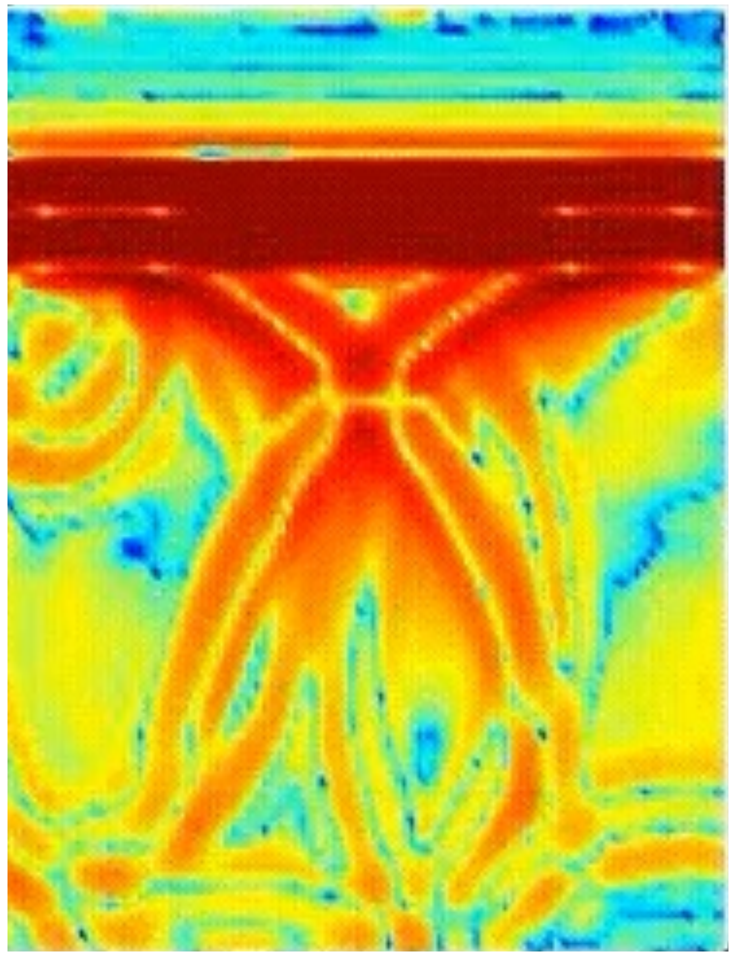
- Material deformation, plasma evolution, tissue imaging



Deformation



Fusion



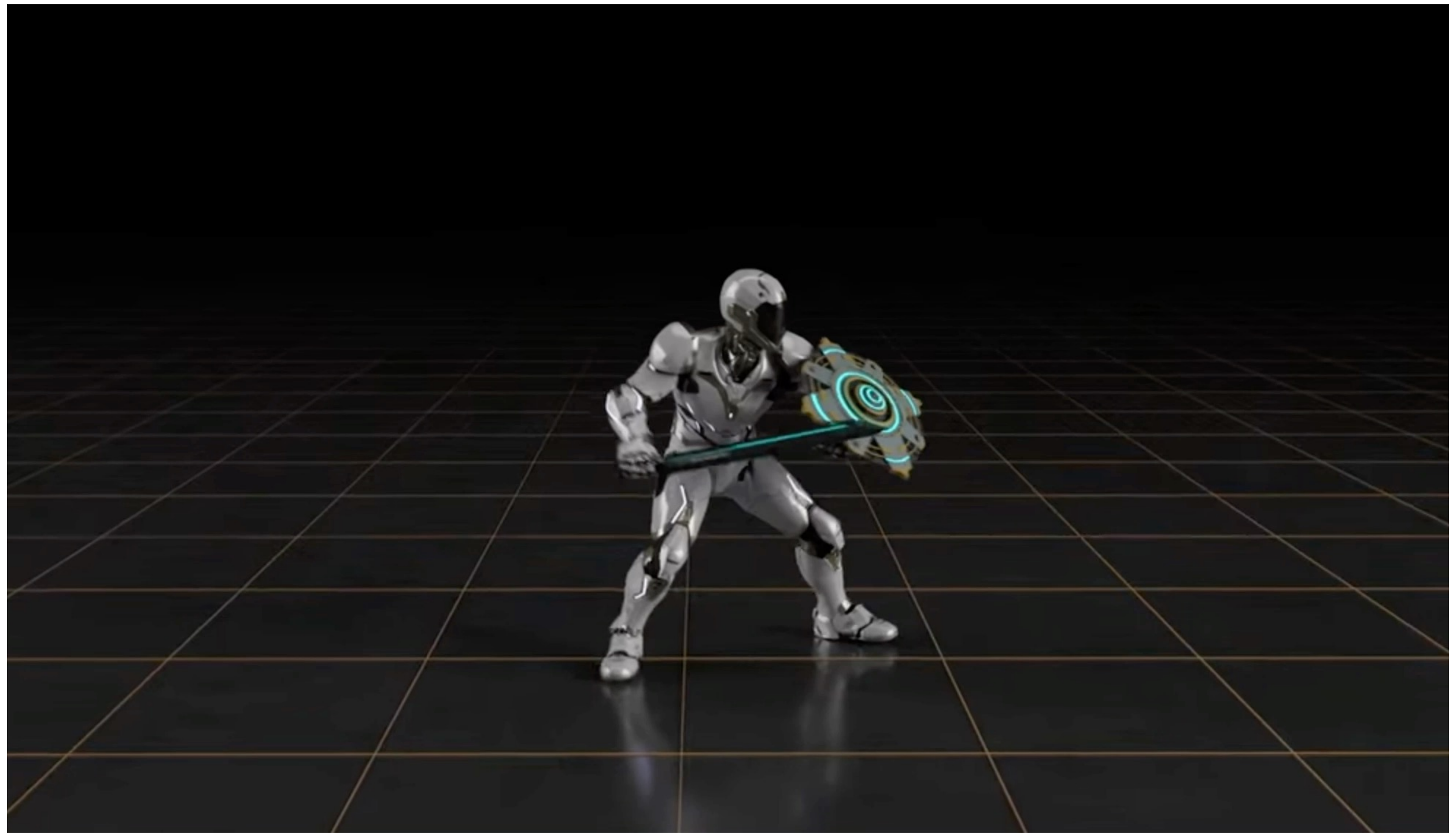
Ultrasound

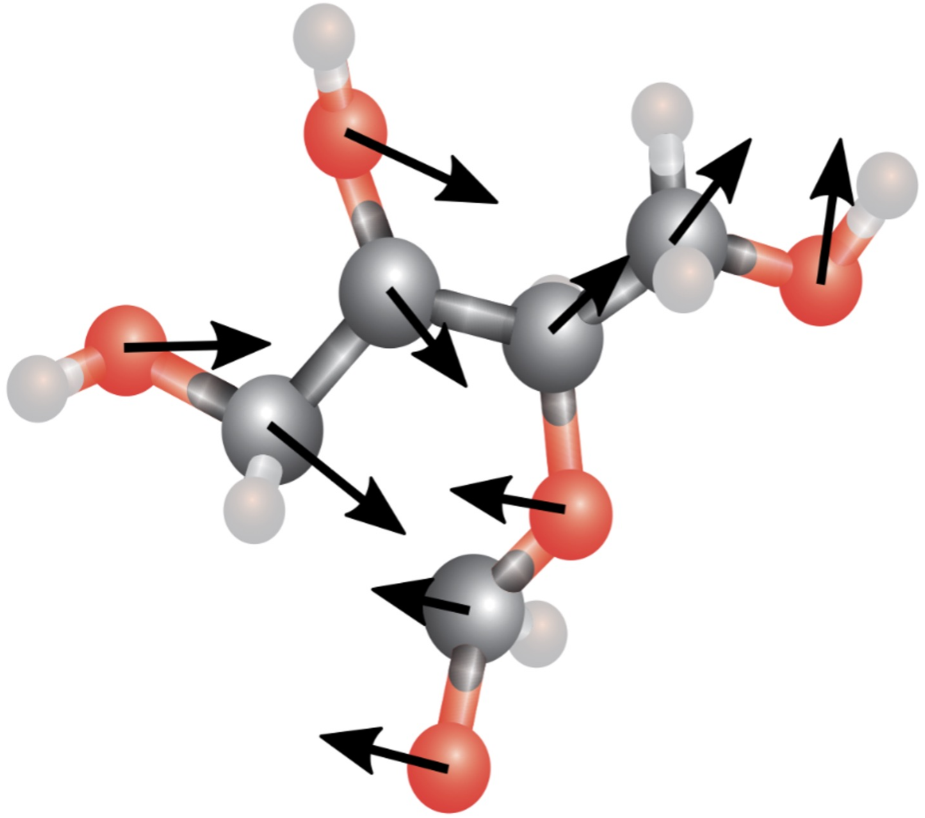
REAL WORLD DATA

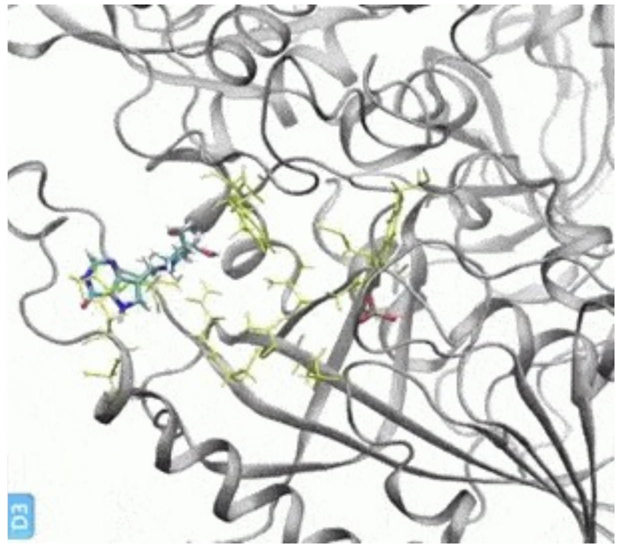
Geophysics, seismology, earth systems, ocean, climate

Domain is function → data is function, e.g.,

- Molecular dynamics, protein engineering, humanoid and robotics





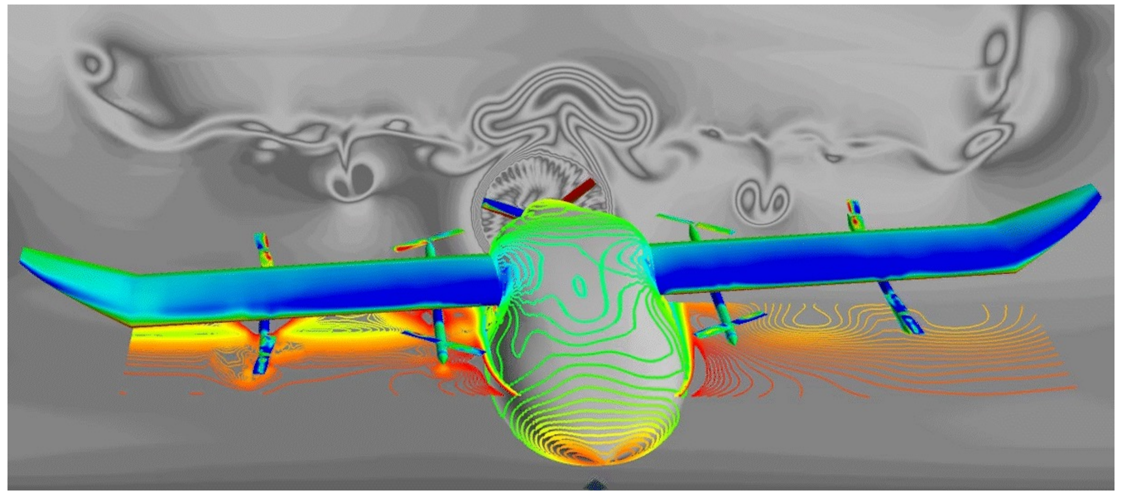


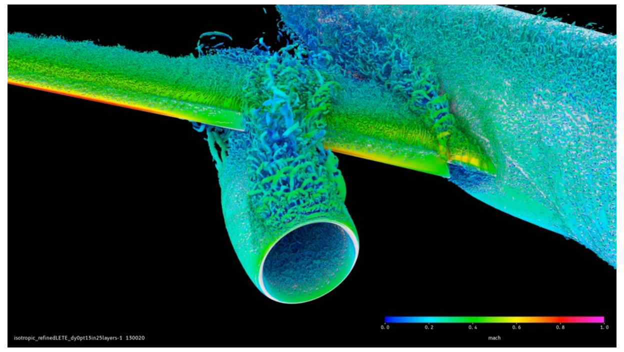
REAL WORLD DATA

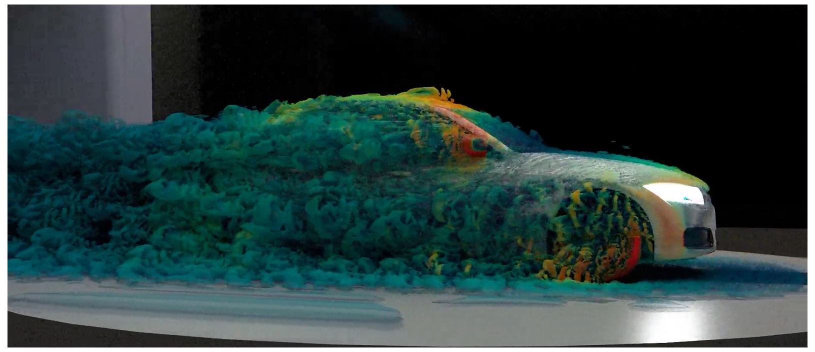
Automative industry, aviation industry

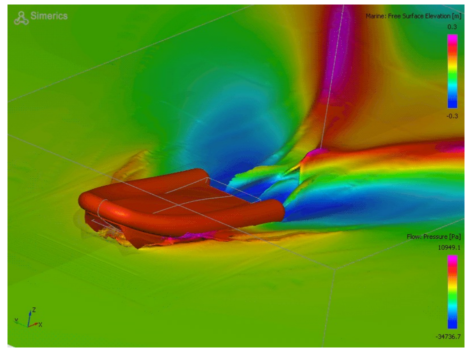
Domain is function → data is function, e.g.,

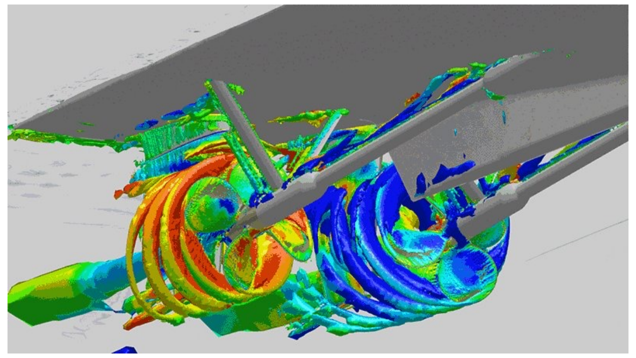
- Fluid dynamics dynamics











REAL WORLD DATA

Natural Sciences and engineering

Domain is function → data is function, e.g.,

- Biochemistry
- Carbon dioxide deposit
- Climate mitigation
- Water reservoir
- Medicine
- Sustainability
- …

Data is function, it is visualized, but these are not

- Pictures
- Sequential time series.

TRADITIONAL METHODS

Natural Sciences and engineering

How are we used to tackling these computational problems?

Model the phenomena using differential and algebraic equations (e.g., PDEs)

Schrödinger, Darcy, Maxwell, Navier-Stokes, fluid dynamics, laws of thermodynamics, Helmholtz, etc.,

$$\begin{aligned} \nabla \cdot \mathbf{E} &= \frac{\rho}{\varepsilon_0} \\ \nabla \cdot \mathbf{B} &= 0 \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{B} &= \mu_0 \left(\mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right) \end{aligned}$$

$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = \left[-\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x,t) \right] \Psi(x,t)$$

$$\rho \ddot{\mathbf{u}} = \mathbf{f} + (\lambda + 2\mu) \nabla (\nabla \cdot \mathbf{u}) - \mu \nabla \times (\nabla \times \mathbf{u}).$$

$$\begin{aligned} \partial_t u(x,t) + u(x,t) \cdot \nabla u(x,t) + \nabla p(x,t) &= \nu \Delta u(x,t) + f(x), \\ \nabla \cdot u(x,t) &= 0, \\ u(x,0) &= u_0(x), \end{aligned}$$

Develop suitable conventional solvers to solve these equations at certain ***resolutions*,**

Finite difference, elements, volume methods, spectral, etc.

TRADITIONAL METHODS

Natural Sciences and engineering

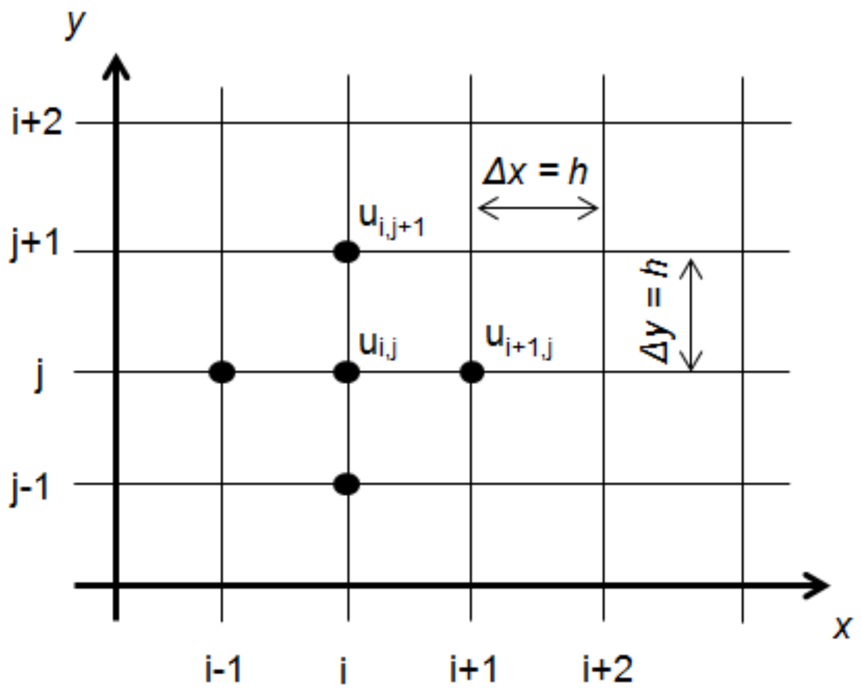
Develop conventional methods and solvers for solving these equations at certain ***resolutions,***

Finite difference, elements, volume methods, spectral, etc.

E.g., in Darcy's flow

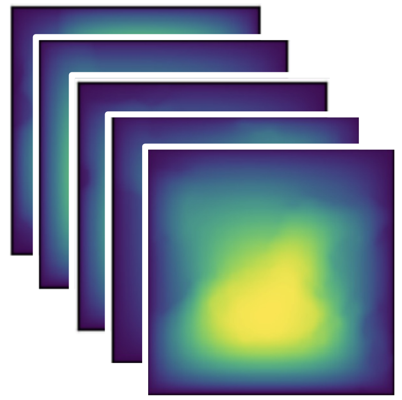
$$-\nabla.\big(a(x)\nabla u(x)\big) = f(x)$$

$$\frac{u_{i+1,j} - u_{ij}}{\Delta x}$$



Solution operator

$\mathcal{G}$



Input function: diffusion coefficients, $a's$

Output functions: solutions, $u's$

Finer discretization → more accurate solution, and more accurate method,

Also more compute

TRADITIONAL METHODS

Modeling is hard to impossible, computation is massive

Why not continue this direction and hand design solution operator?

Modeling real world using equations is hard, e.g., weather forecast,

Parametrizing behavior of clouds, ocean waves, mountains, etc. For some, we don't have much idea how to do them.

$$\frac{\partial U}{\partial t} + \frac{1}{a\cos^2\theta}\left\{U\frac{\partial U}{\partial \lambda} + V\cos\theta\frac{\partial U}{\partial \theta}\right\} + \dot{\eta}\frac{\partial U}{\partial \eta} - fV + \frac{1}{a}\left\{\frac{\partial \phi}{\partial \lambda} + R_{\mathrm{dry}}T_{\mathrm{v}}\frac{\partial}{\partial \lambda}(\ln p)\right\} = P_U + K_U$$

$$\frac{\partial V}{\partial t} + \frac{1}{a\cos^2\theta}\left\{U\frac{\partial V}{\partial \lambda} + V\cos\theta\frac{\partial V}{\partial \theta} + \sin\theta(U^2+V^2)\right\} + \dot{\eta}\frac{\partial V}{\partial \eta}$$
$$+ fU + \frac{\cos\theta}{a}\left\{\frac{\partial \phi}{\partial \theta} + R_{\mathrm{dry}}T_{\mathrm{v}}\frac{\partial}{\partial \theta}(\ln p)\right\} = P_V + K_V$$

$$\frac{\mathrm{d}\vec{V}}{\mathrm{dt}} = -\alpha\vec{\nabla}p - \vec{\nabla}\Phi + \vec{F} - 2\Omega\times\vec{V}$$

$$\frac{\partial \rho}{\partial t} = -\vec{\nabla}\cdot\left(\rho\vec{V}\right)$$

$$p\alpha = \mathrm{RT}$$

$$Q = C_p\frac{\mathrm{dT}}{\mathrm{dt}} - \alpha\frac{\mathrm{dp}}{\mathrm{dt}}$$

$$\frac{\partial \rho q}{\partial t} = -\vec{\nabla}\cdot\left(\rho\vec{V}q\right) + \rho(E-C)$$

$$\delta_t U + A(U) - fV + \frac{1}{a}\left\{\frac{\partial \phi}{\partial \lambda} + R_{\mathrm{dry}}T_{\mathrm{v}}\frac{\partial}{\partial \lambda}(\ln p)\right\}$$
$$= P_U + K_U - \frac{\beta}{2a}\Delta_{tt}\left\{[\gamma]\frac{\partial T}{\partial \lambda} + R_{\mathrm{dry}}T^{\mathrm{ref}}\frac{\partial}{\partial \lambda}(\ln p_{\mathrm{s}})\right\}$$

$$\delta_t V + A(V) + \frac{\sin\theta}{a\cos^2\theta}(U^2+V^2) + fU + \frac{\cos\theta}{a}\left\{\frac{\partial \phi}{\partial \theta} + R_{\mathrm{dry}}T_{\mathrm{v}}\frac{\partial}{\partial \theta}(\ln p)\right\}$$
$$= P_V + K_V - \frac{\beta\cos\theta}{2a}\Delta_{tt}\left([\gamma]\frac{\partial T}{\partial \theta} + R_{\mathrm{dry}}T^{\mathrm{ref}}\frac{\partial}{\partial \theta}(\ln p_{\mathrm{s}})\right)$$

$$\delta_t T + A(T) - \frac{\kappa T_{\mathrm{v}}\omega}{(1+(\delta-1)q)p} = P_T + K_T - \frac{\beta}{2}\Delta_{tt}([\tau]D)$$
$$\delta_t q + A(q) = P_q + K_q$$

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - w\frac{\partial u}{\partial z} + \frac{\mathrm{uv}\tan\phi}{a} - \frac{\mathrm{uw}}{a} - \frac{1}{\rho}\frac{\partial p}{\partial x} - 2\Omega(w\cos\phi - v\sin\phi) + \mathrm{Fr}_x$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - w\frac{\partial v}{\partial z} - \frac{u^2\tan\phi}{a} - \frac{\mathrm{uw}}{a} - \frac{1}{\rho}\frac{\partial p}{\partial y} - 2\Omega u\sin\phi + \mathrm{Fr}_y$$

$$\frac{\partial w}{\partial t} = -u\frac{\partial w}{\partial x} - v\frac{\partial w}{\partial y} - w\frac{\partial v}{\partial z} - \frac{u^2+v^2}{a} - \frac{1}{\rho}\frac{\partial p}{\partial z} + 2\Omega u\cos\phi - g + \mathrm{Fr}_z$$

$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} - v\frac{\partial T}{\partial y} + (\gamma - \gamma_d)w + \frac{1}{c_p}\frac{\mathrm{dH}}{\mathrm{dt}}$$

$$\frac{\partial \rho}{\partial t} = -u\frac{\partial \rho}{\partial x} + -v\frac{\partial \rho}{\partial y} - w\frac{\partial \rho}{\partial z} - \rho\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)$$

$$\frac{\partial q_v}{\partial t} = -u\frac{\partial q_v}{\partial x} - v\frac{\partial q_v}{\partial y} - w\frac{\partial q_v}{\partial z} + Q_v$$

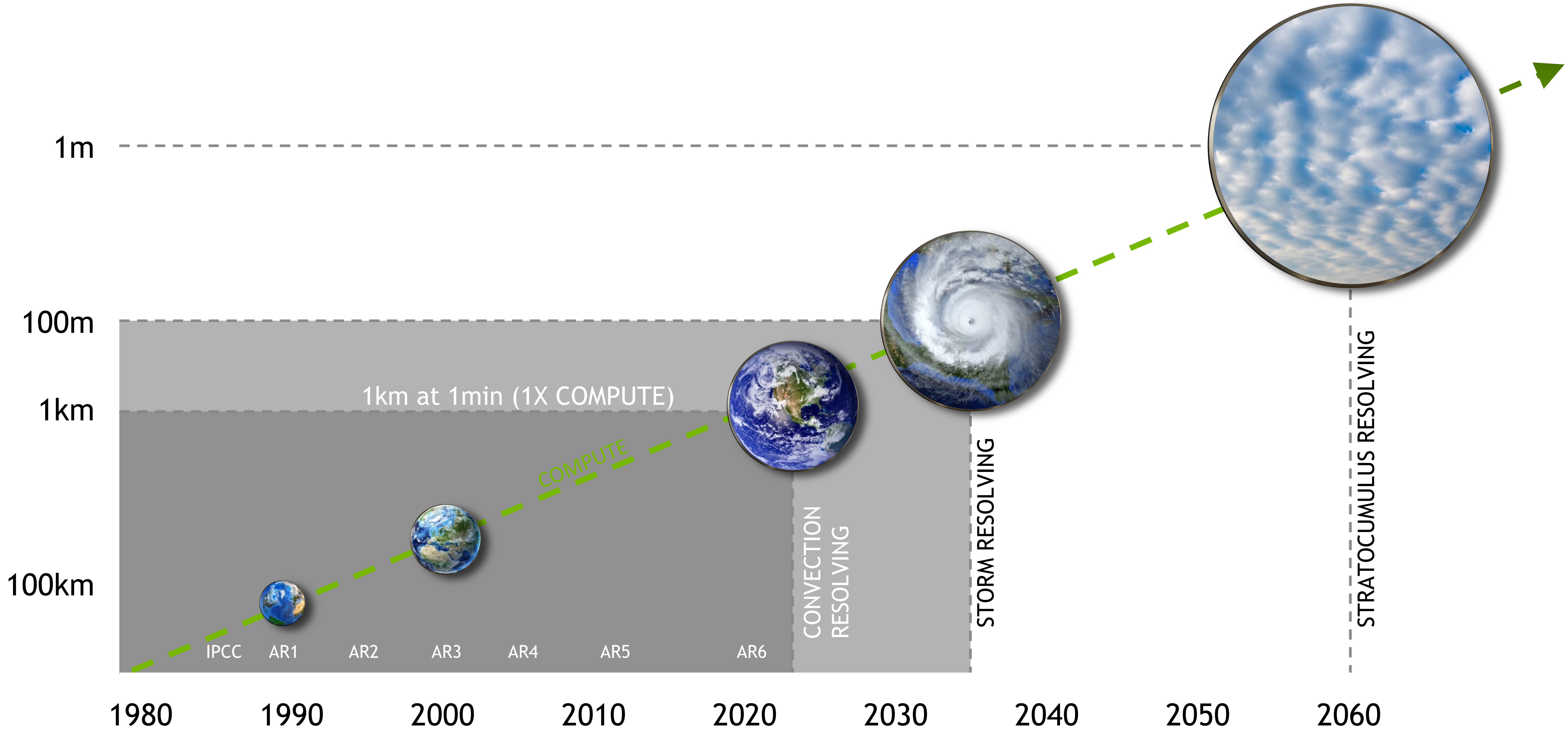
$$p\alpha = \mathrm{RT}$$

...

TRADITIONAL METHODS

Computational constraints limit model resolution

Reasonable solution operator requires high resolution → much more computes



TRADITIONAL METHODS

Modeling is hard to impossible, computation is massive

Why not continue this direction and hand design solution operator?

Human digestible modeling is challenging and limiting,

Modeling unknown physics

Error due to parameterization

Differentiability for invers problems

Barrier to entry

Often hard to incorporate expert knowledge

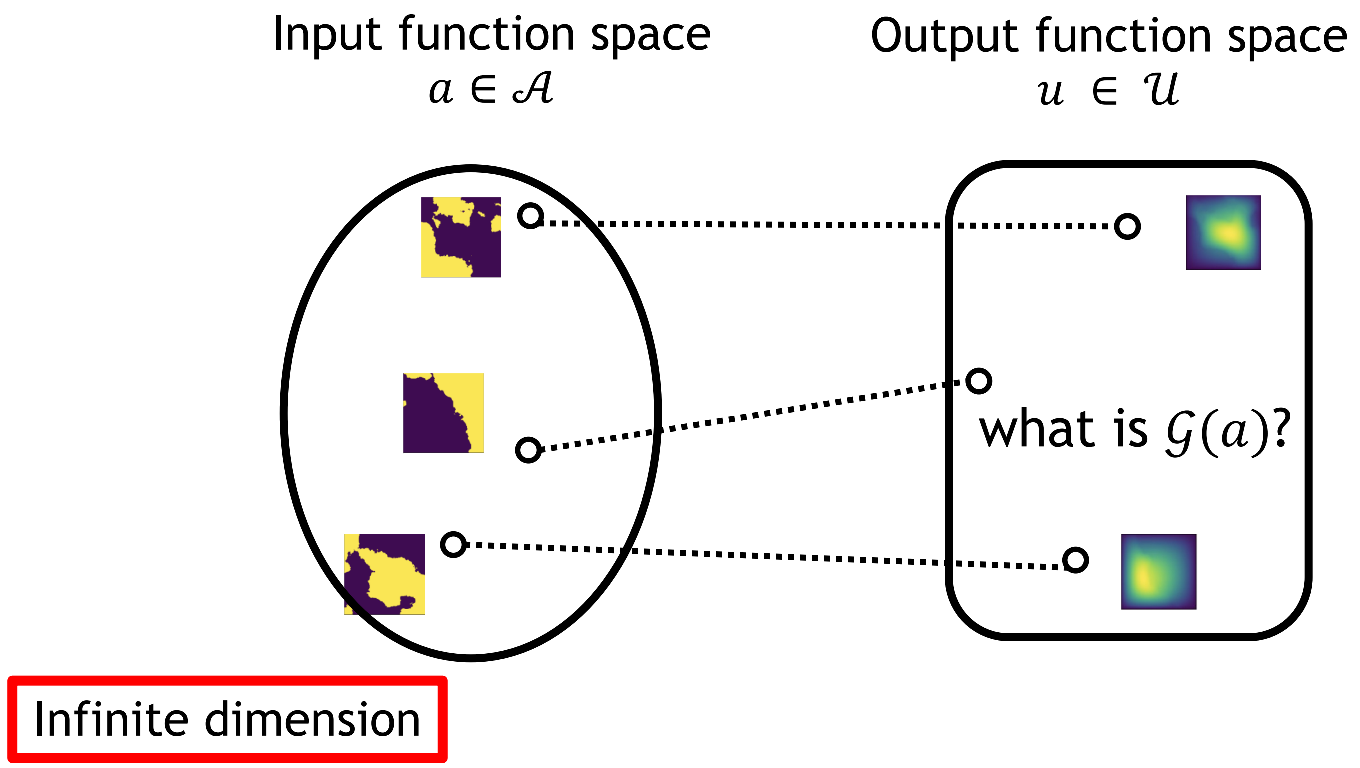
Hard to incorporate real data and experiments

Massive computation

...

How about we learn the solution operator?

Given a, predict u



Machine learning provides tool to advance science and complement the conventional methods

TRADITIONAL METHODS

AI/ML to enable the leap in performance

Why not continue this direction and hand design solution (

Human digestible modeling is challenging and limiting,

Modeling unknown physics

Error due to parameterization

Differentiability for invers problems

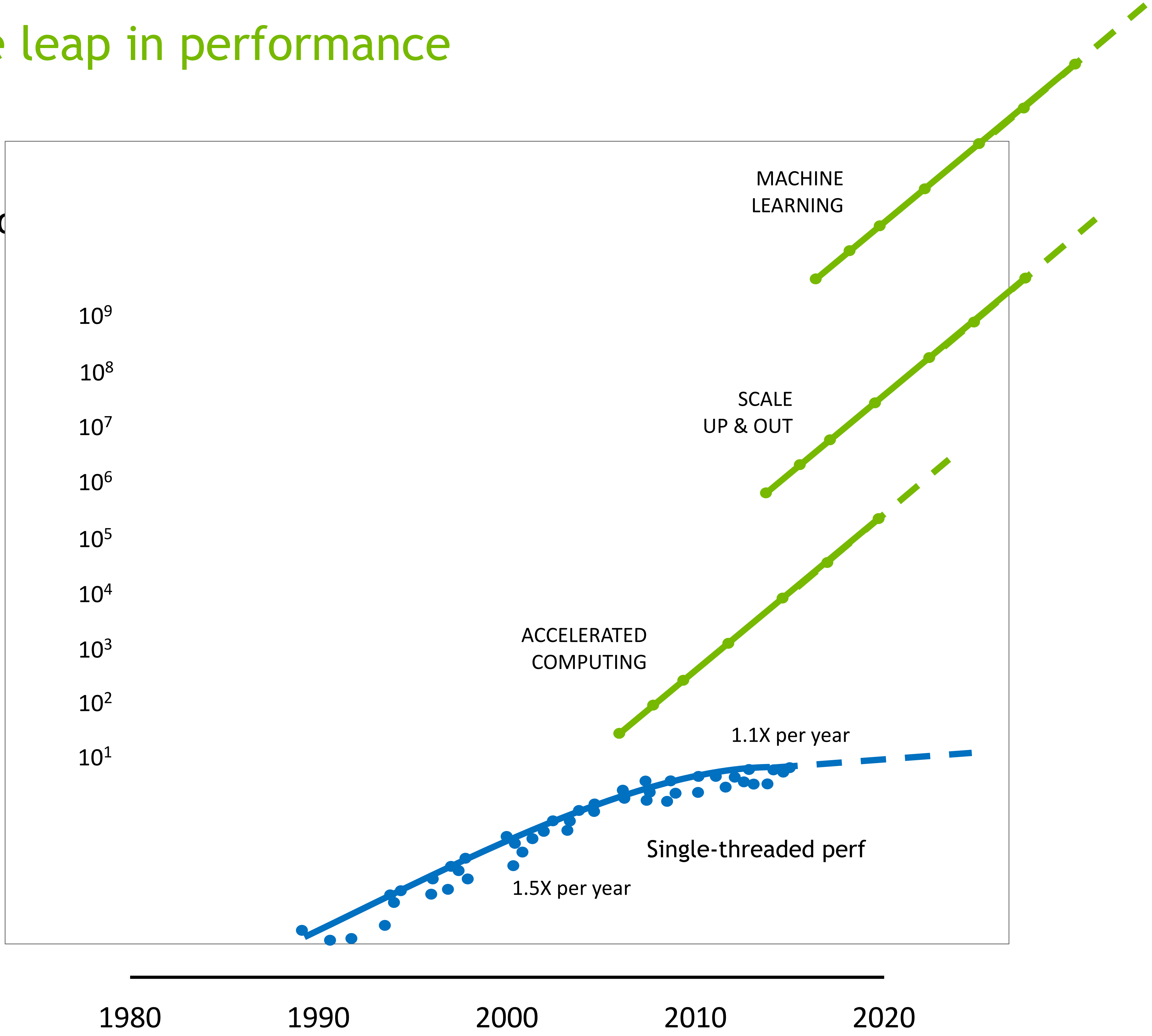
Barrier to entry

Often hard to incorporate expert knowledge

Hard to incorporate real data and experiments

Massive computation

…

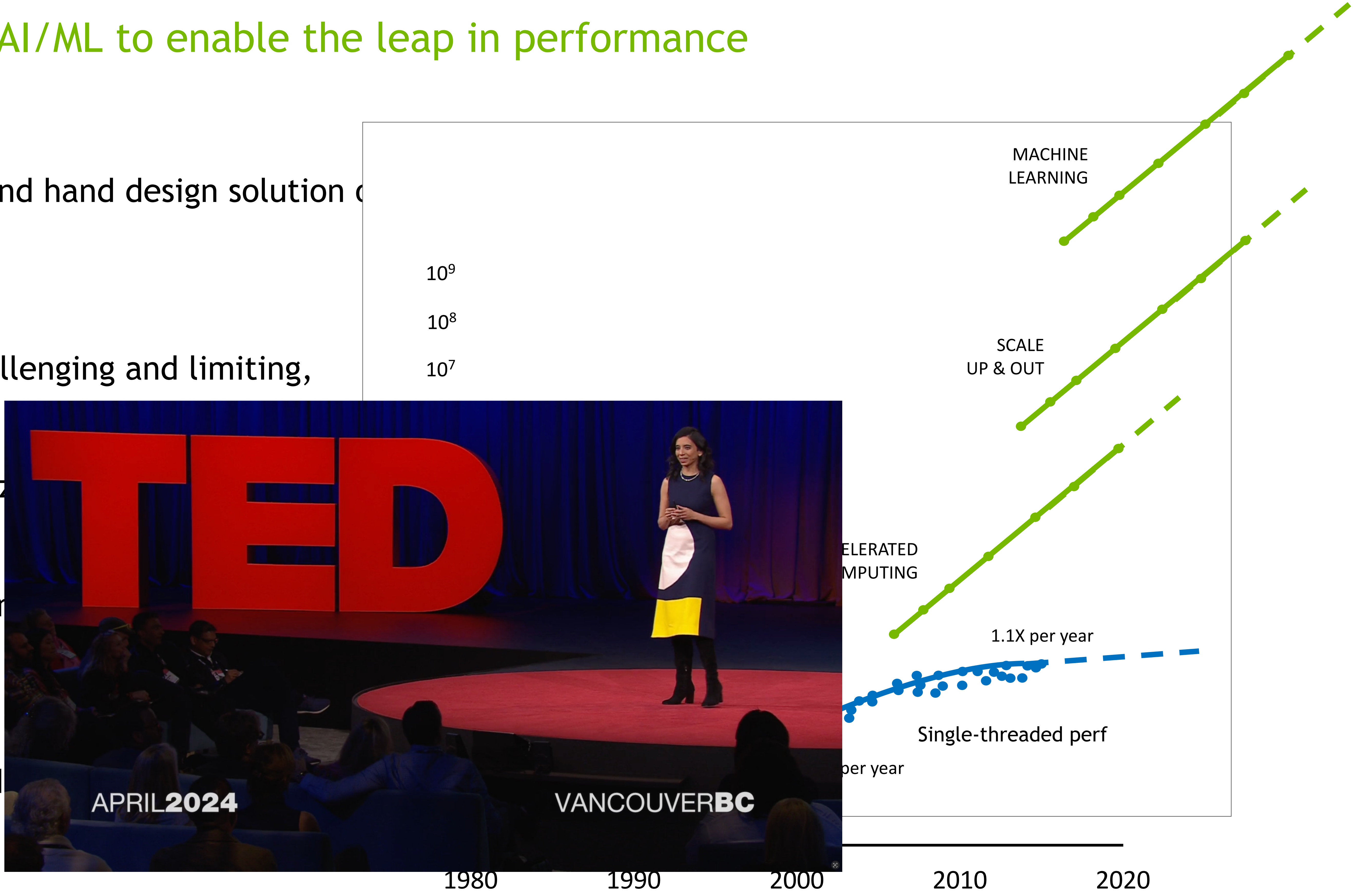


Machine learning provides tools to advance science and complement the conventional methods

TRADITIONAL METHODS

AI/ML to enable the leap in performance

Why not continue this direction and hand design solution o



Human digestible modeling is challenging and limiting,

Modeling unknowns

Approximation due to parameteriz

Massive computation

Differentiability for invers probler

Barrier to entry

Often hard to incorporate expert

Hard to incorporate real data and

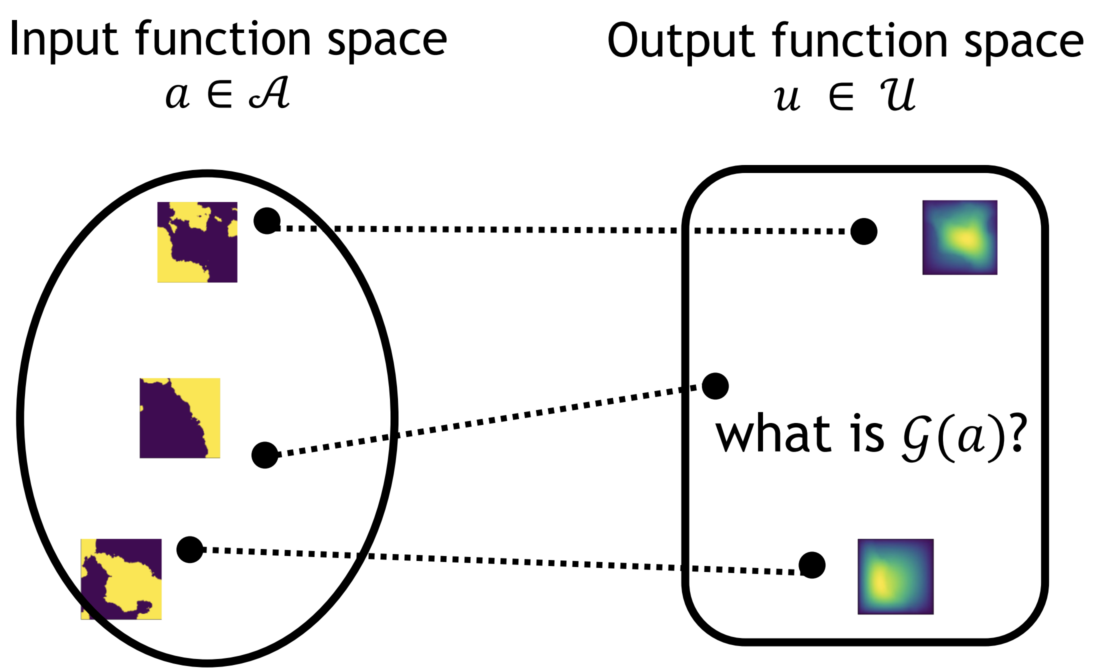
…

https://www.ted.com/talks/anima_anandkumar_ai_that_connects_the_digital_and_physical_worlds

MACHINE LEARNING ON FUNCTION SPACES

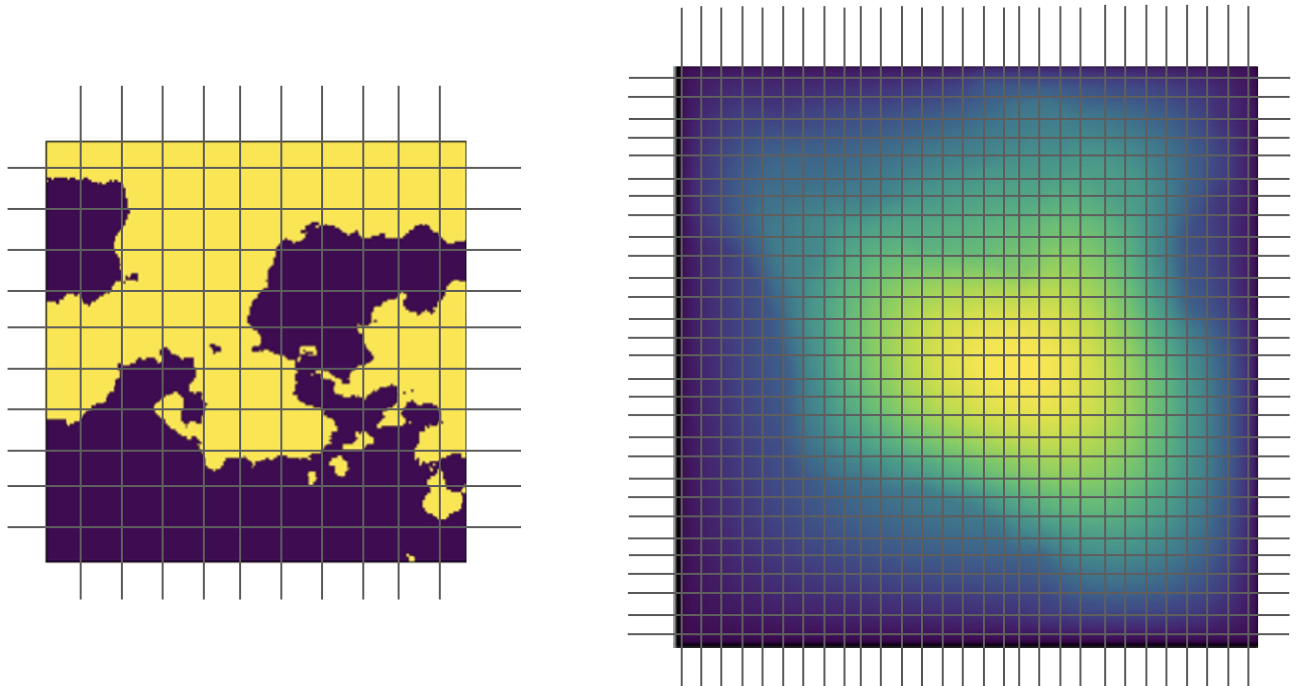
MACHINE LEARNING ON FUNCTIONS

Data and discretization



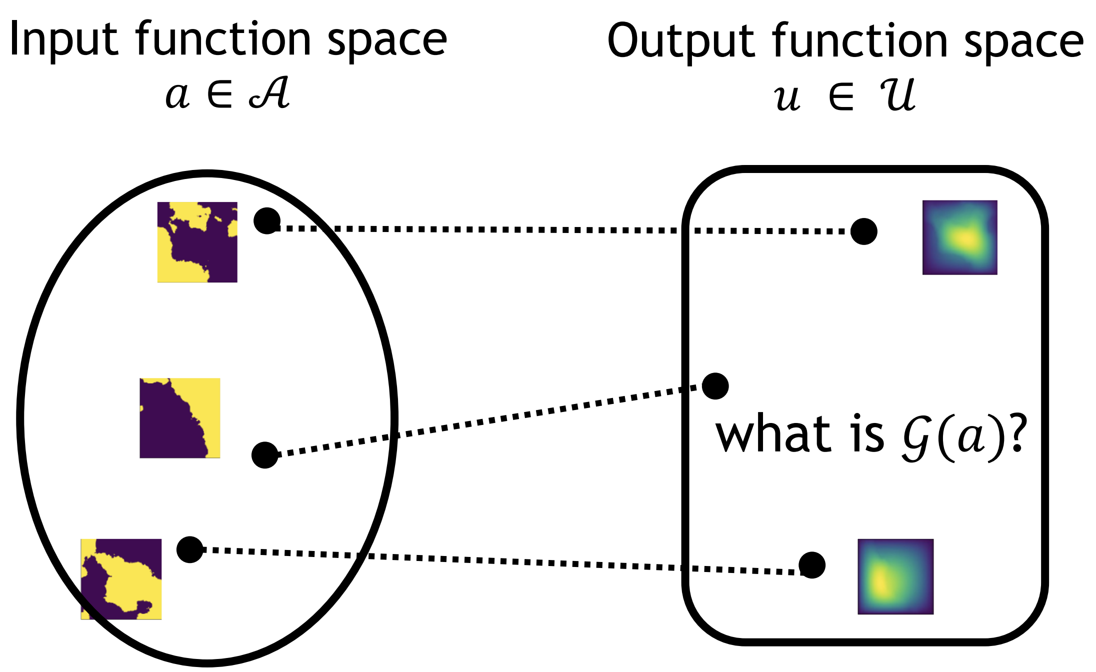
First step challenge

- Input data is given at various resolutions
- Output data is given at various resolution
- The model output needs to be function (derivatives and integrals in physics)



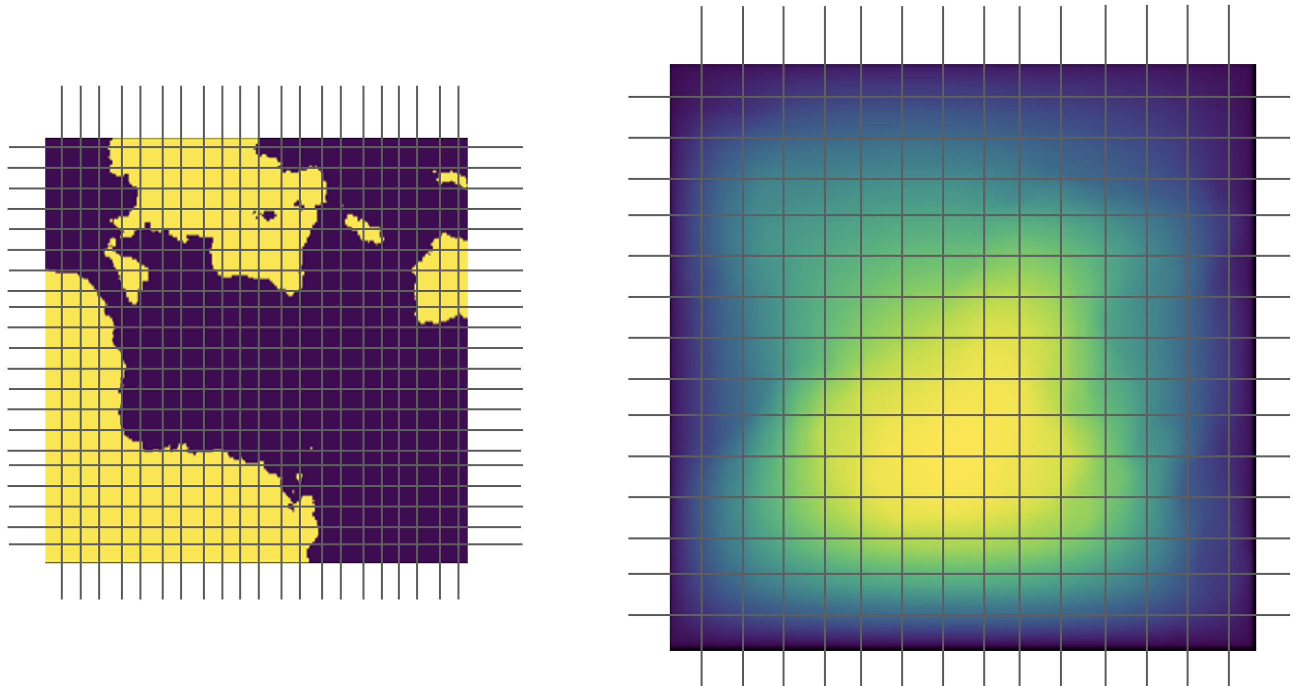
MACHINE LEARNING ON FUNCTIONS

Data and discretization



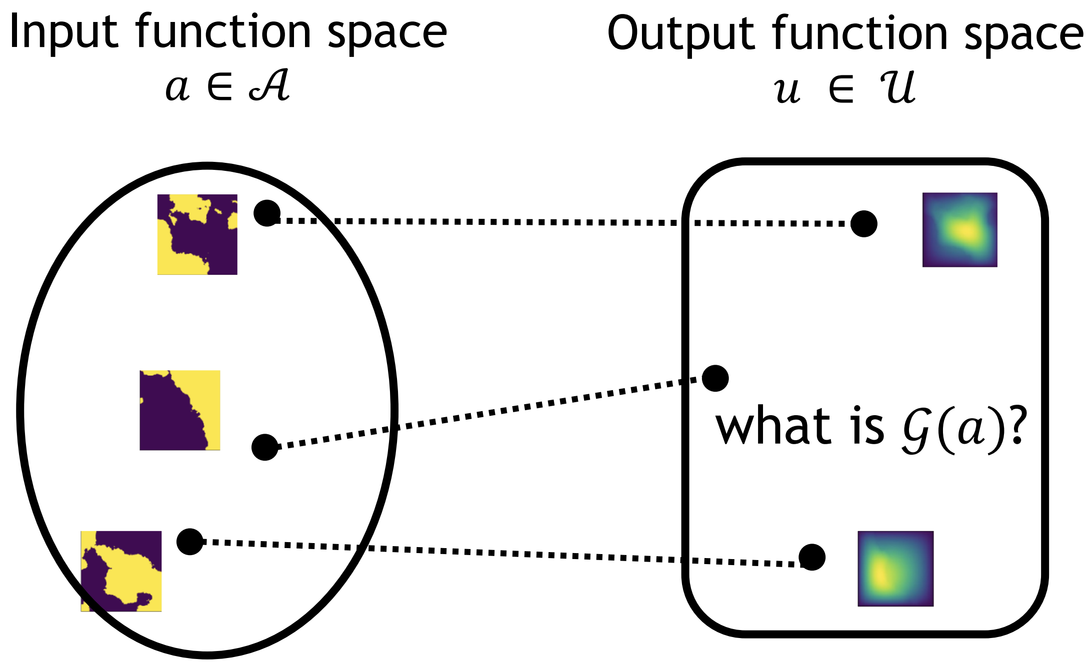
First step challenge

- Input data is given at various resolutions
- Output data is given at various resolution
- The model output needs to be function (derivatives and integrals in physics)



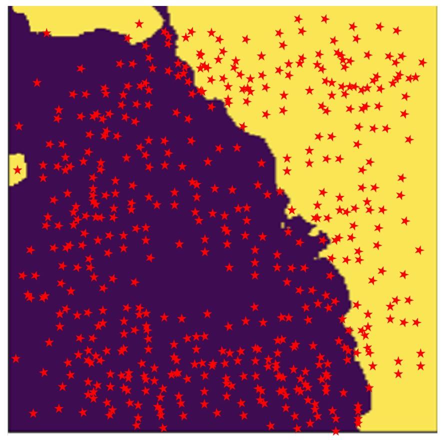
MACHINE LEARNING ON FUNCTIONS

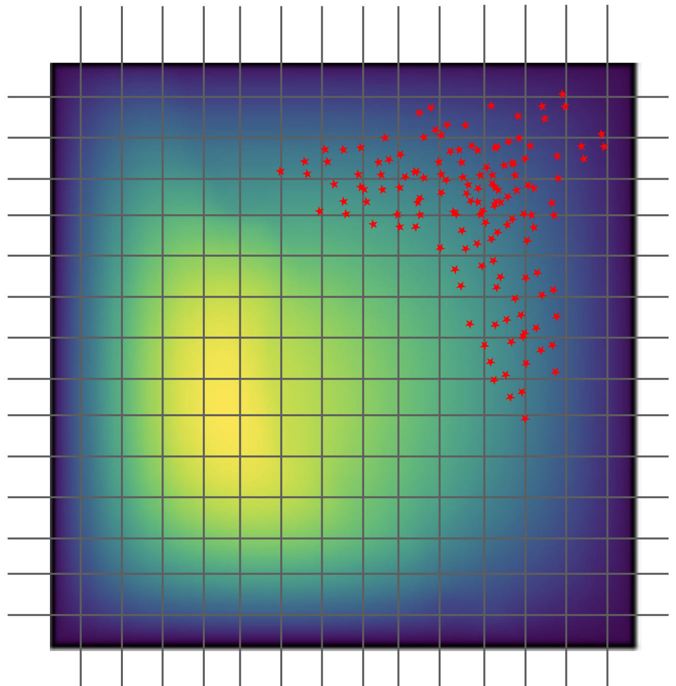
Data and discretization



First step challenge

- Input data is given at various resolutions
- Output data is given at various resolution
- The model output needs to be function (derivatives and integrals in physics)



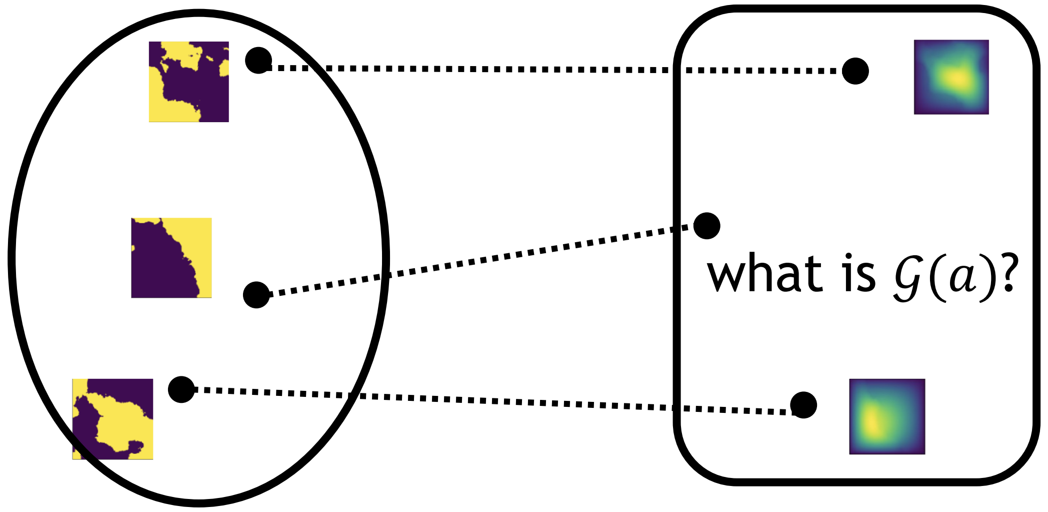


MACHINE LEARNING ON FUNCTIONS

Data and discretization

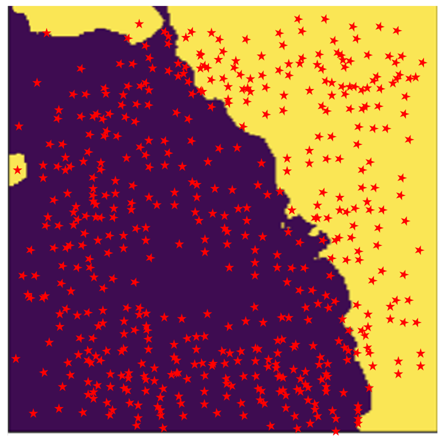
Input function space
$a \in \mathcal{A}$

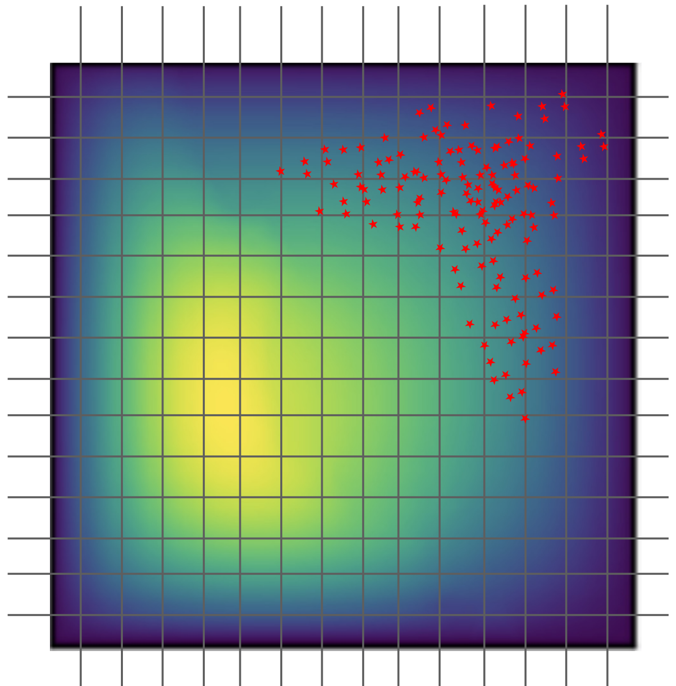
Output function space
$u \in \mathcal{U}$

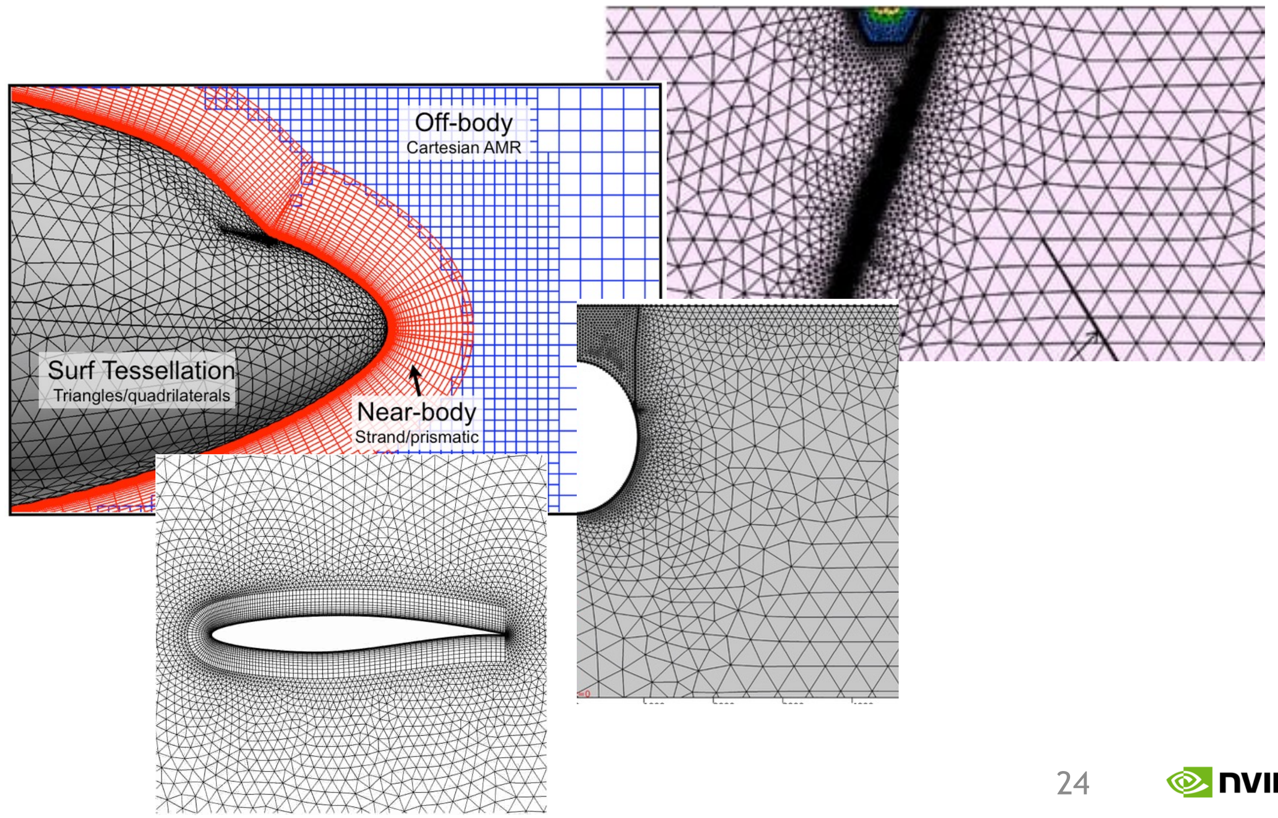


First step challenge

- Input data is given at various resolutions
- Output data is given at various resolution
- The model output needs to be function (derivatives and integrals in physics)

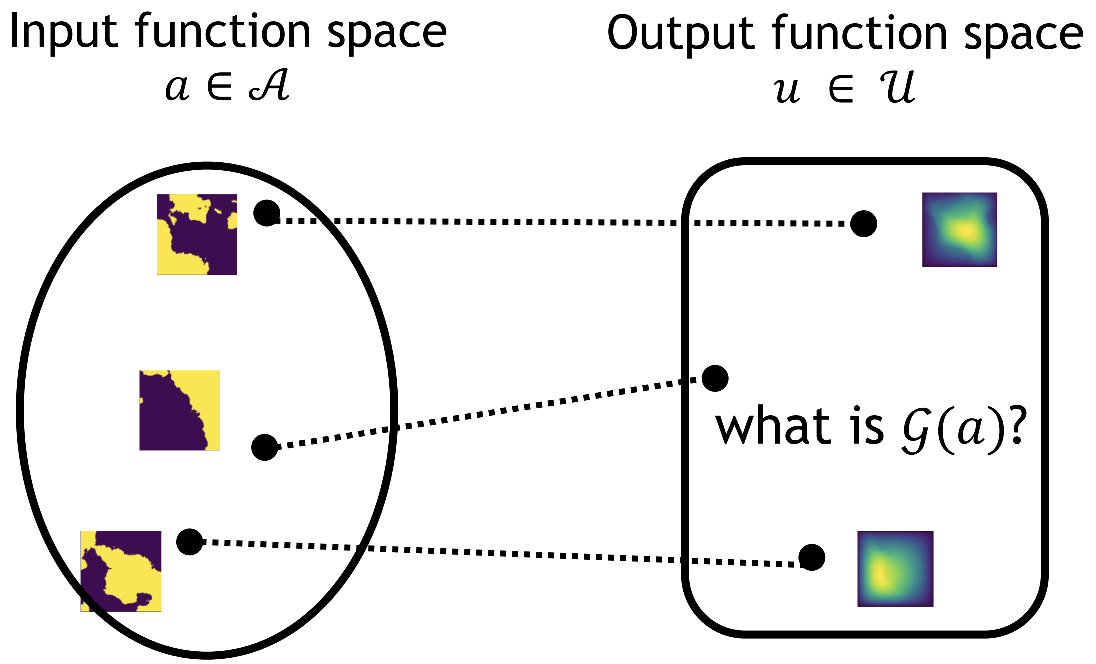






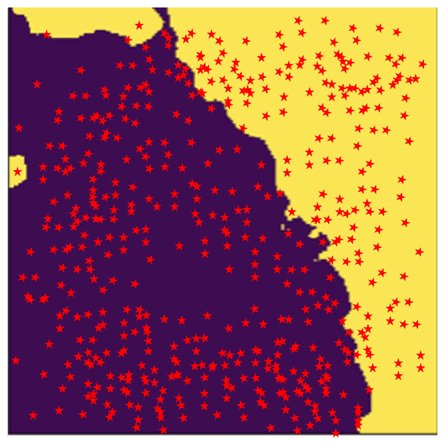
MACHINE LEARNING ON FUNCTIONS

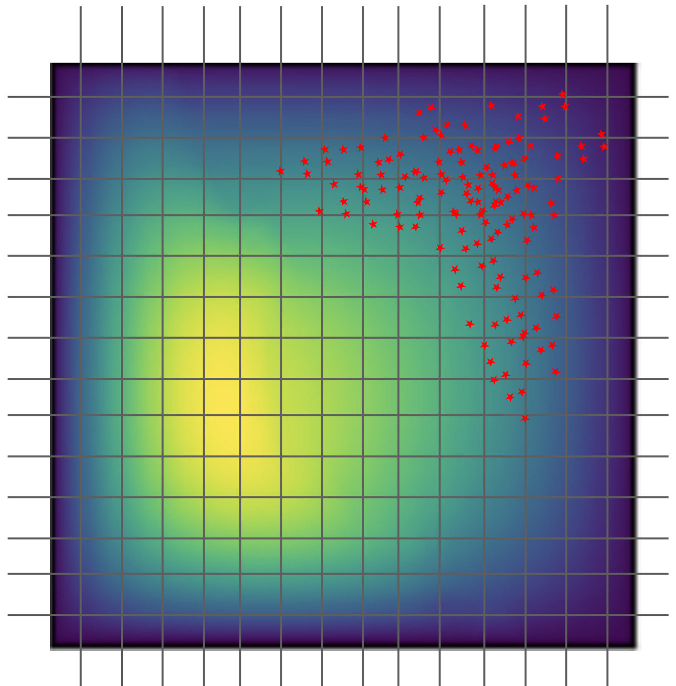
Data and discretization

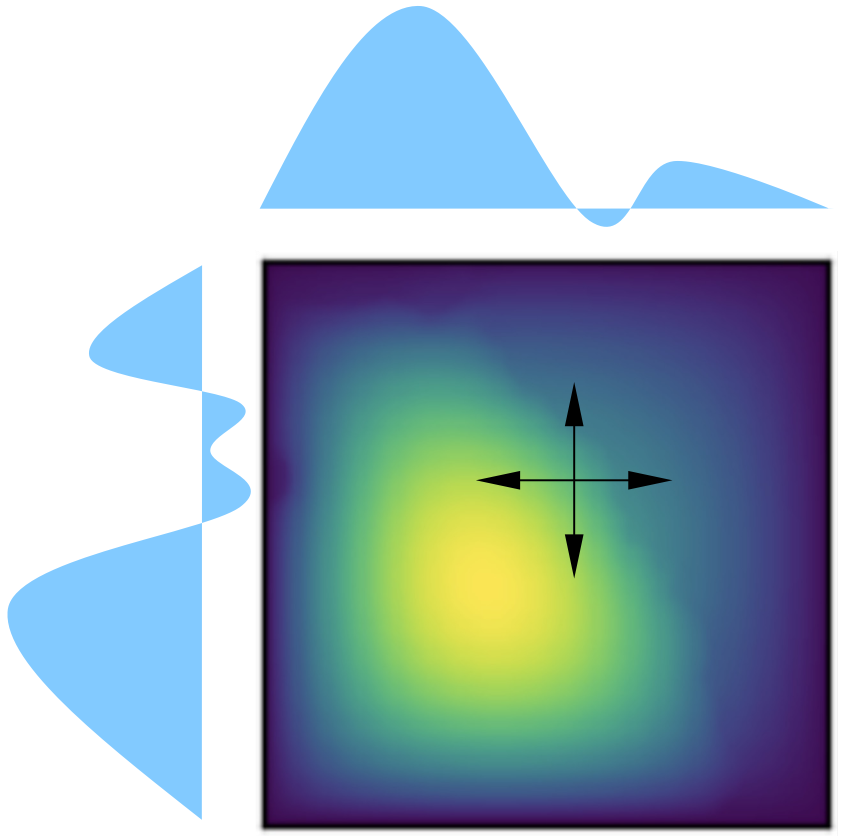


First step challenge

- Input data is given at various resolutions
- Output data is given at various resolution
- The model output needs to be function (derivatives and integrals in physics)







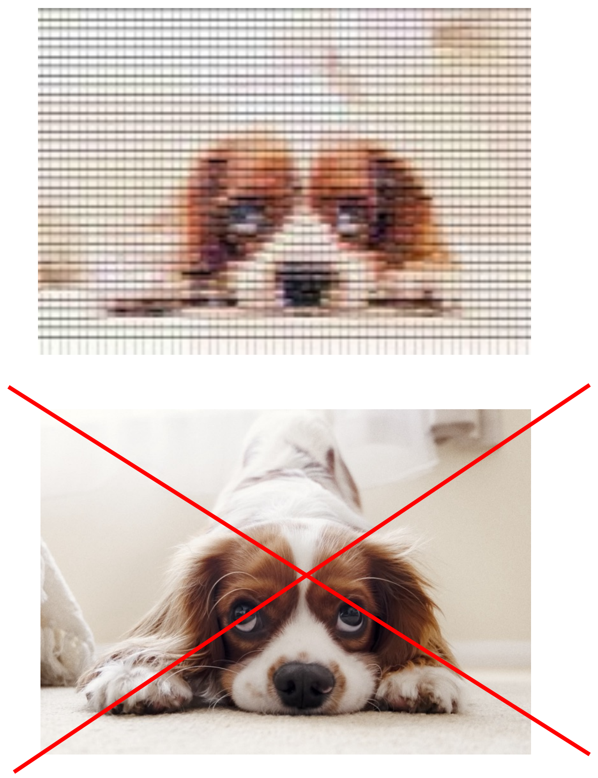
NEURAL OPERATOR

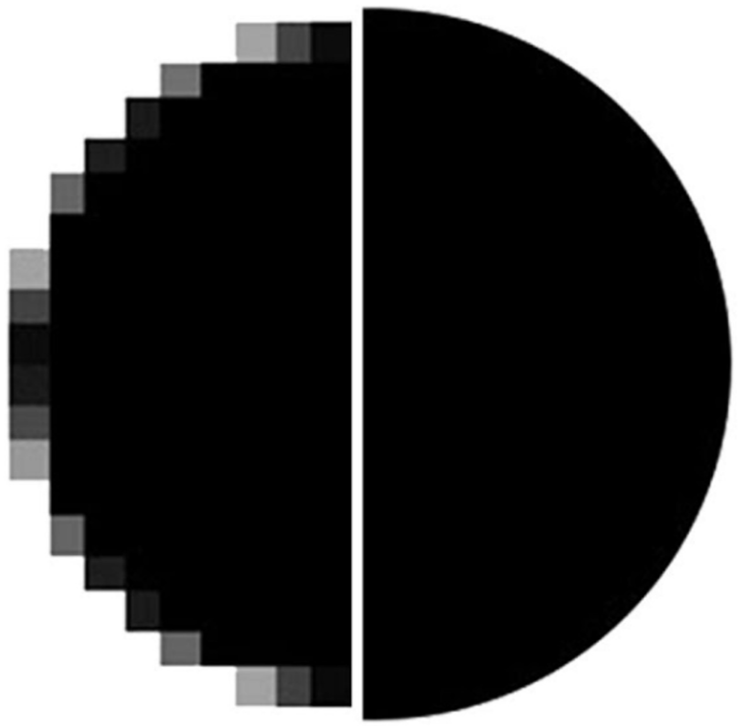
DISCRETIZATION AGNOSTIC LEARNING

One ML model for any discretization

Neural Network

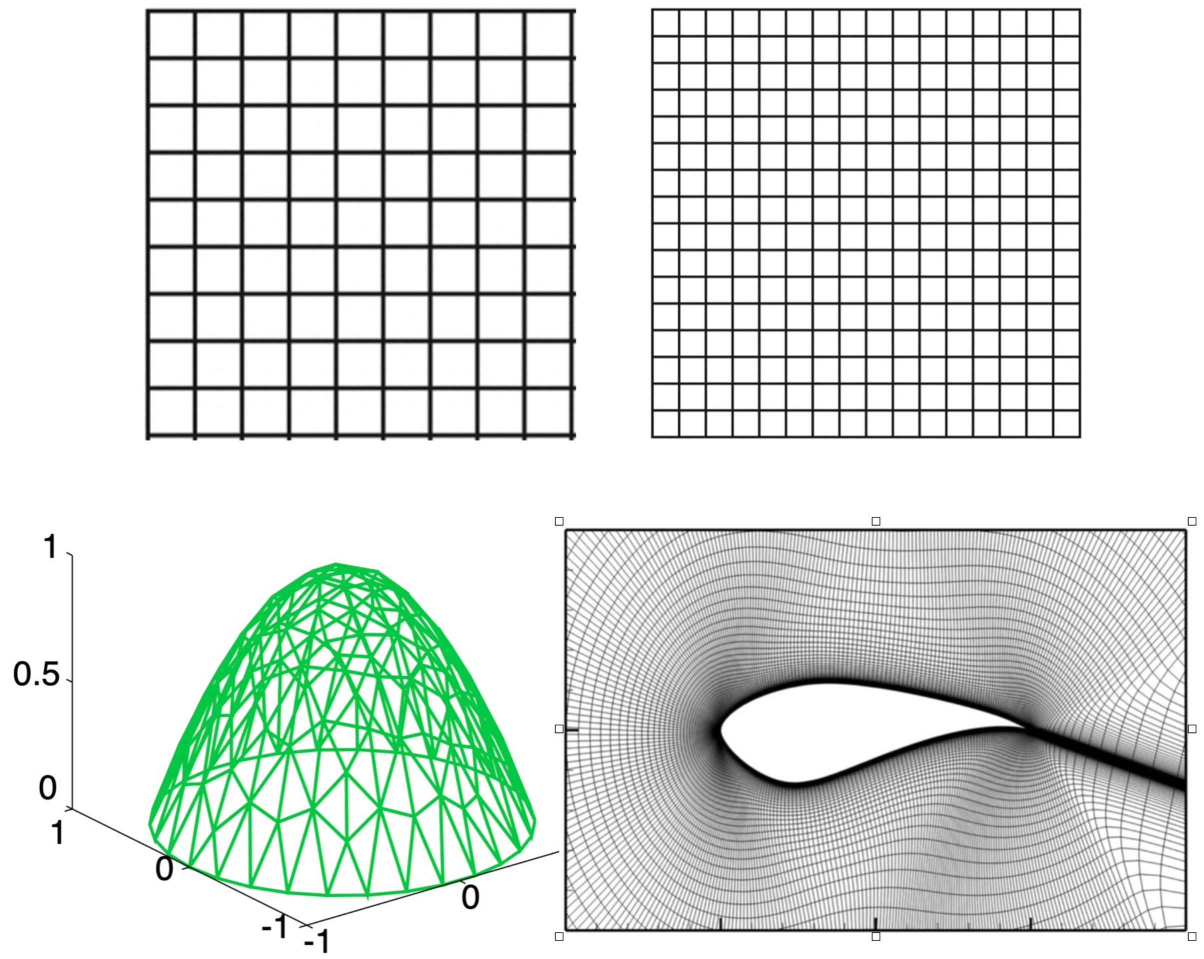
Input and output at fixed resolution





Neural Operator

Input and output at any resolution



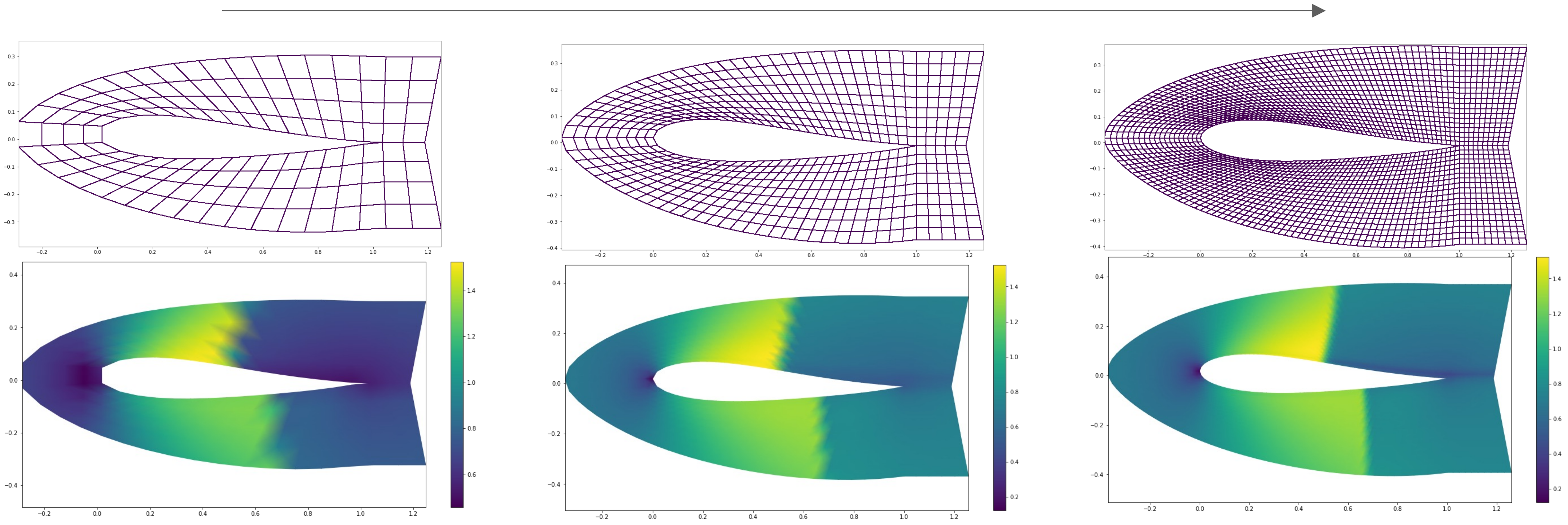
NEURAL OPERATOR: DISCRETIZATION AGNOSTIC

One ML model for any discretization

Definition: a trained AI model is discretization-convergent if

- We can query at any point.
- Converges upon mesh refinement to a limit.

Mesh refinement



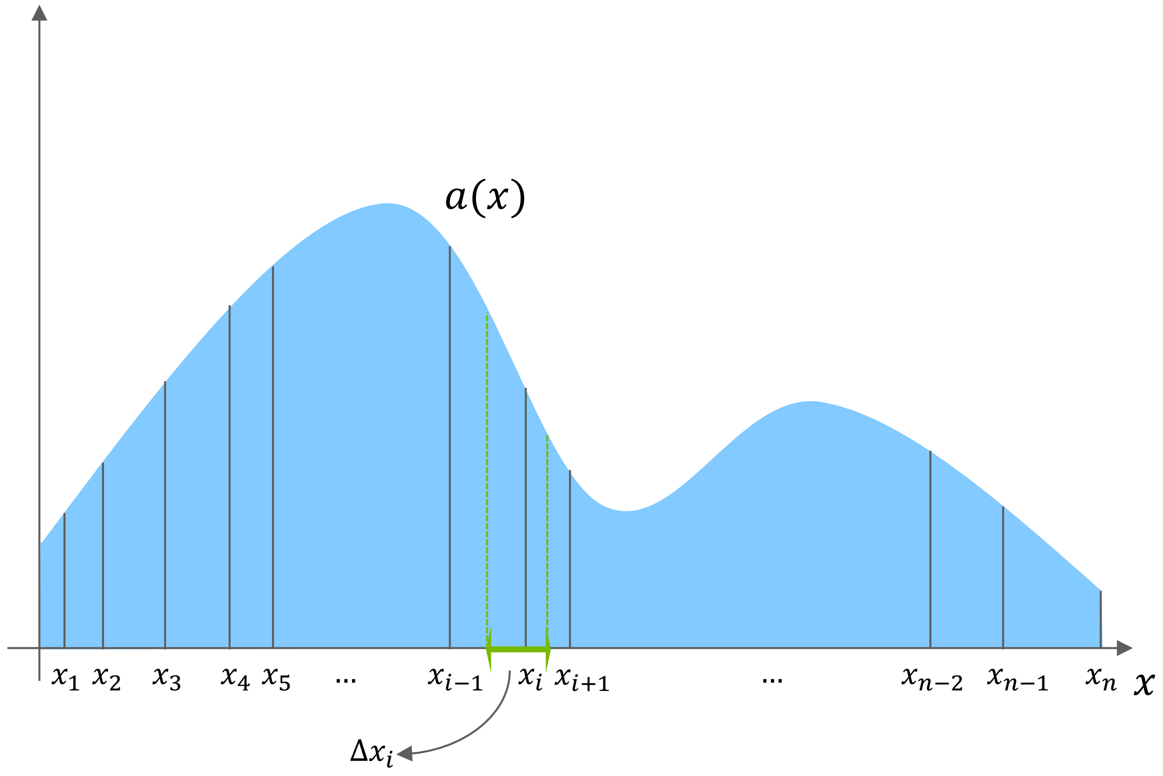
Converging solution

PRE-REQ

MACHINE LEARNING ON FUNCTIONS

Integral and discretization

Pre-req for ML on function spaces



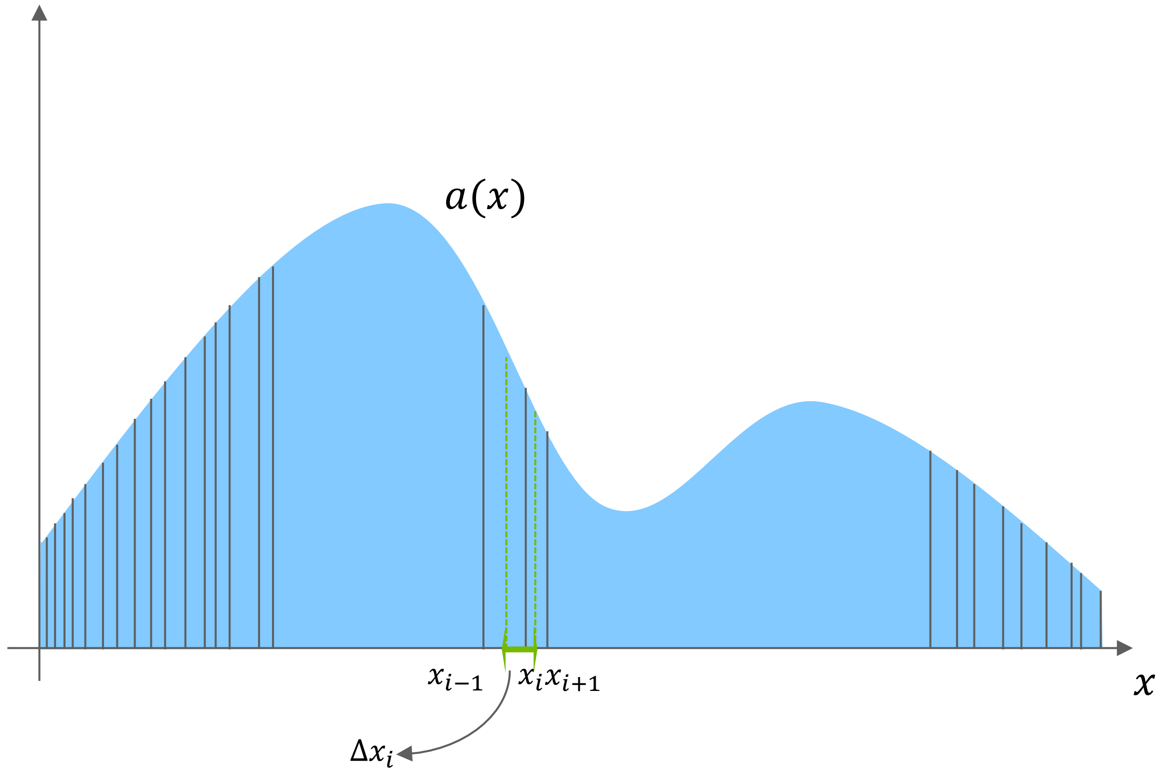
$$\int a(x)dx \approx \sum_{i}^{n} a(x_i)\Delta \mathrm{x_i}$$

Riemannian sum

MACHINE LEARNING ON FUNCTIONS

Integral and discretization

Pre-req for ML on function spaces



$$\int a(x)dx \approx \sum_{i}^{n} a(x_i)\Delta \mathrm{x_i}$$

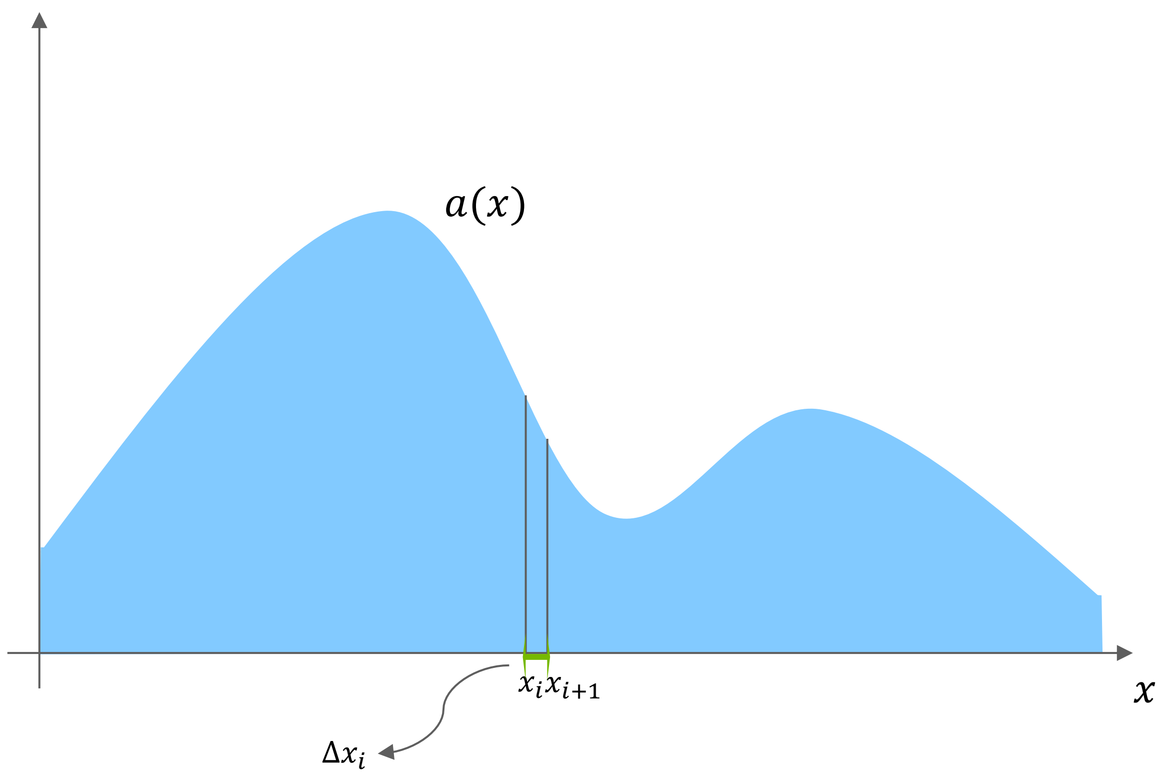
Riemannian sum

Finer mesh → better approximation of integral

MACHINE LEARNING ON FUNCTIONS

Derivative and discretization

Pre-req for ML on function spaces



$$\left.\frac{da}{dx}\right|_x \approx \frac{a(x_{i+1}) - a(x_i)}{\Delta x}$$

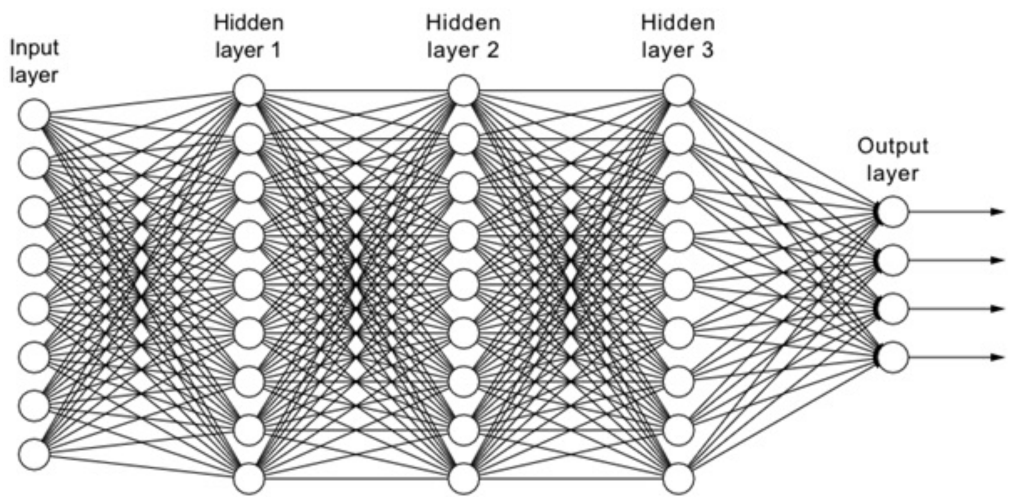
$$\left.\frac{da}{dx}\right|_x \approx \frac{a(x_{i+1}) - a(x_i)}{2\Delta x} + \frac{a(x_i) - a(x_{i-1})}{2\Delta x}$$

Finer mesh → better approximation of derivatives

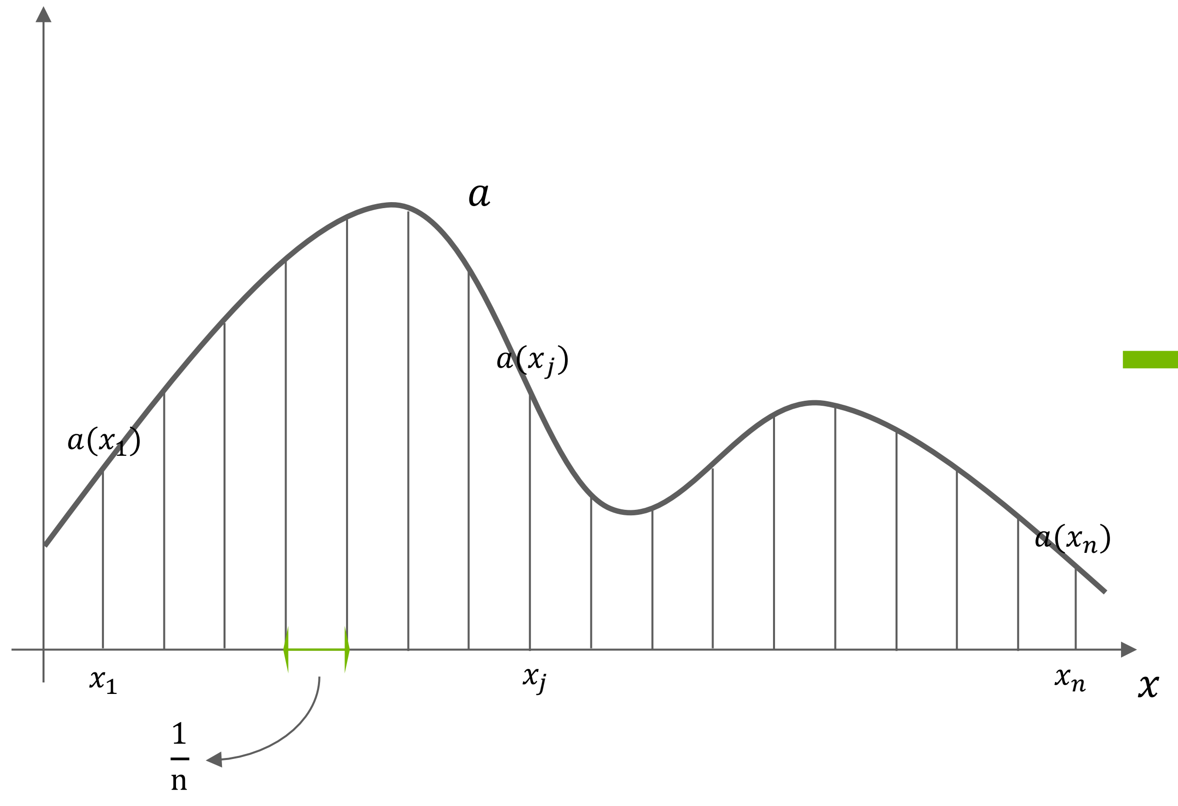
FROM NEURAL NETWORKS TO NEURAL OPERATORS

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



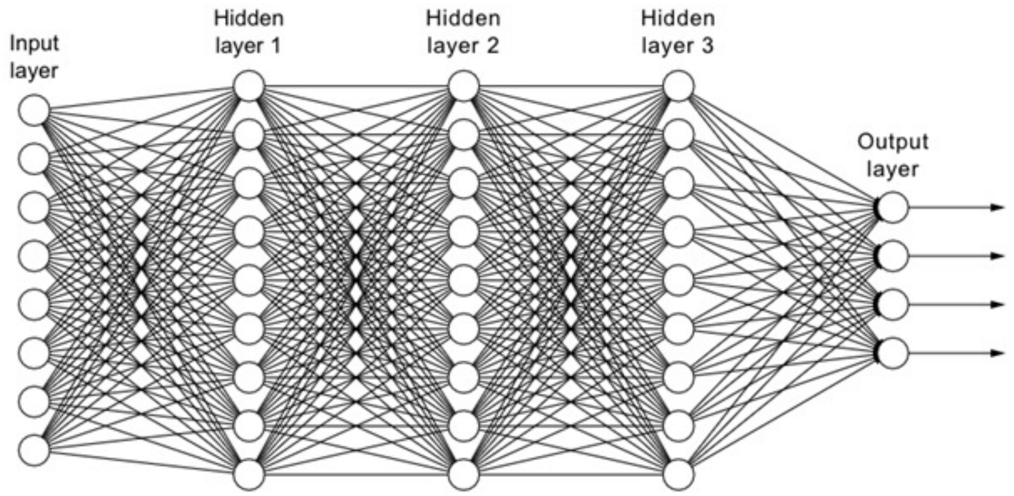
Consider a basic feed forward neural network layer



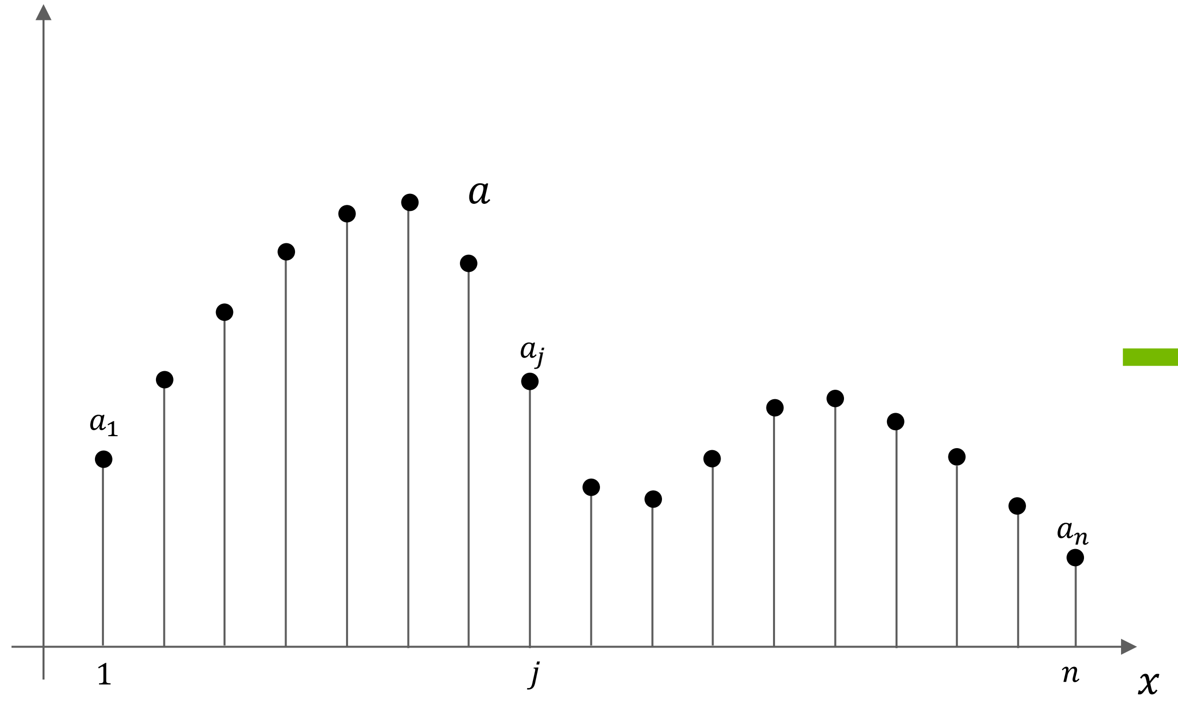
$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



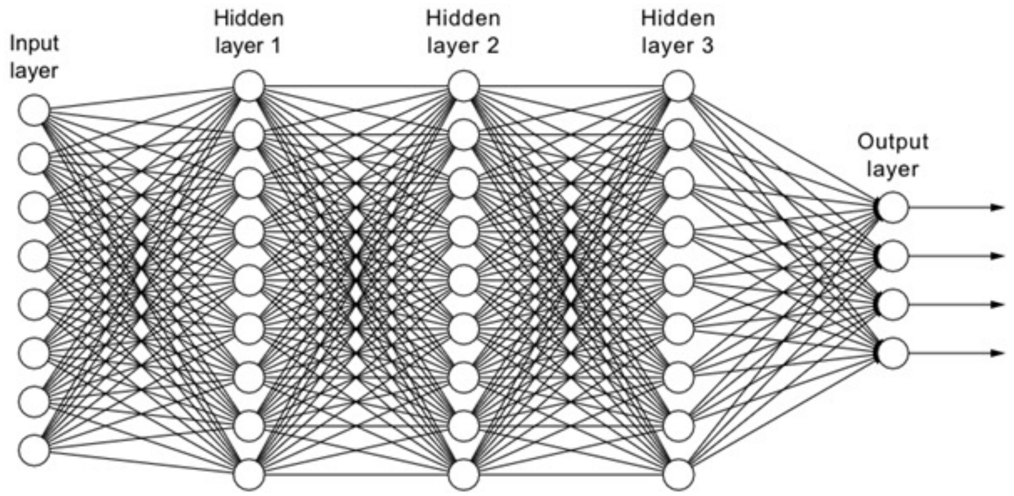
Consider a basic feed forward neural network layer



$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$

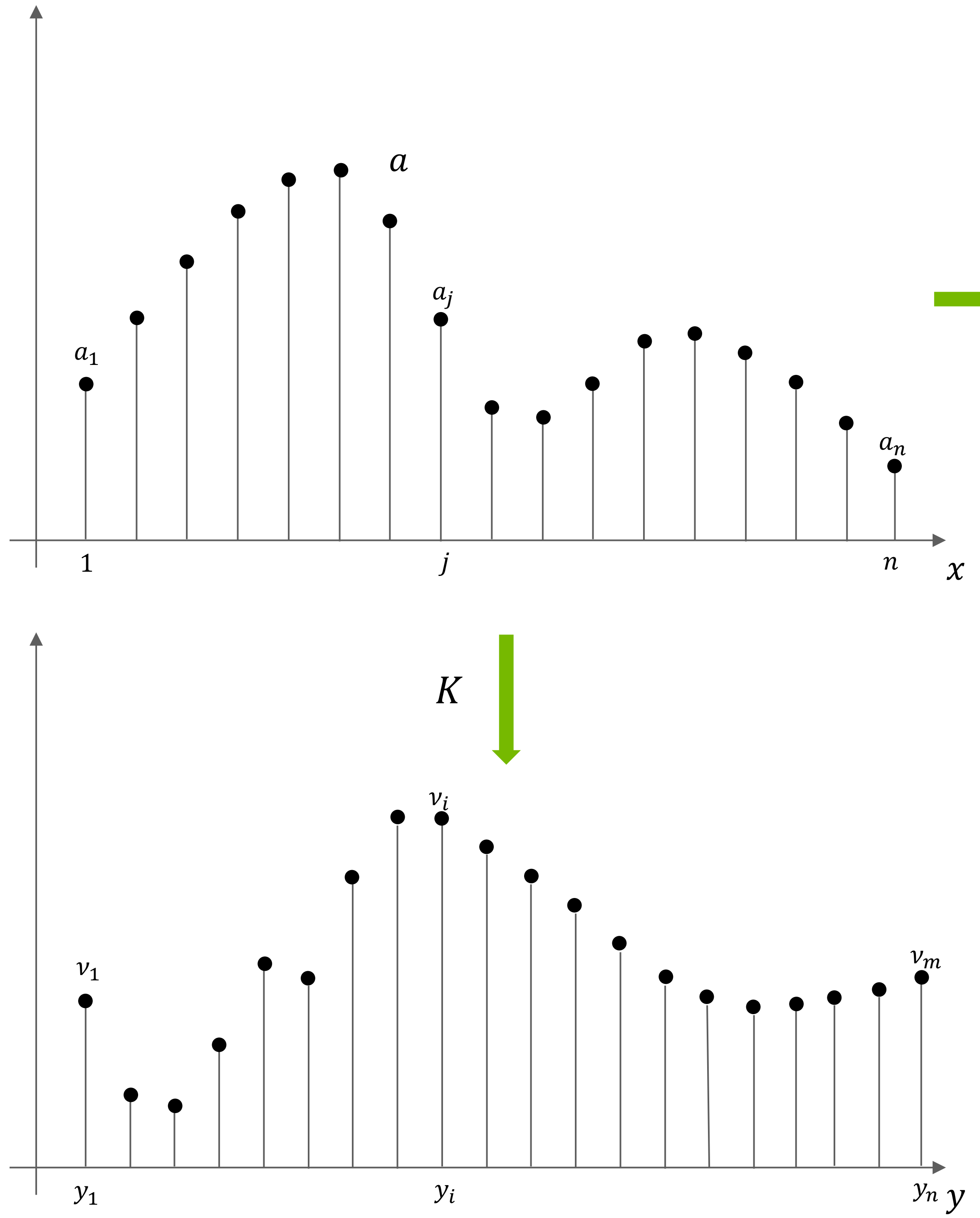
MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



K

Consider a basic feed forward neural network layer



$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$

First layer

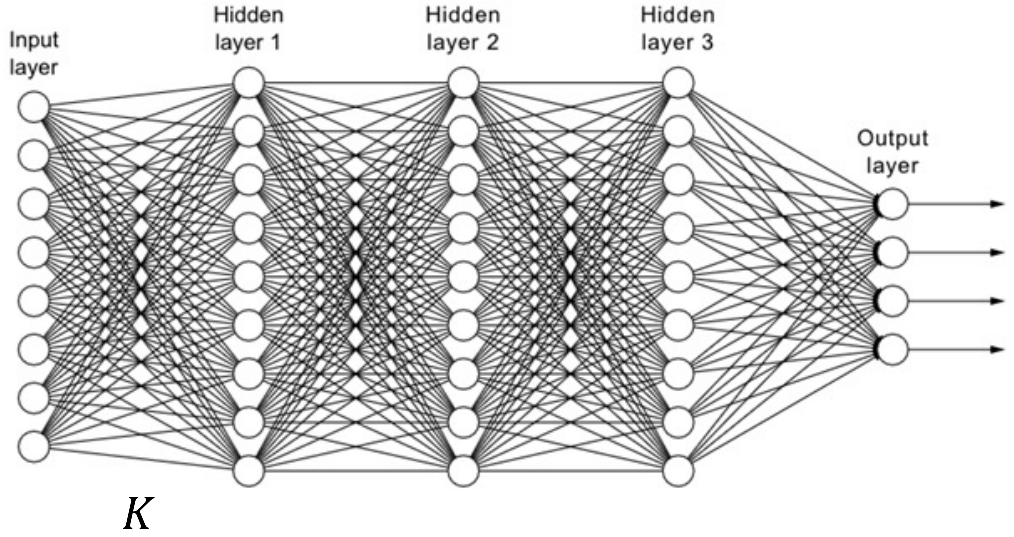
$$v_i = \sigma\left(\frac{1}{n}\sum_{j}^{n} K_{ij} a_j \right)$$

Linear model in conventional ML

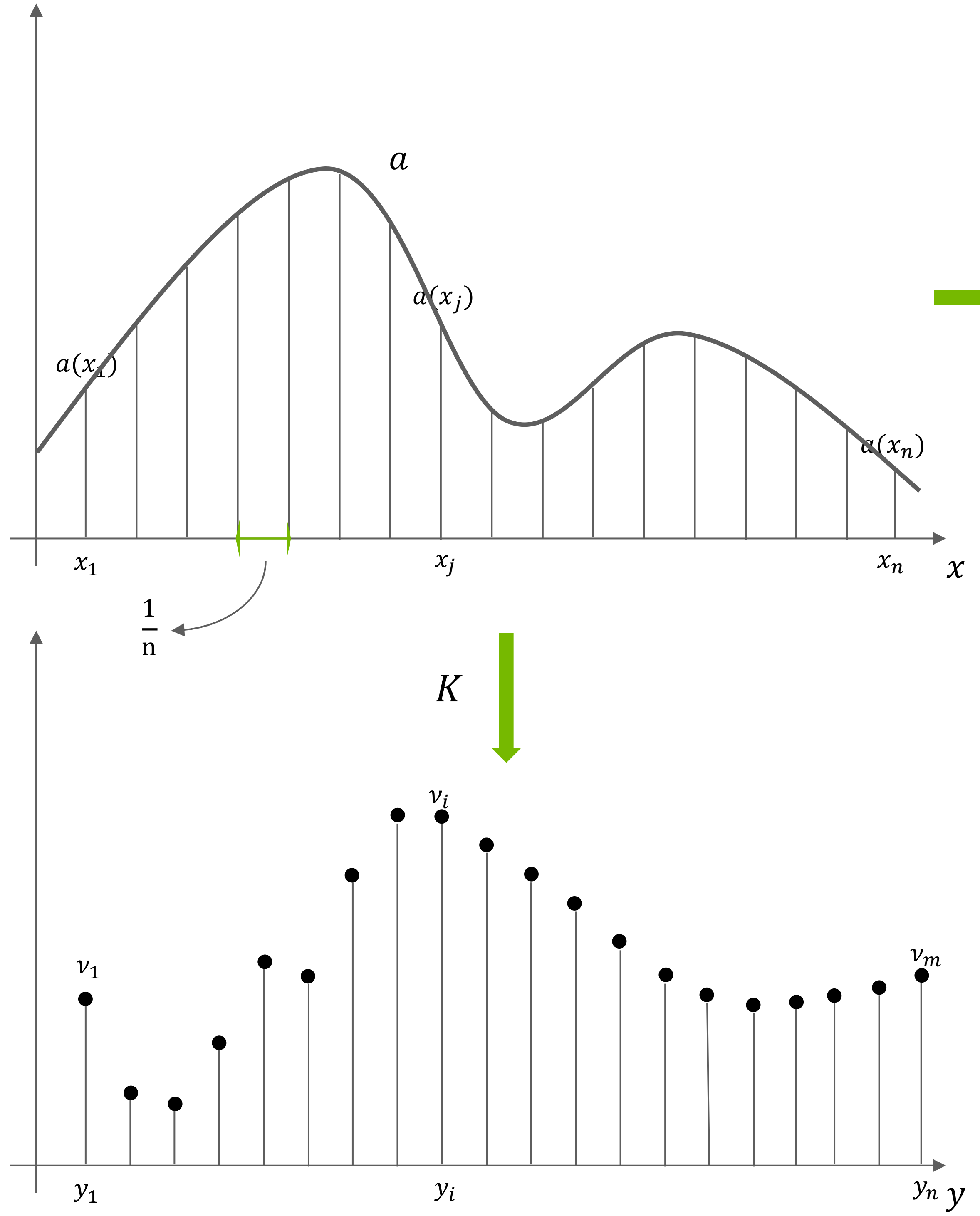
(finite dimensional ML)

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



Consider a basic feed forward neural network layer



$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$

First layer

$$v_i = \sigma\left(\frac{1}{n}\sum_j^n K_{ij} a_j\right)$$

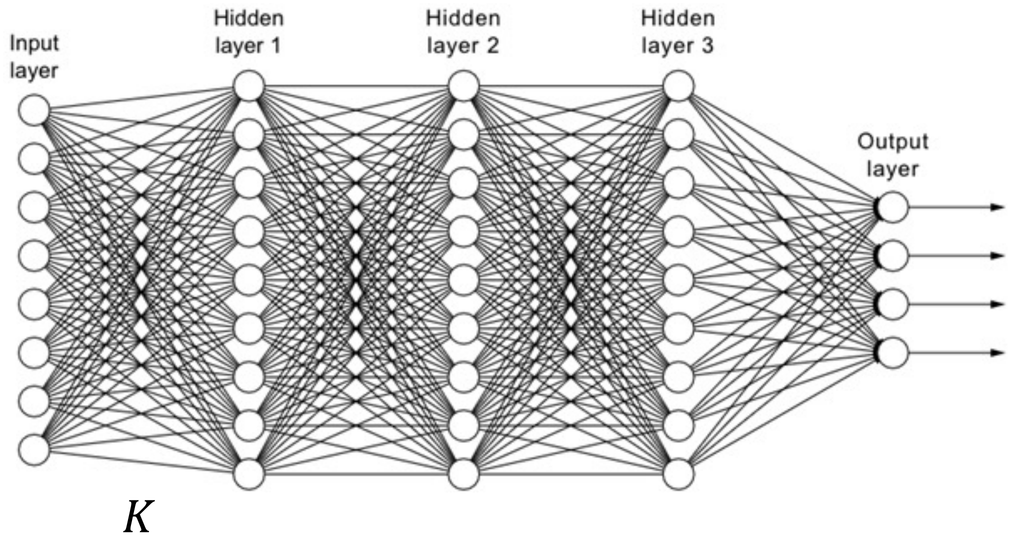
$$= \sigma\left(\frac{1}{n}\sum_j^n K_{ij} a(x_j)\right)$$

Linear model in conventional ML

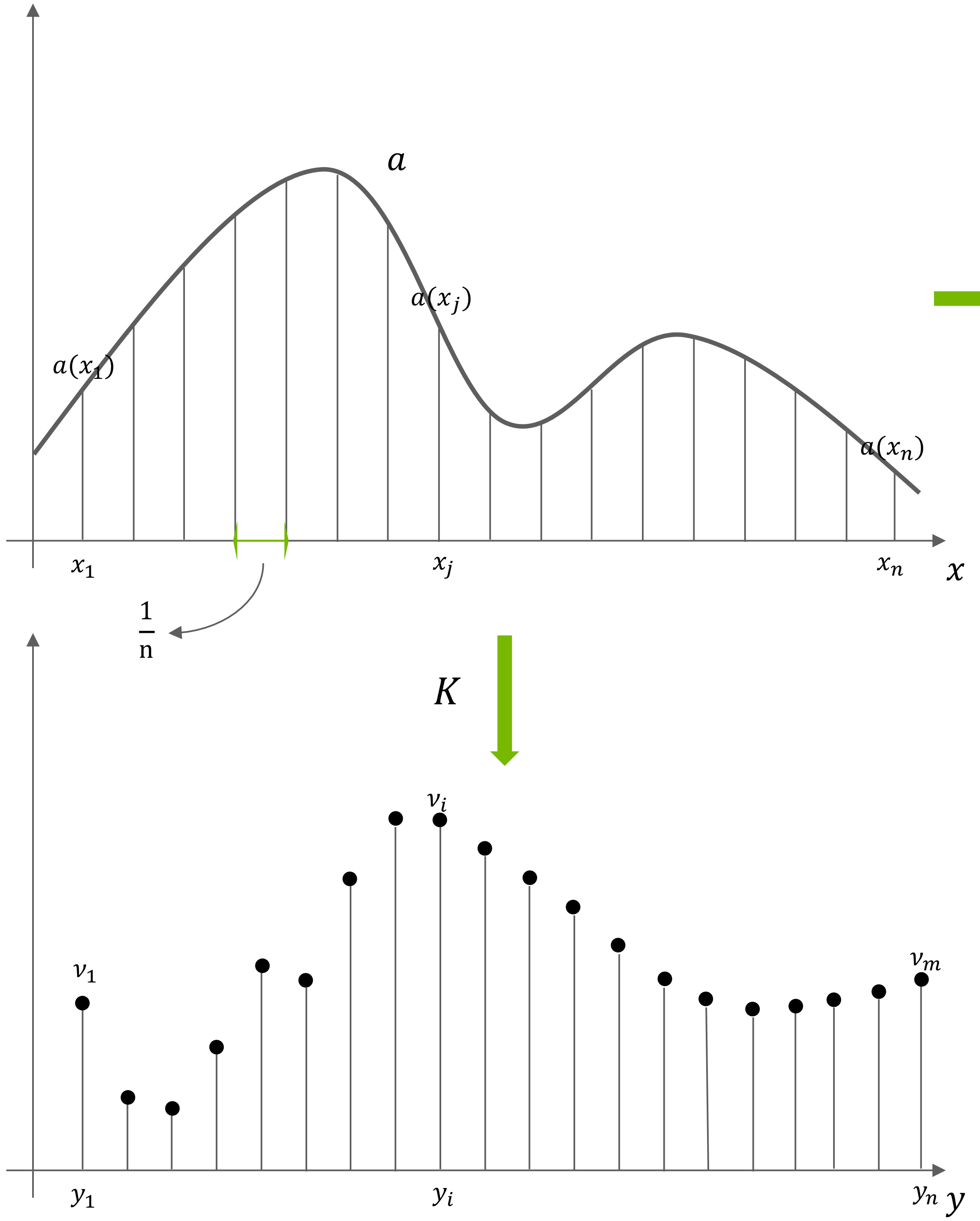
(finite dimensional ML)

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



Consider a basic feed forward neural network layer



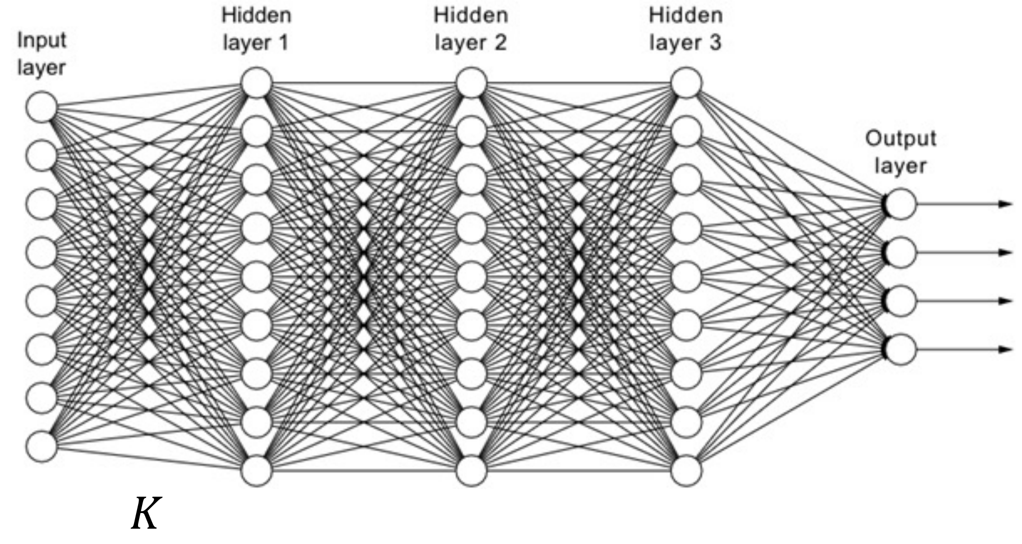
$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$

First layer

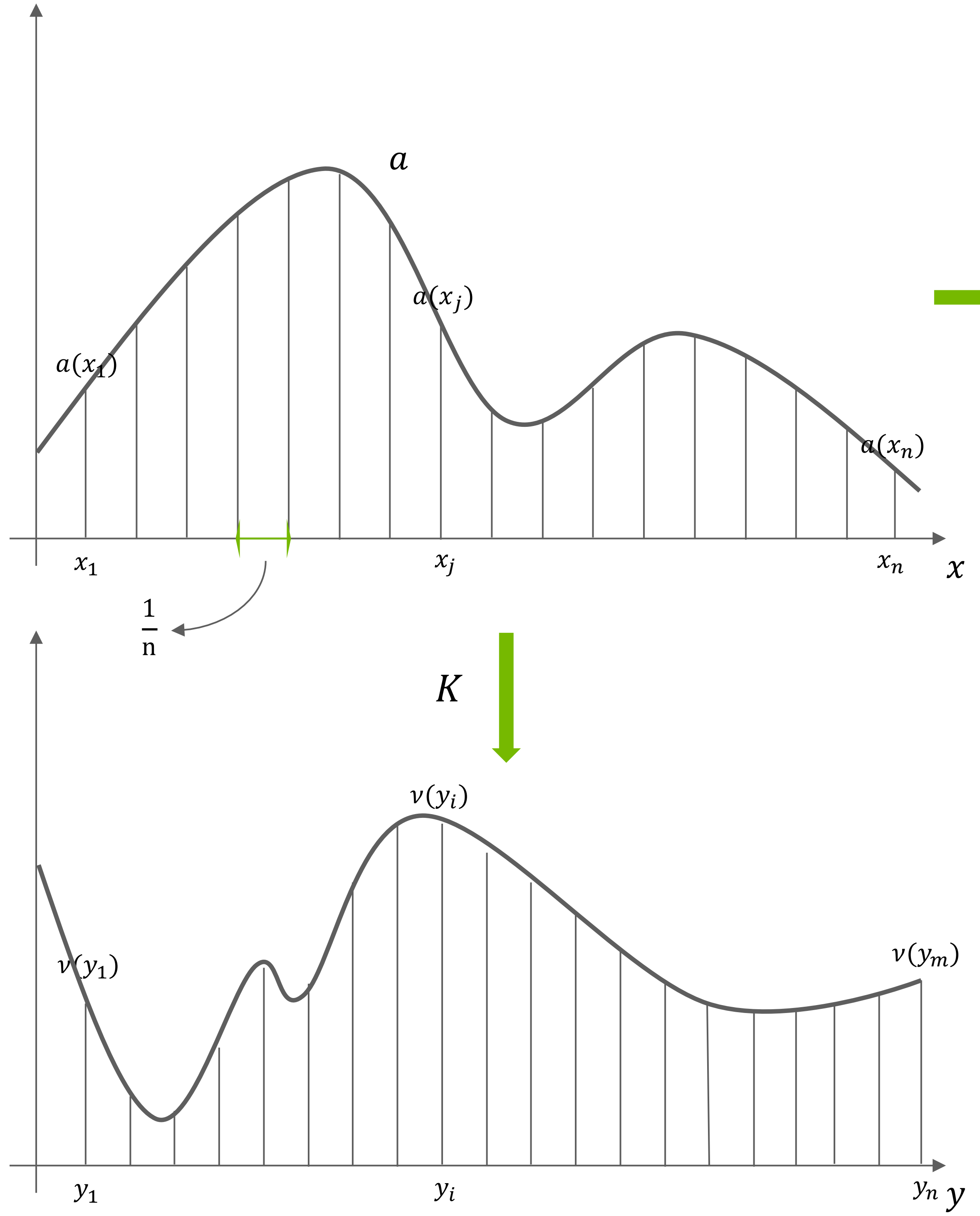
$$v(y_i) = \sigma\left(\frac{1}{n}\sum_{j}^{n} K_{ij}\, a(x_j)\right)$$

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



Consider a basic feed forward neural network layer



$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$

First layer

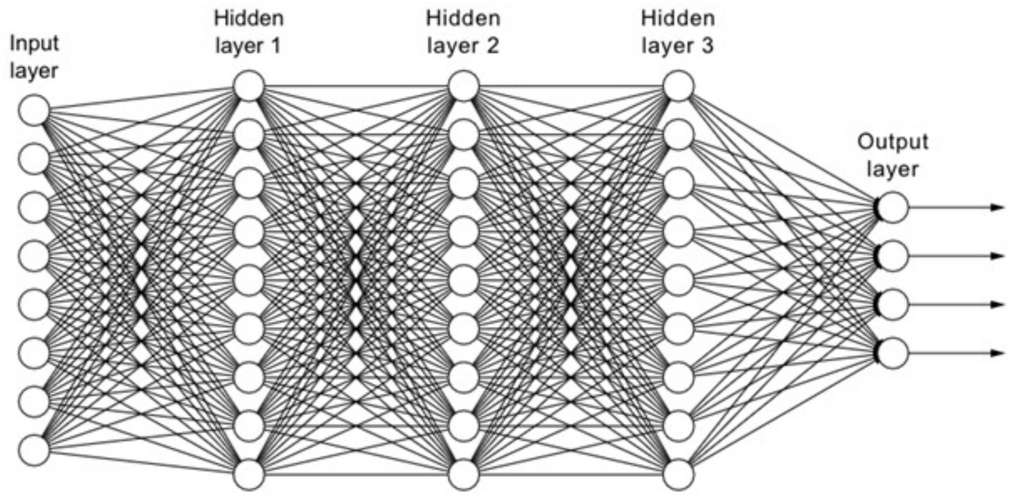
$$v(y_i) = \sigma\Big(\frac{1}{n}\sum_{j}^{n} K_{ij}\, a(x_j)\Big)$$

$$= \sigma\Big(\frac{1}{n}\sum_{j}^{n} \kappa(y_i, x_j)\, a(x_j)\Big)$$

NVIDIA

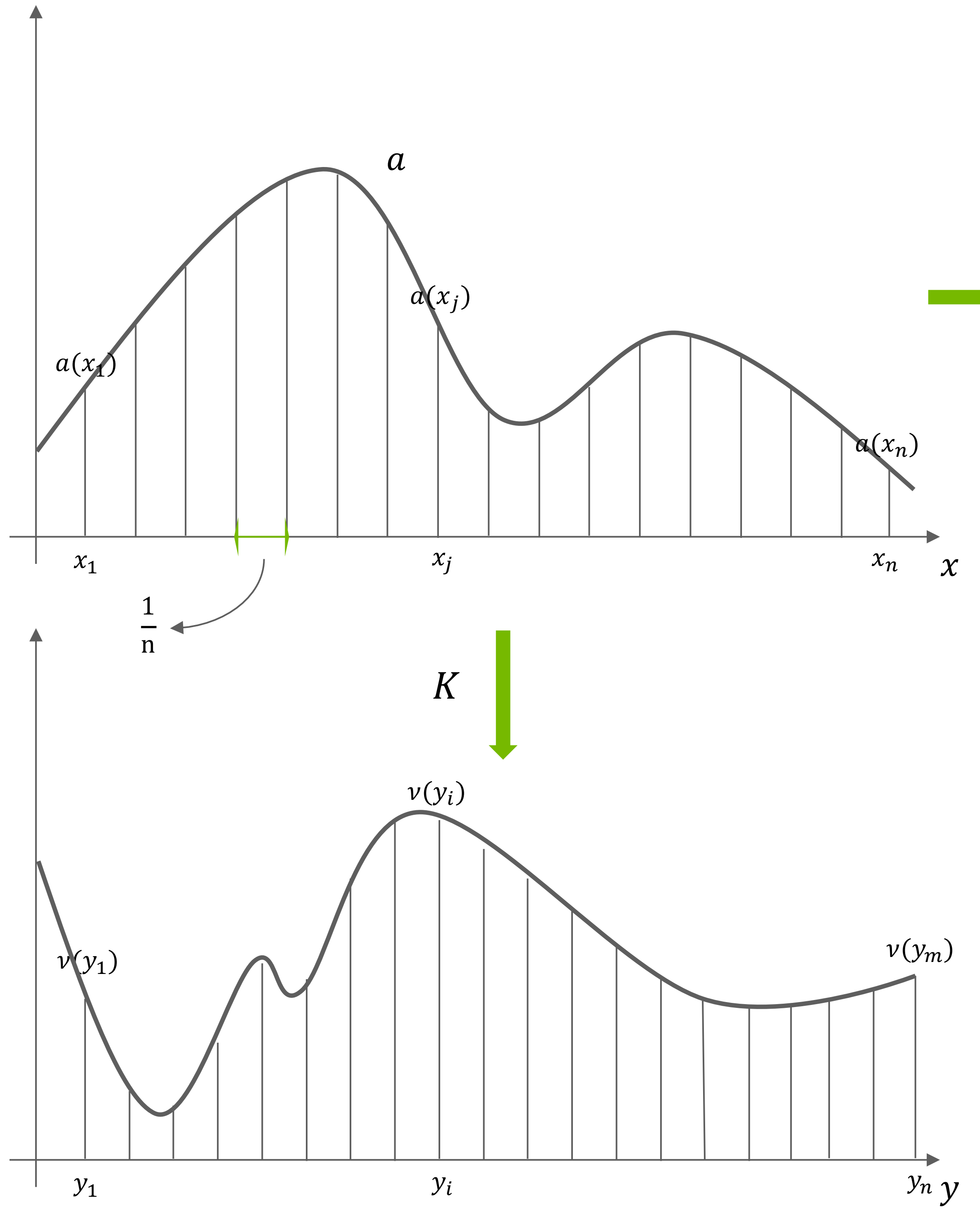
MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



K

Consider a basic feed forward neural network layer



$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ . \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$

First layer

$$v(y_i) = \sigma\left(\frac{1}{n}\sum_{j}^{n} K_{ij}\, a(x_j)\right)$$

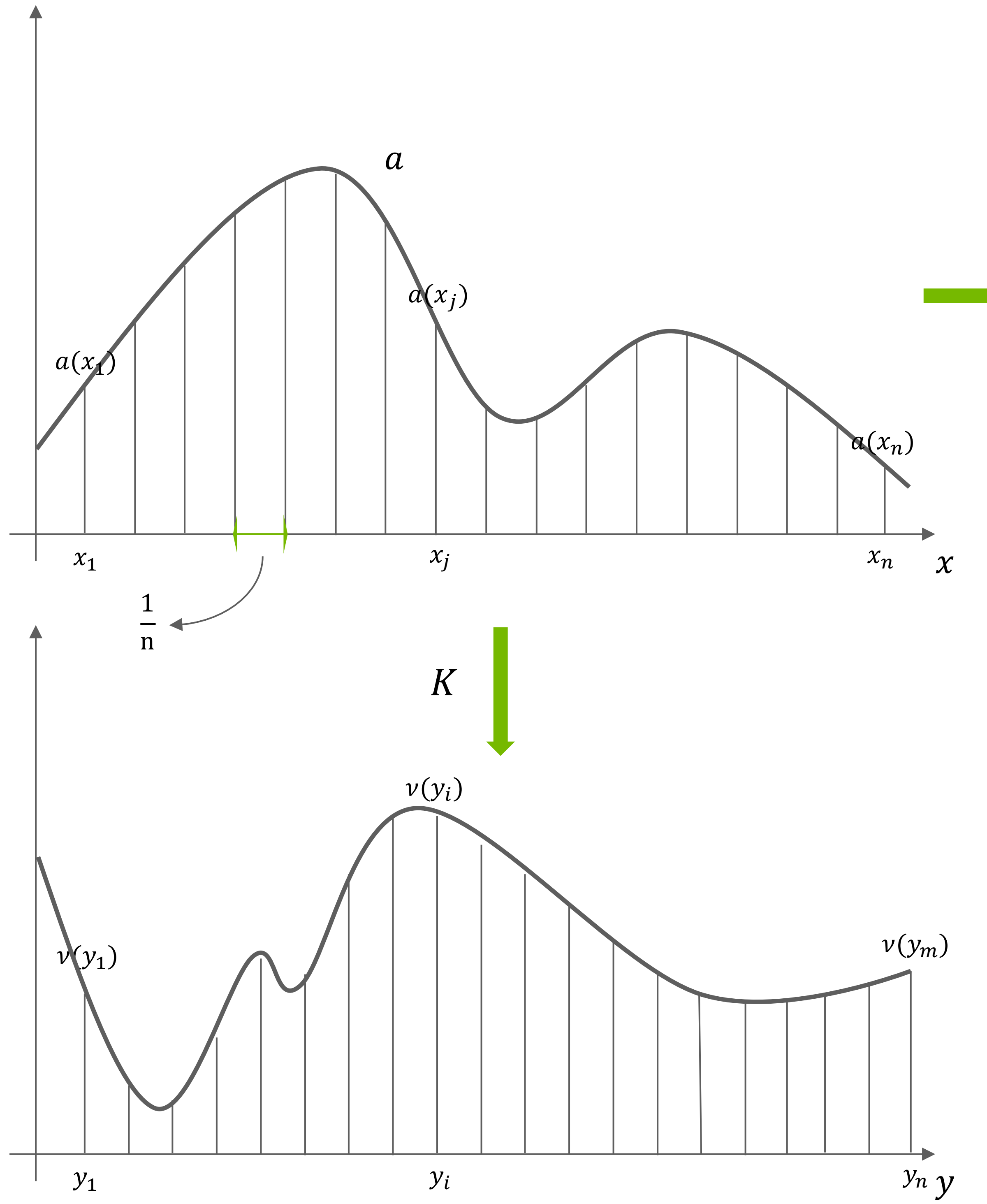
$$= \sigma\left(\frac{1}{n}\sum_{j}^{n} \kappa(y_i, x_j)\, a(x_j)\right)$$

$$= \sigma\left(\sum_{j}^{n} \kappa(y_i, x_j)\, a(x_j)\ \Delta x_j\right)$$

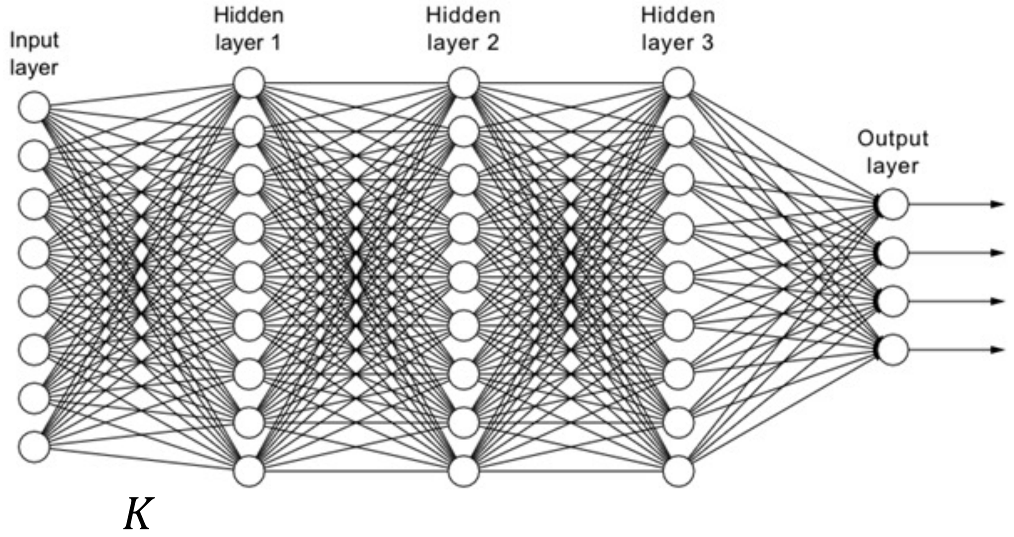
MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators

Consider a basic feed forward neural network layer



$$a \approx \begin{bmatrix} a(x_1) \\ a(x_2) \\ \vdots \\ a(x_{n-1}) \\ a(x_n) \end{bmatrix}$$



First layer

$$v(y_i) = \sigma\Big(\frac{1}{n}\sum_{j}^{n} K_{ij}a(x_j)\Big)$$

Linear model in conventional ML

(finite dimensional ML)

$$= \sigma\Big(\frac{1}{n}\sum_{j}^{n} \kappa(y_i, x_j)a(x_j)\Big)$$

$$= \sigma\Big(\sum_{j}^{n} \kappa(y_i, x_j)a(x_j)\ \Delta x_j\Big)$$

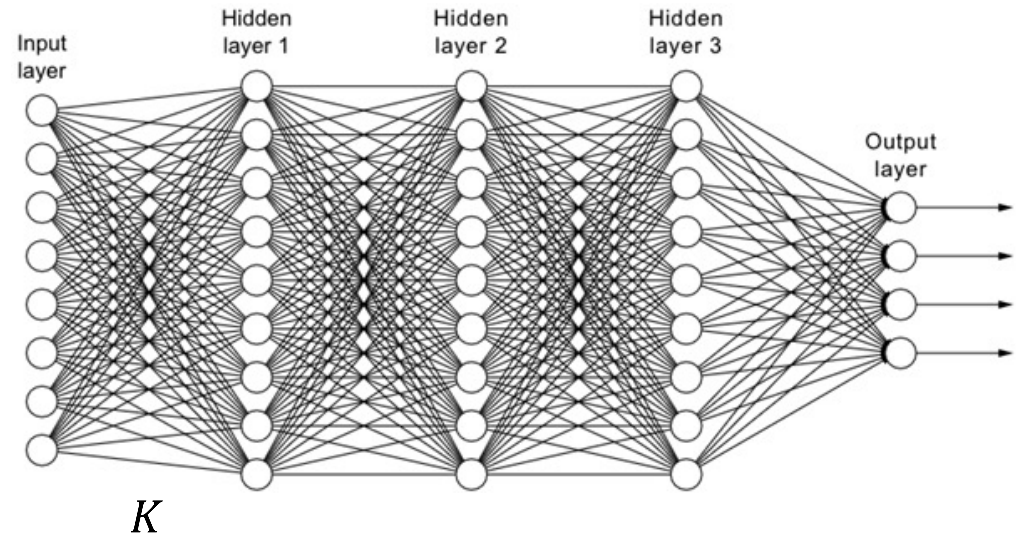
$$v(y) = \sigma\Big(\int \kappa(y, x)a(x)\, \mathrm{d}x\Big)$$

Linear integral operator in function spaces

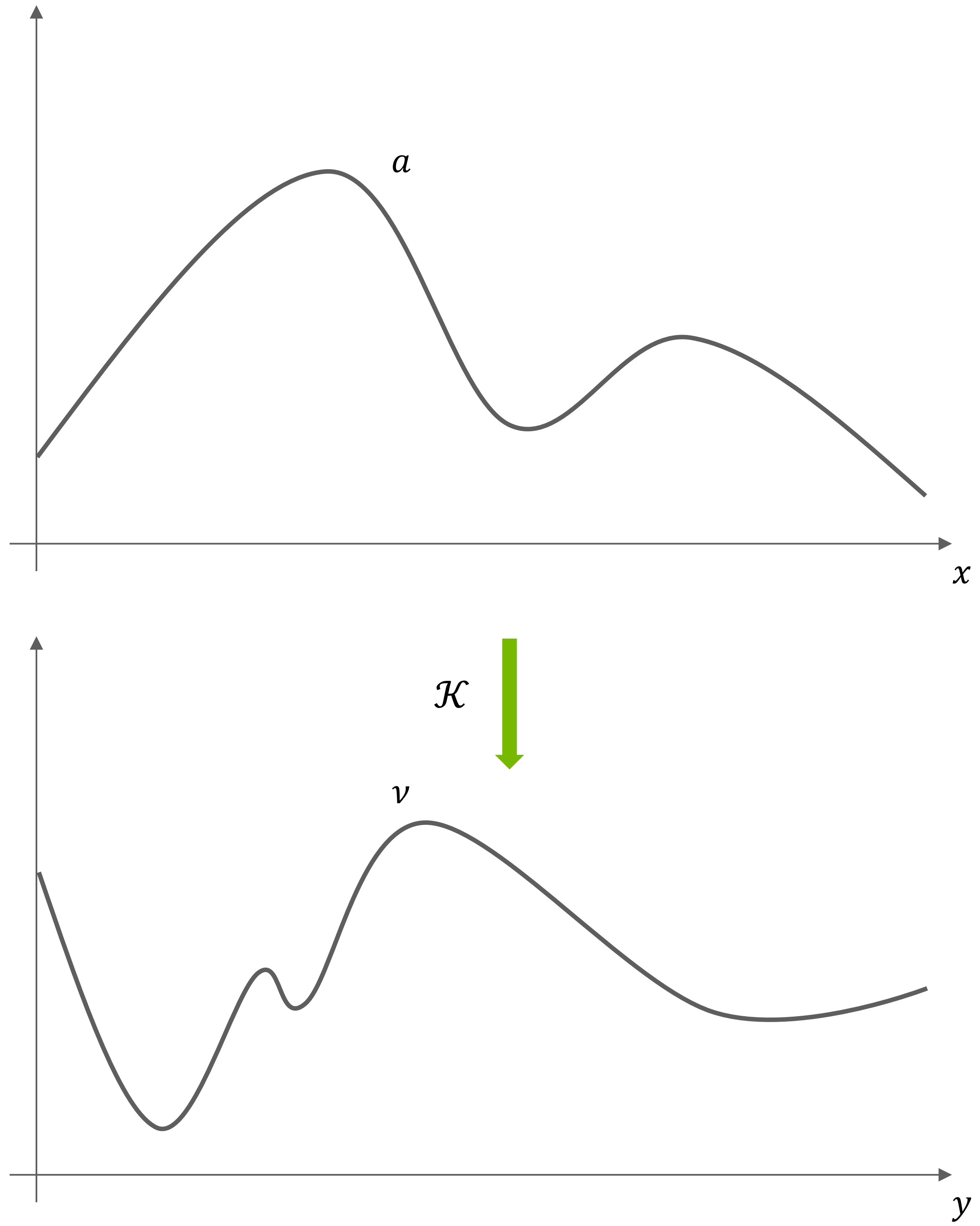
(infinite dimensional ML)

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



Basic neural operator layer



$$\mathcal{K}(a)(y) = \int \kappa(y, x) a(x)\, \mathrm{d}x$$

Note: Similar to linear model $y = Ax$ that is ubiquitous,

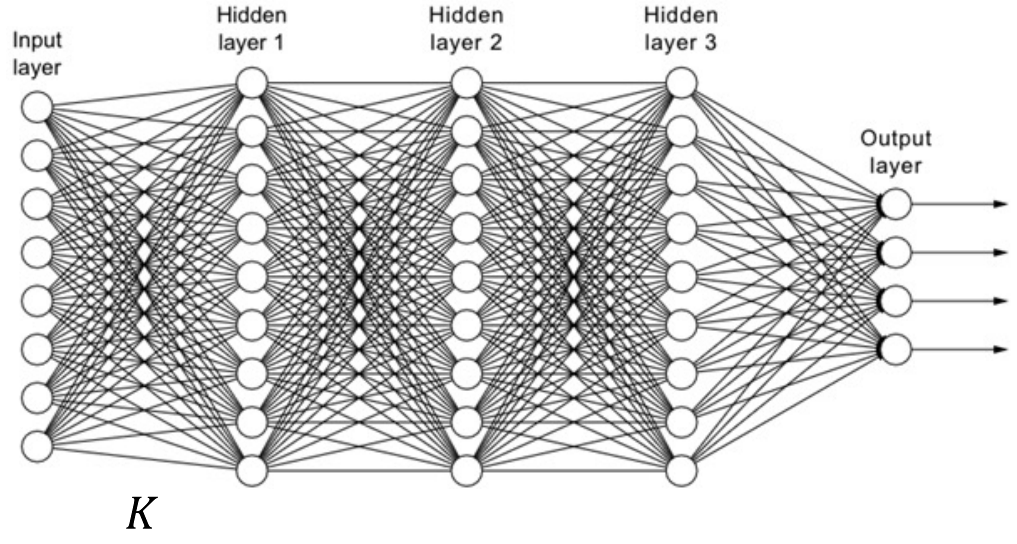
The linear integral operator $\int \kappa(y, x) a(x)\, \mathrm{d}x$ is also familiar and ubiquitous:

- Impulse response
- Green's function
- Frequency response, convolution
- …

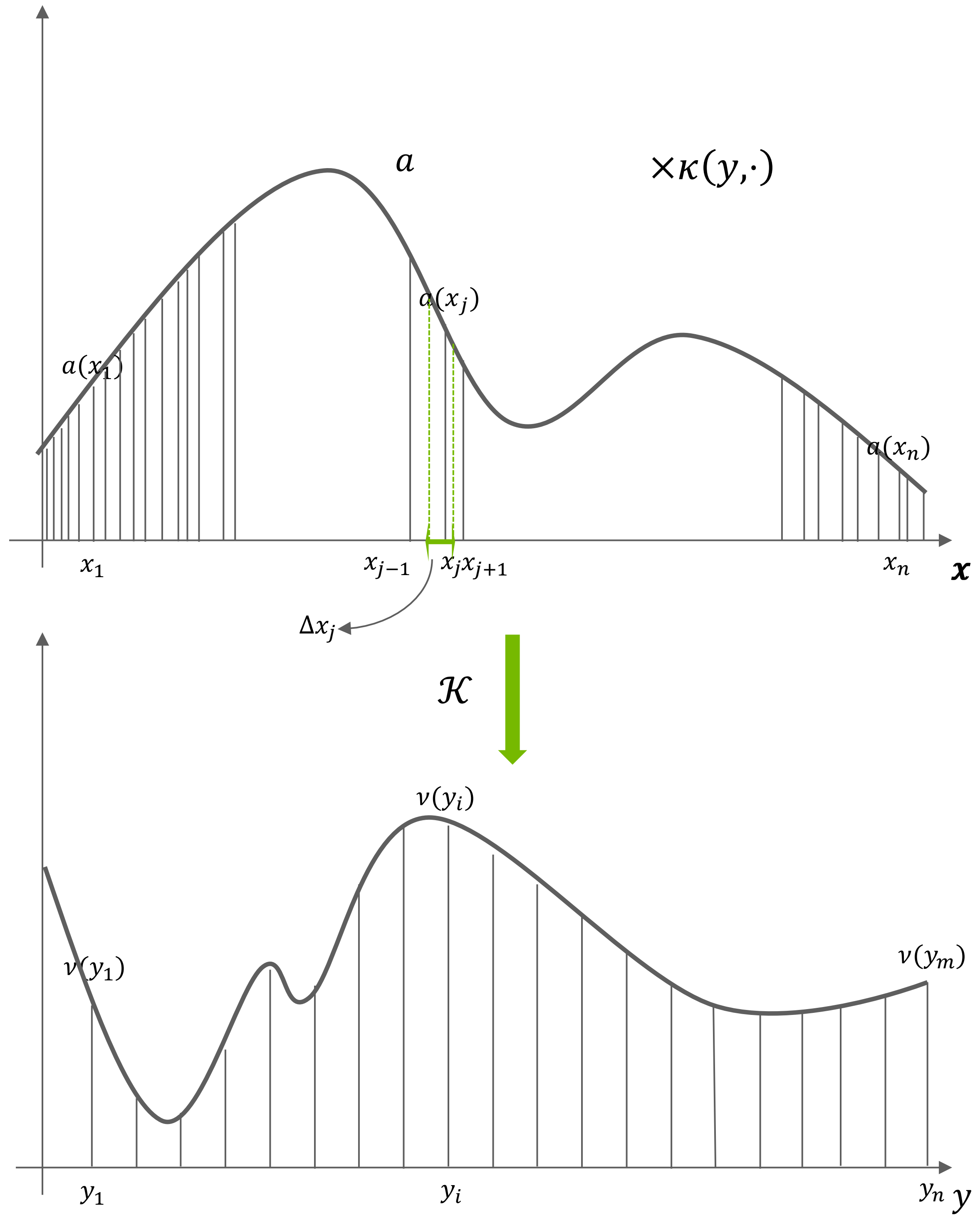
- Poisson eq, stationary 3D Schrödinger eq, wave eq, diffusion eq, harmonic oscillator, Gravity eq, heat eq, relativity eq, Feynman eq, …

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



Basic neural operator layer



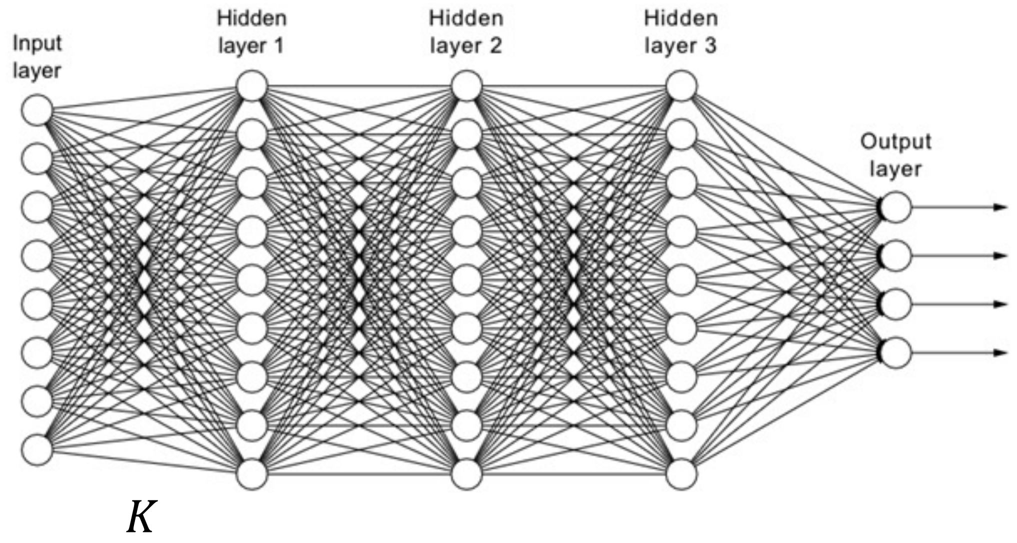
Output function can be evaluated at any point

$$v(y) = \sigma(\mathcal{K}(a))(y) = \sigma\left(\int \kappa(y,x)a(x)\,\mathrm{d}x\right) \approx \sigma(\sum_{j}^{n} \kappa(y,x_j)a(x_j)\ \Delta x_j)$$

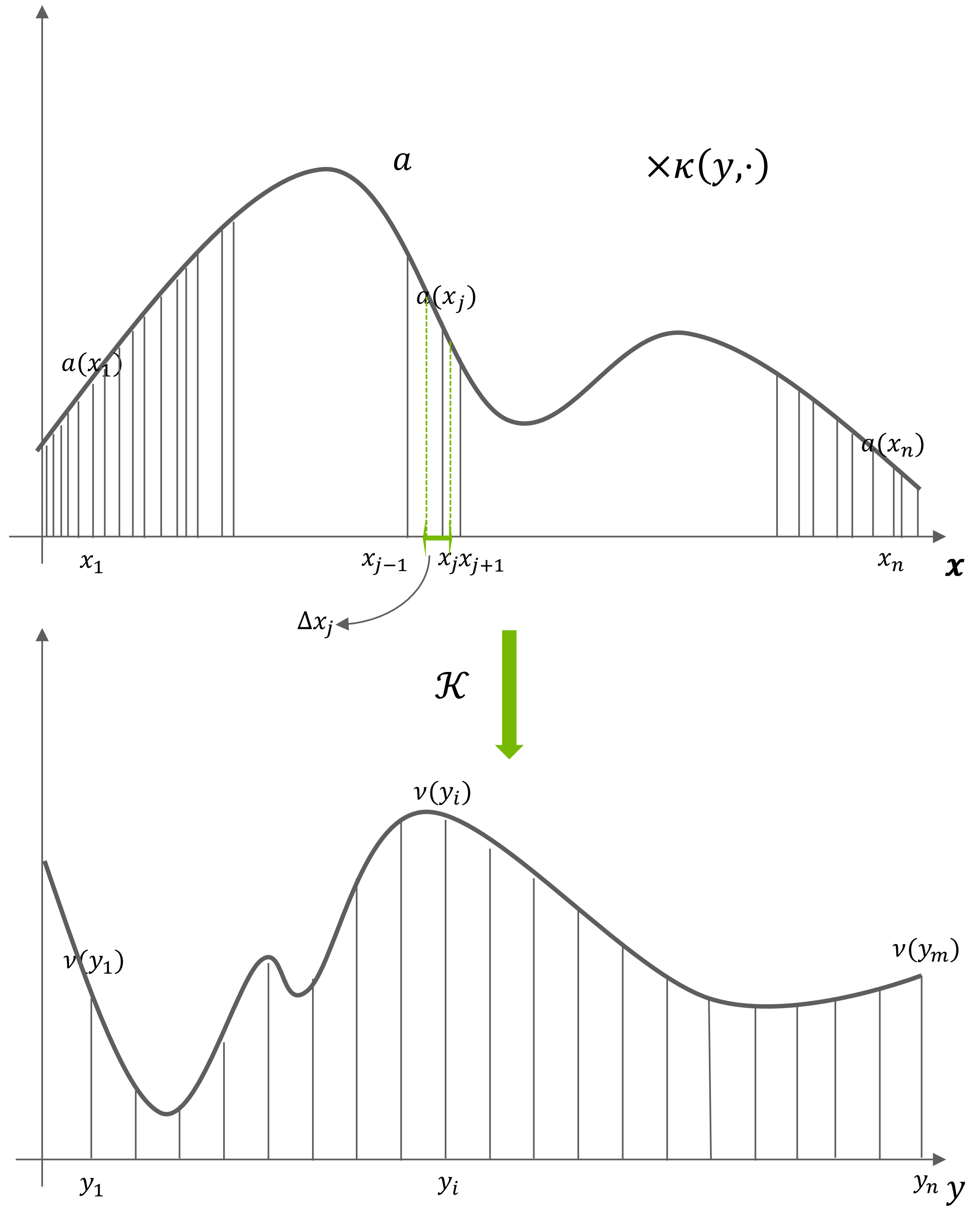
Input function at any resolution

MACHINE LEARNING ON FUNCTIONS

From neural networks to neural operators



Basic neural operator layer



Add more basic components

Proper measure/distribution

Residual connection

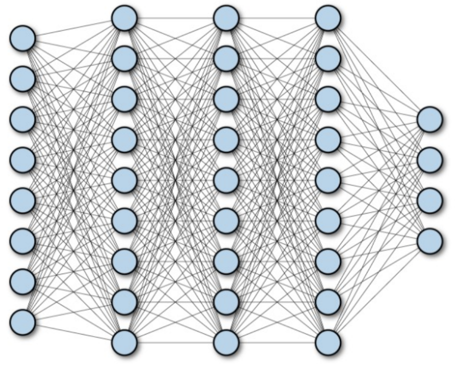
$$v(y) = \sigma\big(\mathcal{K}(a)\big)(y) = \sigma\Big(\int \kappa(y,x)a(x)\,\mathrm{d}\mu + Wa(y) + b(y)\Big)$$

Bias function

For simplicity, we keep x and y to belong to same space

NEURAL OPERATORS

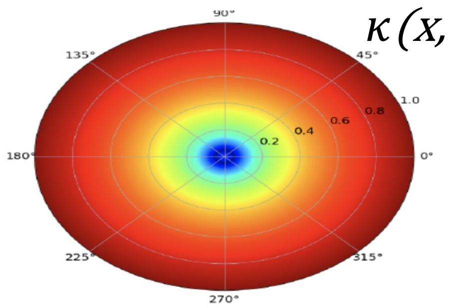
From neural networks to neural operators



Neural Networks, learn function $y = f(x)$

$Ax + w \cdot x + b$

Input:
Feature vector x → Linear function → Non-linearity → • • → Output:
Label vector y



$\kappa(x, y)$

Neural Operators, learn operator $\mathrm{u} = \mathcal{G}(a)$

$\int \kappa(x, y) a(y) d\mu + W(a(y)) + b(y)$

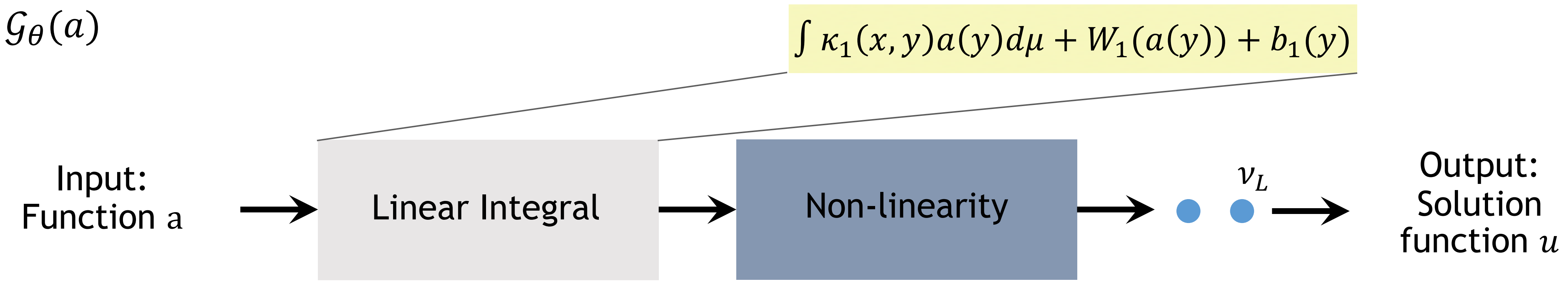
Input:
Function a → Linear Integral → Non-linearity → • • → Output:
Solution
function u

- Integral operator outputs functions (not just finite-dimensional vectors).
- Integral operator is discretization agnostic and discretization convergent.
- Neural Operators are universal approximator of operators.

NEURAL OPERATORS

Function to function map

Neural Operators, learn operator $\mathrm{u} = \mathcal{G}_\theta(a)$



- Composition of linear operators followed by point-wise non linearity

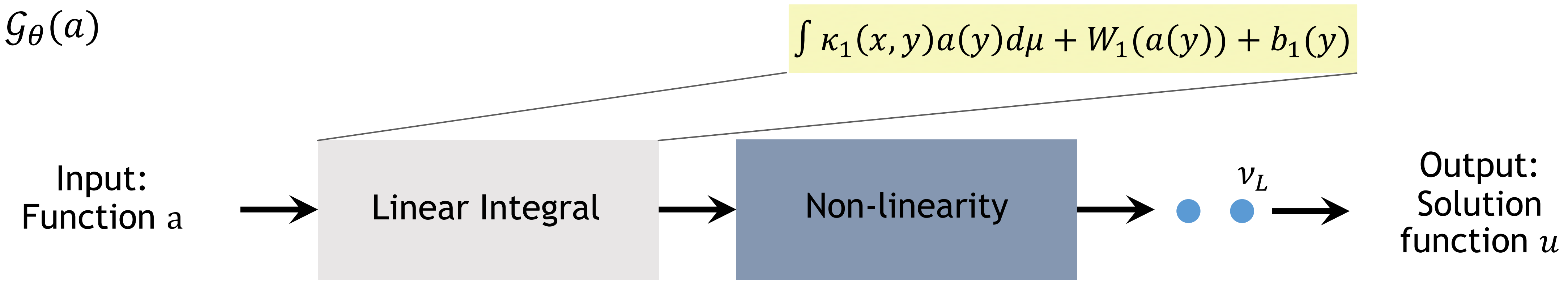
$$\mathrm{u} = \mathcal{G}(a) = \sigma\left(\mathcal{K}_L\left(\sigma\left(\mathcal{K}_{L-1}\left(\dots\left(\sigma(\mathcal{K}_1(a))\right)\dots\right)\right)\right)\right)$$

Recall neural networks

NEURAL OPERATORS

Function to function map

Neural Operators, learn operator $u = \mathcal{G}_\theta(a)$



- Integral operator outputs functions (not just finite-dimensional vectors).

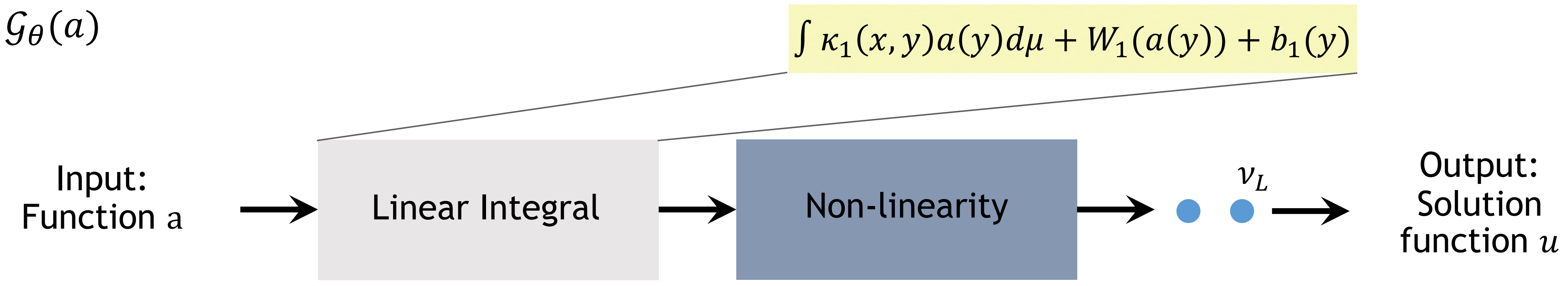
$$u(y) = \sigma(\mathcal{K}_L(v_L))(y) = \sigma\left(\int \kappa_L(y,x)v_L(x)\,\mathrm{d}x\right) \approx \sigma\Big(\sum_j^n \kappa_L(y,x_j)v_L(x_j)\ \Delta x_i\Big)$$

Output function can be evaluated at any point

NEURAL OPERATORS

Function to function map

Neural Operators, learn operator $\mathrm{u} = \mathcal{G}_\theta(a)$



- Integral operator is discretization agnostic and discretization convergent.

$$v_1(y) = \sigma(\mathcal{K}_1(a))(y) = \sigma\left(\int \kappa_1(y,x)a(x)\,\mathrm{d}x\right) \approx \sigma\Big(\sum_j^n \kappa_1(y,x_j)a(x_j)\ \Delta x_i\Big)$$

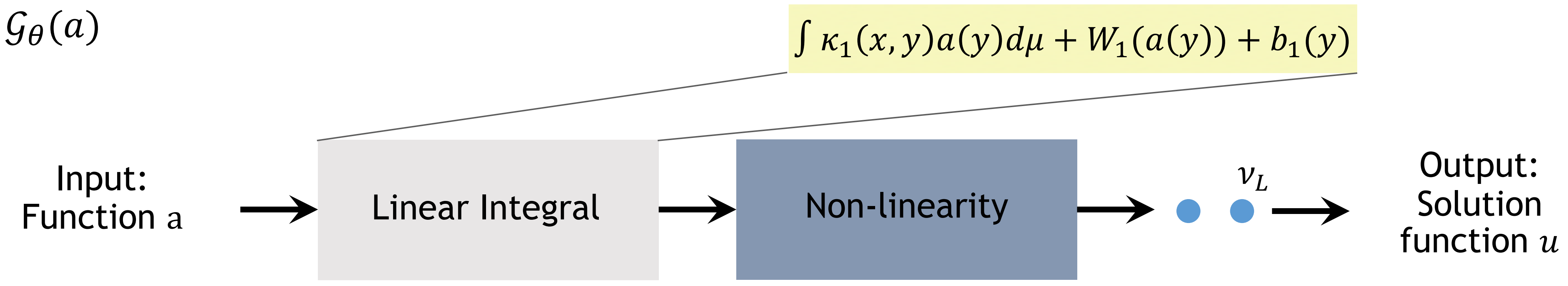
Input function can be provided at any discretization

As discretization gets finer (no matter what way), the operator converges to a unique operator in continuum.

NEURAL OPERATORS

Function to function map

Neural Operators, learn operator $\mathrm{u} = \mathcal{G}_\theta(a)$



- Neural Operators are universal approximator of operators.

$$\|\hat{\mathcal{G}}_\theta(D_L, a|_{D_L}) - \mathcal{G}^\dagger(a)\|_{\mathcal{U}} \leq \underbrace{\|\hat{\mathcal{G}}_\theta(D_L, a|_{D_L}) - \mathcal{G}_\theta(a)\|_{\mathcal{U}}}_{\text{discretization error}} + \underbrace{\|\mathcal{G}_\theta(a) - \mathcal{G}^\dagger(a)\|_{\mathcal{U}}}_{\text{approximation error}}.$$

Theorem (Universal approximation theorem of neural operators) :

Under a mild regularity condition, for any given arbitrary operator between general function spaces $\mathcal{G}^\dagger$, and any $\epsilon > 0$, there exist a neural operator $\mathcal{G}_\theta$, such that,

$$\sup_a \|\mathcal{G}^\dagger(a) - \mathcal{G}(a)\|_{\mathcal{U}} \leq \epsilon.$$

NEURAL OPERATORS

Function to function map

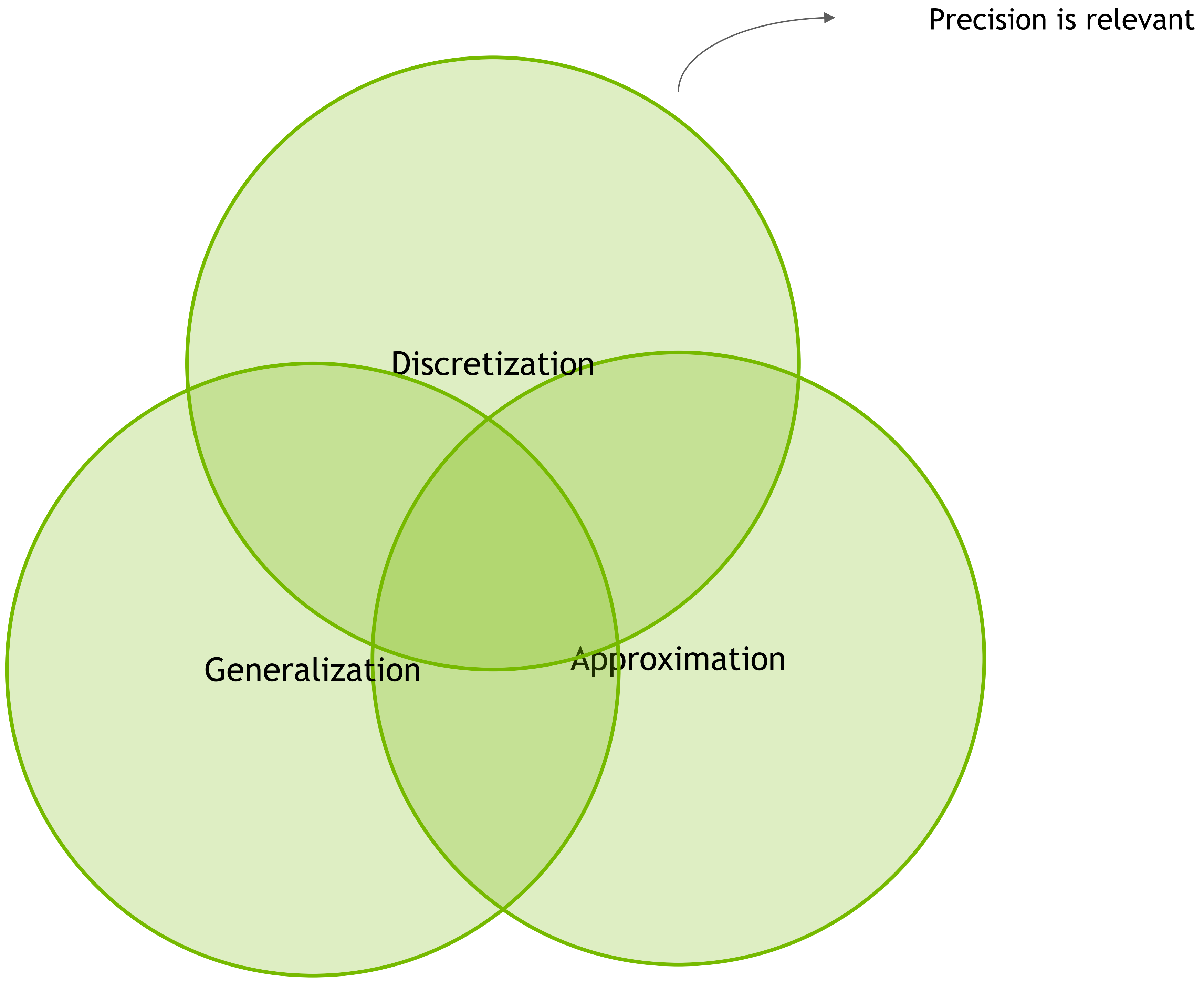
Conventional deep learning error analysis

- Generalization
- Approximation

Error analysis in operator learning

- Generalization
- Approximation
- Discretization

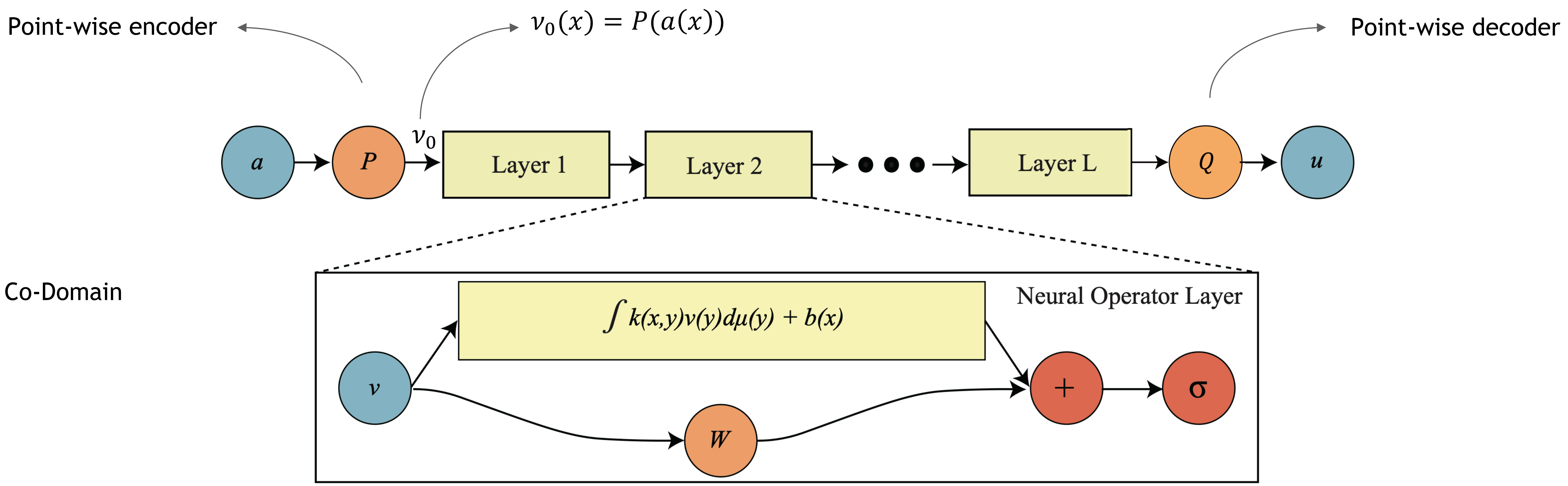
Learning happens on discretized data



NEURAL OPERATORS

Architectures

Neural Operator architecture



Domain

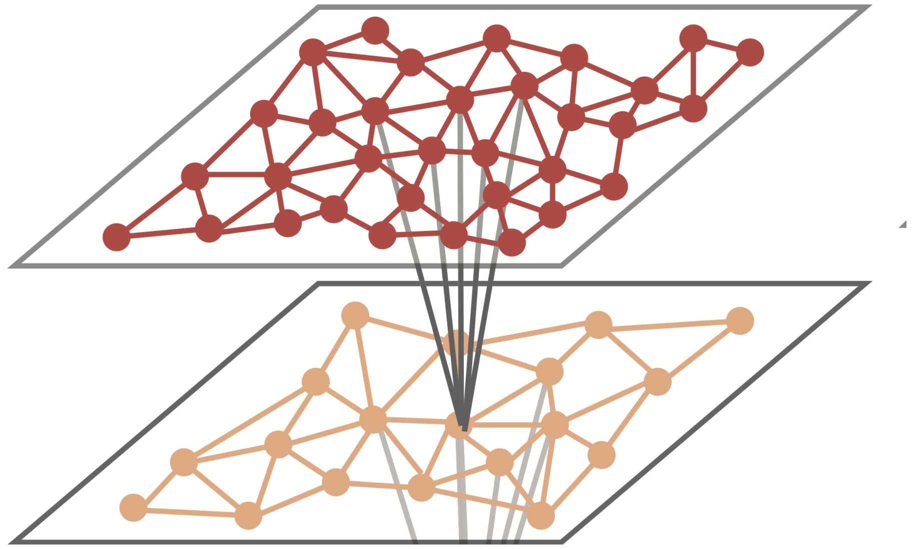
Co-Domain

$a: D \to \mathbb{R}^d$

E.g., weather:
Temperature, velocity, humidity, land mask, vorticity, precipitation, …

NEURAL OPERATORS

Architectures



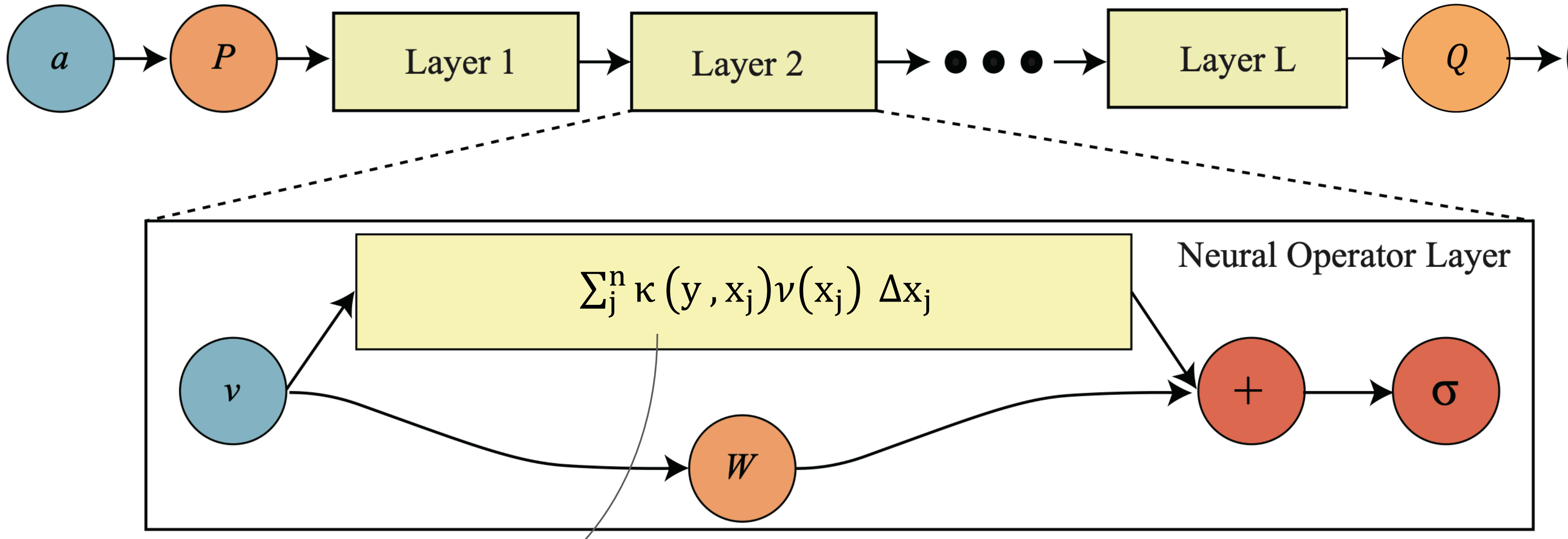
Neural Operator architecture

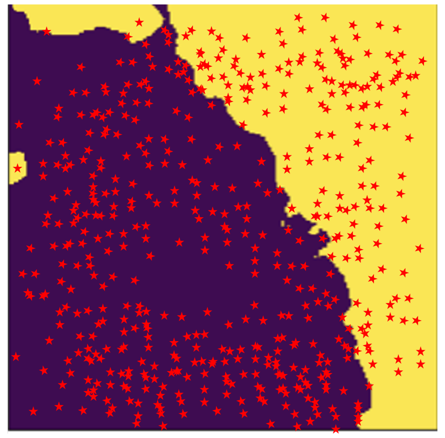
Represent the kernel using a neural network

Graph Neural Operator (GNO) “also referred to as kernel NO)

Or $\sum_j^n \kappa\left(y, x_j, \nu(x_j)\right)\nu(x_j)\ \Delta x_j$,

and many more variants

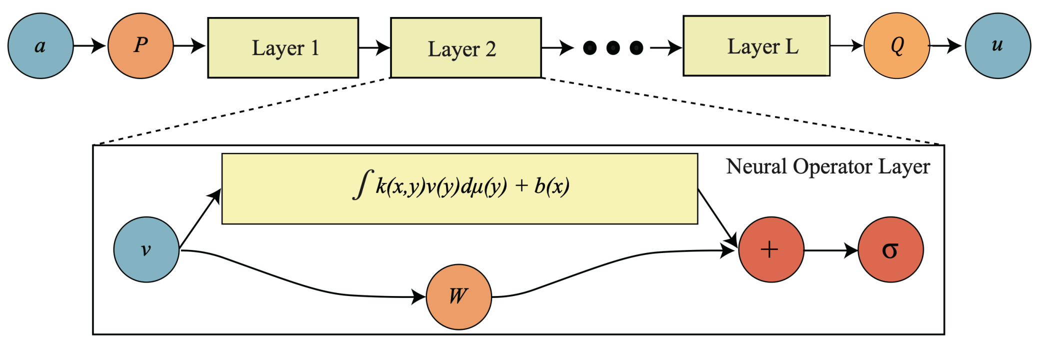




Learnable neural network

Local or Global

Generalization of GNNs to neural operators



NEURAL OPERATORS

Architectures

ψ_k Rep. functions for output space

Project input function on ϕ_k

What are the ways to compute integrals?

$$\int \kappa(y,x)a(x)\,\mathrm{d}x = \sum_k (R_k \cdot < \phi_k, a >)\,\psi_k(y)$$

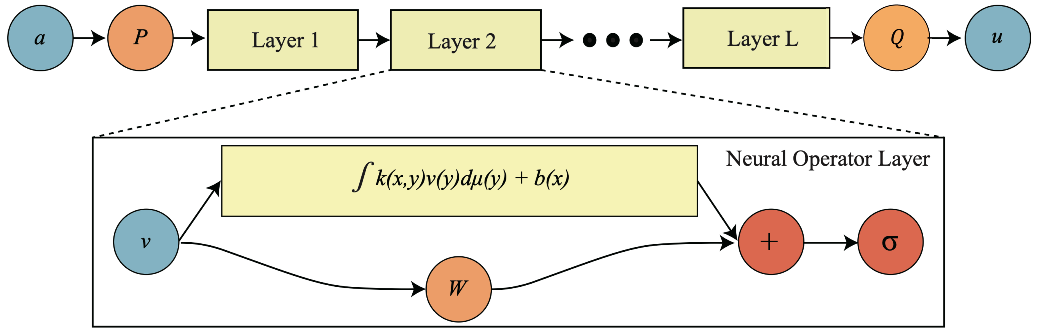
Project onto ϕ_k and project back to ψ_k

$$= \sum_k \left(R_k \cdot \int \phi_k(x)a(x)dx \right) \psi_k(y)$$

$$\approx \sum_k \left(R_k \cdot \sum_j^n \phi_k(x_j)a(x_j)\Delta x_j \right) \psi_k(y)$$

Residual connection

$$v(y) = \sigma(\mathcal{K}(v))(y) = \sigma(\int \kappa(y,x)v(x)\,\mathrm{d}\mu + Wv(y) + b(y))$$



NEURAL OPERATORS

Architectures

What are the ways to compute integrals?

$\int \kappa(y,x)a(x)\,\mathrm{d}x$ Integral kernel operator

When ϕ_k and ψ_k are Fourier bases → similarity to convolution

(inspired by Fluid dynamics where Fourier bases or ubiquitous)

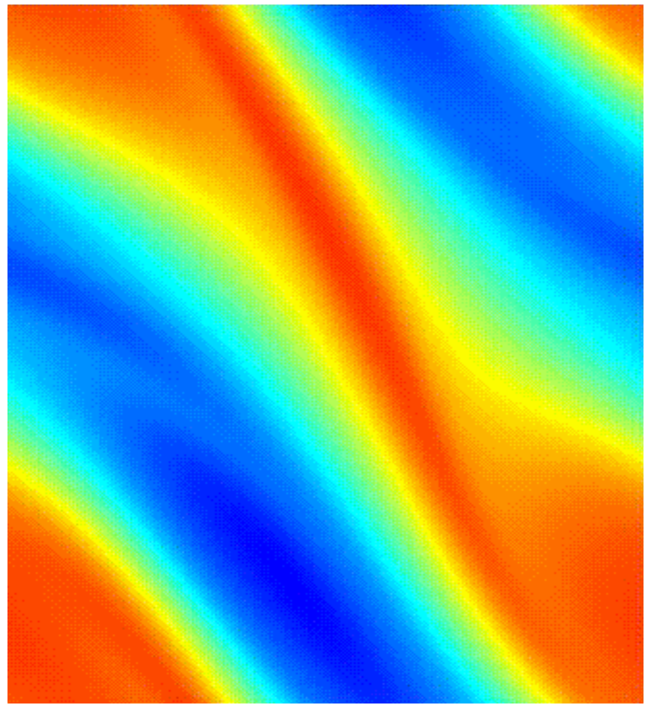
$\int \kappa(y-x)a(x)\,\mathrm{d}x$ Convolution

Learn κ function

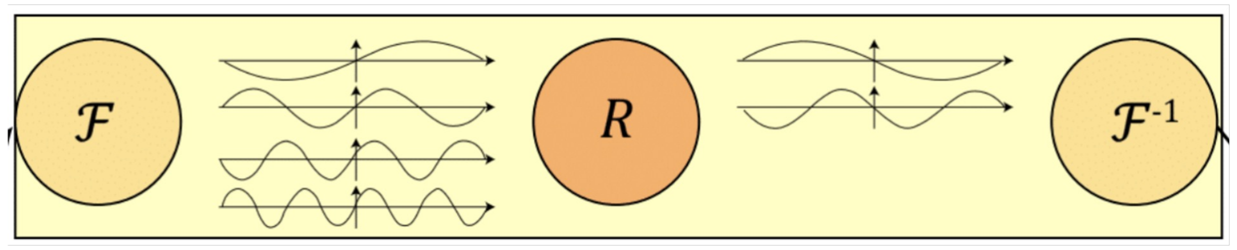
$\mathcal{F}^{-1}\big(\mathcal{F}(\kappa)\cdot\mathcal{F}(a)\big)$ Fourier Domain

Learn R matrix

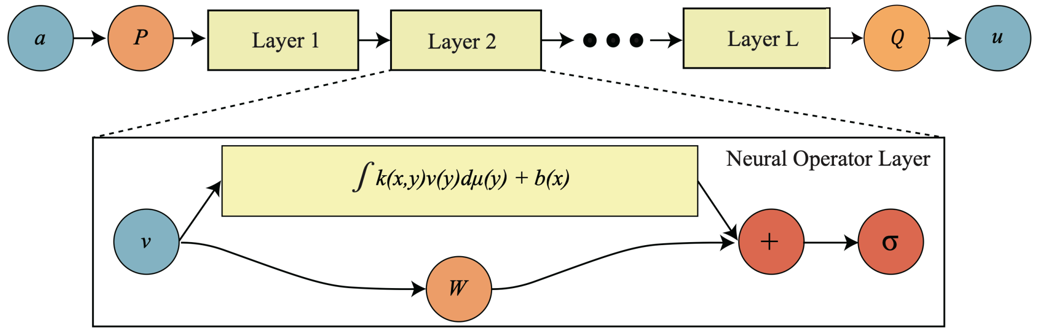
$\mathcal{F}^{-1}\big(R\cdot\mathcal{F}(a)\big)$ Fourier Domain



Learn weights R in Fourier Domain



Generalization of CNN with large kernels to neural operators



NEURAL OPERATORS

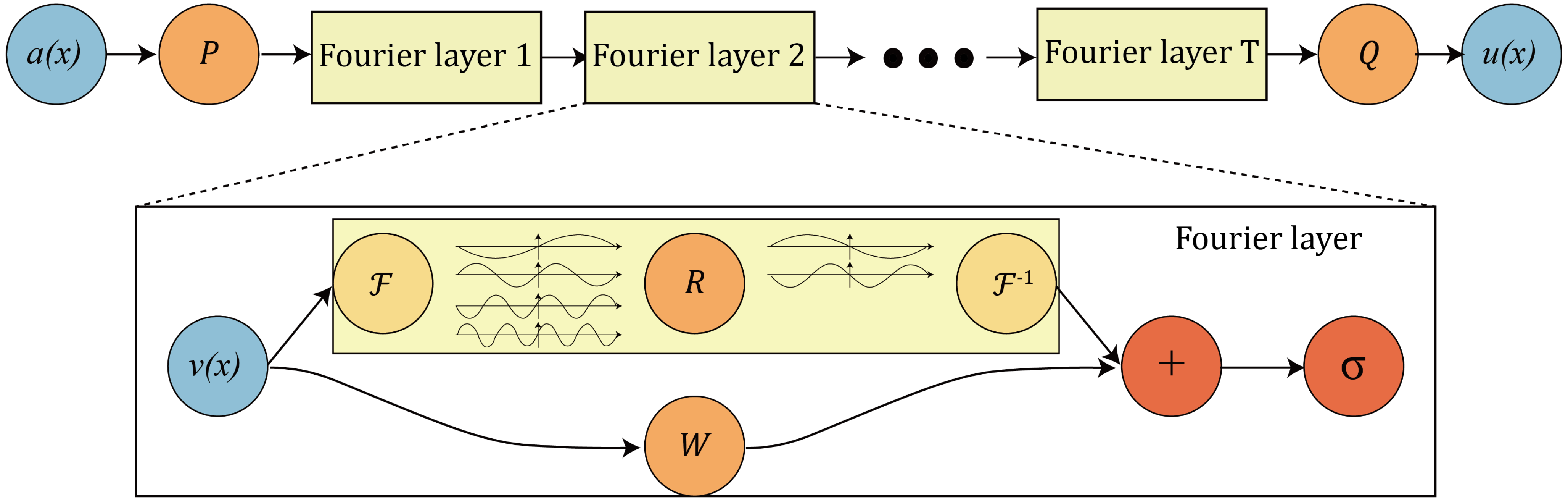
Architectures

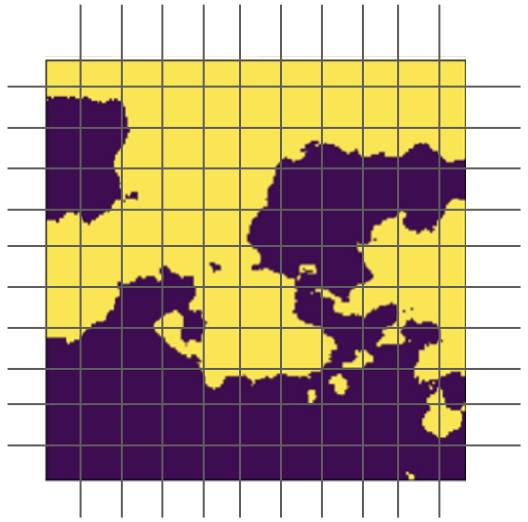
What are the ways to compute integrals?

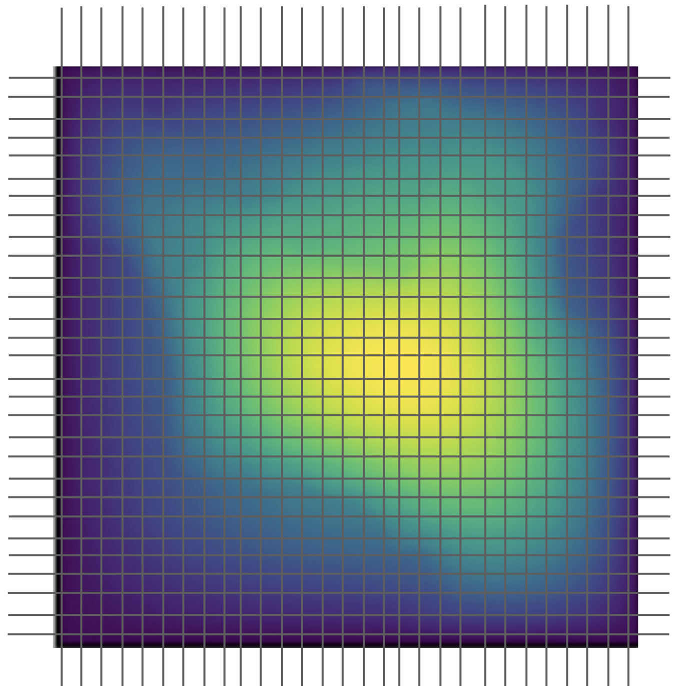
Fourier Neural Operator (FNO)

When the input function is given on a regular grid,

The integral is approximated using FFT







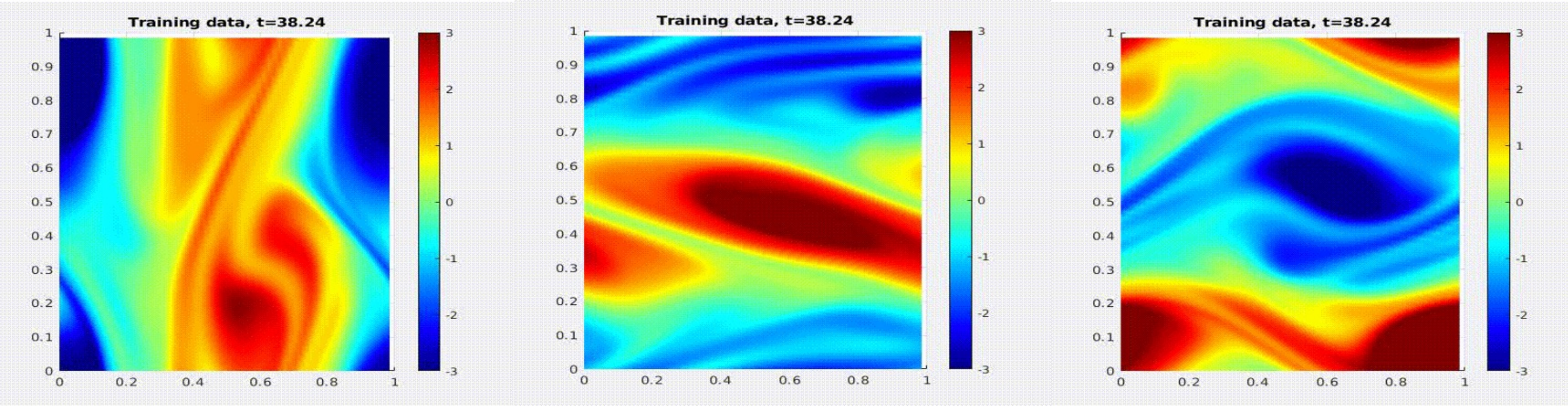
$$\int e^{-i\omega_k x}\, a(x)dx \approx \frac{1}{n}\sum_{j}^{n} e^{-i\omega_k x_j} a(x_j)$$

Discrete Fourier transform → Fast Fourier transform

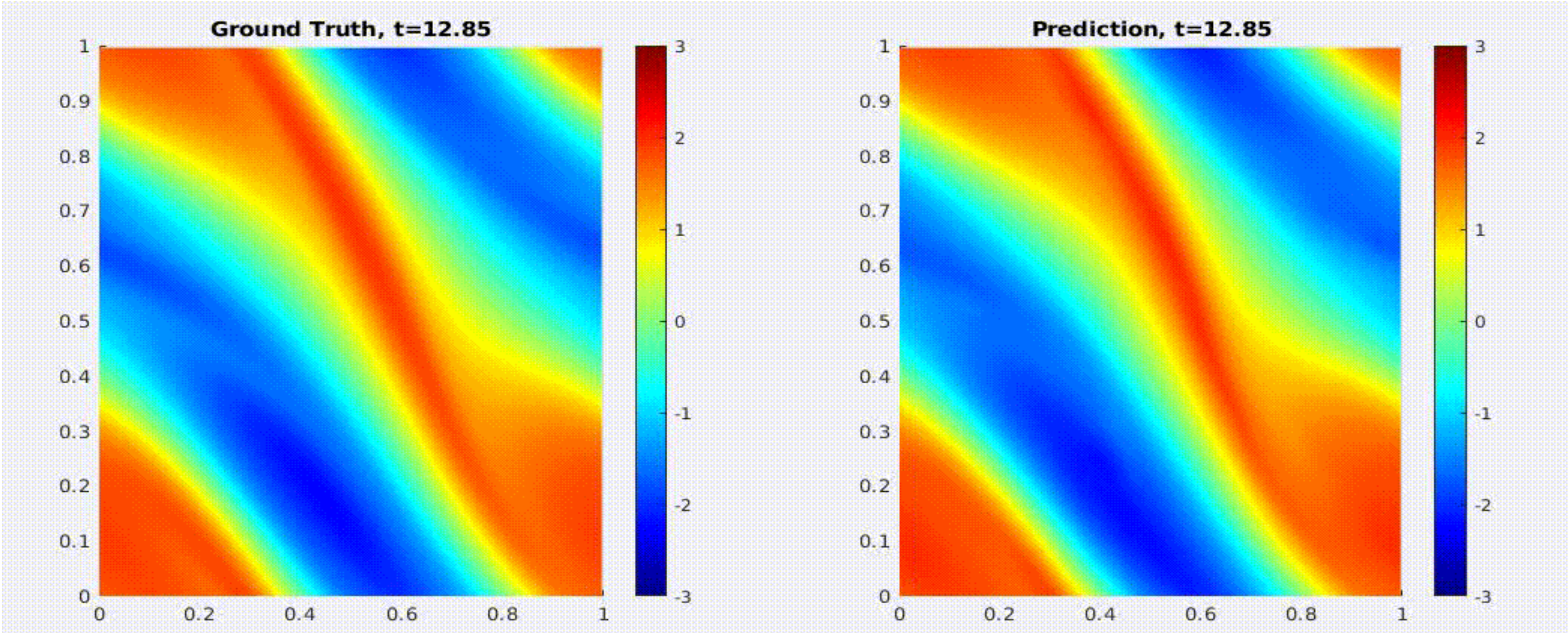
NEURAL OPERATORS

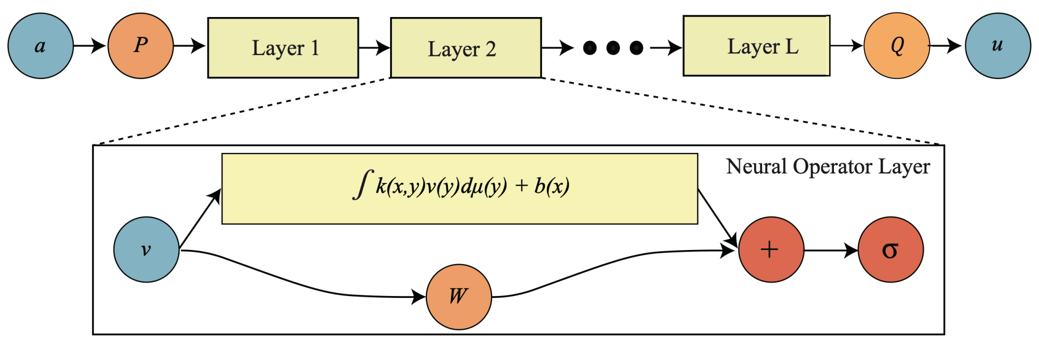
Discretization agnostic, zero shot super resolution

Train using coarse resolution data



Directly evaluate on higher resolution (no re-training)





NEURAL OPERATORS

Architectures

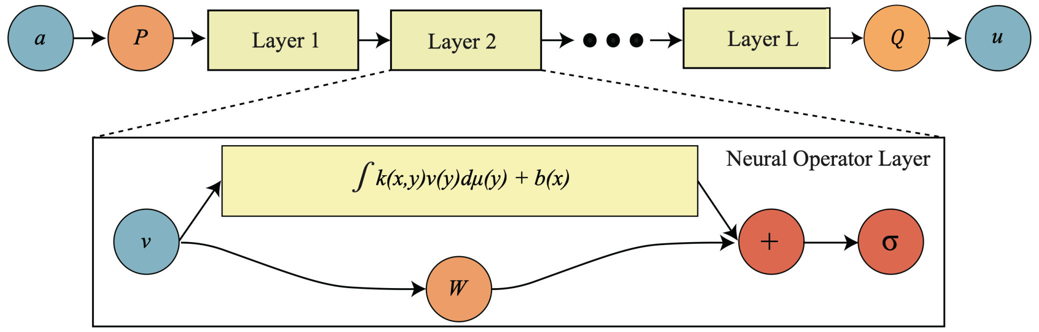
What are the ways to compute integrals?

- Wavelet (WNO)
- PCA (PCA-NO)
- Laplace (LNO)
- Represented by an implicit neural network (GNO, DeepONet-NeuralOperator)

$$\int \kappa(y,x)a(x)\,\mathrm{d}x = \sum_k (R_k \cdot < \phi_k, a >)\, \phi'_k(y)$$

$$= \sum_k \left(R_k \cdot \int \phi_k(x)a(x)dx \right) \phi'_k(y)$$

$$\approx \sum_k \left(R_k \cdot \sum_j^n \phi_k(x_j)a(x_j)\Delta x_j \right) \phi'_k(y)$$



NEURAL OPERATORS

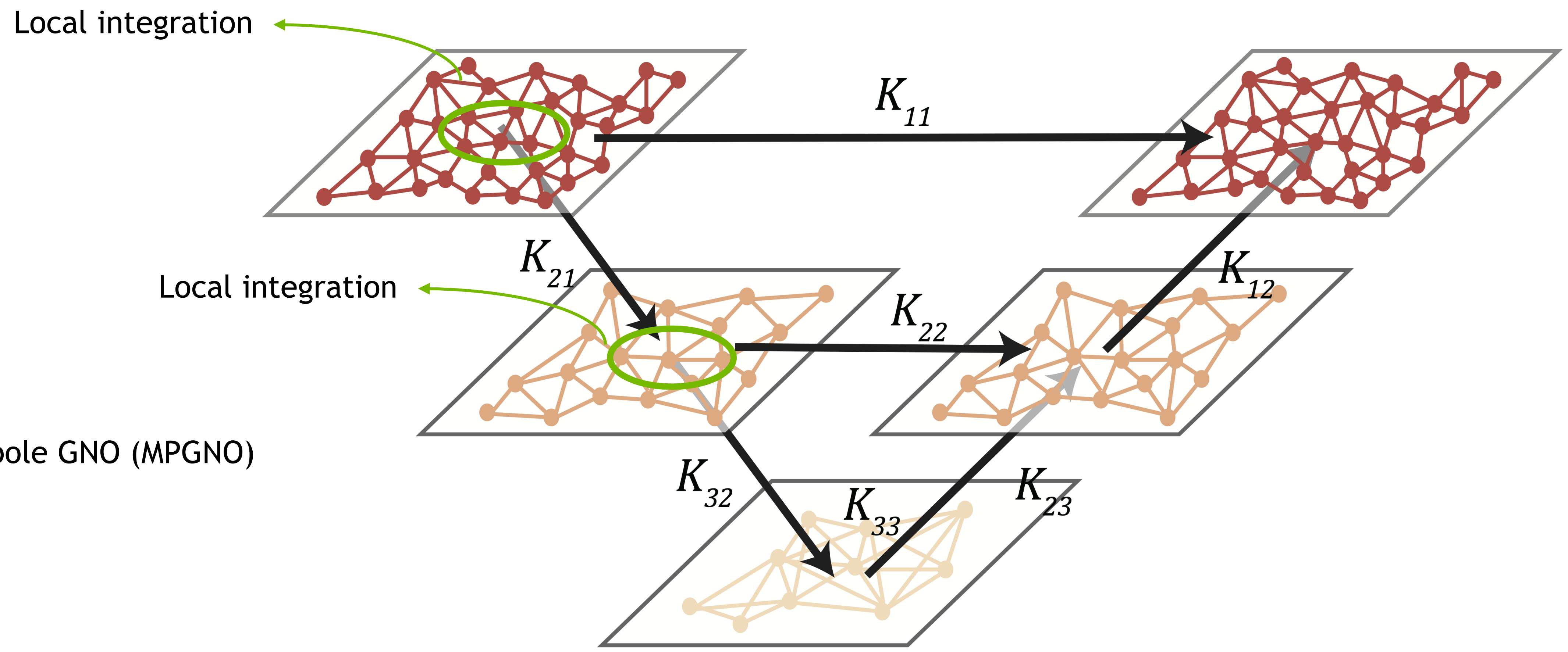
Architectures

What are the ways to compute integrals?

$$\int \kappa(y,x)v(x)\,\mathrm{d}x \approx \sum_{j}^{n} \kappa\left(y, x_j\right)v\left(x_j\right)\,\Delta x_i$$

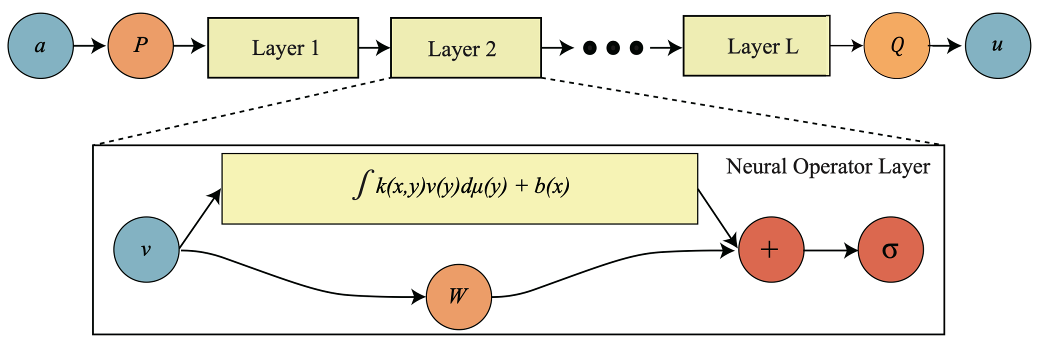
In the last few centuries, many methods are developed to compute integrals

- Gaussian quadrature ($\omega_i \Delta x_i$)
- Galerkin method (Gaussian Pyramid)
- Muti-grid method (Pyramid+Yolo)
- Muti-pole method (UNet)
- …



Multi-pole GNO (MPGNO)

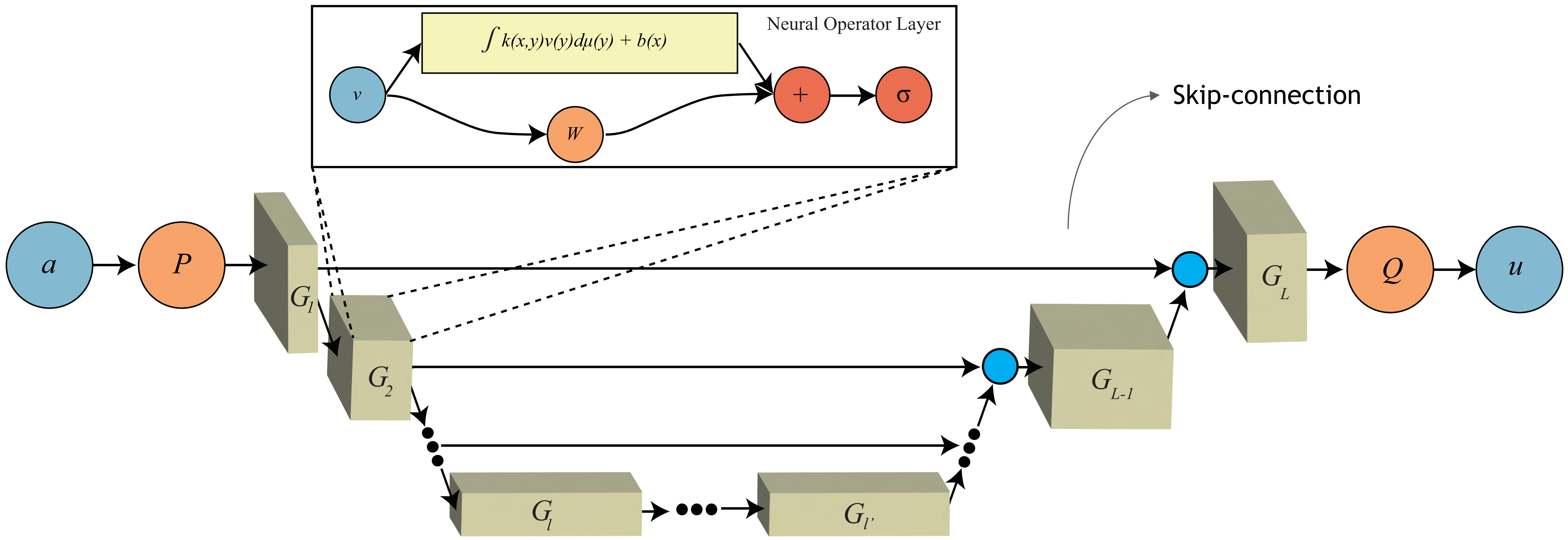
Generalization of UNet to graph based Neural Operators

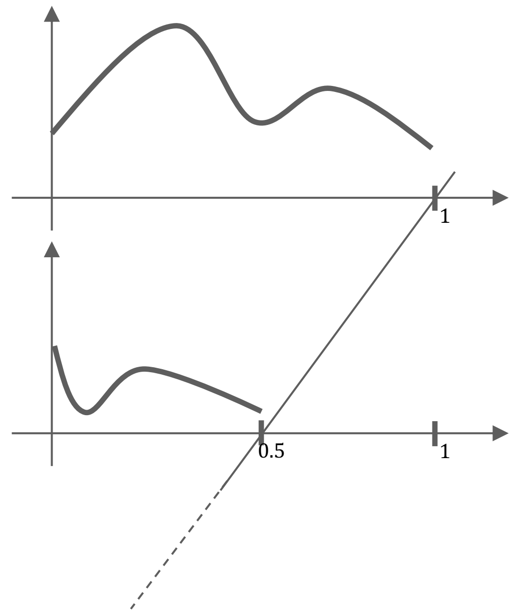


NEURAL OPERATORS

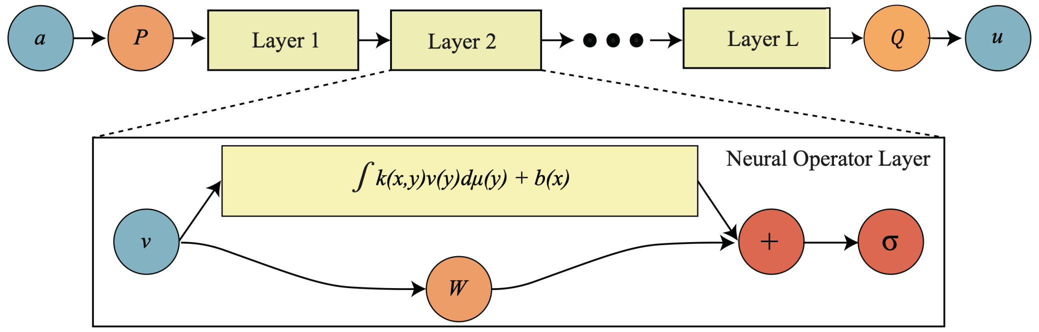
Architectures

U-shaped neural operator (UNO)





Shrinking domain size



NEURAL OPERATORS

Architectures

Key

Query

Value

Transformer Neural Operator

$$\sum_j \frac{\exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right)}{\sum_j \exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right)} v(\nu(y_i)),$$

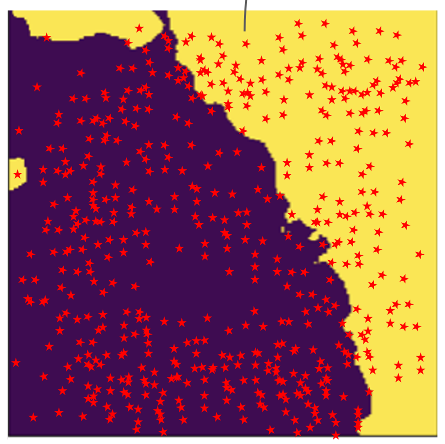
From neural networks to neural operators

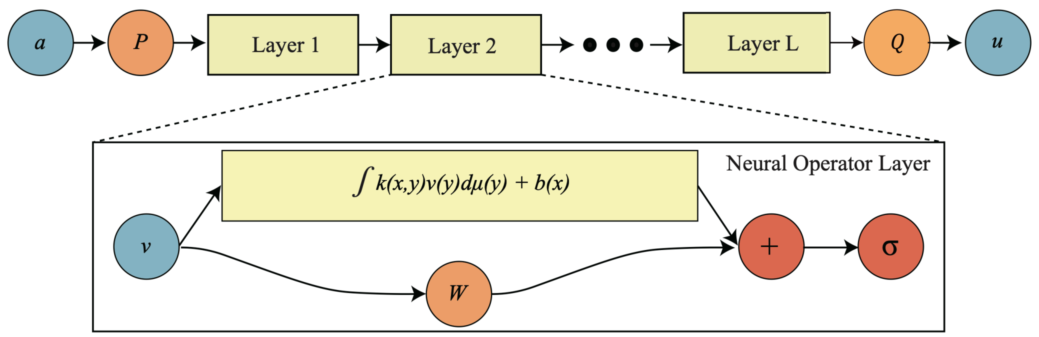
$$\int_x \frac{\exp\left(k(\nu(x))^\top q(\nu(y))\right)}{\int_x \exp\left(k(\nu(x)^\top q(\nu(y))\right) dx} v(\nu(y))dx,$$

Discretization

Tokens after $P \rightarrow \nu(x)$

$$\sum_j \frac{\exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right)}{\sum_j \exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right) \Delta x_j} v(\nu(y_i))\Delta x_j,$$





NEURAL OPERATORS

Architectures

When grid/mesh is regular

Transformer Neural Operator

Key

Query

Value

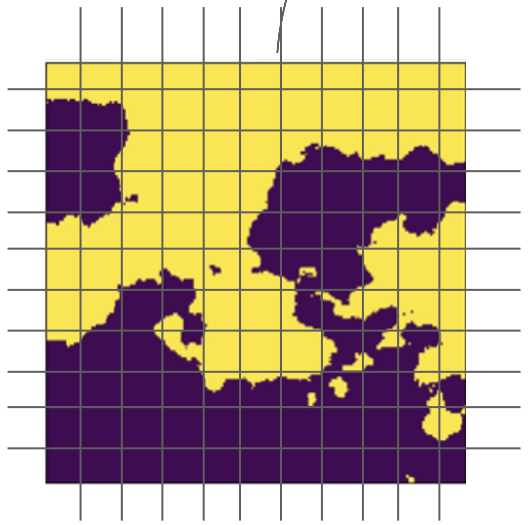
$$\sum_j \frac{\exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right)}{\sum_j \exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right)} v(\nu(y_i)),$$

$$\int_x \frac{\exp\left(k(\nu(x))^\top q(\nu(y))\right)}{\int_x \exp\left(k(\nu(x))^\top q(\nu(y))\right) dx} v(\nu(y))dx,$$

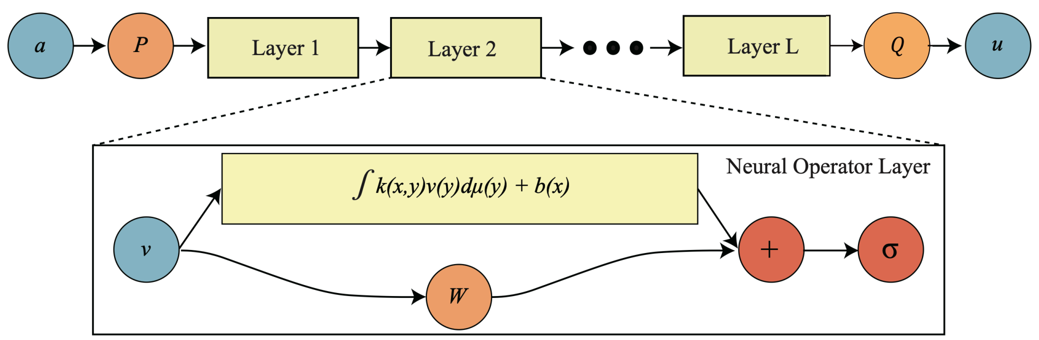
$$\sum_j \frac{\exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right)}{\sum_j \exp\left(k(\nu(x_j))^\top q(\nu(y_i))\right) \cancel{\Delta x_j}} v(\nu(y_i))\cancel{\Delta x_j},$$

$$\frac{1}{n}$$

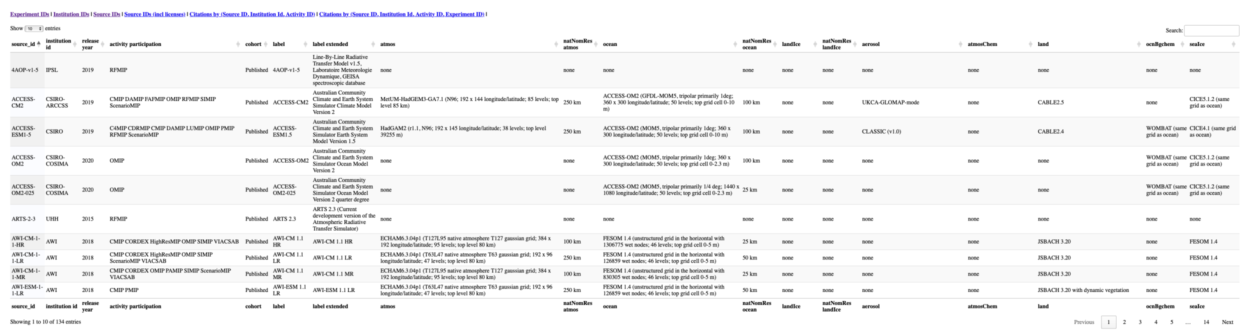
Tokens after $P \rightarrow \nu(x)$



NEURAL OPERATORS

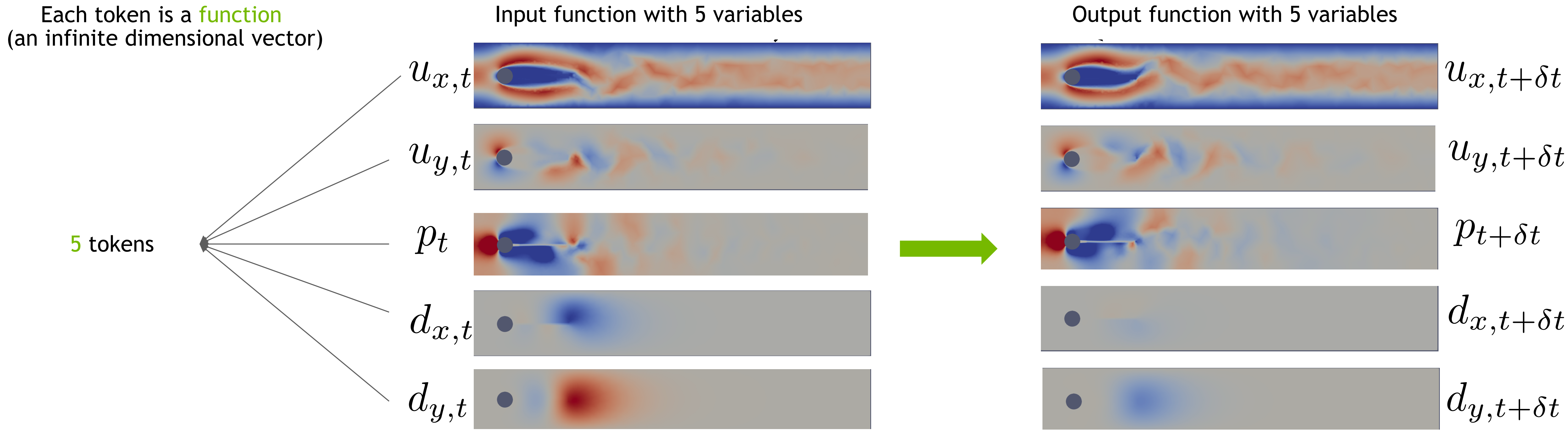


Architectures

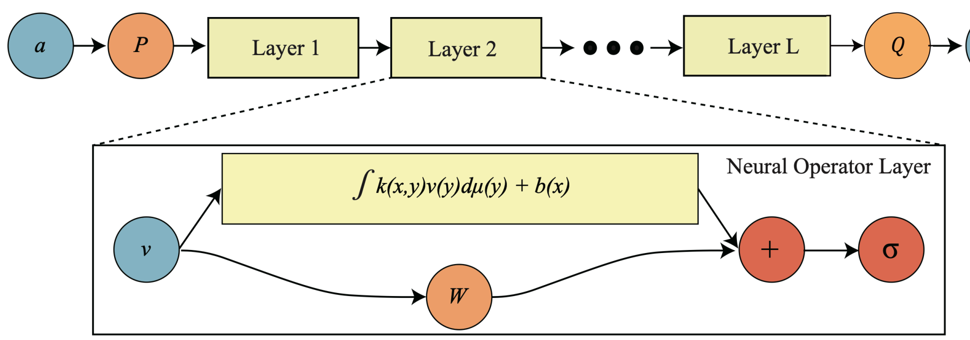


Many real-world datasets have different number of variables, e.g., coupled PDEs, climate data (collaboration between many countries) …

Co-domain Attention Neural Operator (CoDANO)

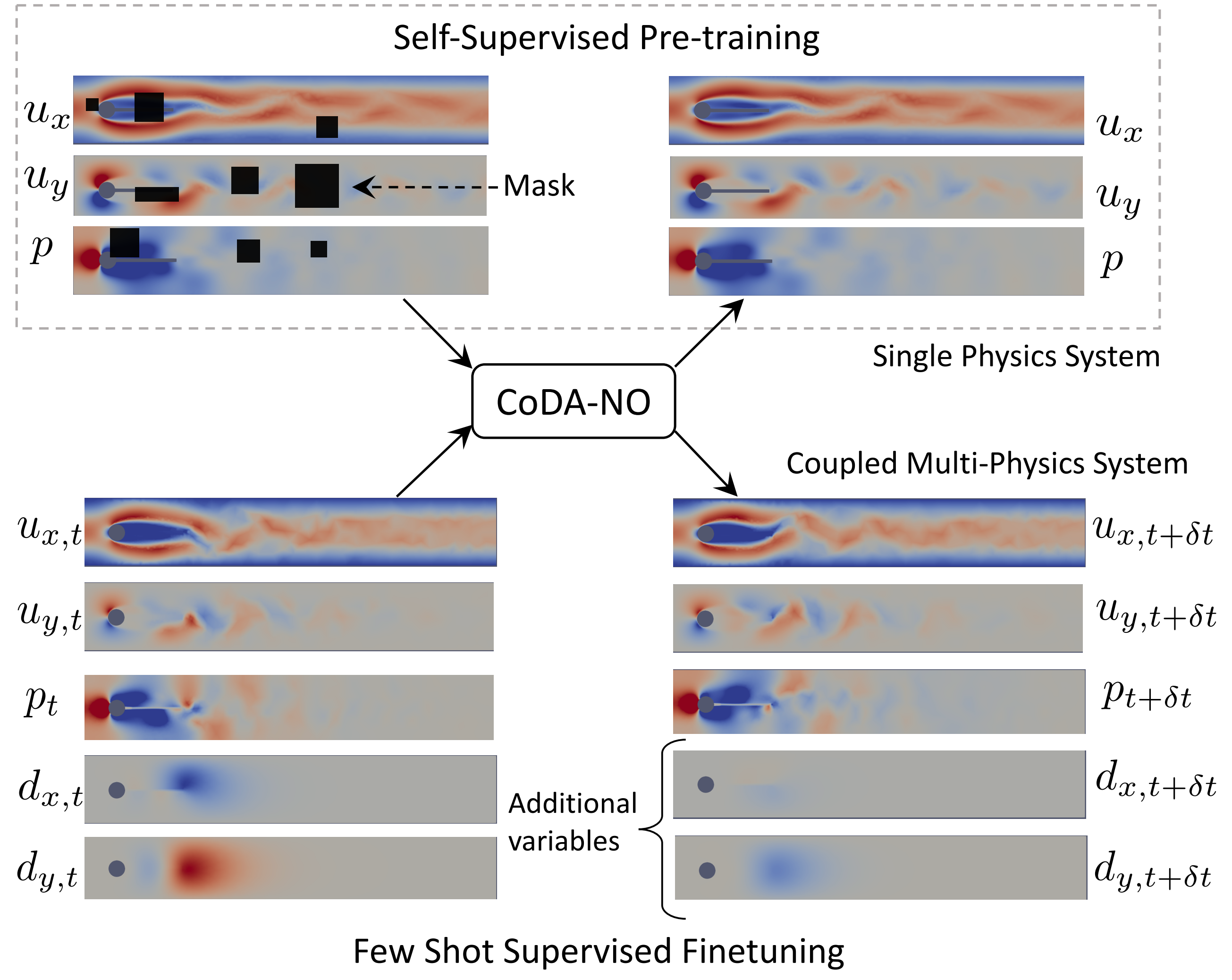


Query, key, value matrices in transformers become become integral operators in CoDANO.



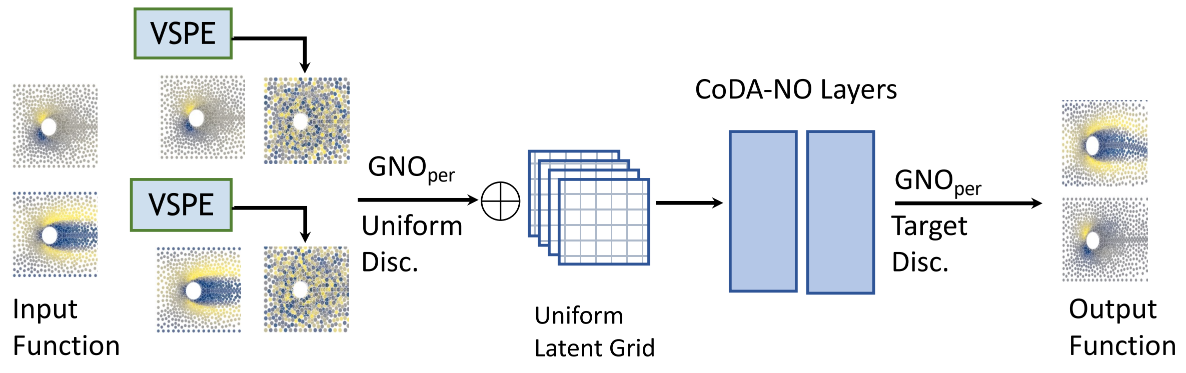
NEURAL OPERATORS

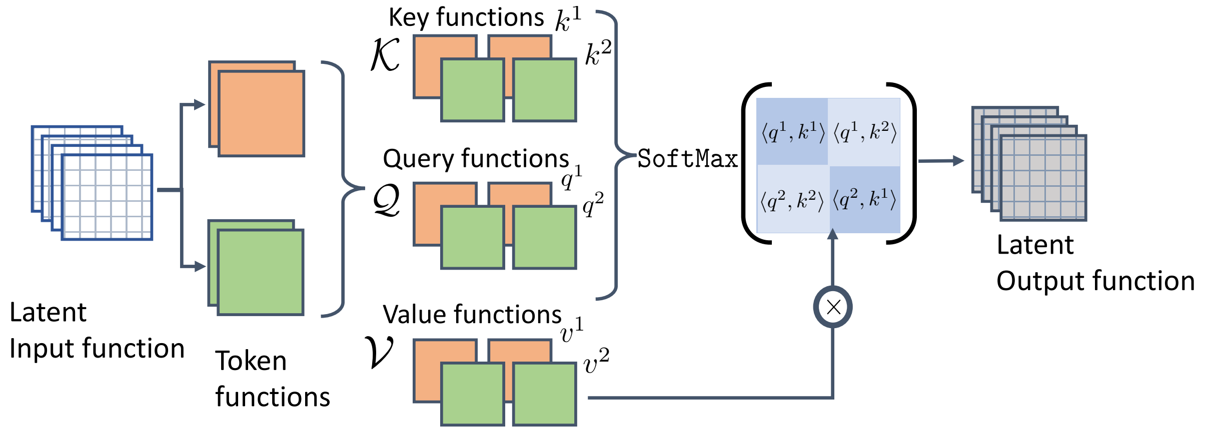
Architectures

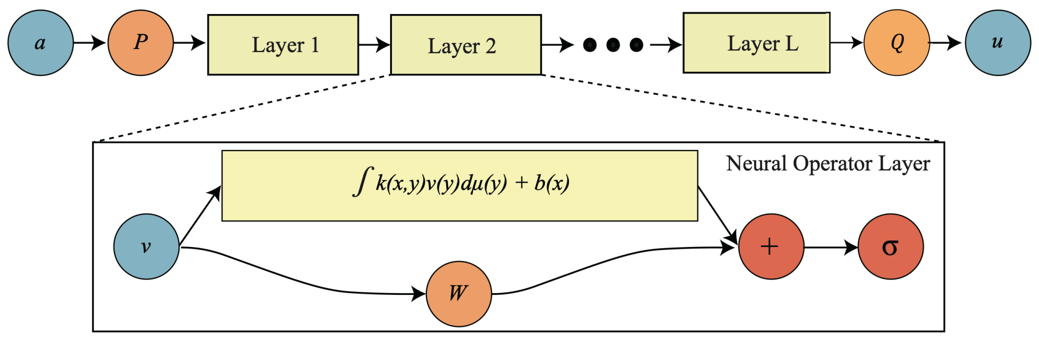


Co-domain Attention Neural Operator (CoDANO)

- Self-supervised learning
- Generalize to new unseen functions
- Prompt learning for fine tunning
- Variable specific positional encoding







NEURAL OPERATORS

Architectures

Main components so far,

- Integration, global (FNO) or local (GNO)
- Residual connection (point-wise)
- Bias function (point-wise)
- What else?

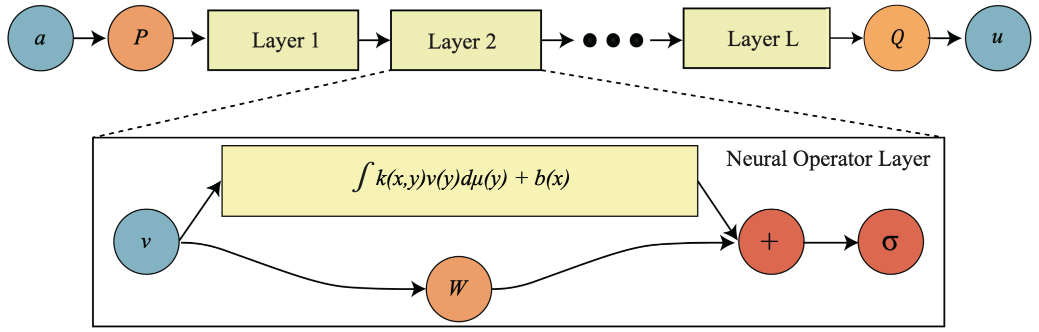
Remember Darcy's flow

$$-\nabla.\big(a(x)\nabla u(x)\big) = f(x)$$

Consider the following inverse problem,

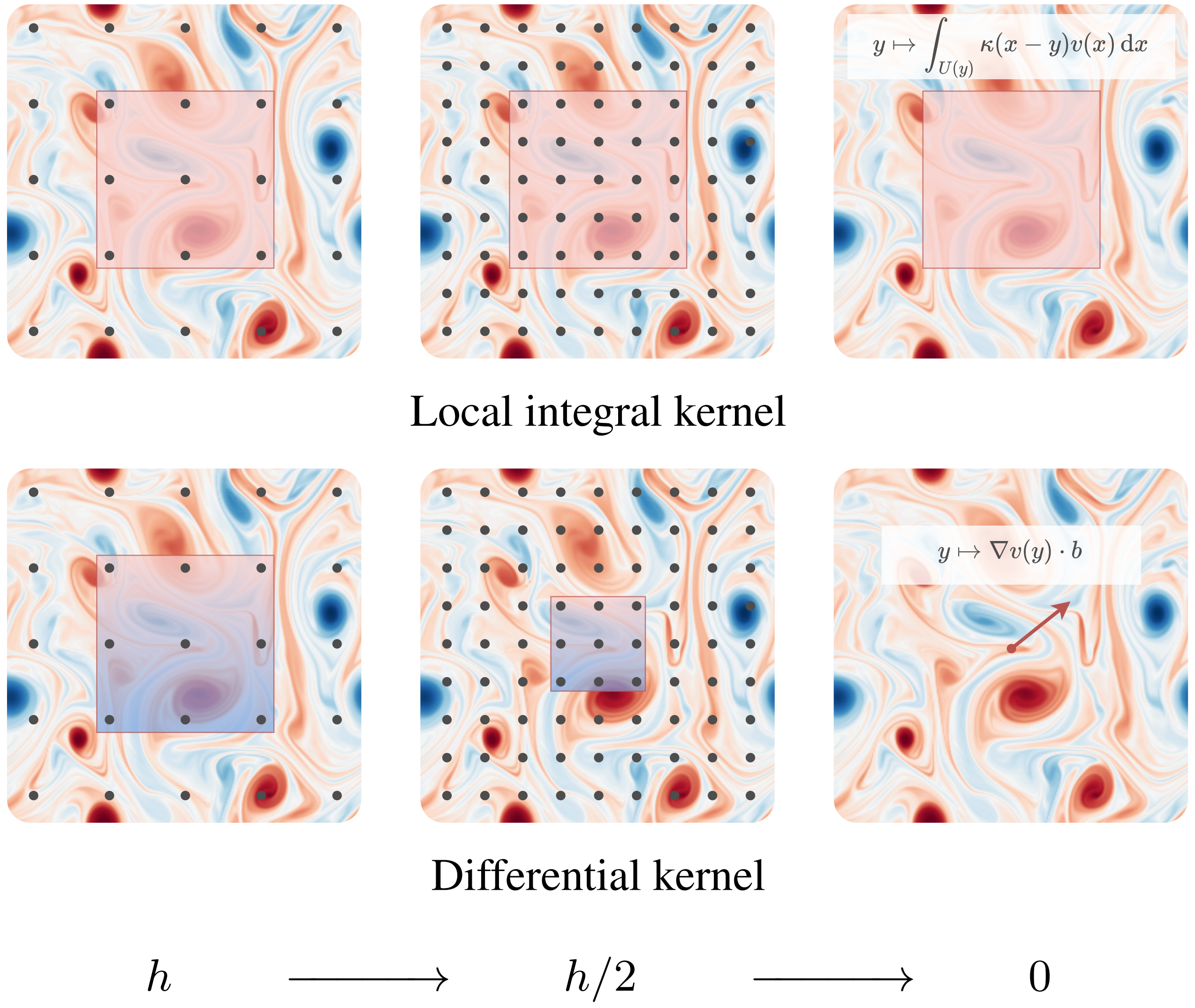
Fix a , input is u and output is f, then is integration efficient?

We need derivatives in the architecture $\frac{\mathrm{d}}{\mathrm{d}t}, \frac{\mathrm{d}}{\mathrm{d}x}, \frac{\mathrm{d}^2}{\mathrm{d}t^2}, \frac{\mathrm{d}^2}{\mathrm{d}x^2}, \ldots$



NEURAL OPERATORS

Architectures



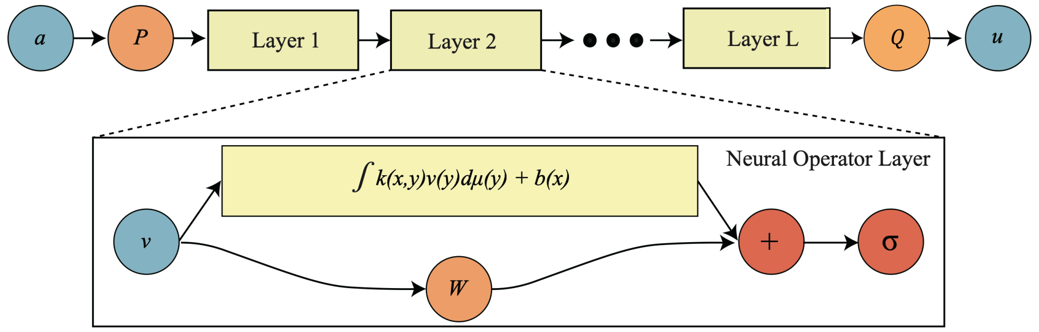
Main components so far,

- Integration, global (FNO) or local (GNO, CNO)
- Residual connection (point-wise)
- Bias function (point-wise)
- Derivatives

FNO/CNO corresponds to large kernel CNNs,

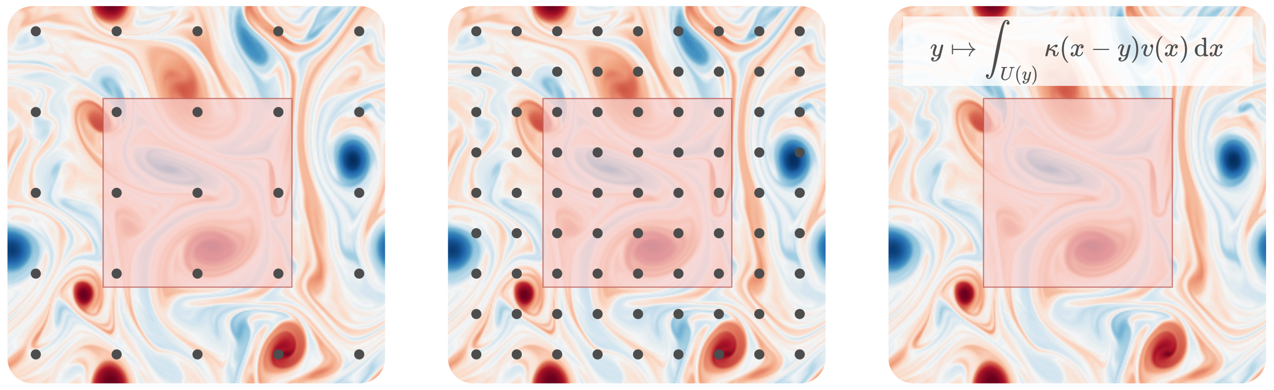
Derivative layers corresponds to small (e.g., 3x3) kernel CNNs

Convolutional Neural Operators for robust and accurate learning of PDEs

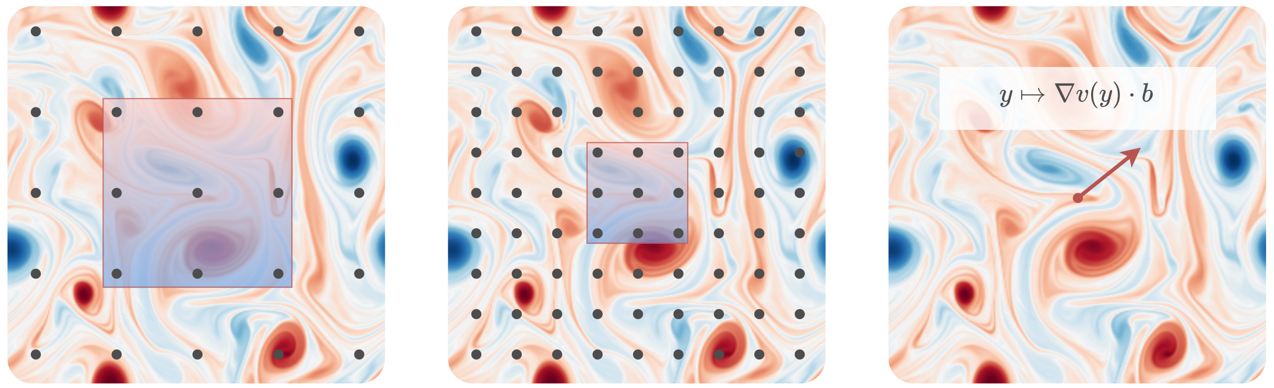


NEURAL OPERATORS

Architectures



Local integral kernel



Differential kernel

$$h \longrightarrow h/2 \longrightarrow 0$$

Main components so far,

- Integration, global (FNO) or local (GNO, CNO)
- Residual connection (point-wise)
- Bias function (point-wise)
- Derivatives

FNO/CNO corresponds to large kernel CNNs,

Derivative layers corresponds to small (e.g., 3x3) kernel CNNs

Recall derivative in CNNs:

The coefficients sum to zero

0	1	0
-1	0	1
0	-1	0

Discretization agonistic derivative

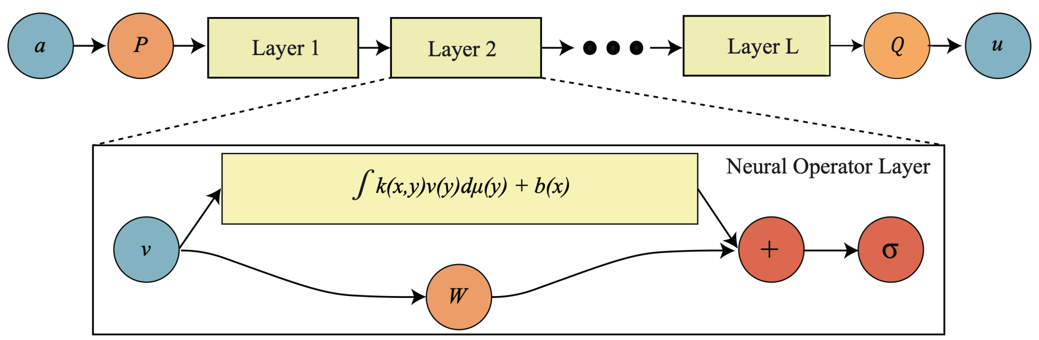
0	$1/h$	0
$-1/h$	0	$1/h$
0	$-1/h$	0

General directional derivative

k_{11}	k_{12}	k_{13}
k_{21}	k_{22}	k_{23}
k_{31}	k_{32}	k_{33}

$\sum_{ij} k_{ij} = 0$ If we double the resolution, $k_{ij} \leftarrow 2k_{ij}$

Irregular grids: GNNs to compute derivatives



NEURAL OPERATORS

Architectures

Main components so far,

- Integration, global (FNO) or local (GNO, CNO)
- Residual connection (point-wise)
- Bias function (point-wise)
- Derivatives
- What else? what other ways of information aggregation? Max pooling? We need consistency in some sense
- What else? (yet to be solved)

Global integral operator

Local integral operator

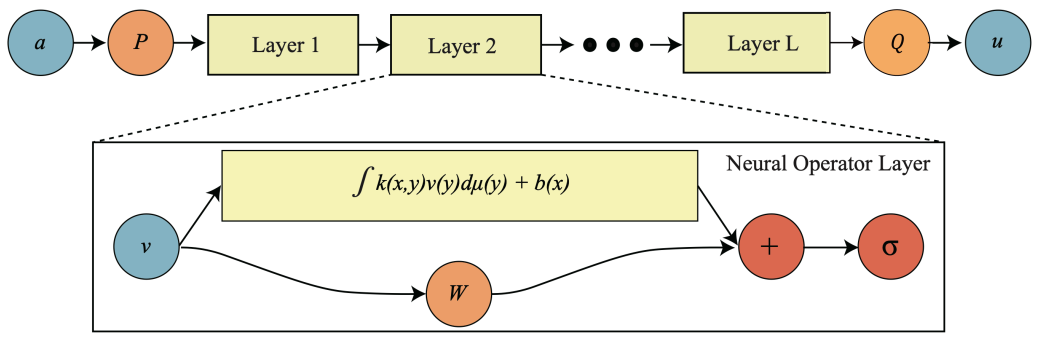
Differential operator

Residual connection

$$v(y) = \sigma(\mathcal{K}(a))(y) = \sigma\left(\int \kappa_{Global}(y,x)v(x)\,\mathrm{d}\mu + \int \kappa_{Local}(y,x)v(x)\,\mathrm{d}\mu' + \mathfrak{D}\,a\Big|_y(y) + Wa(y) + b(y)\right)$$

Bias function

NEURAL OPERATORS AND SCIENTIFIC COMPUTING



NEURAL OPERATORS

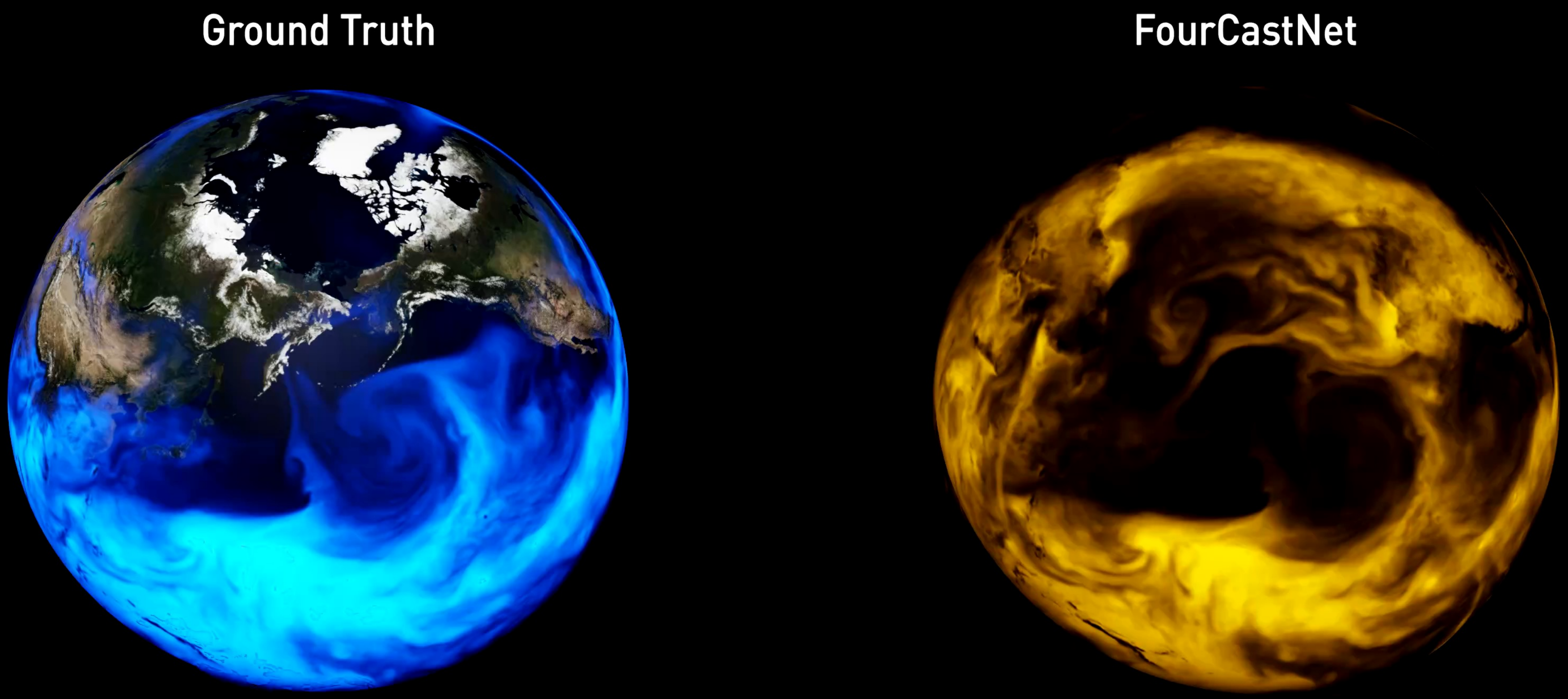
Architectures

In the past many decades, for each domain, CV, language, etc, we developed algorithms, architectures, benchmarks, datasets, metrics, …

We need to do the same for each domain of scientific computing domains, e.g.,

- Automative industry,
- Weather/climate,
- Molecular dynamics
- Seismology
- Electromagnetic
- Material design
- Drug discovery
- …

These are not applications, these are domains, no plug and play!

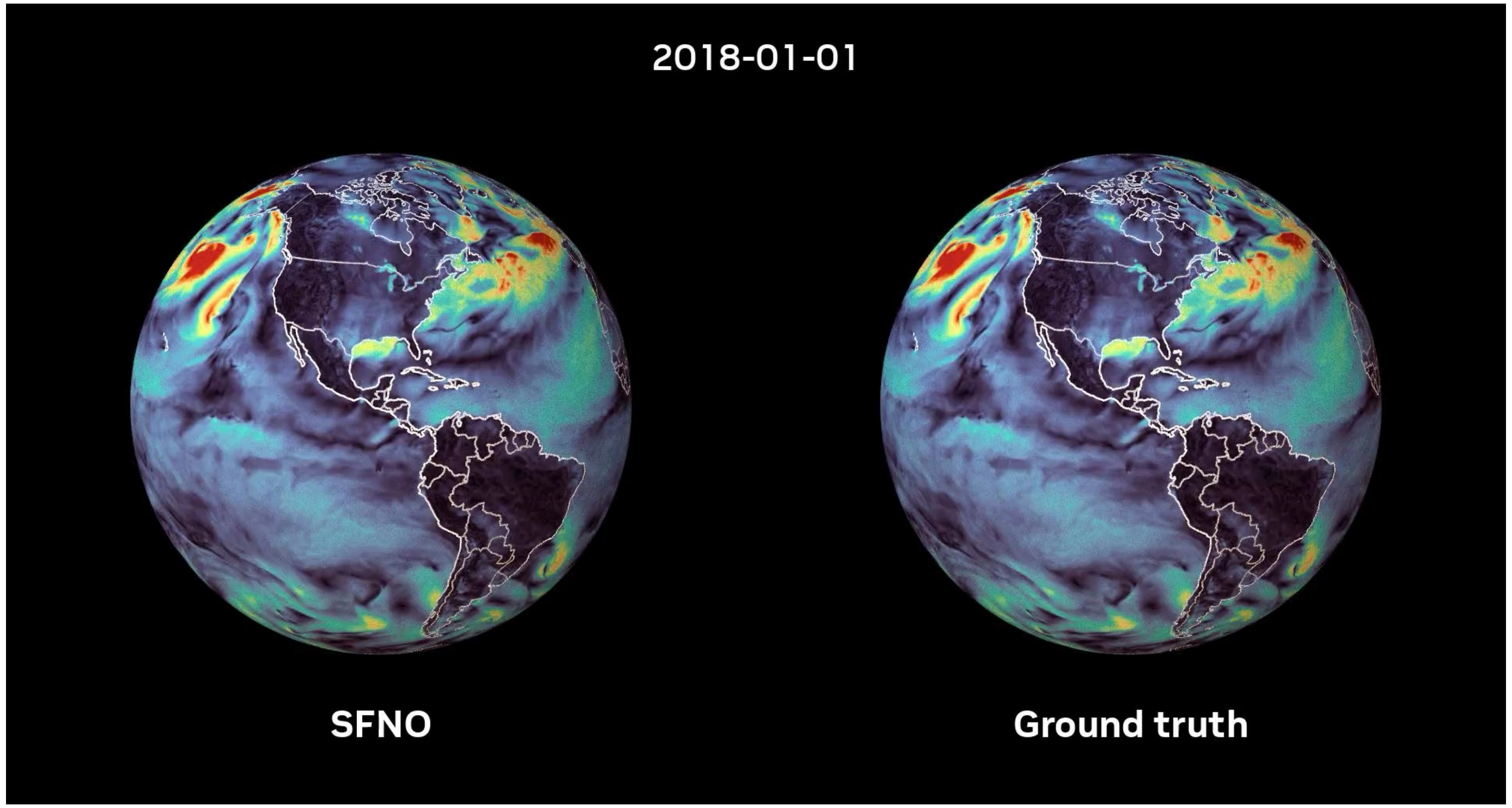


FourCastNet is **45,000 times** faster than current weather models

FOURCASTNET FOR WEATHER PREDICTION

Domain: Weather/Climate/Ocean

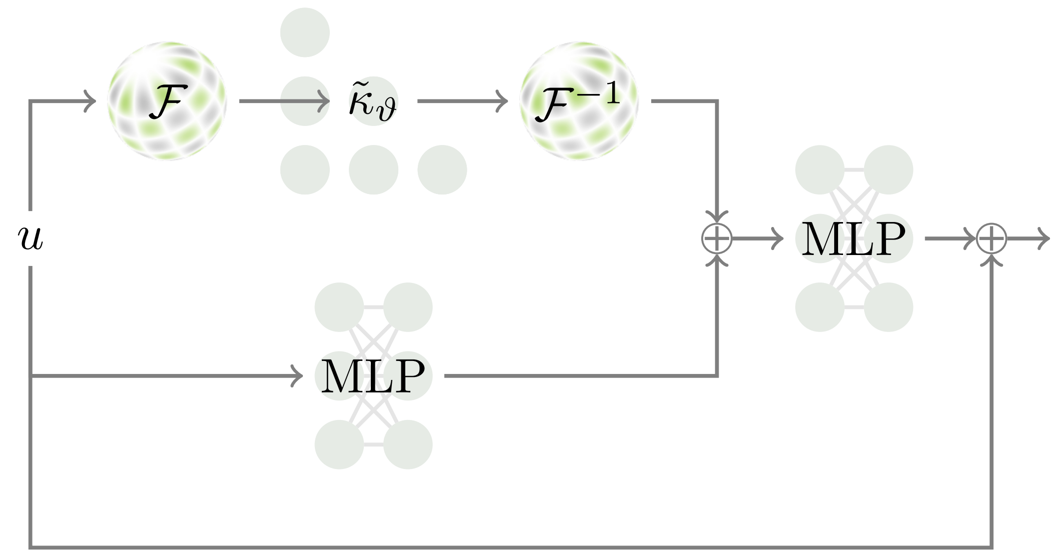
- Dataset: 10 TB of weather data
- Metrics: L2 on function space, ACC, expert analysis
- Architecture: Adaptive FNO (AFNO)
- **45,000x** speedup
- **25000x** smaller energy footprint.



Domain inspired advanced architecture

Spherical-harmonic Fourier Neural Operator (SFNO)

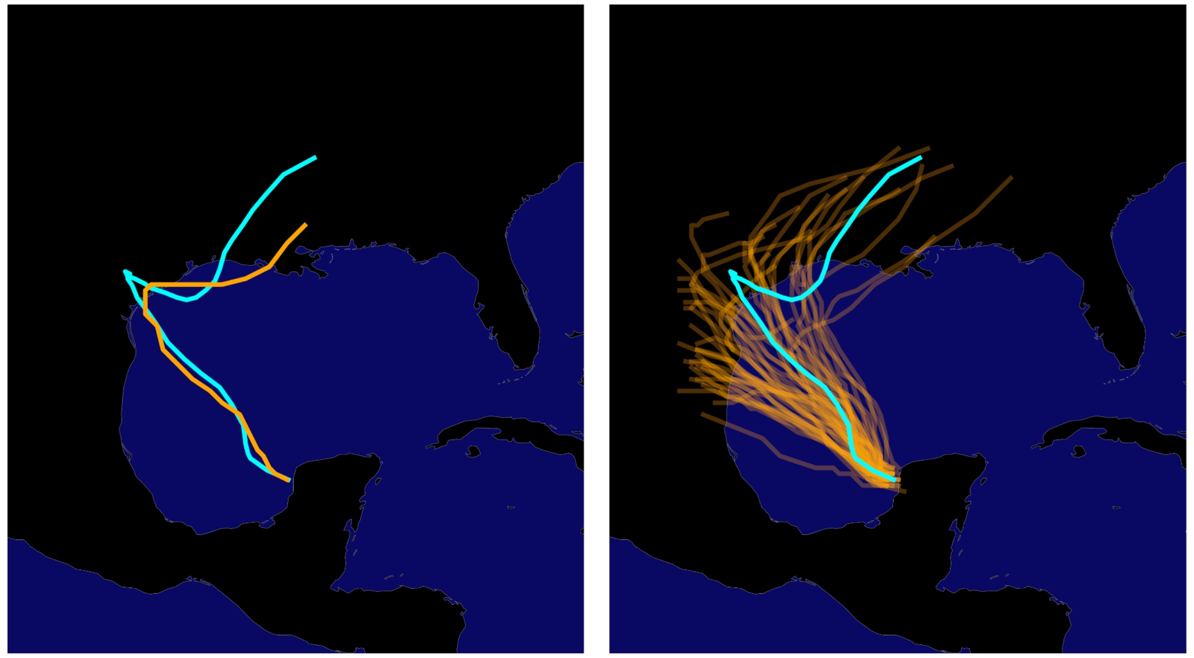
- Bases functions: spherical harmonic
- Gaussian quadrature for sum
- MLP based residual connections



FOURCASTNET FOR WEATHER PREDICTION

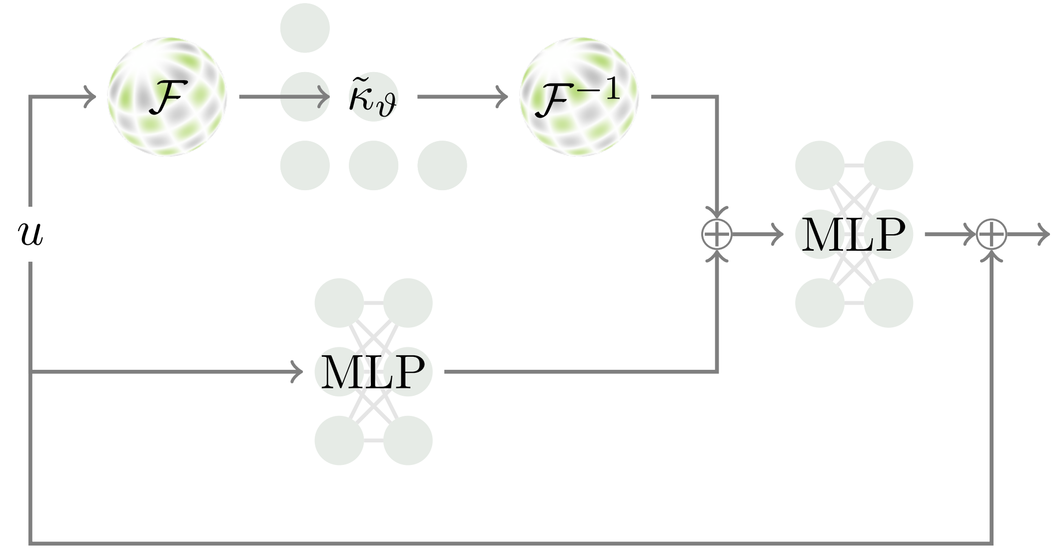
Domain: Weather/Climate/Ocean

- Dataset: 10 TB of weather data
- Metrics: L2 on function space, ACC, expert analysis
- Architecture: Adaptive FNO (AFNO)
- **45,000x** speedup
- **25000x** smaller energy footprint.
- 1000-member ensemble in a few seconds



Spherical-harmonic Fourier Neural Operator (SFNO)

- Bases functions: spherical harmonic
- Gaussian quadrature for sum
- MLP based residual connections

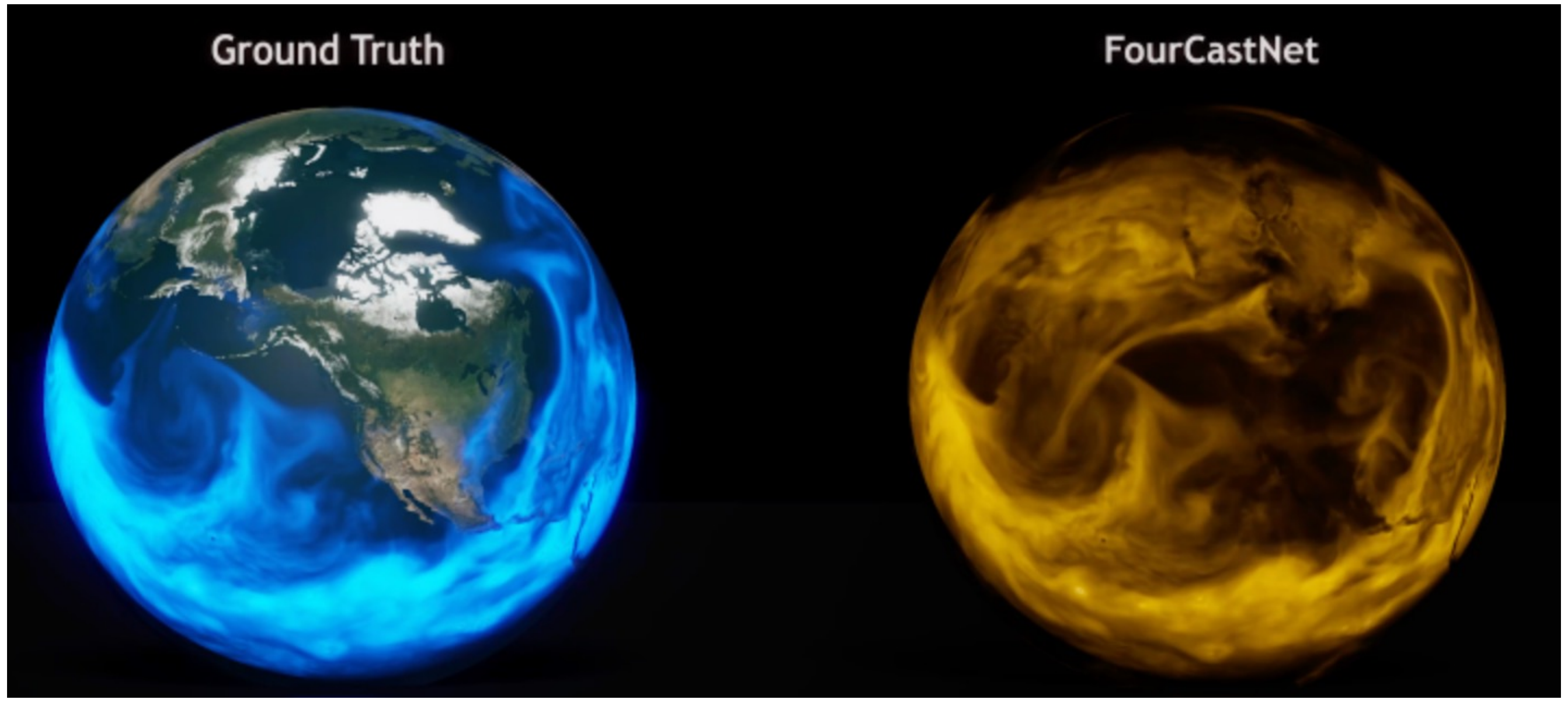


WEATHER FORECAST

Domain: Weather/Climate/Ocean

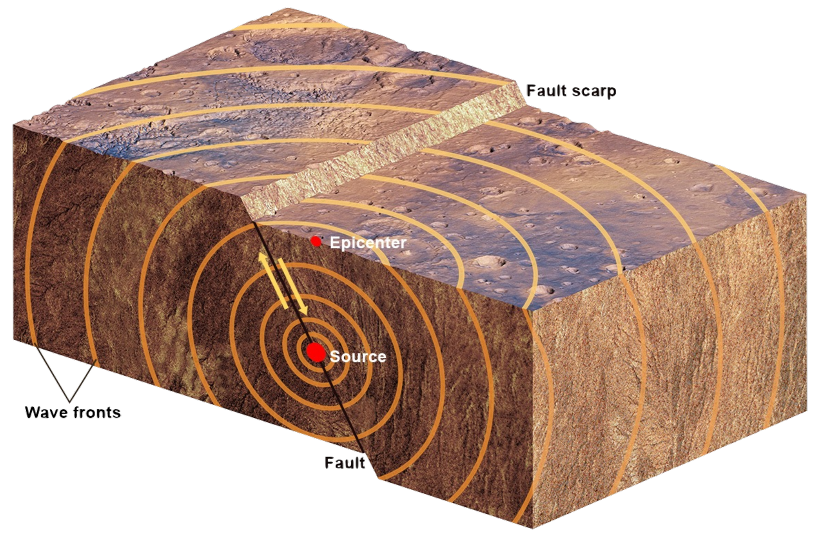
Open problems:

- How to make the accuracy absolute?
- How to deal with discretization error?
- How to deal with chaotic nature of the problem?
- How to deal with uncertainty and probabilistic nature of the problem?
- How to scale up?
- How to train on multiple datasets?
- How to incorporate physics and domain knowledge?
- What are the right metrics here (F^2ID?)?



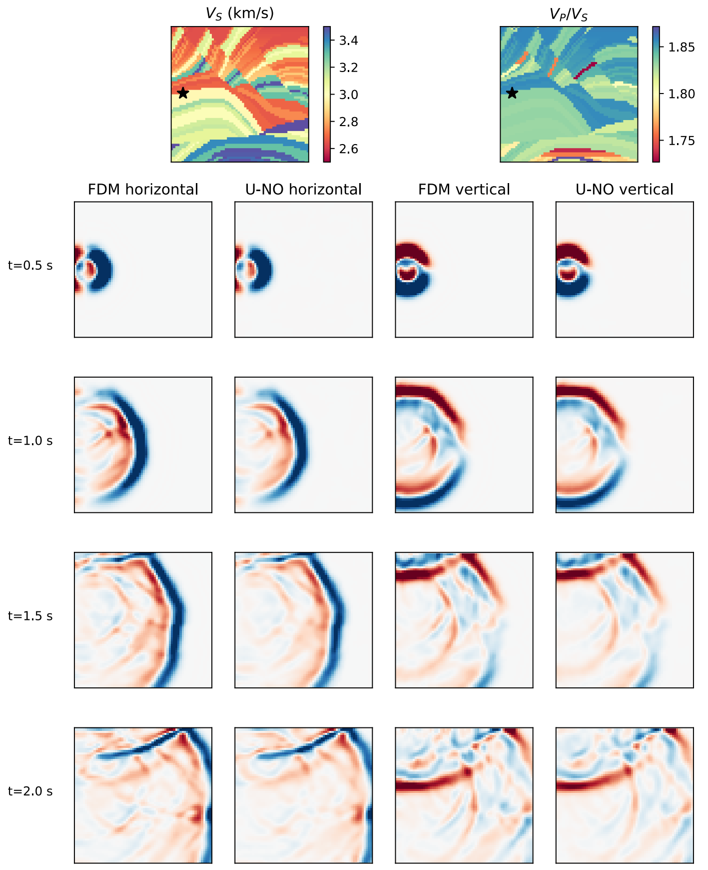
A very important domain of study of ML on function space, which many open problems

GEOPHYSICS



Domain: Seismology, Earth Sciences

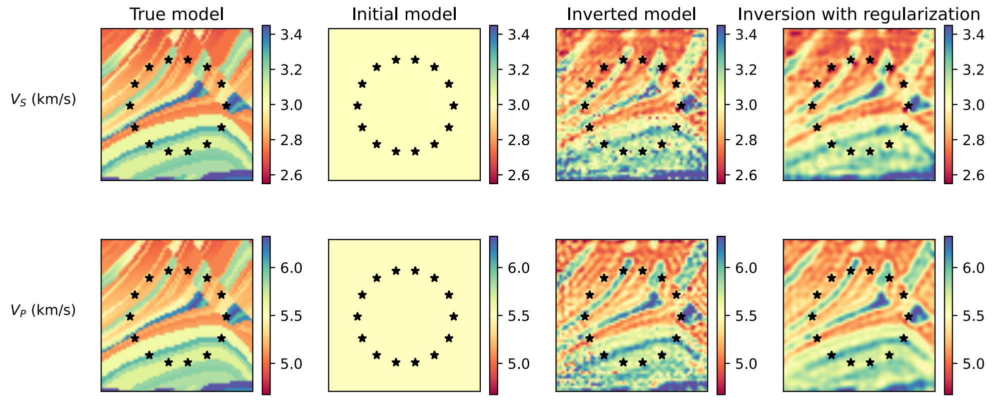
Given earth structure, earthquake location, and shaking profile, how the wave propagates?



Given the wave observation on the surface, what is Earth structure and earthquake source?

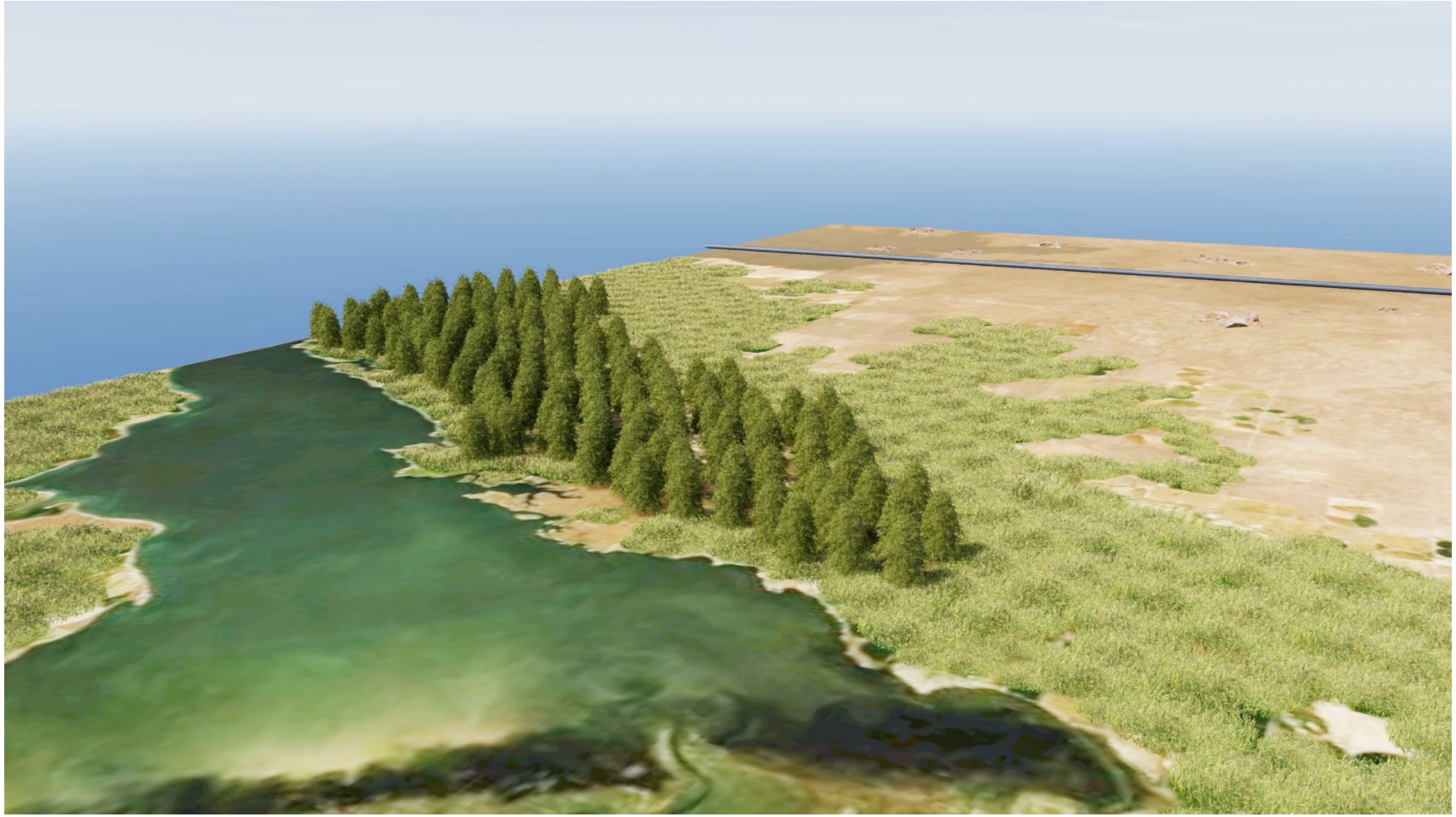
- Neural operators are fast to query and differentiable → fast inverse solvers

$$u = \mathcal{G}_\theta(a)$$



CLIMATE CHANGE MITIGATION: MODELING CO2 STORAGE

Domain: Sub-surface flow

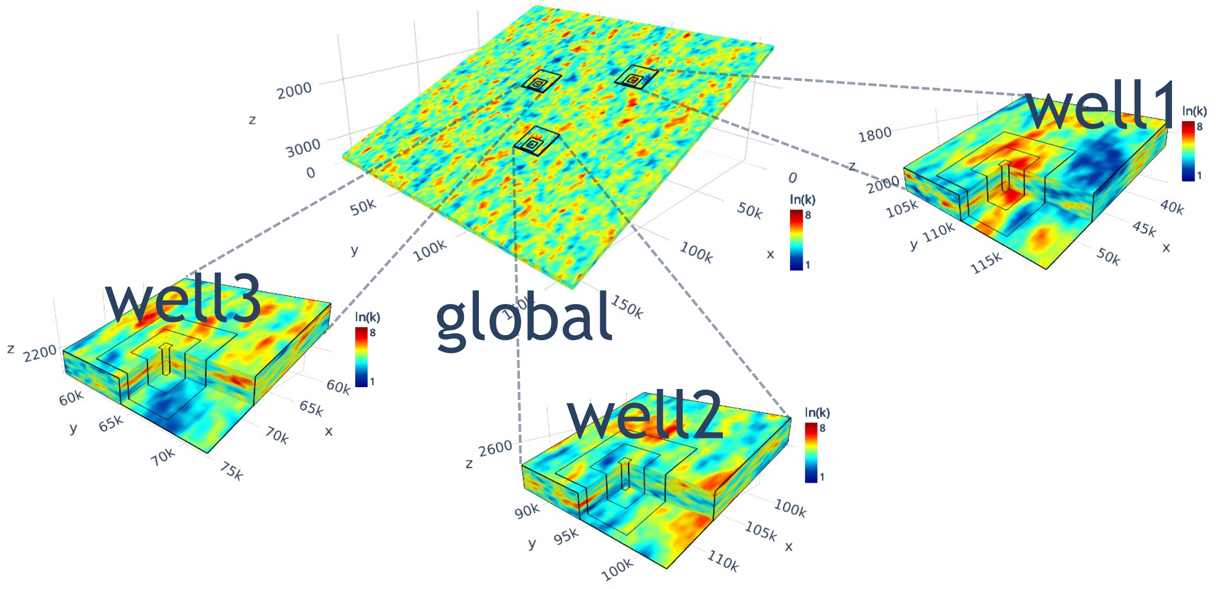


CCS plumes spanning ~ 1km

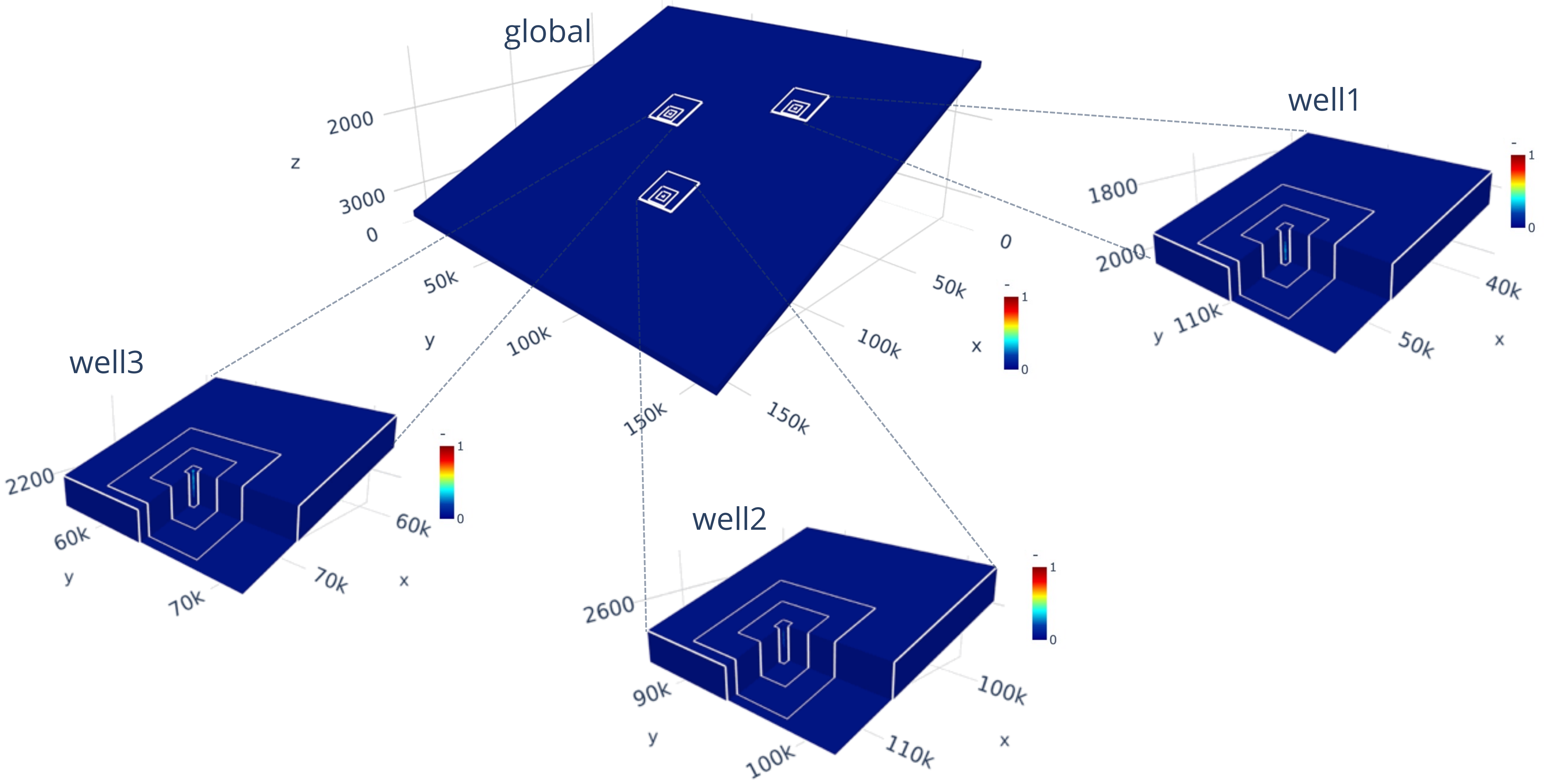
Omniverse visualization by Marius Koch

NETZERO CLIMATE: CO_2 MODELING WITH AI (FNO)

ML to accelerates CCS by 700,000 times using Nvidia GPU - A100



Permeability Heat Map



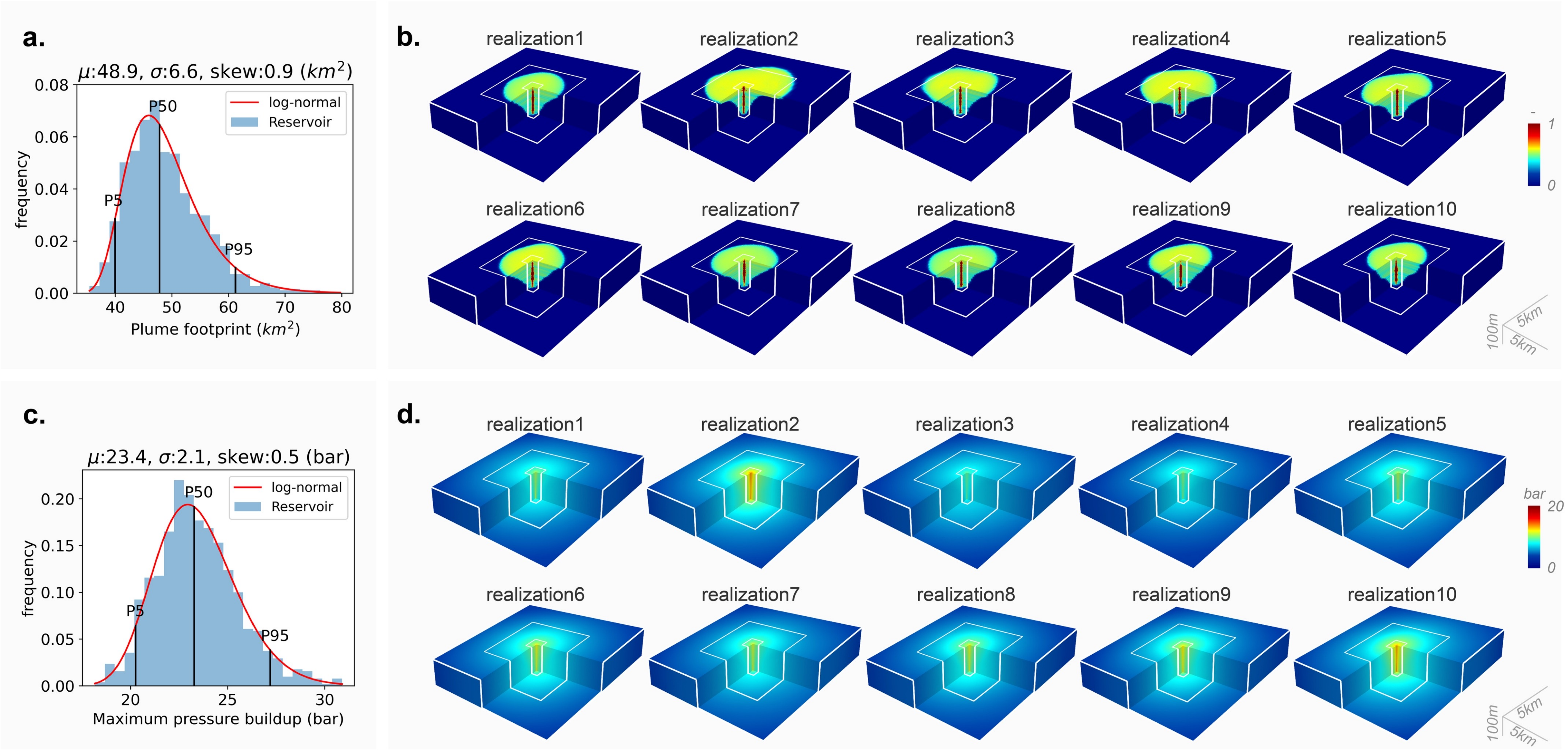
Pred, t=10 day

Challenges:

- Multi physics, scale, phase
- 30yr to 1000yr
- Seismic monitoring

FOUR-DIMENSIONAL CCS MODELING WITH AI (FNO)

Uncertainty quantification 20 years to 20 second



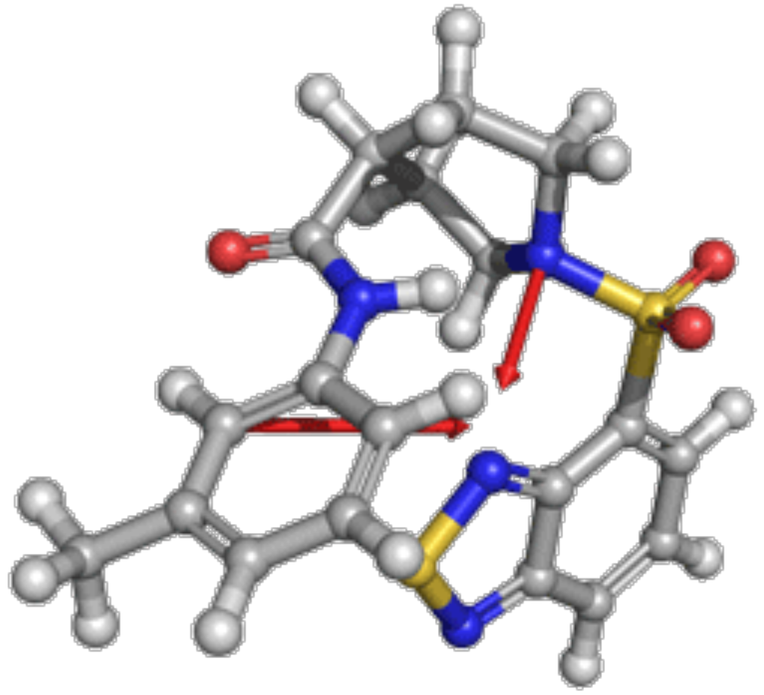
TIME SERIES OR A FUNCTION IN TIME?

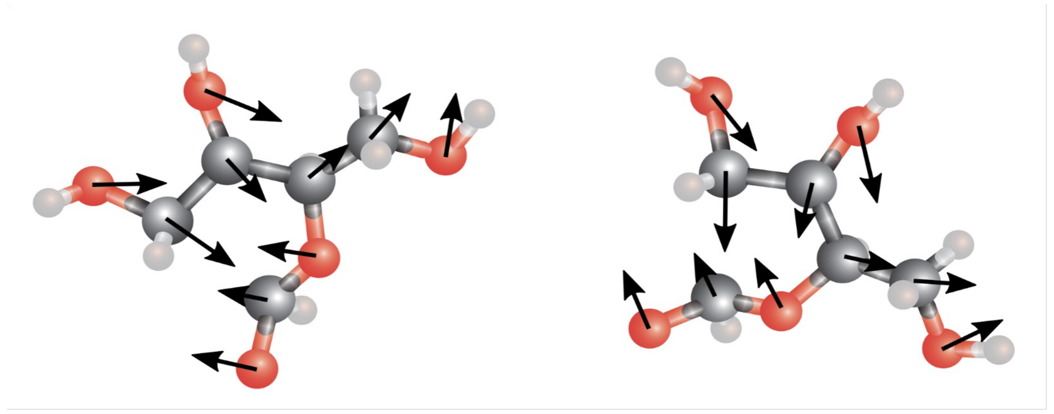
Molecular dynamics

- Simulate stable structure of molecular geometries
- Computationally costly quantum mechanical calculations
- Equivariant to rotation and translation
- Schrödinger equation

$$\hat{H}\Psi = E\Psi$$

Method	Complexity
Hartree Fock	$O(n^3) - O(n^4)$
Density Functional Theory	$O(n^3) - O(n^4)$
MP2	$O(n^5)$
CCSD	$O(n^6)$
CCSD(T)	$O(n^7)$
Full CI	$O(n!)$



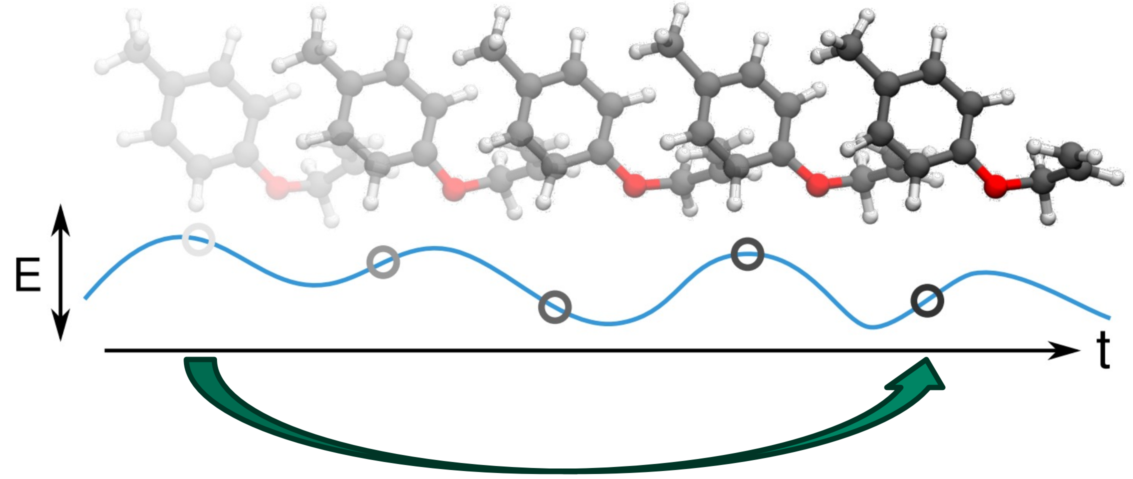


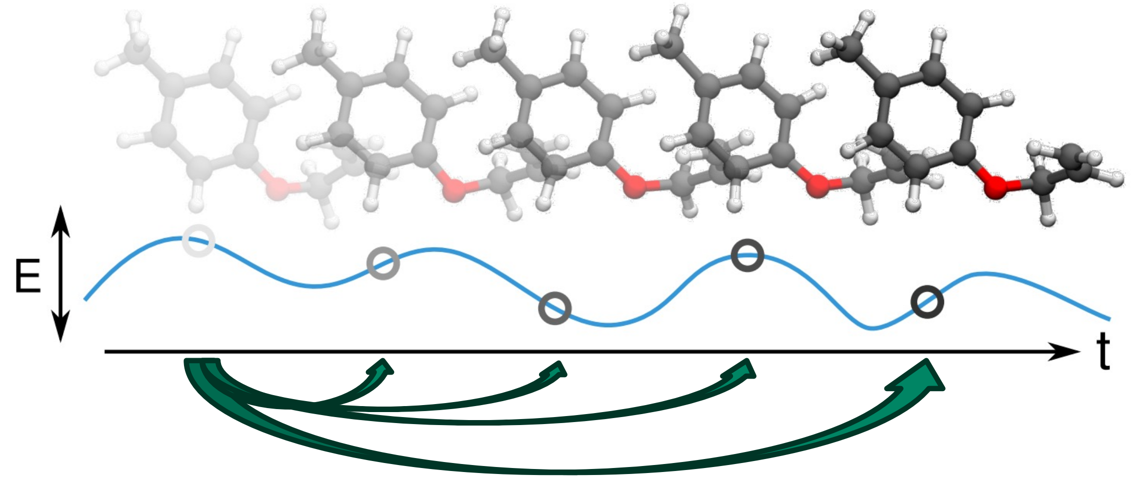
Rotate and translate

TIME SERIES OR A FUNCTION IN TIME?

Molecular dynamics

- Neural network approach: data is a time series
  - Equivariant graph neural network
- Nature of MD is continuous in time
  - Not a discrete sequence of considerably different events
- Spatial graph in space and temporal neural operator in time
- Potentially infinitely more supervision
- True emulator, capturing force and higher order physics





When dealing with time series looking data, we borrow domain expert glasses and check whether it is discrete time series or a continuous function in time

TIME SERIES OR A FUNCTION IN TIME?

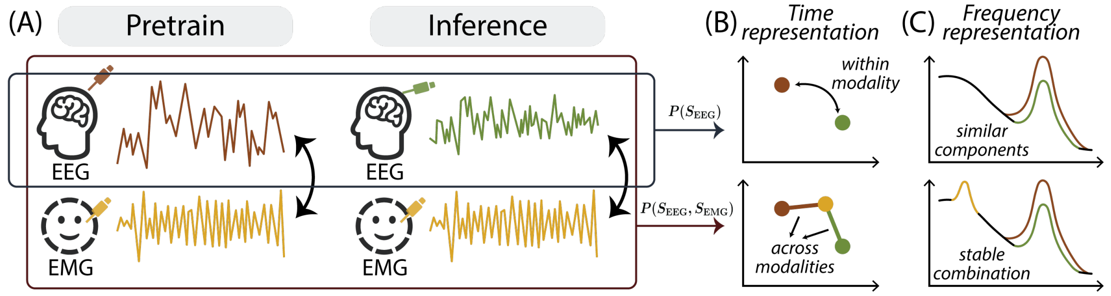
Diffusion models distillation

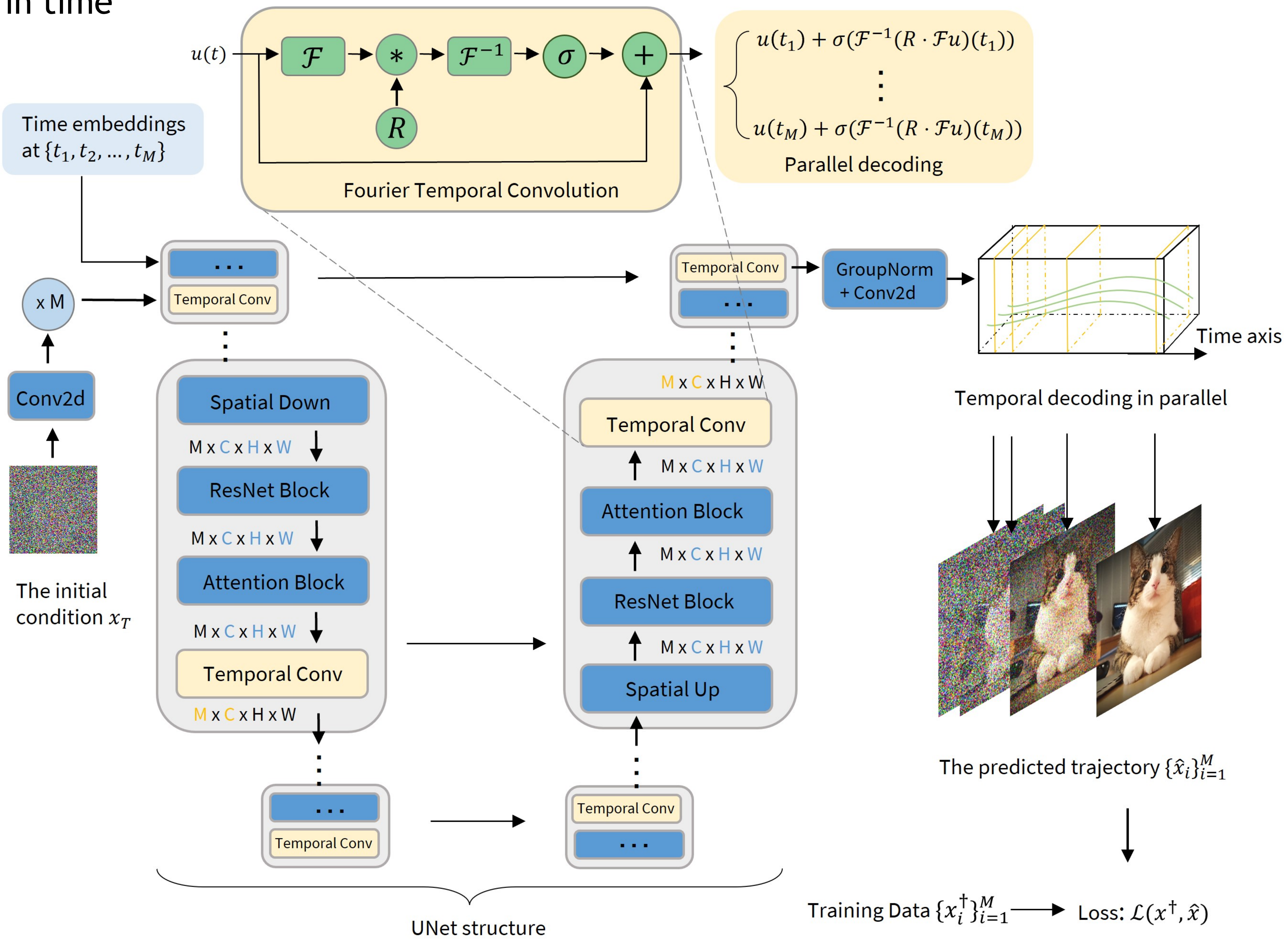
From noise to image, there is a path, which is continuous in time

Map the noise to the function in time

Infinite supervision

Other domains, finance, speech, bio …

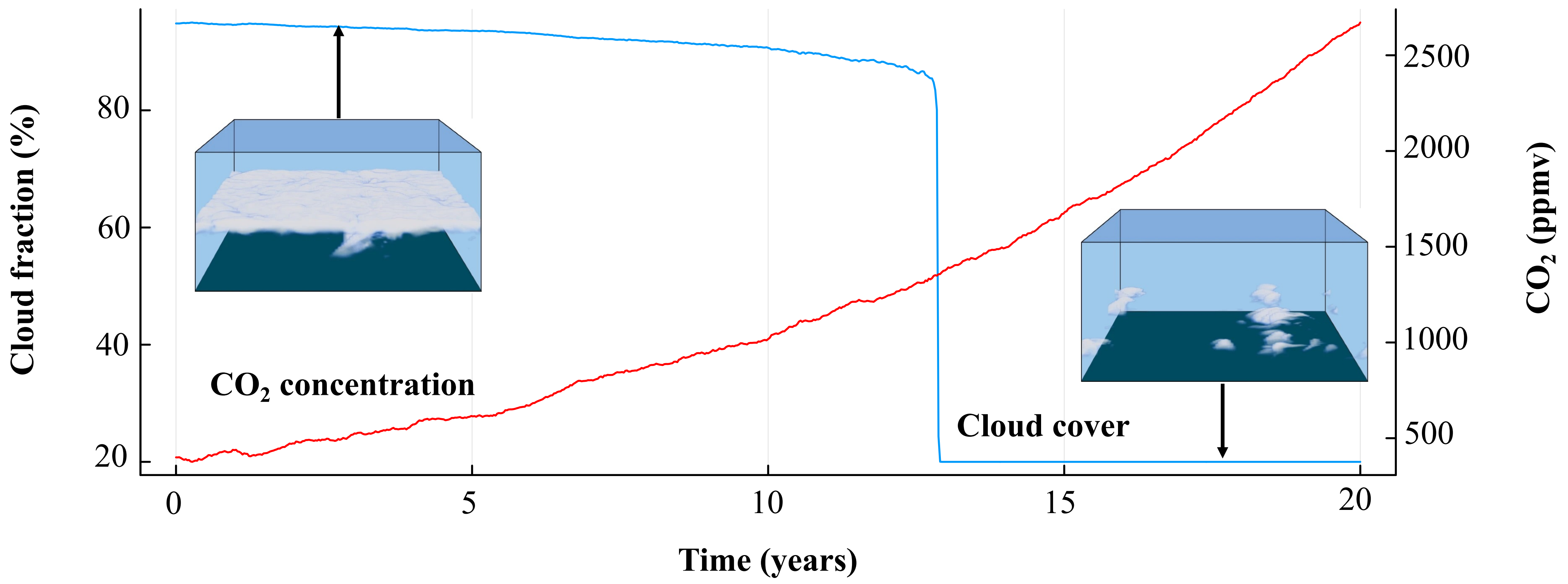




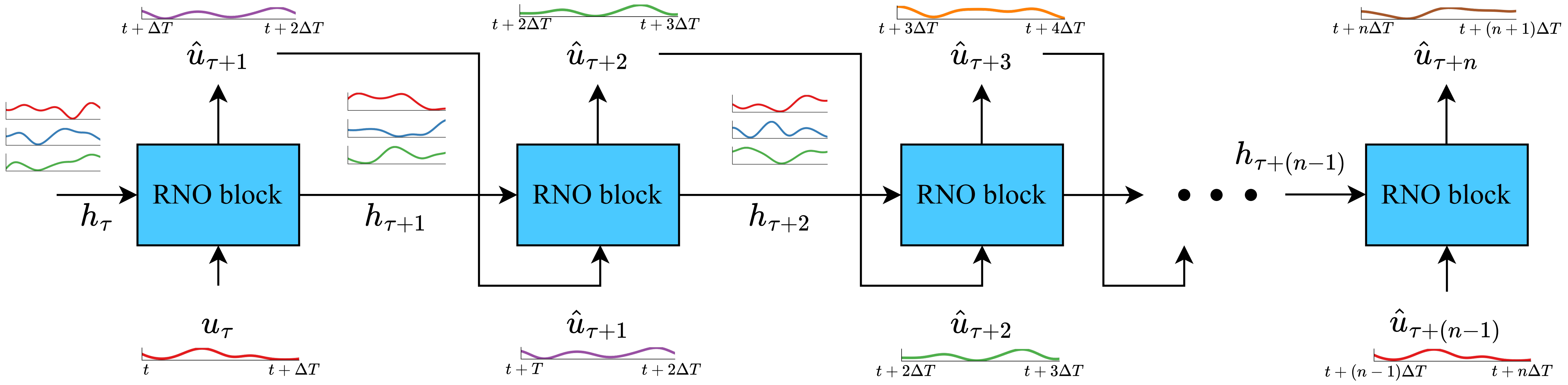
Frequency-Aware Masked Autoencoders for Multimodal Pretraining on Biosignals

TIPPING POINT FORECASTING ON FUNCTION SPACES

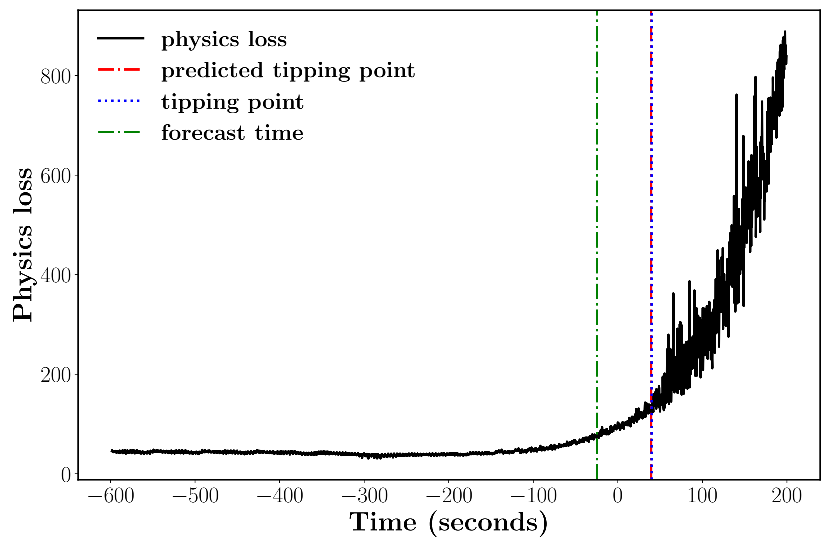
Tipping points: abrupt, drastic, and often irreversible changes in the evolution of non-stationary and chaotic dynamical systems.

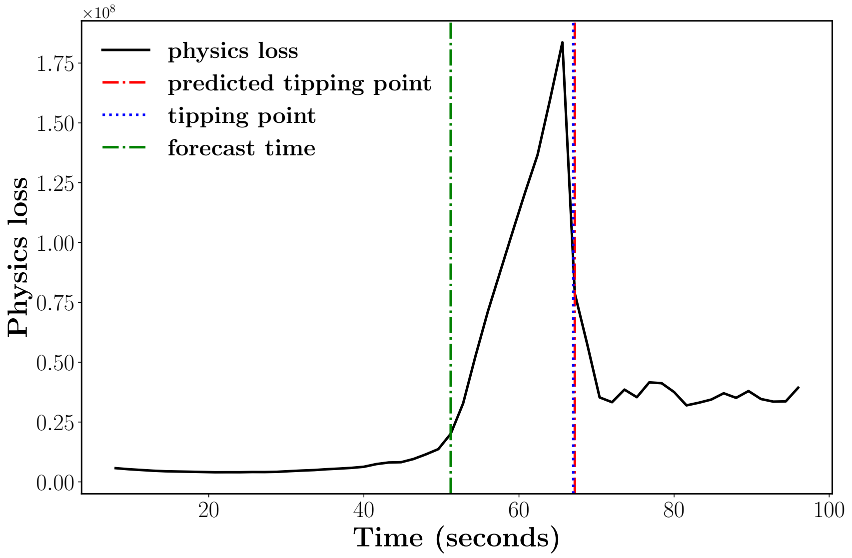


TIPPING POINT FORECASTING ON FUNCTION SPACES



- Recurrent neural operator (RNO) to forecast evolution of non-stationary systems
- RNO maps time interval to time interval
- Tipping point forecast through tracking physics constraint violation

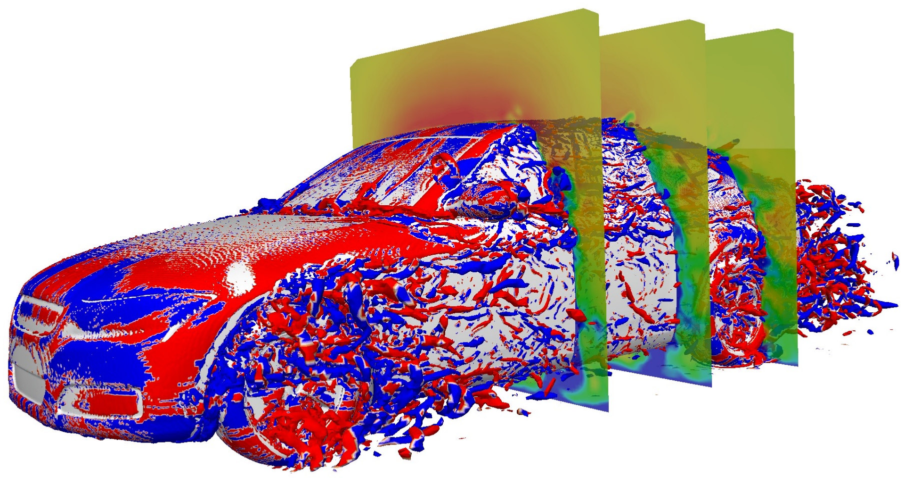




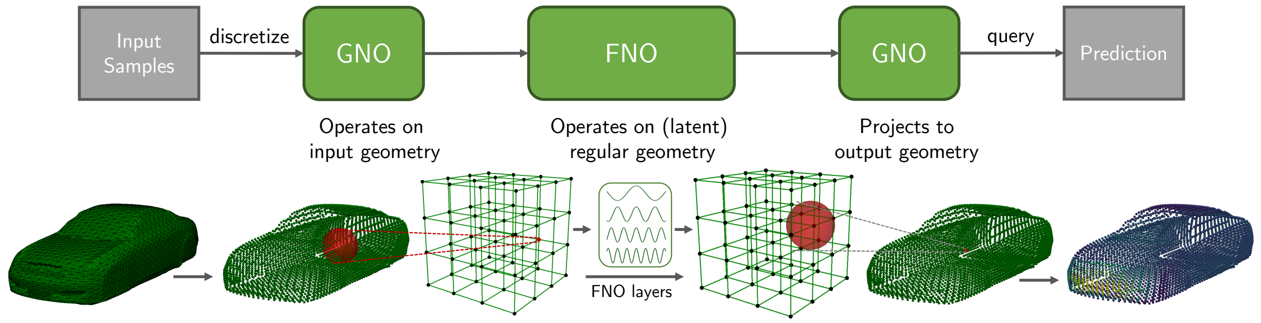
AUTOMATIVE INDUSTRY

140,000 speed up on CFD of real cars

- Complex fluid dynamics problem
- 3D solvers to compute pressure and velocity
- Geometry-informed Neural Operator (GINO)



Model	training error	test error
GNO	18.16%	18.77%
Geo-FNO (sphere)	10.79%	15.85%
UNet (interp)	12.48%	12.83%
FNO (interp)	9.65%	9.42%
GINO (encoder-decoder)	7.95%	9.47%
GINO (decoder)	6.37%	**7.12%**



An open problem in all the mentioned domains: The accuracy is not good enough yet → need same or more amount of work as we did in CV and NLP

PHYSICS INFORMED NEURAL OPERATORS

PHYSICS

Supervision in supervised learning

Supervision in conventional machine learning:

- Data (real, simulation)
- Domain knowledge

Supervision in Scientific computing and Neural operators?

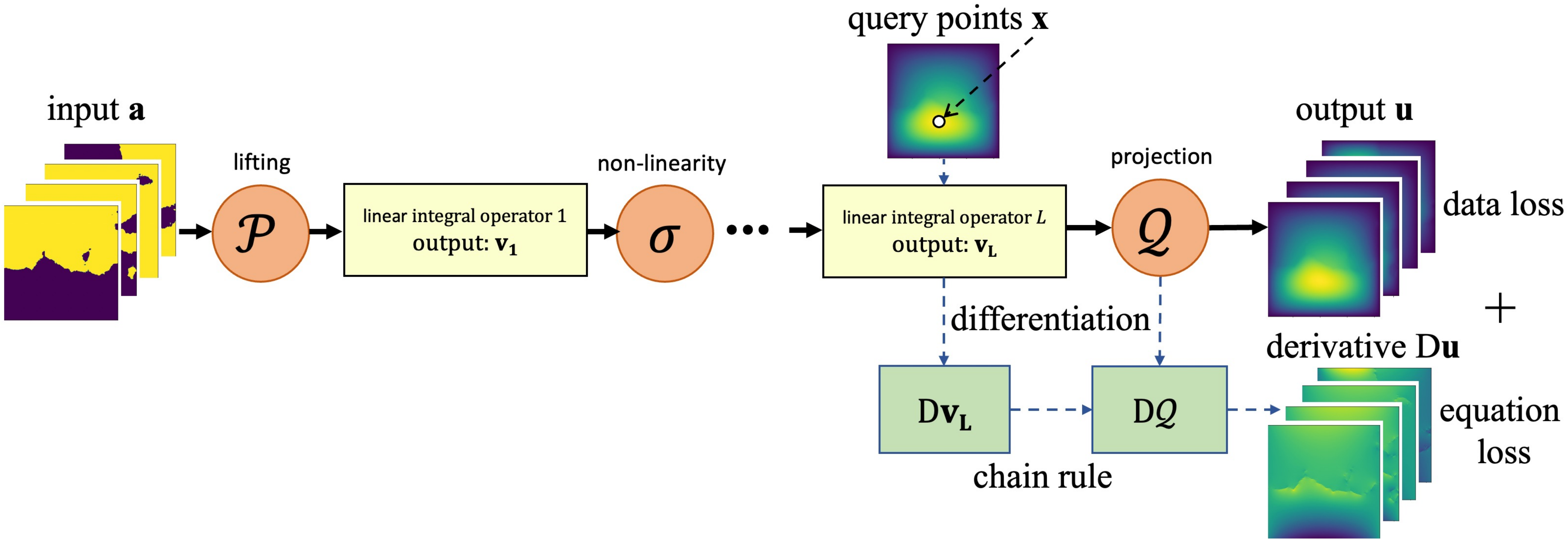
- Data (real, simulation, simulation, simulation)
- Domain knowledge
- Physics

How to use physics in operator learning?

Neural operators output functions → accurate derivatives and integrals

PINO: PHYSICS-INFORMED NEURAL OPERATOR

Infinite supervision from physics

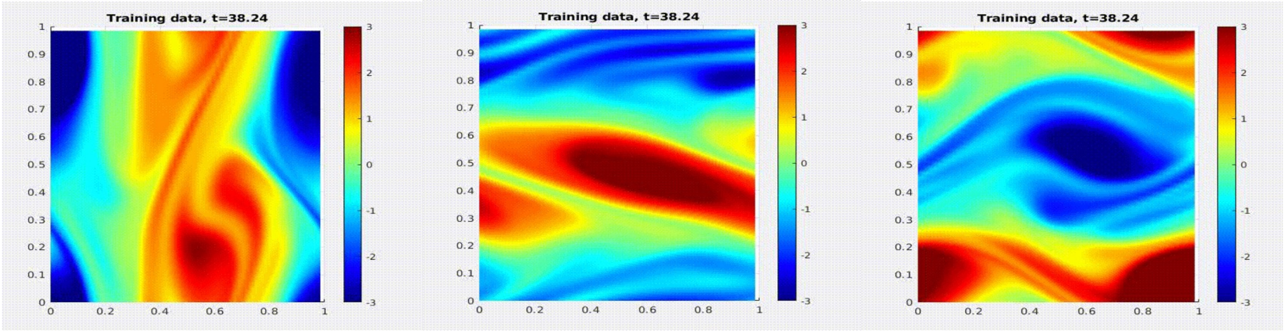


E.g., for a fixed f, → $-\nabla.\left(a(x)\nabla u(x)\right) - f(x) = 0$

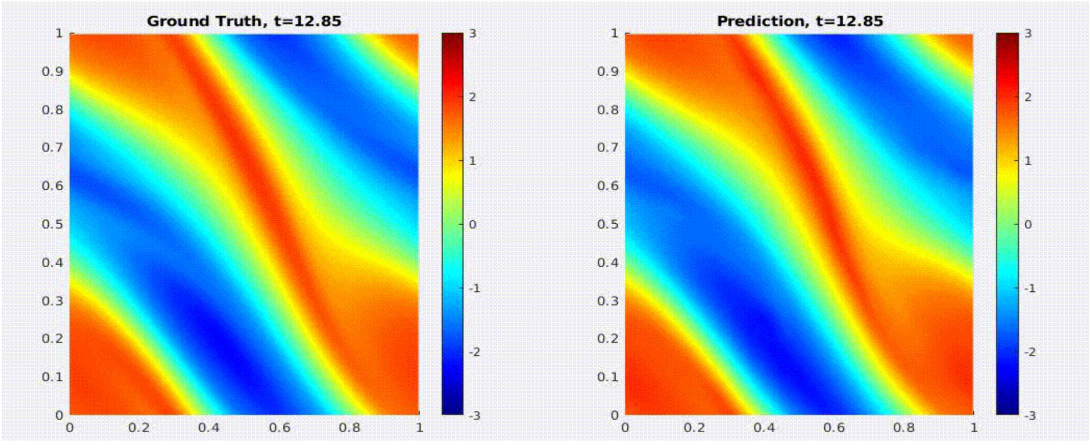
PINO: PHYSICS-INFORMED NEURAL OPERATOR

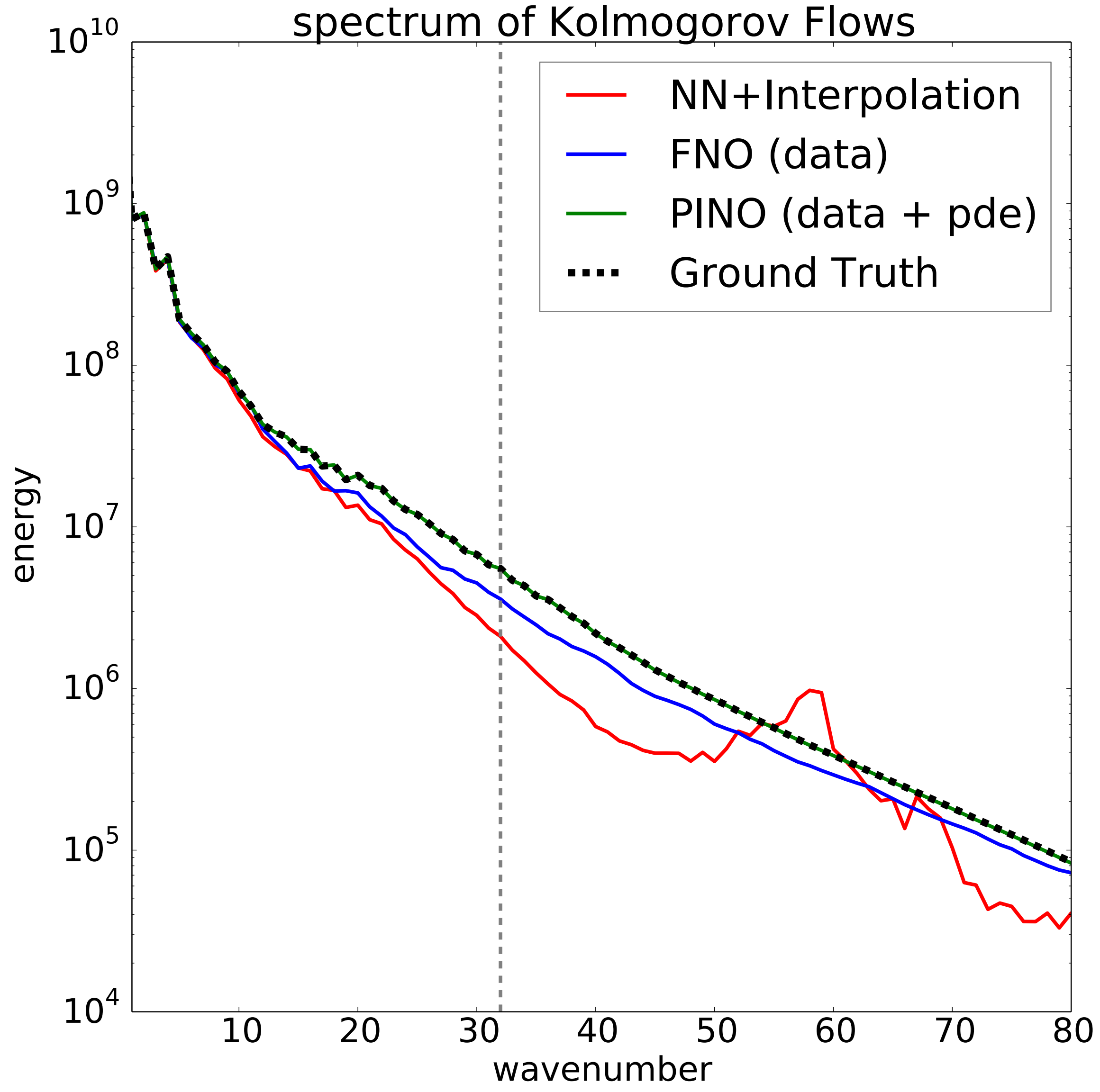
Infinite supervision from physics

Train at low resolution



Use PINO principle to further tune → test at higher resolution



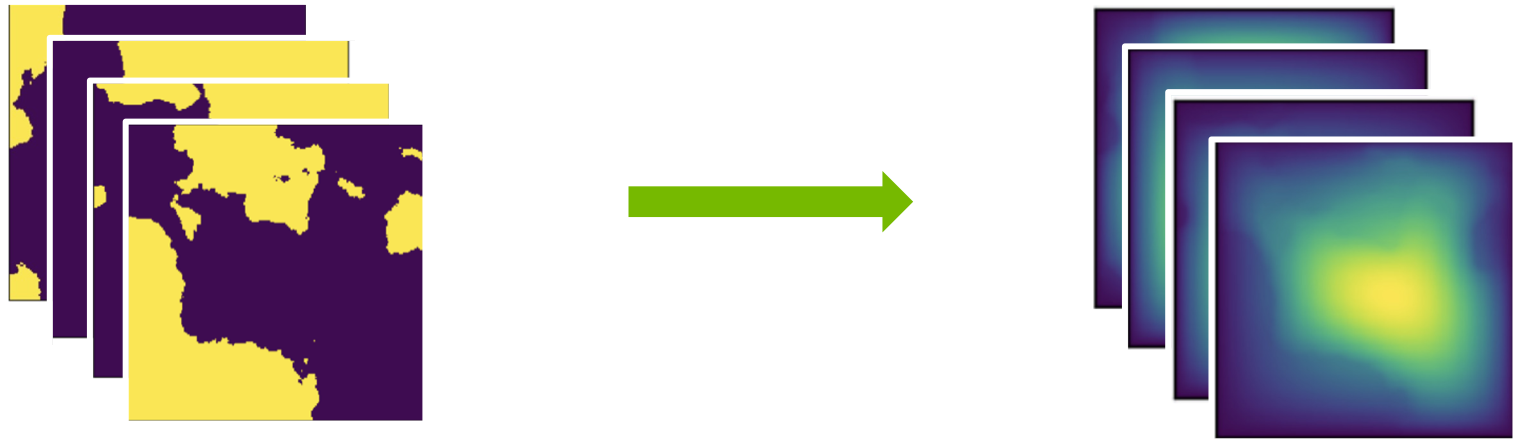


PINO enables generalization to unseen high resolutions, where we don't have data

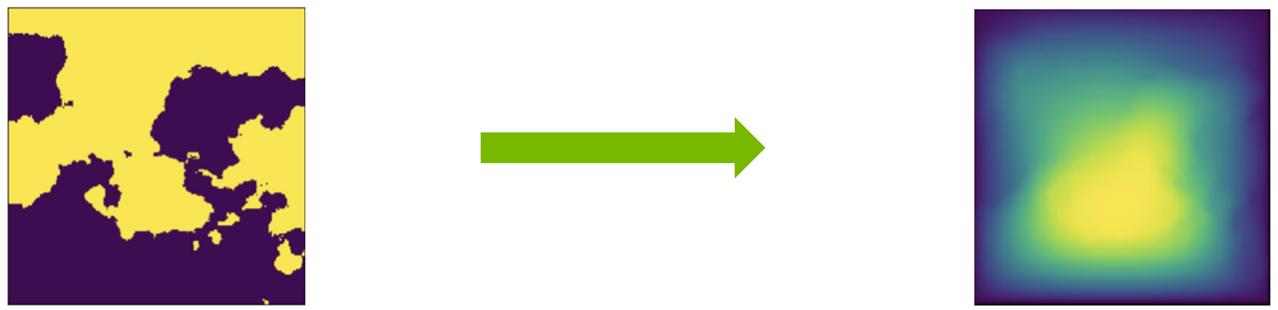
PINO: PHYSICS-INFORMED NEURAL OPERATOR

Solution operator for a family of equations and fine-tune on an instance

Operator learning



Instance-wise finetuning

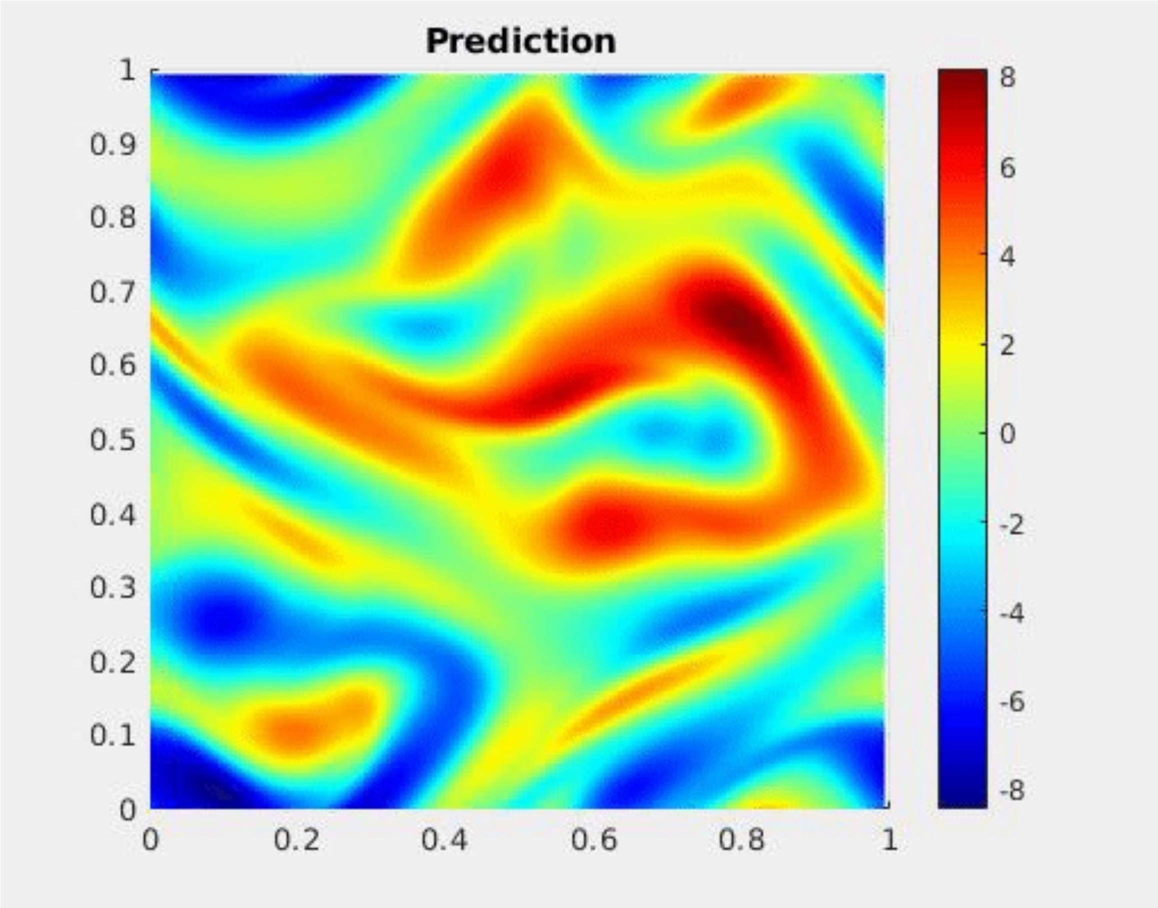


Resembles physics informed neural network, with good preconditioner

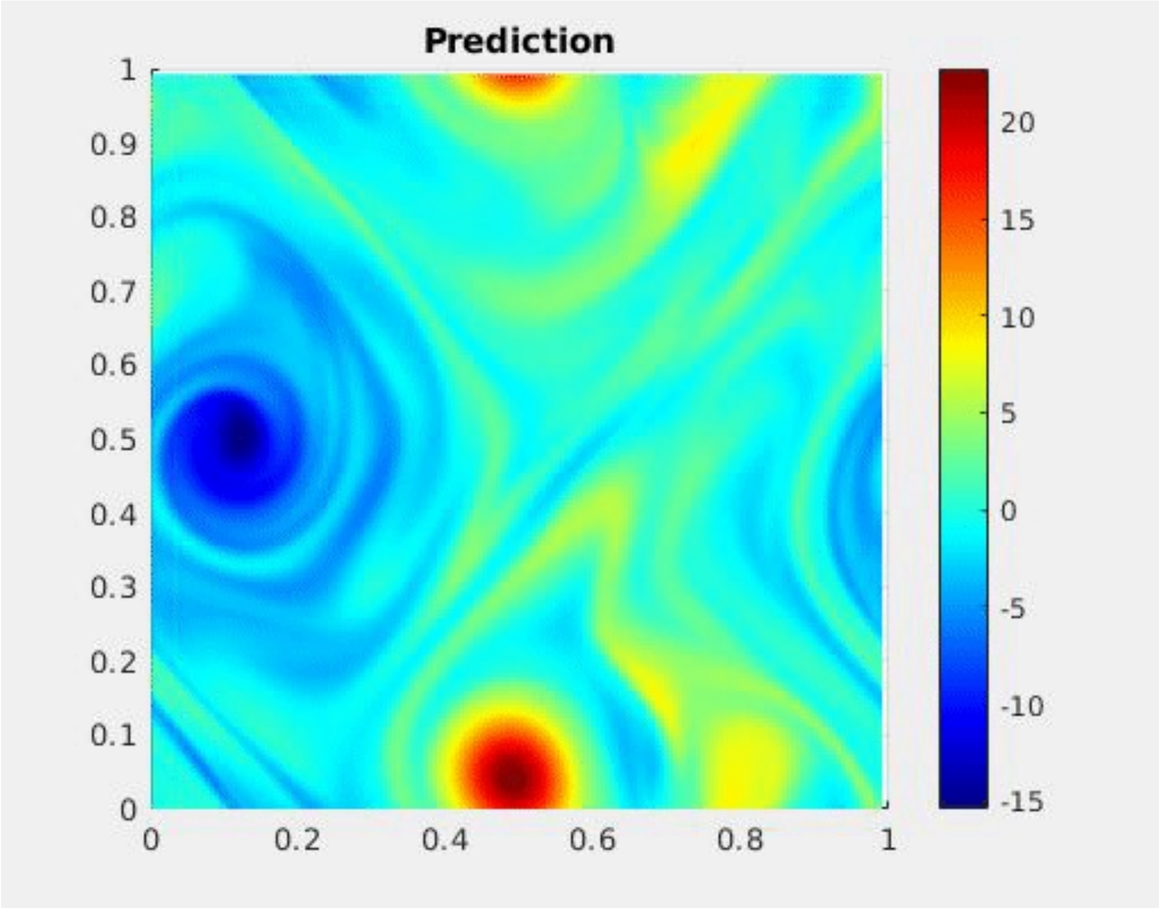
PINO: PHYSICS-INFORMED NEURAL OPERATOR

Transfer Learning with PINO

Operator learned on Re100, fine-tune to Re500. Converges 3x faster.



Reynolds number 100



Reynolds number 500

PINO: PHYSICS-INFORMED NEURAL OPERATOR

Inverse problems with PINO

Inverse problem: $\mathrm{u} = \mathcal{G}(a)$

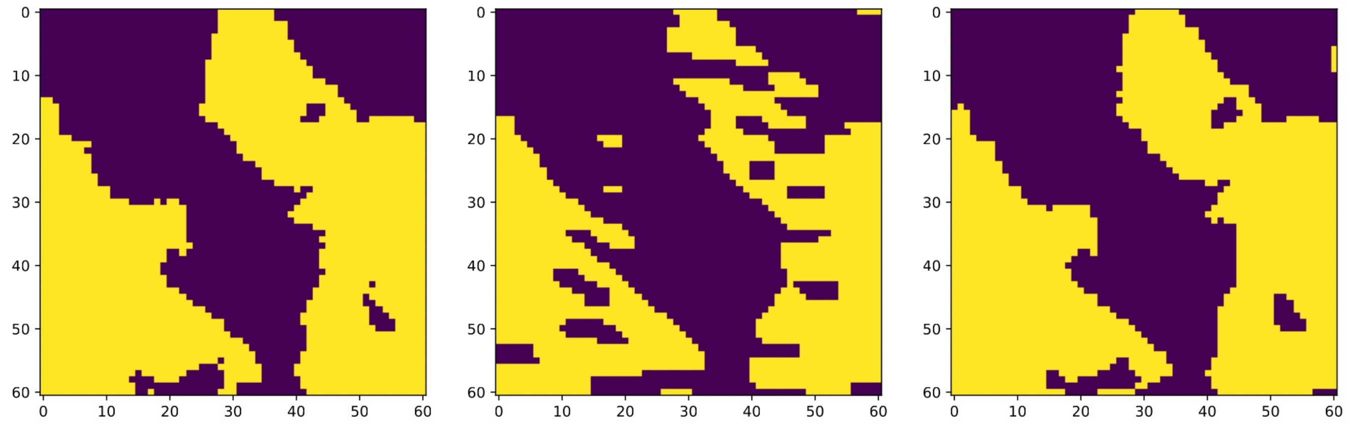
Given the output u, what was the corresponding input a?

Gradient descent find a solution → it might not be physically valid

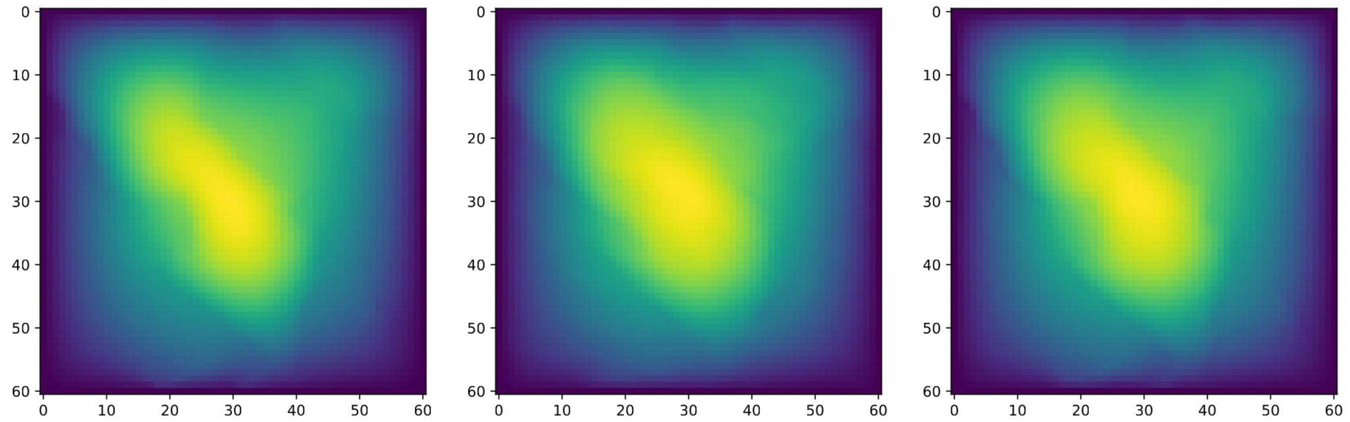
Run Gradient descent with physics in mind.

$$\min_{\mathrm{a}} \|\mathrm{u}^* - \mathcal{G}(a)\|$$

$$-\nabla.\left(a(x)\nabla u(x)\right) = f(x)$$



(a) Ground truth input a (b) Inversion using only data constraint (c) Inversion using data and PDE constraints



(d) Observed output function (e) Output function of inversion using only data constraint (f) Output function of inversion using data and PDE constraints

PINO: PHYSICS-INFORMED NEURAL OPERATOR

How to make PINO principle really practical

Open problems:

- How to impose partial physics
- How to preserve physics quantities
- How to speed/scale up the PINO paradigm
- How to reduce error to sufficient degree
- How to outperform solvers in terms of accuracy

UNCERTAINTY QUANTIFICATION

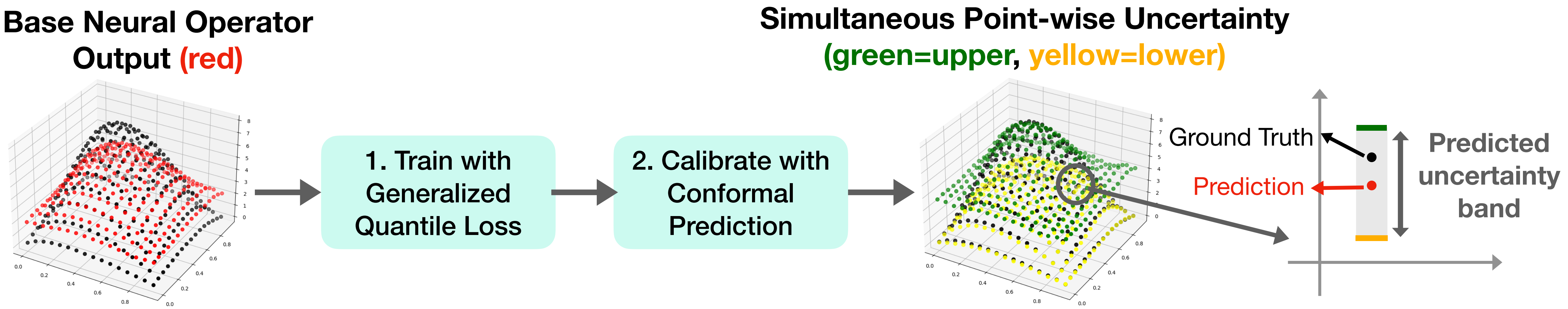
UNCERTAINTY QUANTIFICATION

UQ is ubiquitous in science

Report: 3.63 ± 0.04

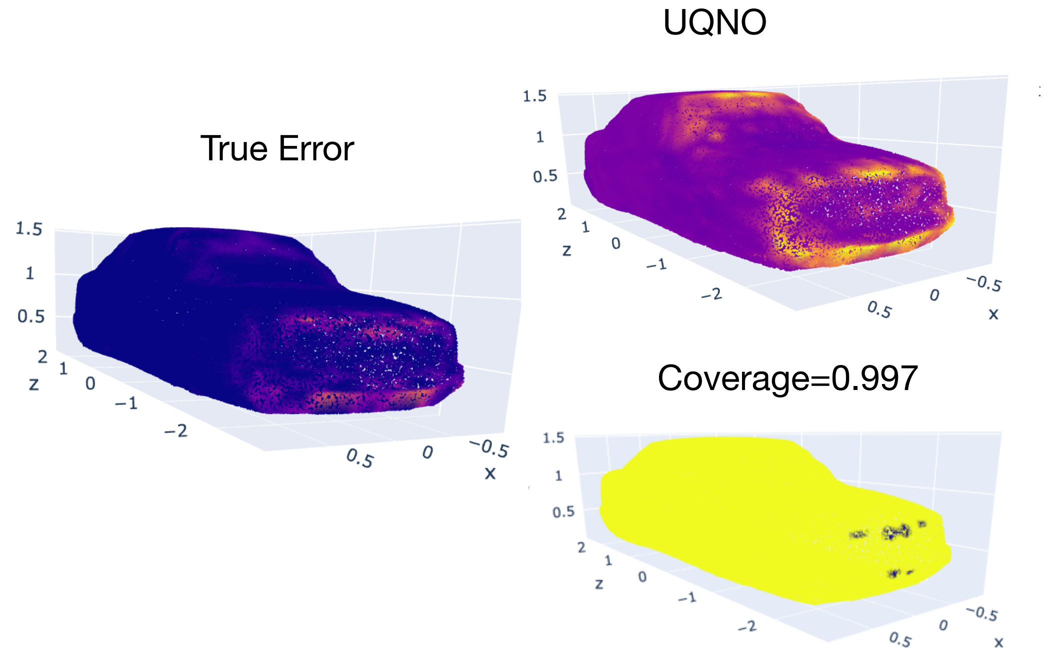
In science, reporting numbers without error bar is almost always unadvised (remember, we learned error analysis first in science classes)

UQ on function spaces is an important area, with only few studies, e.g., generalization of conformal prediction to function spaces



Train a predative neural operator

Train a risk-controlling quantile neural operator



GENERATIVE MODELS IN FUNCTION SPACES

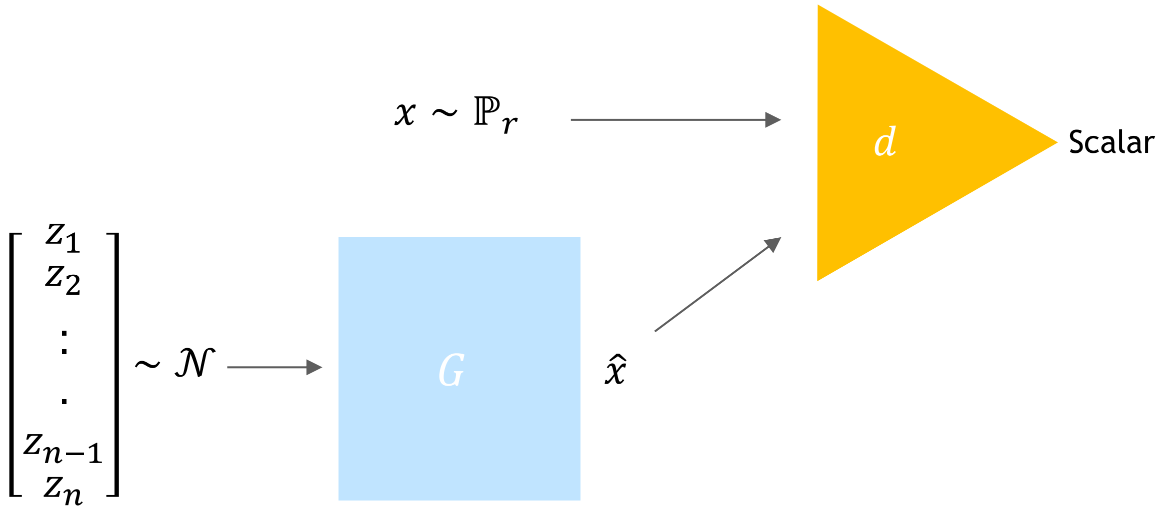
GANO: GENERATIVE ADVERSARIAL NEURAL OPERATOR

Generative model for function spaces

Let's see how we can generalize Wasserstein GAN to function spaces

GAN

G: Generator neural network
d: Discriminator neural network



$x, z, \hat{x}$: Finite dimensional objects

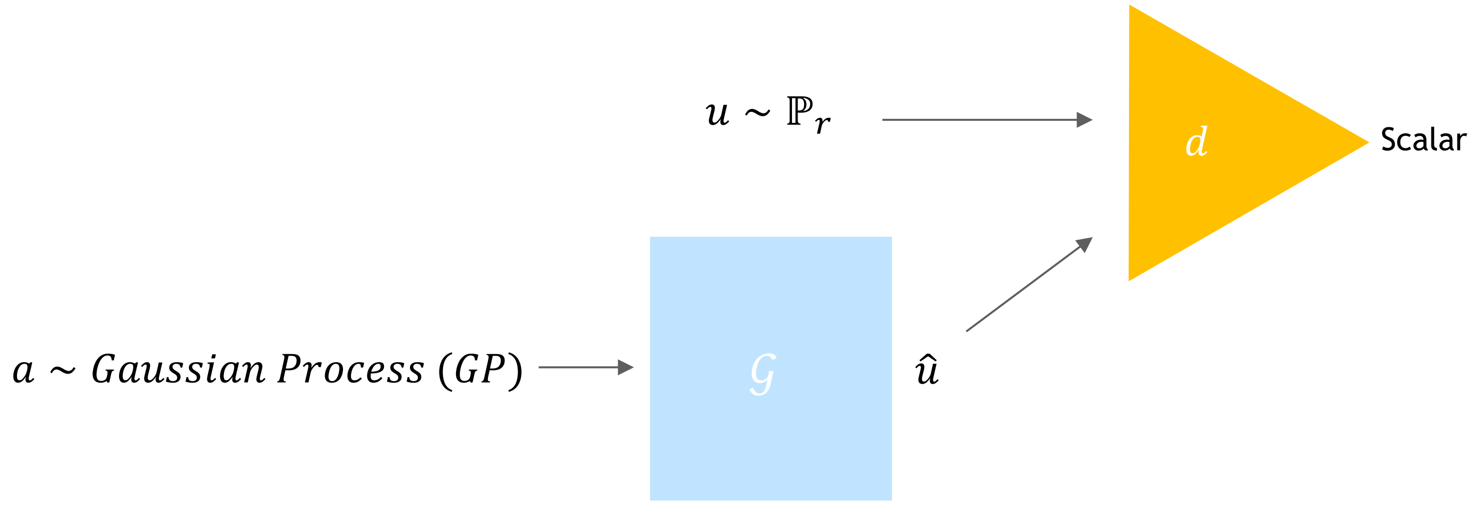
Minimize Wasserstein distance

$$\sup_{\|d\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[d(x)] - \mathbb{E}_{\hat{x} \sim G(z)}[d(\hat{x})]$$

Gradient penalty: $\|\nabla d\| \leq 1$

GANO

$\mathcal{G}$: Generator neural operator
d: Discriminator neural functional



$a, u, \hat{u}$: Infinite dimensional objects

Minimize Wasserstein distance

$$\sup_{\|d\|_L \leq 1} \mathbb{E}_{u \sim \mathbb{P}_r}[d(u)] - \mathbb{E}_{\hat{u} \sim G(a)}[d(\hat{u})]$$

Gradient penalty: $\|\partial d\|_{\mathcal{U}^*} \leq 1$

Fréchet derivative

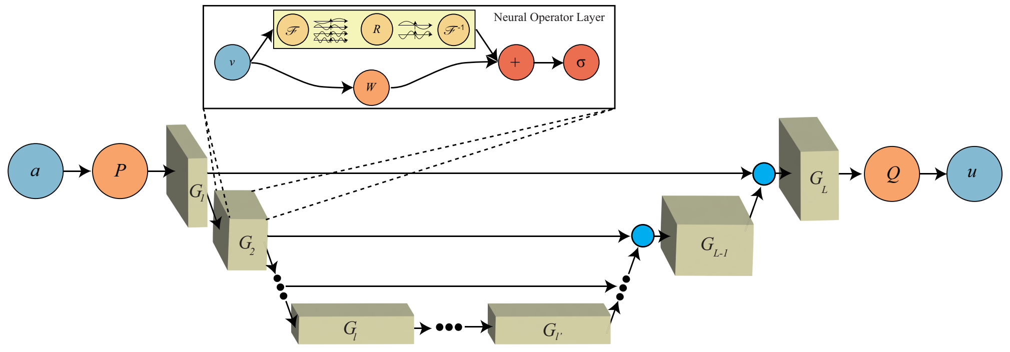
Dual space

GANO: GENERATIVE ADVERSARIAL NEURAL OPERATOR

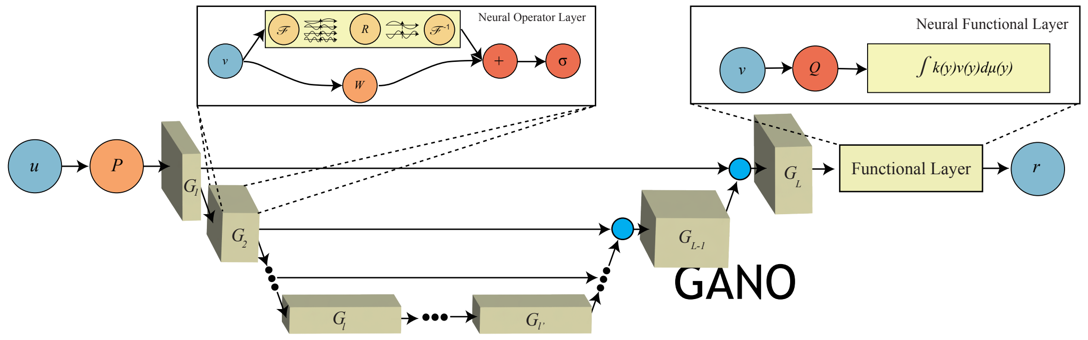
Train and generate at different resolutions

GANO in practice

Base model: UNO-FNO



$\mathcal{G}$: Generator neural operator



d: Discriminator neural functional

Gradient penalty: $\|\partial d\|_{\mathcal{U}^*} \leq 1$

- $1D$ domain: If $u|$ presented on a regular grid of size m → $\|\nabla d(u|_m)\| \leq \frac{1}{\sqrt{m}}$
- $2D$ domain: If $u|$ presented on a regular grid of size $m_1 \times m_2$ → $\|\nabla d(u|_m)\| \leq \frac{1}{\sqrt{m_1 m_2}}$
- For irregular grid, the law needs to be calculated for each datapoint

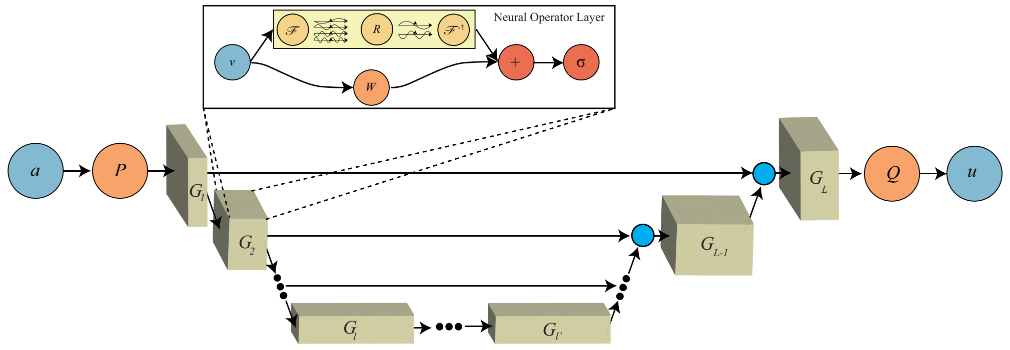
Models	**GANO**	**GAN**
Input/output spaces	Function Spaces	Euclidean spaces
Input measure	Gaussian Random Fields	Multivariate random variables
Controls	length scale, variance, energy, etc.	dimension, variance, etc.

GANO: GENERATIVE ADVERSARIAL NEURAL OPERATOR

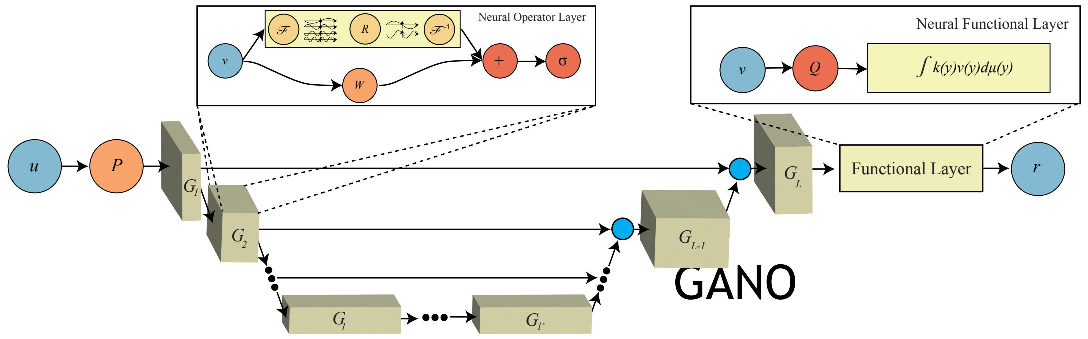
Train and generate at different resolutions

GANO in practice

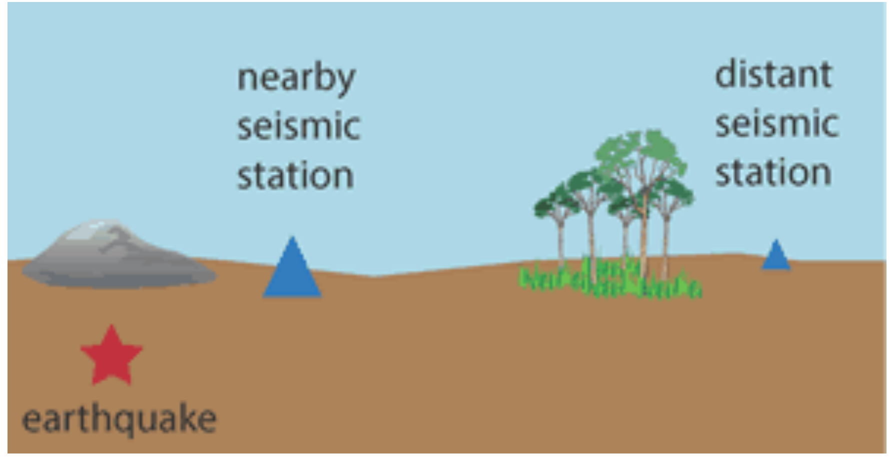
Base model: UNO-FNO



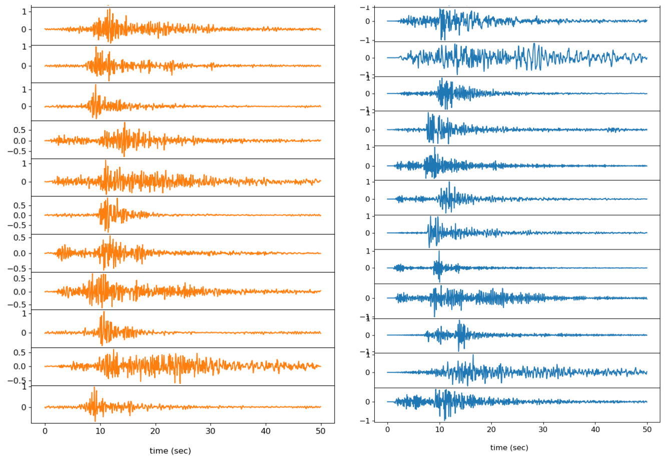
$\mathcal{G}$: Generator neural operator



d: Discriminator neural functional

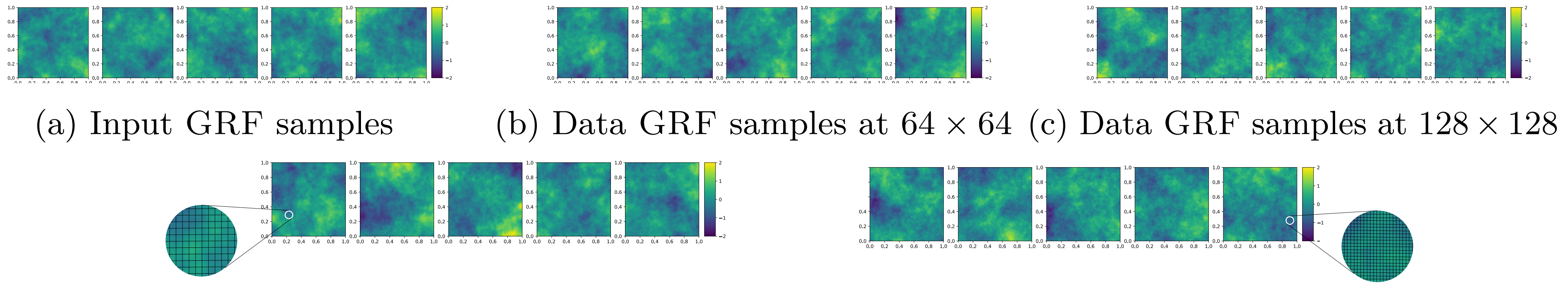


Generated



Seismogram

Train and generation at different resolutions



(a) Input GRF samples (b) Data GRF samples at 64 × 64 (c) Data GRF samples at 128 × 128

(d) GANO generated samples on 64 × 64 (e) GANO generated samples on 128 × 128

DENOISING DIFFUSION OPERATOR (DDO)

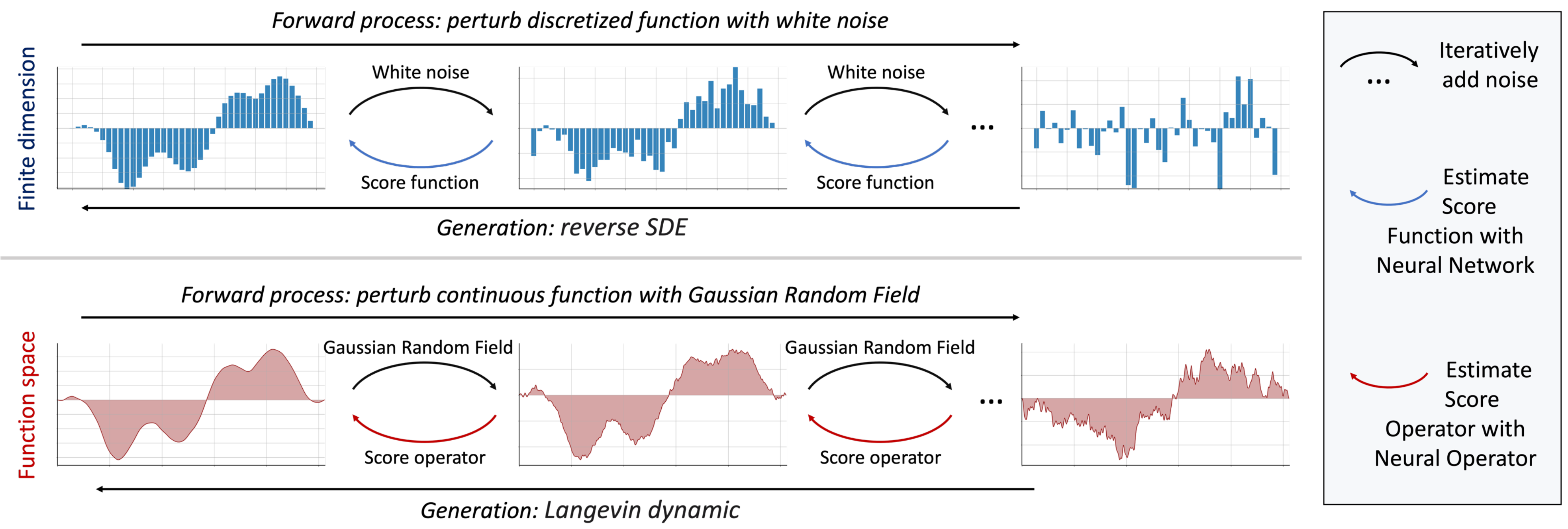
Score based generative model for function spaces

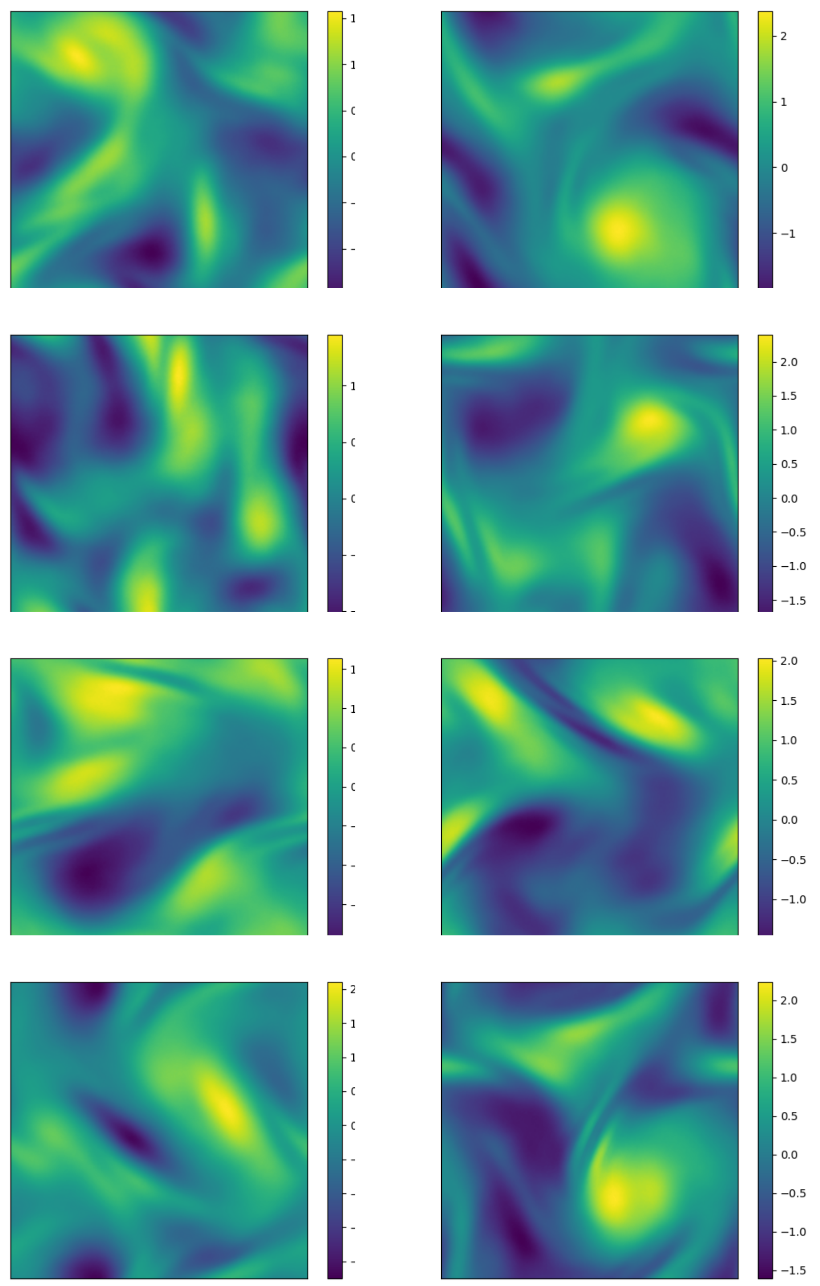
Theory of SDE on function spaces

Score operator vs score function

Multivariate Gaussian for noise, vs GP

Training loss on function space





Training on 128x128, generation on 1024x1024

Score-based Diffusion Models in Function Space
Diffusion generative models in infinite dimensions
Infinite-dimensional diffusion models for function spaces
Functional Diffusion
Infinite-Dimensional Diffusion Models
Conditional score-based diffusion models for Bayesian inference in infinite dimensions
Multilevel diffusion: Infinite dimensional score-based diffusion models for image generation

GENERATIVE MODEL ON FUNCTION SPACES

Variational Autoencoding Neural Operators (VAE to function spaces)

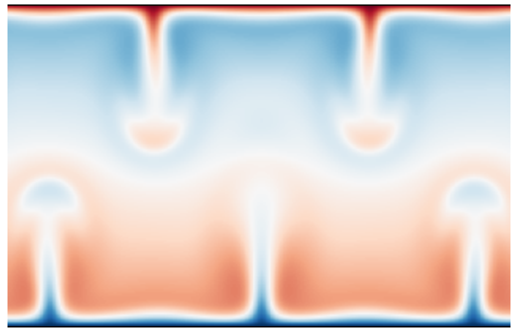
Universal Functional Regression with Neural Operator Flows (Normalizing flow to function spaces)

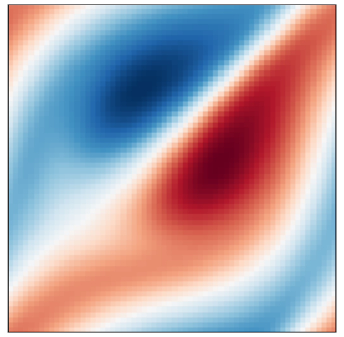
Functional Flow Matching (Flow matching to function spaces

…

UNIVERSAL FUNCTION REGRESSION

Neural Operators to generalize GP regression

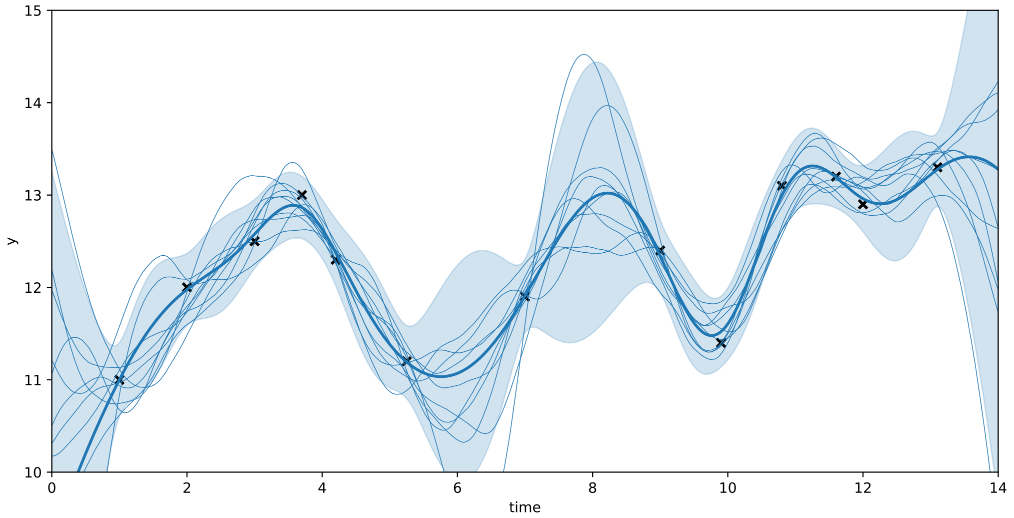


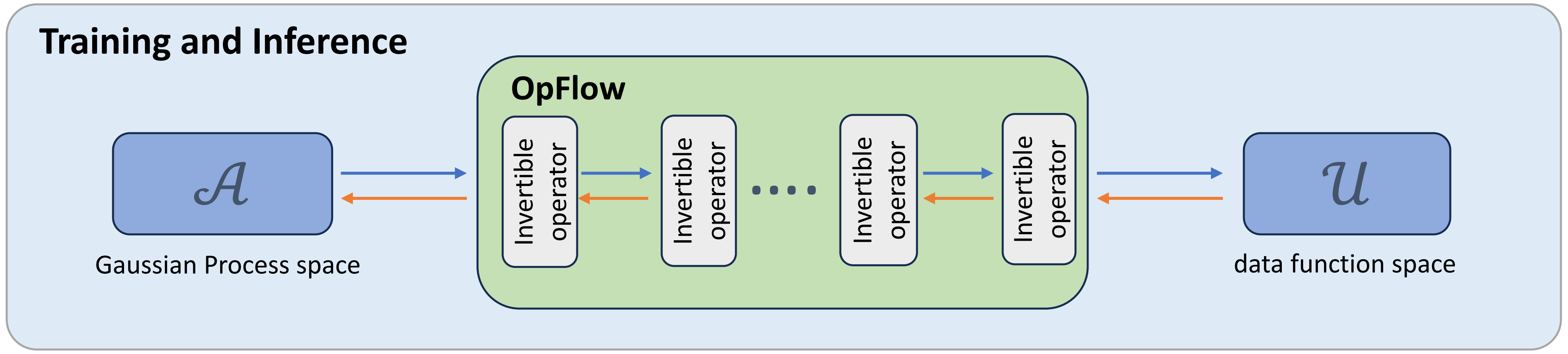


GP regression → Function prior is Gaussian

In real world, function data is not Gaussian

Can we learn maps between stochastic processes?





$$\log p_\theta(u|_D) = \log p(a|_{D_\mathcal{A}}) + \sum_{i=1}^{s} \log|\det(\frac{\partial(v^i|_{D^i_\mathcal{V}})}{\partial(v^{i-1}|_{D^{i-1}_\mathcal{V}})})|.$$

Models	Fast training	Fast inference	Bijective architecture	Exact likelihood estimation
GANO	✗	✓	✗	✗
DDO	✗	✗	✗	✗
VANO	✓	✓	✗	✗
OpFlow	✓	✓	✓	✓

OPEN PROBLEMS

OPEN PROBLEM

Architecture and training

Takeaway message → the whole field is absolutely open

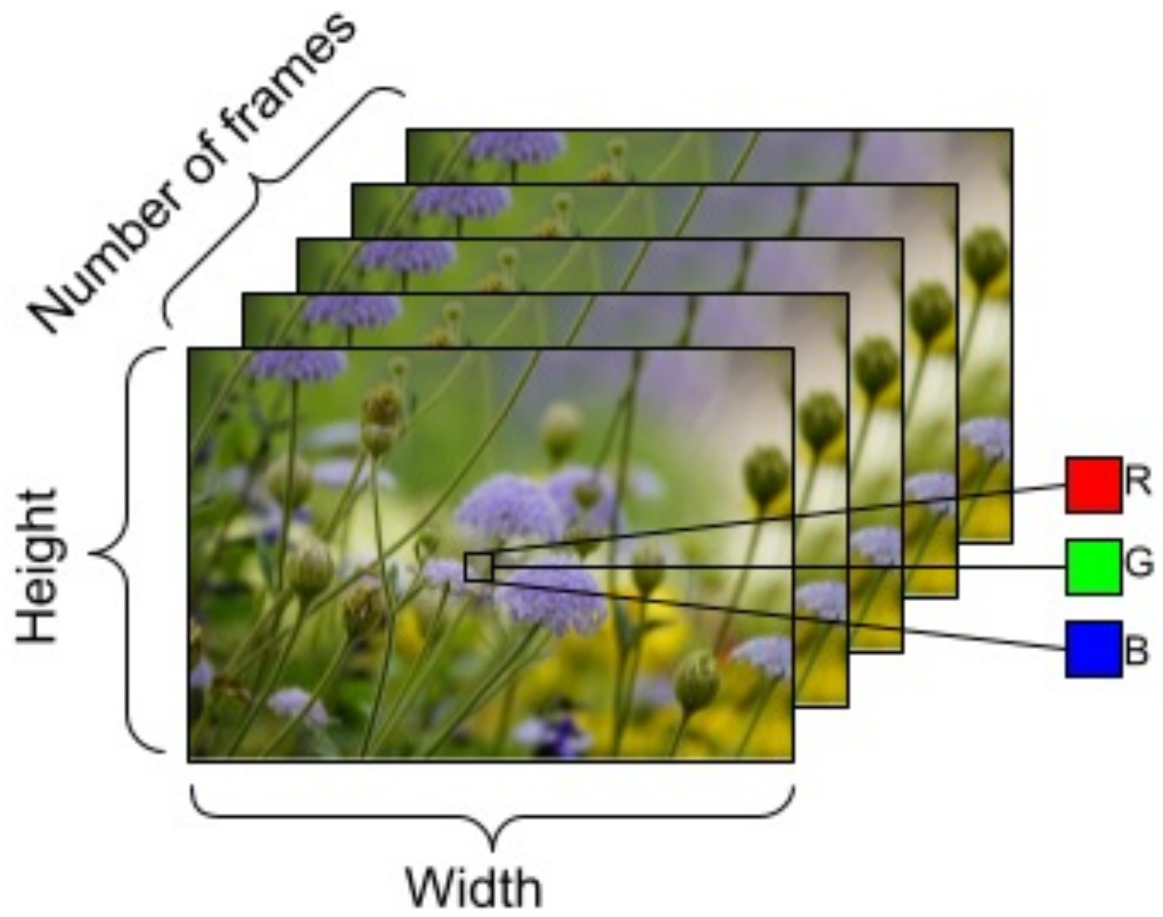
- The neural operator architectures are still primitive
- Also, what other ways of information aggregation? Max pooling? We need consistency in some sense
- Necessary components yet to be explored
- Scaling up is a big challenge
- The accuracies of supervised models are yet limited
- PINO is the future, and very challenging
- What are the best methods for inverse problem
- What happens if we train on low res and test on high res, and vice versa? Theory and practice
- What is the deep learning theory for neural operators? What is width?
- The resolution in the intermediate layers is designer choice, how it should be done?
- The last layer of Neural Operator models is similar to NeRF, how we can bridge between these two?

OPEN PROBLEM

Computer vision, image as a function

Takeaway message → the whole field is absolutely open

- Image as a spatial function
- Video as spatial temporal function



FNOSEG3D: RESOLUTION-ROBUST 3D IMAGE SEGMENTATION WITH FOURIER NEURAL OPERATOR
DFU: scale-robust diffusion model for zero-shot super-resolution image generation

OPEN PROBLEM

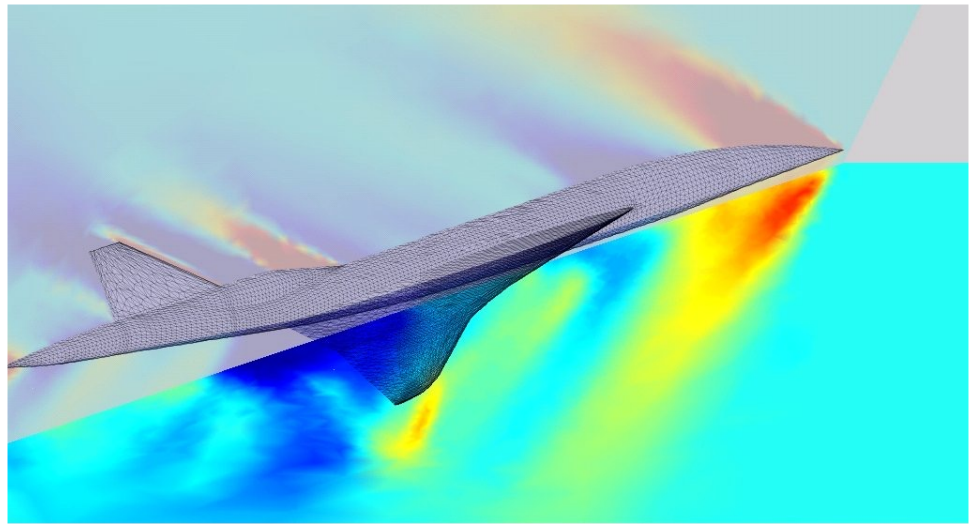
Active learning and UQ

Takeaway message → the whole field is absolutely open

- Data collection and data generation (simulation) are expensive
- Active learning method for data collection and training
- Each scientific domain needs its own active learning method
- UQ is essential in science; what are the principles for UQ in function spaces
- UQ is essential for inverse problem and optimization

Real world wind tunnel experiment

Simulations takes days to months



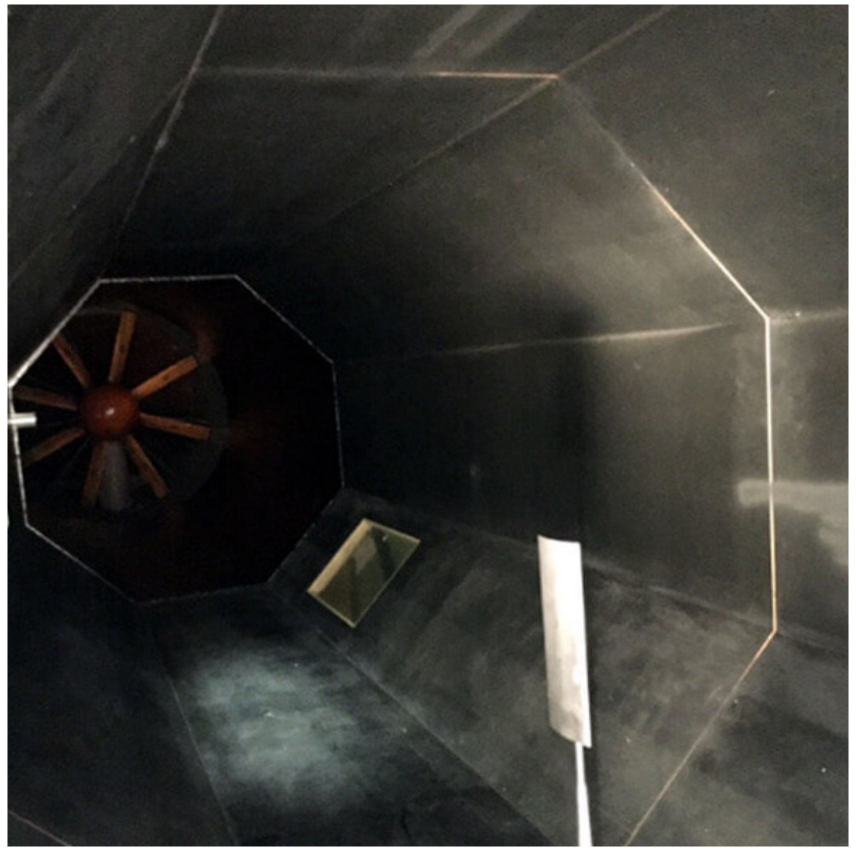
OPEN PROBLEM

RL in function spaces

Takeaway message → the whole field is absolutely open

RL is essential in scientific computing and engineering

- Fluid control
- Wind farm control
- Climate navigation
- Material design
- Drug discovery
- Flow current stabilization
- Online learning in function spaces,

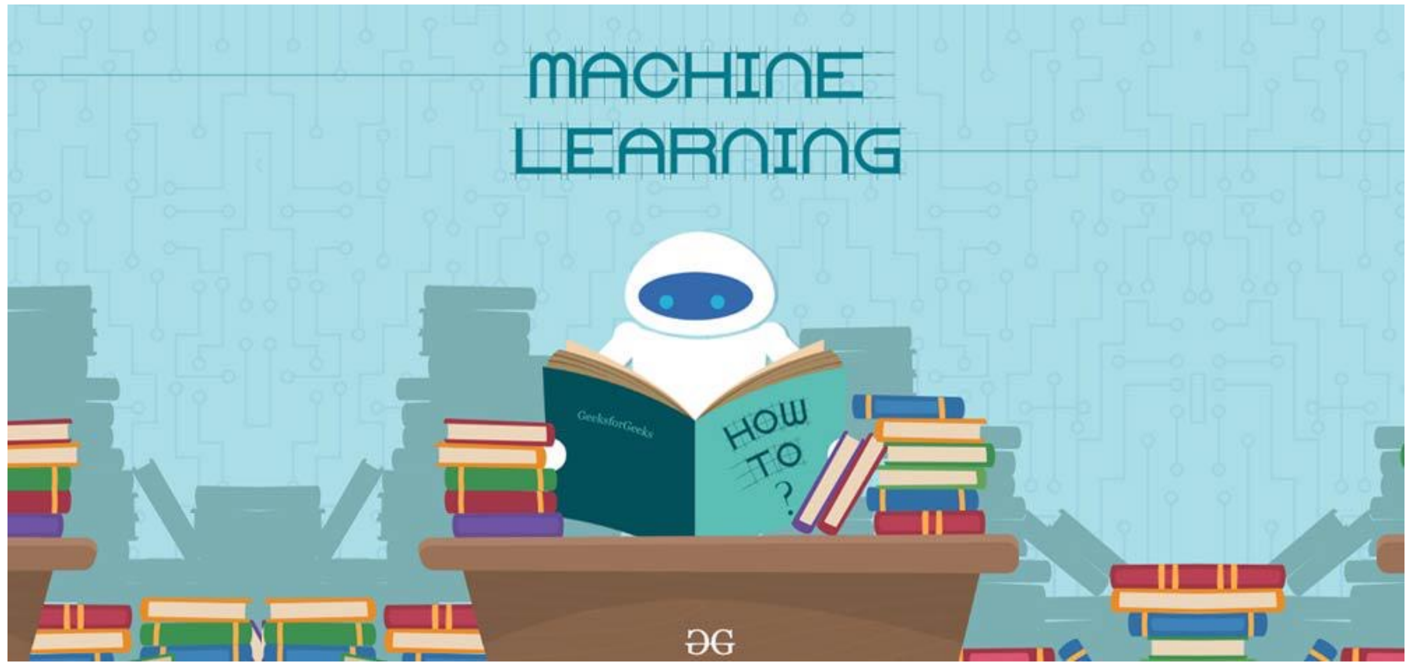


OPEN PROBLEM

Write all of ML on function spaces, with domain in mind

Takeaway message → the whole field is absolutely open

- Unsupervised learning in function spaces,
- Transfer learning and domain adaptation in function spaces,
- Adversarial robustness in function spaces,
- Anomaly Detection in function spaces,
- Self supervised learning in function spaces
- Meta learning in function spaces,
- …



There is urgent need for curating massive datasets for all these domains

COLLABORATION

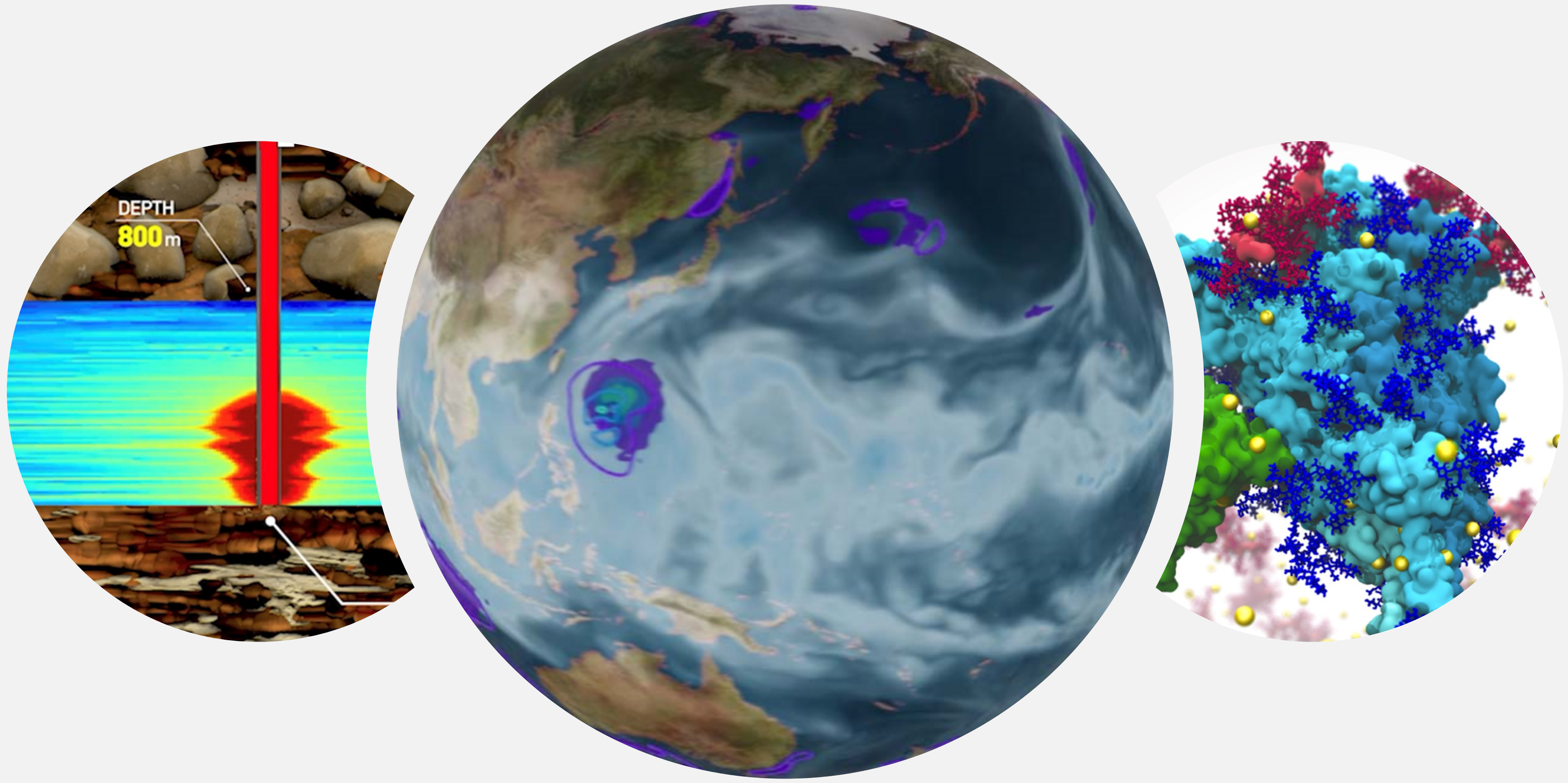
How to collaborate with domain experts

Be aware of challenges:

- Often domain experts are pessimistic about ML
- Initial ML methods are often not as good as the existing paradigms
- Domain experts don’t know much ML as ML-ists don’t know much other field
- Domain experts often don’t have right metrics
- Needs joint development
- Needs building language bridge
- The data is not generated having ML in mind

CONCLUSION

- AI4science is the future of science
- Principled algorithms for zero-shot generalization
- Neural operator extends neural networks to learning in infinite dimensional spaces
- Orders of magnitude speedup while maintaining accuracy



RESOURCES

Neural operator library :

https://neuraloperator.github.io/neuraloperator/dev/index.html

Nvidia Modulus

https://docs.nvidia.com/modulus/index.html

RESOURCES

Nature: - Neural Operators for Accelerating Scientific Simulations and Design

GNO: Neural Operator: - Graph Kernel Network for Partial Differential Equations

Multi-pole: - Multipole Graph Neural Operator for Parametric Partial Differential Equations

FNO: - Fourier Neural Operator for Parametric Partial Differential Equations

PINO: - Physics-informed machine learning: case studies for weather and climate modelling

: - Physics-Informed Neural Operators with Exact Differentiation on Arbitrary Geometries

Multi-grid: - Multi-Grid Tensorized Fourier Neural Operator for High-Resolution PDEs

Theory: - Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs

UNO: - U-NO: U-shaped Neural Operators

Differential operator: - Neural Operators with Localized Integral and Differential Kernels

DNO: - Fast Sampling of Diffusion Models via Operator Learning

Precision: - Guaranteed Approximation Bounds for Mixed-Precision Neural Operators

CoDANO: - Pretraining Codomain Attention Neural Operators for Solving Multiphysics PDEs

MD: - Equivariant Graph Neural Operator for Modeling 3D Dynamics

UQ: - Calibrated Uncertainty Quantification for Operator Learning via Conformal Prediction

Material science: - A learning-based multiscale method and its application to inelastic impact problems

Tipping points: - Tipping Point Forecasting in Non-Stationary Dynamics on Function Spaces

GINO: - Geometry-Informed Neural Operator for Large-Scale 3D PDEs

Seismology: - Seismic wave propagation and inversion with Neural Operators

: - Rapid Seismic Waveform Modeling and Inversion with Universal Neural Operators

FourcastNet: - Fourcastnet: A global data-driven high-resolution weather model using adaptive Fourier neural operators

- Calibration of Large Neural Weather Models

- Spherical Fourier Neural Operators: Learning Stable Dynamics on the Sphere

CCS: - U-FNO - an enhanced Fourier neural operator based-deep learning model for multiphase flow

: - Real-time high-resolution CO2 geological storage prediction using nested Fourier neural operators

GANO: - Generative Adversarial Neural Operators

F^2ID: - PaCMO: Partner Dependent Human Motion Generation in Dyadic Human Activity using Neural Operators

DDO: Score-based Diffusion Models in Function Space

SOURCES OF FIGURES

Figures

https://freecontent.manning.com/neural-network-architectures
https://images.app.goo.gl/64BSh361KX78THzXA
https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
https://towardsdatascience.com/u-net-explained-understanding-its-image-segmentation-architecture-56e4842e313a
https://d2l.ai/chapter_convolutional-modern/resnet.html
https://arxiv.org/abs/1706.03762
https://pubs.geoscienceworld.org/ssa/srl/article-abstract/90/3/1268/568984/Broadband-0-5-Hz-Fully-Deterministic-3D-Ground?redirectedFrom=fulltext
https://engys.com/applications/automotive
https://videos.mentor-cdn.com/mgc/videos/5400/d7dd0086-eb28-4302-a67e-17c10c13eeca-en-US-video.mp4
https://www.nbcnewyork.com/weather/weather-stories/storms-threaten-tri-state-monday-as-temps-soar-near-90-strong-winds-possible/3869922/
https://gifs.com/gif/molecular-dynamics-simulation-of-a-drug-entering-into-the-binding-site-of-a-target-protein-vQDNLq
https://www.youtube.com/watch?v=8oIQy6fxfCA
https://www.linkedin.com/pulse/really-useful-add-so-many-ai-units-gaming-graphics-cards/
https://iopscience.iop.org/article/10.1088/1741-4326/ad313a/pdf
https://www.nasa.gov/aeronautics/nasa-simulates-a-smooth-ride-to-stabilize-air-taxis/
https://8020engineering.com/cfd-for-marine-design-and-hull-propeller-optimization/
https://x.com/CadenceCFD/status/1739956538368987433
https://www.youtube.com/watch?app=desktop&v=TXMPE5mtXcw
http://petersengineering.blogspot.com/2017/05/finite-difference-method.html
https://www.lancaster.ac.uk/stor-i-student-sites/thomas-newman/2022/05/05/gaussian-processes-in-regression/
https://opidesign.net/landscape-architecture/landscape-architecture-fun-facts/
https://www.businessinsider.com/boeing-777x-starts-wind-tunnel-testing-2013-12
https://magazine.viterbi.usc.edu/fact/75_dryden-wind-tunnel/
https://www.smartdraw.com/collaboration/collaboration.htm
https://www.geeksforgeeks.org/machine-learning/
https://www.superannotate.com/blog/what-is-natural-language-processing
https://www.mdpi.com/2073-4441/15/5/850
https://www.researchgate.net/figure/a-A-portion-of-the-finite-element-mesh-showing-the-magma-chamber-and-an-inward-dipping_fig2_343026921
https://www.researchgate.net/figure/Dual-mesh-near-body-off-body-overset-grid-system_fig1_242397432
https://www.learncax.com/knowledge-base/blog/by-category/cfd/good-looking-mesh-may-not-always-be-good-for-the-simulation
https://www.nasa.gov/aeronautics/research-model-increases-
accuracy/?utm_source=FBPAGE&utm_medium=NASA+Aeronautics&utm_campaign=NASASocial&linkId=477679258&fbclid=IwZXh0bgNhZW0CMTAAAR3DTq9ozrNiYTdNjeUEIfjk96vDOJ78VflWMWYopNTWal3PT5FaxStAvQ4_aem_qLF6HUttvhnI_tQnDiZThA

THANK YOU.

Anima Anandkumar
Kaushik Bhattacharya
Zachary E. Ross
Zongyi Li
Andrew M Stuart
Nikola Kovachki
Burigede Liu
Jean Kossaifi
Md Ashiqur Rahman
Robert W. Clayton
Gege Wen
Sally Benson
Miguel Liu-Schiaffini
Boris Bonev
Weili Nie
Chris Choy
Boyi Li
Hongyu Sun
Christian Hundt

Yaozhong Shi
Marius Koch
Jan Kautz
Mohammad Amin Nabian
Maximilian Stadler
Weiqiang Zhu
Daniel V Leibovici
Renbo Tu
Gennady Pekhimenko
Grigorios Lavrentiadis
Domniki Asimaki
Caifeng Zou
Björn Lütjens
Suyash Bire
David Pitt
Minkai Xu
Jure Leskovec
Arvind Ramanathan

Stefano Ermon
Jiaqi Han
Aaron Lou
Thorsten Kurth
Angela F. Gao
Mogab Elleithy
Anirban Chandra
Suraj Pawar
Aniruddha Panda
Jeroen Snippe
Faruk O. Alpak
Farah Hariri
Clement Etienam
Pandu Devarakota
Detlef Hohl
Robert Joseph George
Andre Graubner
Tapio Schneider

Noah Brenowitz
Clare E. Singer
Akshay Subramaniam
Dale Durran
Maximilian Baust
Farah Hariri
Karsten Kreis
Ricardo Baptista
Jiaming Song
Jae Hyun Lim
Christopher Beckham
Chris Pal
Karthik Kashinath
Haoxuan Chen
Julius Berner
Colin White
Jaideep Pathak
Morteza Mardani

Hongkai Zheng
Peter Harrington
Shashank Subramanian
Philip Marcus
Yan Yang
Arash Vahdat
Mike Pritchard
Sanjeev Raja
Sanjay Choudhry
Yair Cohen