Basis-to-Basis Operator Learning

A Paradigm for Scalable and Interpretable Operator Learning on Hilbert Spaces

Adam Thorpe, Tyler Ingebrand, Somdatta Goswami,
Krishna Kumar, Ufuk Topcu

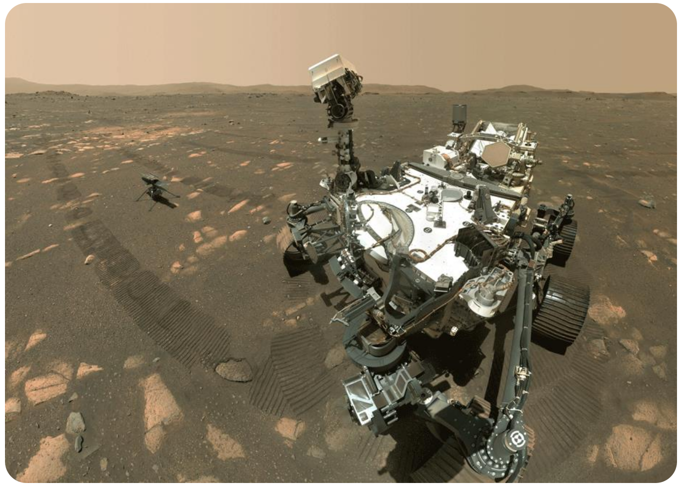
TEXAS The University of Texas at Austin

ODEN INSTITUTE FOR COMPUTATIONAL ENGINEERING & SCIENCES

JOHNS HOPKINS UNIVERSITY

We need algorithms that can **adapt** & **transfer**...

...across domains

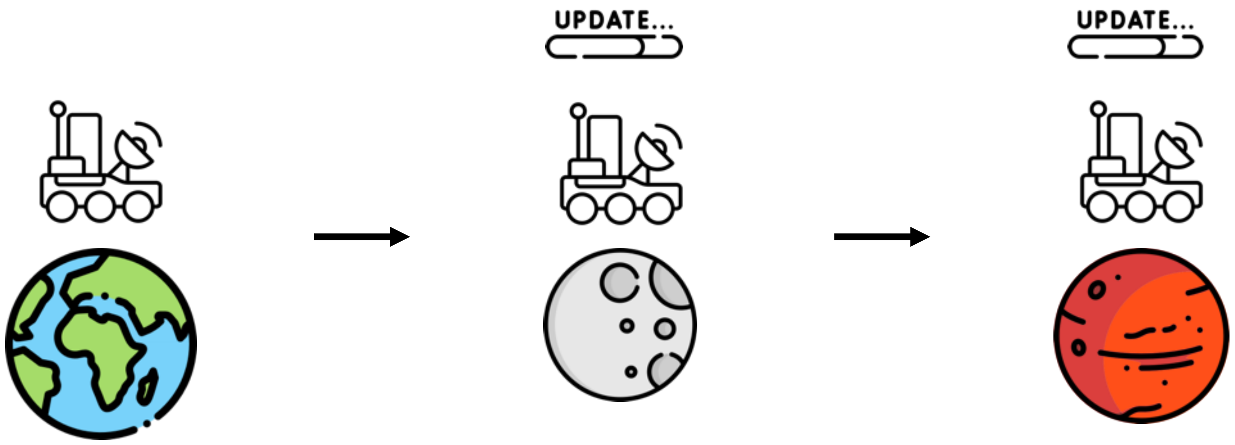


...across tasks

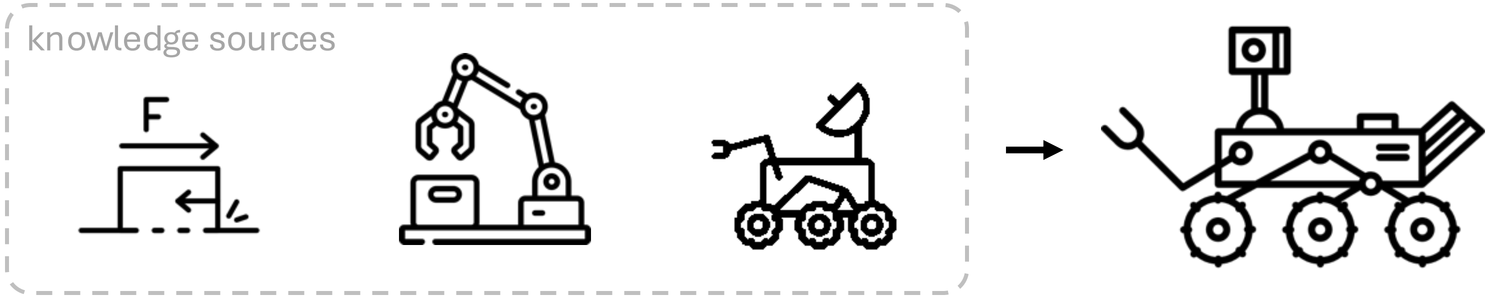
...across platforms

What do I mean by adaptation and transfer?

Adaptation updating or refining learned models using new data

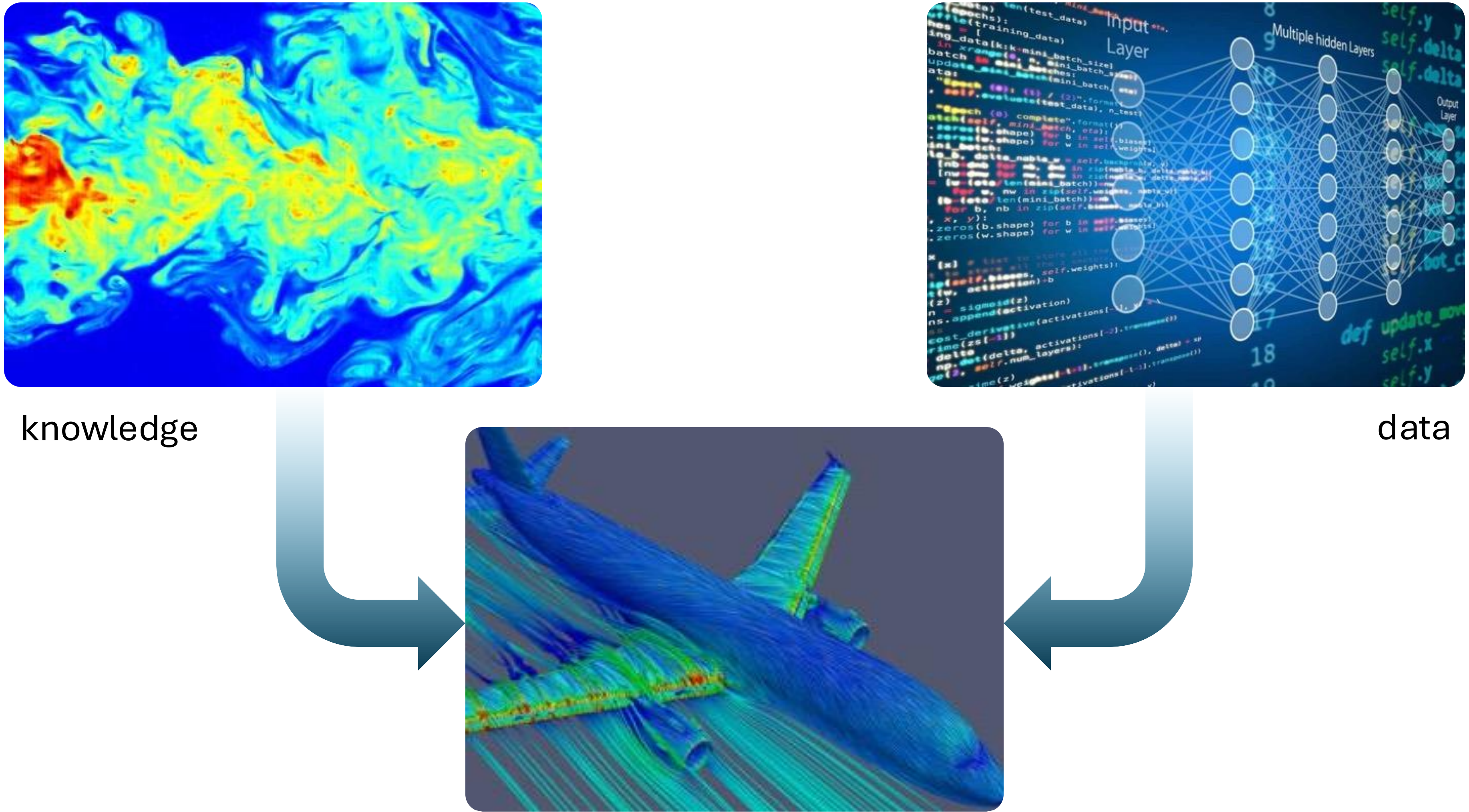


Transfer leveraging knowledge from diverse sources


knowledge sources

Moving beyond black-box learning

Incorporating known physics and mathematical structure

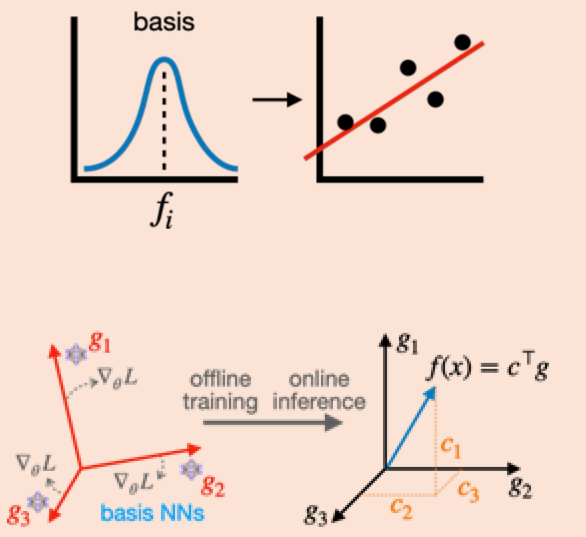


knowledge

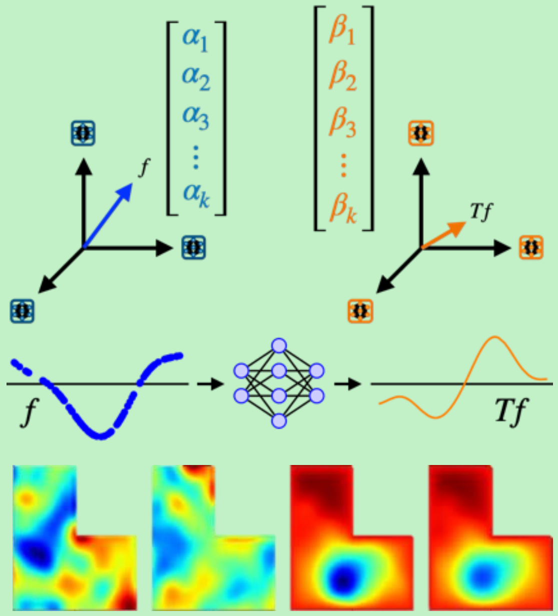
data

Basis-to-basis operator learning

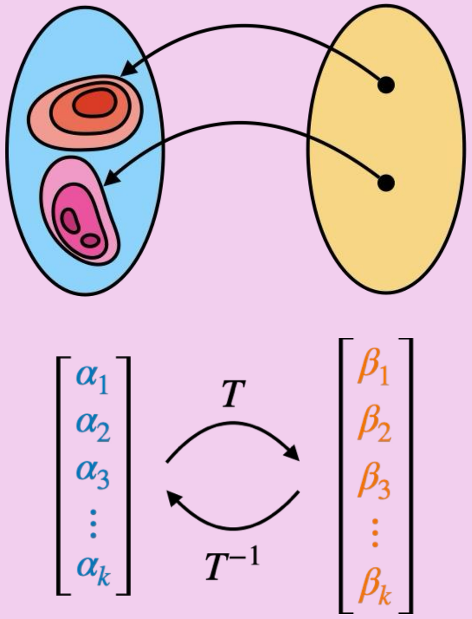
Function Encoders



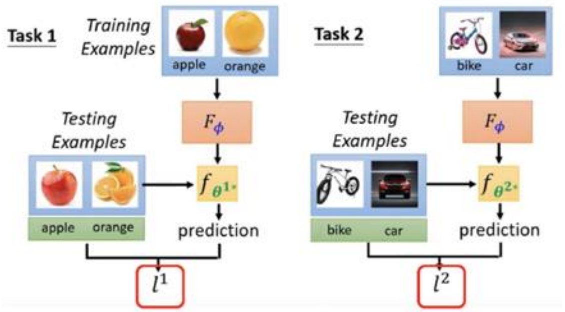
Basis-to-Basis Operator Learning



Inverse Neural Operators

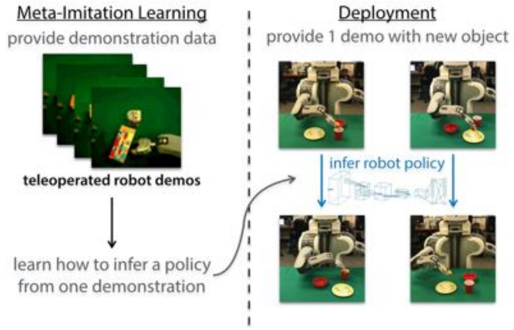


Existing transfer approaches



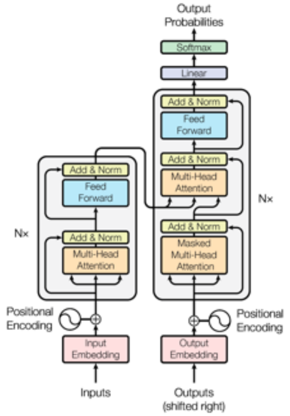
Meta-Learning

Chelsea Finn, Pieter Abbeel, Sergey Levine. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.


Meta-Imitation Learning
provide demonstration data
teleoperated robot demos
learn how to infer a policy from one demonstration
Deployment
provide 1 demo with new object
infer robot policy

Imitation Learning

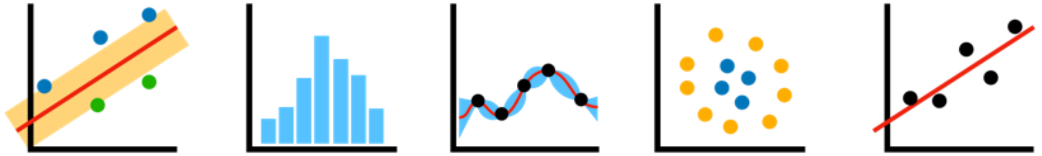
O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., ... & Chen, M. (2024). Open X-Embodiment: Robotic Learning Datasets and RT-X Models



Transformers

Ashish Vaswani, et. al. (2017). Attention is All you Need.

D. Celestini, D. Gammelli, T. Guffanti, S. D'Amico, E. Capello and M. Pavone. (2024). Transformer-Based Model Predictive Control: Trajectory Optimization via Sequence Modeling

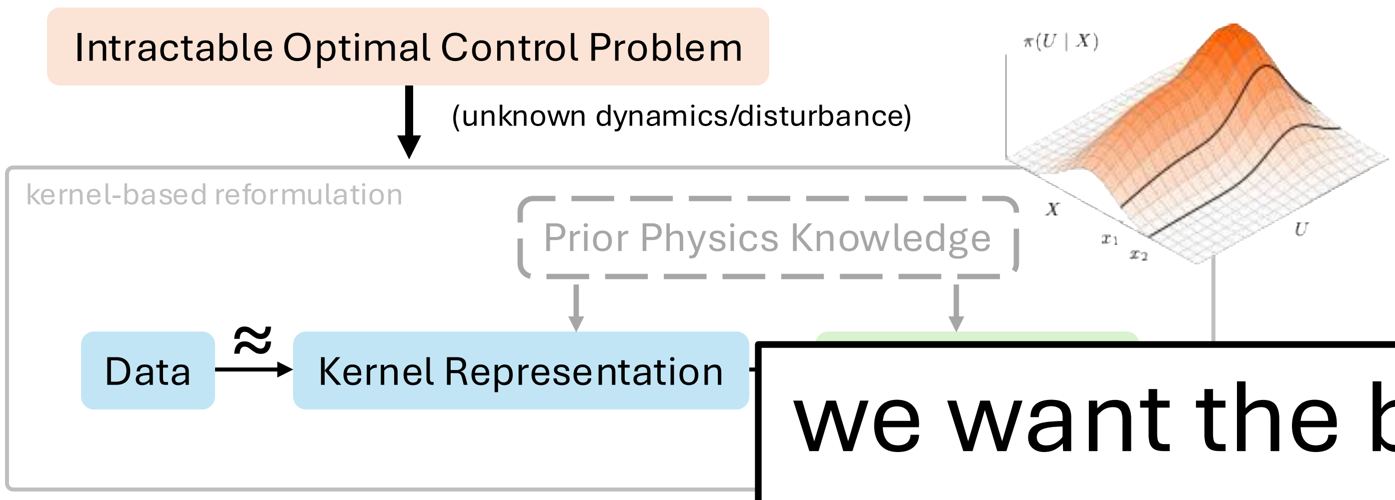


SVMs Density Estimation GPs Kernel PCA Ridge Regression

Hilbert Space Representations

✓ guarantees ✓ interpretable ✓ efficient

Prior work: kernel-based stochastic optimal control


Intractable Optimal Control Problem
(unknown dynamics/disturbance)
kernel-based reformulation
Prior Physics Knowledge
Data
≈
Kernel Representation
$\pi(U \mid X)$
X
x_1
x_2
U

Approximate Policy:

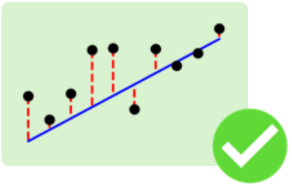
scales with the amount of data

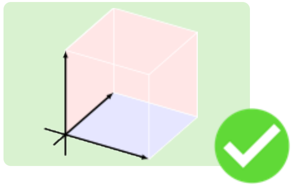
$$\sum_{i=1}^{m} \alpha_i(x) k(u_i, \cdot)$$

we want the best of both worlds

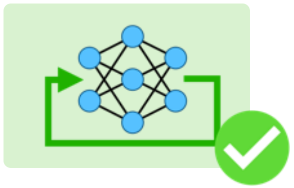
ethods

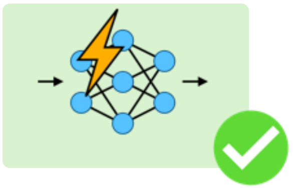
ty

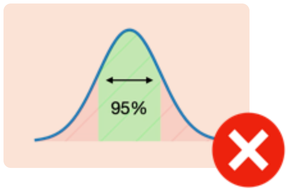

least squares


structure

Neural Networks

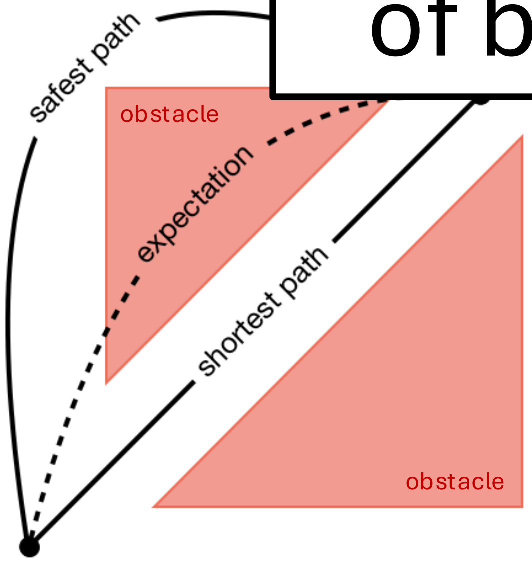

offline training


fast inference



guarantees

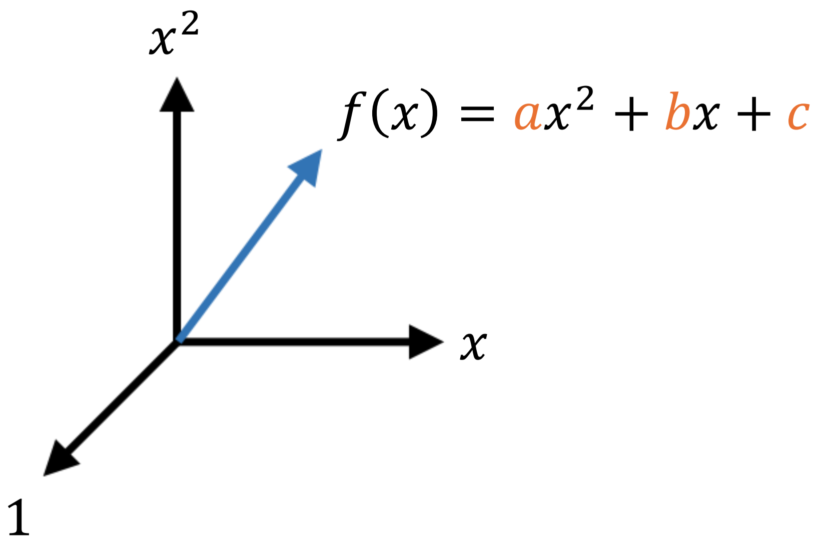
uncertain mass
unknown disturbance
unknown dynamics



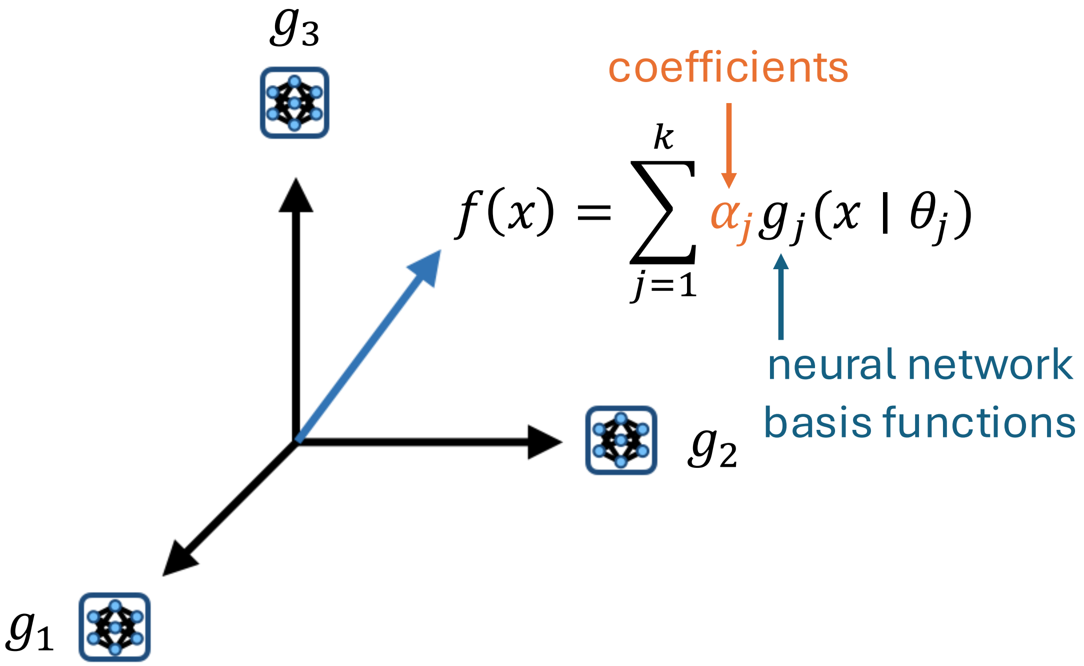
Thorpe, A., Lew, T., Oishi, M. & Pavone, M. (2022). Data-Driven Chance Constrained Control using Kernel Distribution Embeddings. L4DC

Function encoders: combining neural networks and Hilbert spaces

Problem: How can we represent Hilbert spaces?



$$f(x) = ax^2 + bx + c$$



coefficients

$$f(x) = \sum_{j=1}^{k} \alpha_j g_j(x \mid \theta_j)$$

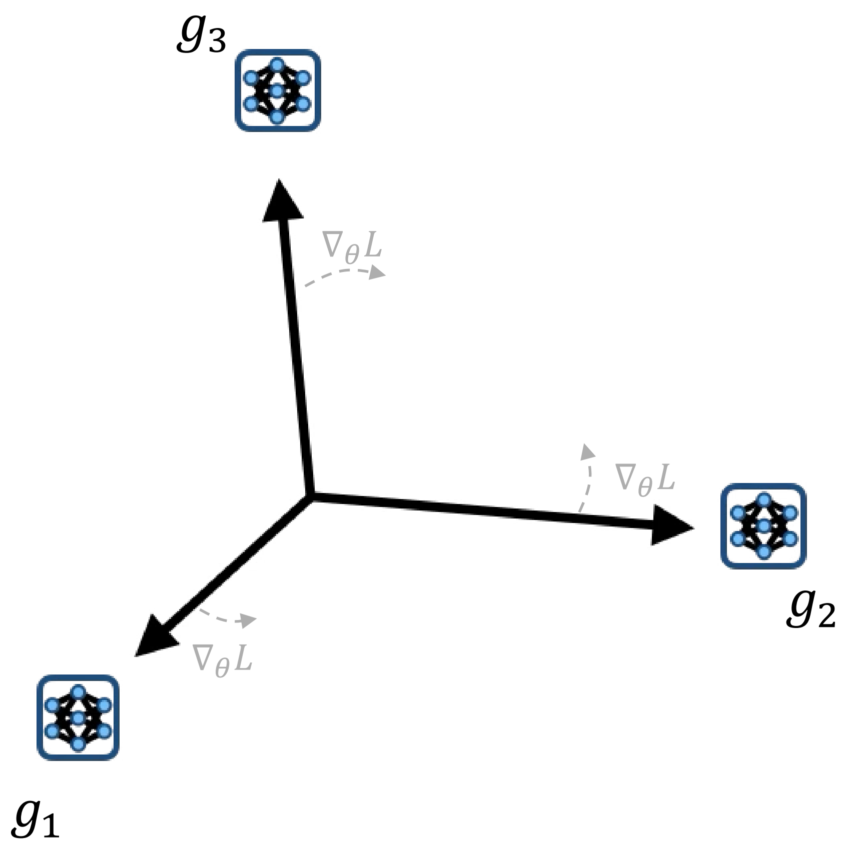
neural network basis functions

	simple polynomial example	**function encoders**
Basis:	$\{1 \quad x \quad x^2\}$	$\{g_1 \quad g_2 \quad g_3 \quad \cdots \quad g_k\}$
Representation:	$[a \quad b \quad c]$	$[\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \cdots \quad \alpha_k]$

Breaking function encoders down: **offline training, online inference**

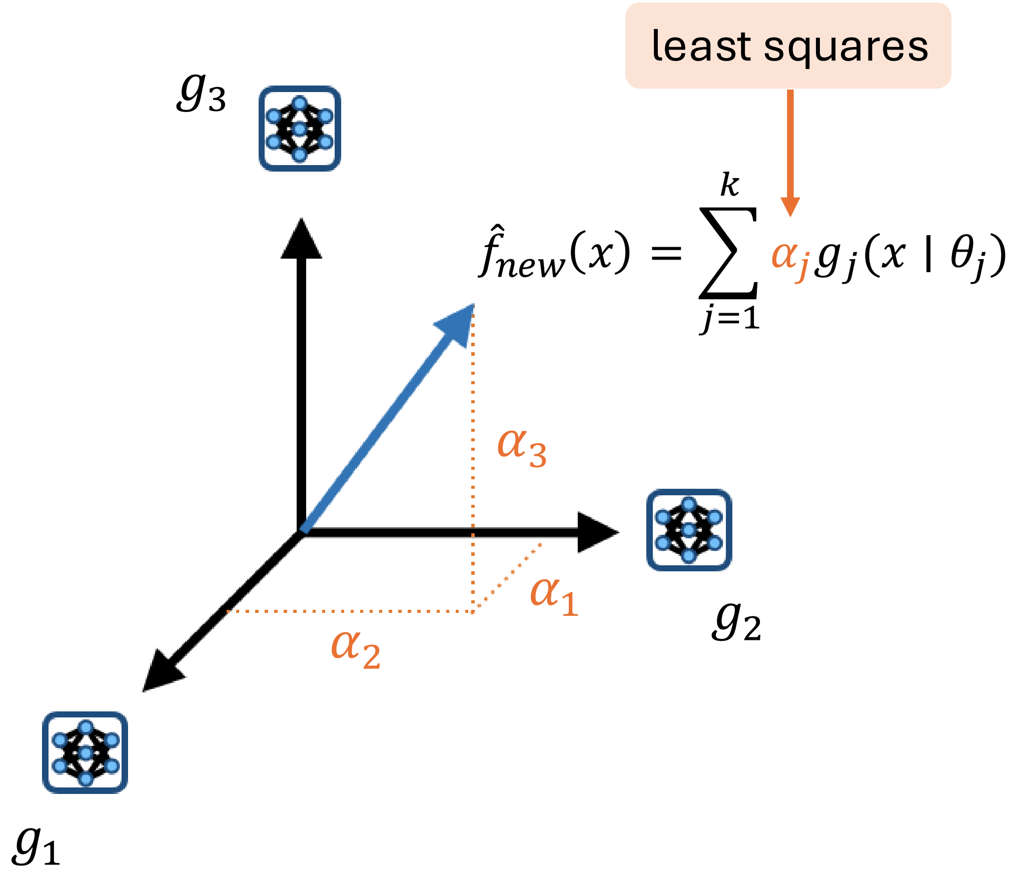
Offline Training

learn the basis functions



Online Inference

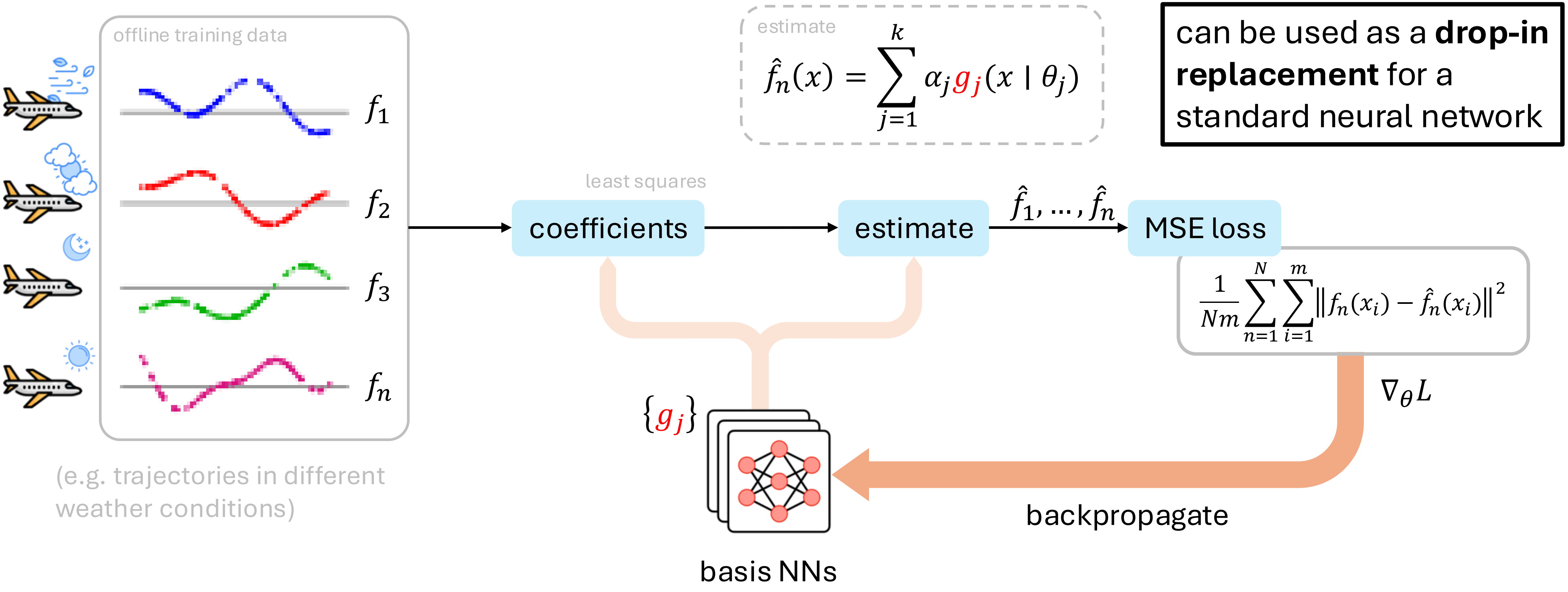
compute the coefficients α



least squares
g_3
$\hat{f}_{new}(x) = \sum_{j=1}^{k} \alpha_j g_j(x \mid \theta_j)$
α_3
α_1
α_2
g_2
g_1

Ingebrand, T., Thorpe, A. J., & Topcu, U. (2025). Function Encoders: A Principled Approach to Transfer Learning in Hilbert Spaces.

Offline Training: Training neural network basis functions



estimate

$$\hat{f}_n(x) = \sum_{j=1}^{k} \alpha_j g_j(x \mid \theta_j)$$

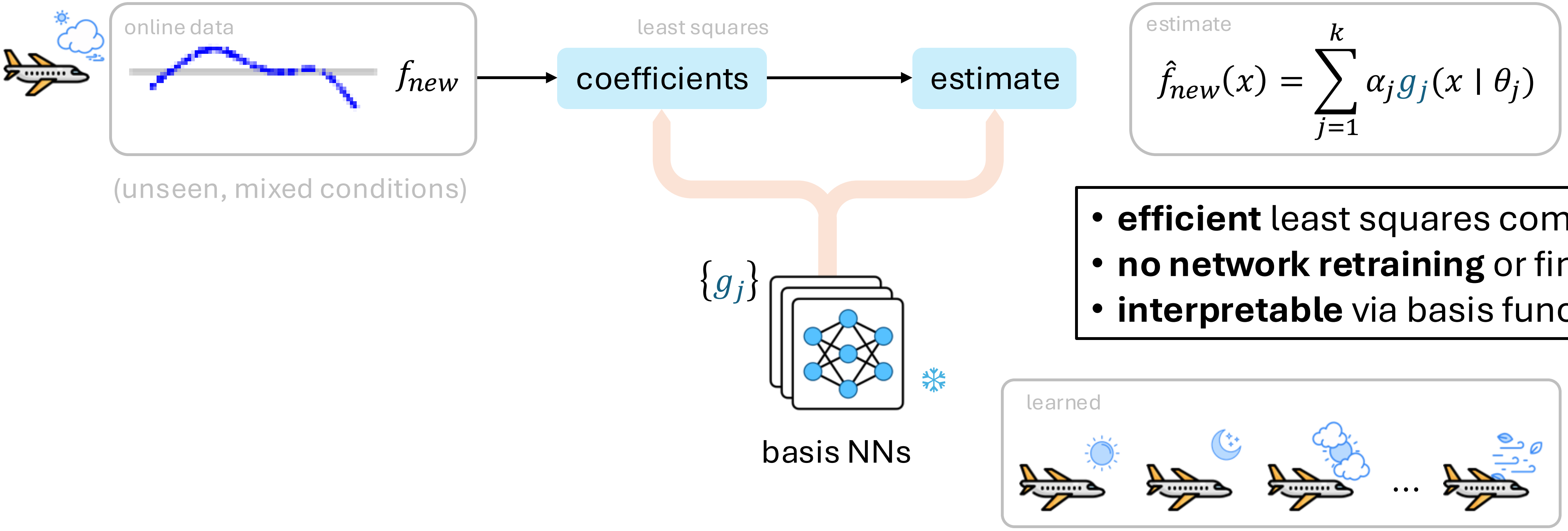
can be used as a **drop-in replacement** for a standard neural network

$$\frac{1}{Nm}\sum_{n=1}^{N}\sum_{i=1}^{m}\left\| f_n(x_i) - \hat{f}_n(x_i) \right\|^2$$

(e.g. trajectories in different weather conditions)

Ingebrand, T., Thorpe, A. J., & Topcu, U. (2025). Function Encoders: A Principled Approach to Transfer Learning in Hilbert Spaces.

Online Inference: Computing coefficients for a new function



$$\hat{f}_{new}(x) = \sum_{j=1}^{k} \alpha_j g_j(x \mid \theta_j)$$

- **efficient** least squares computation
- **no network retraining** or fine-tuning
- **interpretable** via basis functions

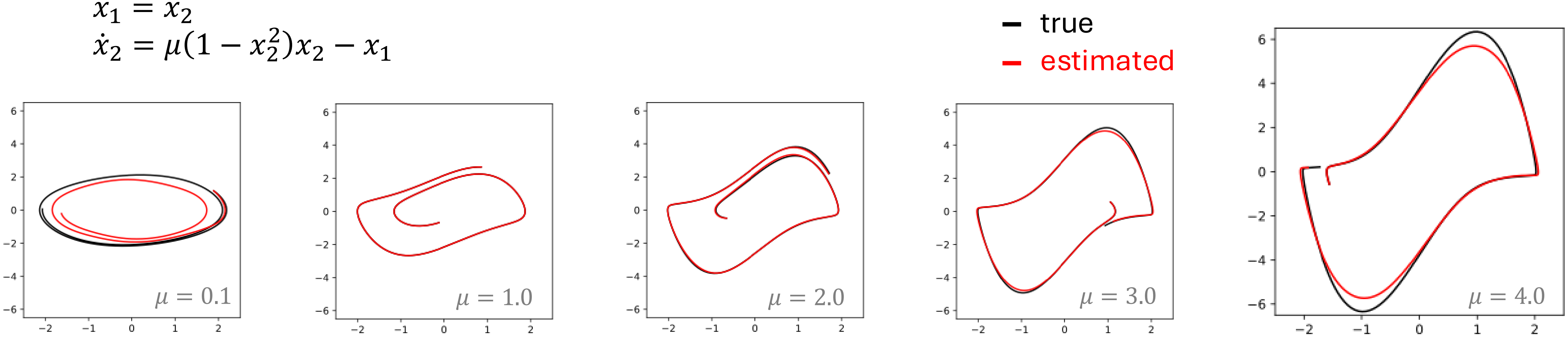
Ingebrand, T., Thorpe, A. J., & Topcu, U. (2025). Function Encoders: A Principled Approach to Transfer Learning in Hilbert Spaces.

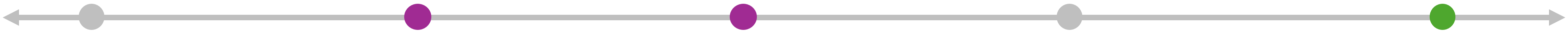
Function encoders enable transfer **beyond** the training data

Using the properties of the Hilbert space for transfer

Van der Pol oscillator:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \mu(1 - x_2^2)x_2 - x_1$$





training data α_1

interpolation in the convex hull of α_1 and α_2

training data α_2

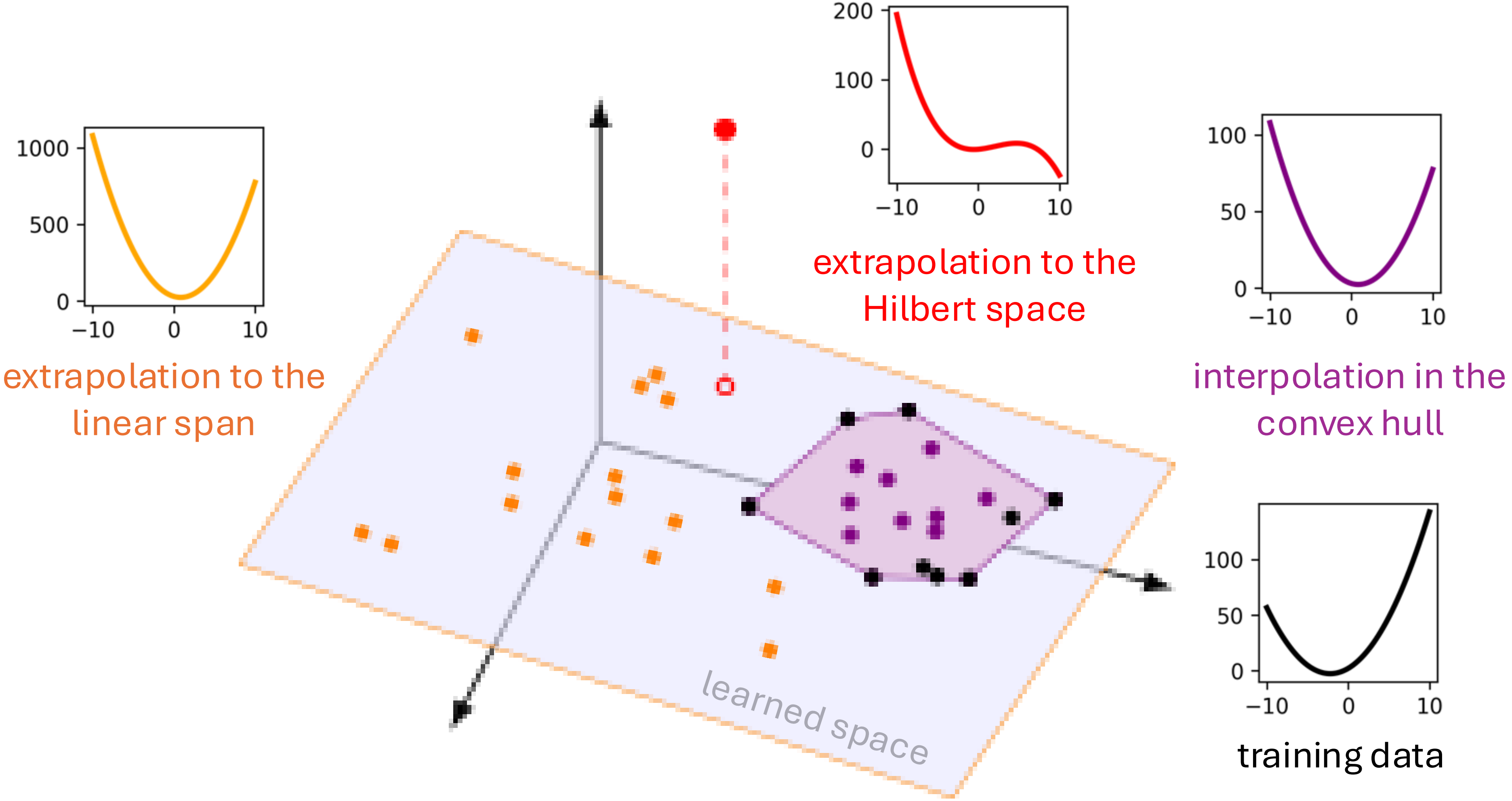
extrapolation outside the convex hull

inside the training data

outside the training data

Ingebrand, T., Thorpe, A. J., & Topcu, U. (2024). Zero-shot transfer of neural ODEs. NeurIPS

A geometric characterization of transfer



extrapolation to the linear span

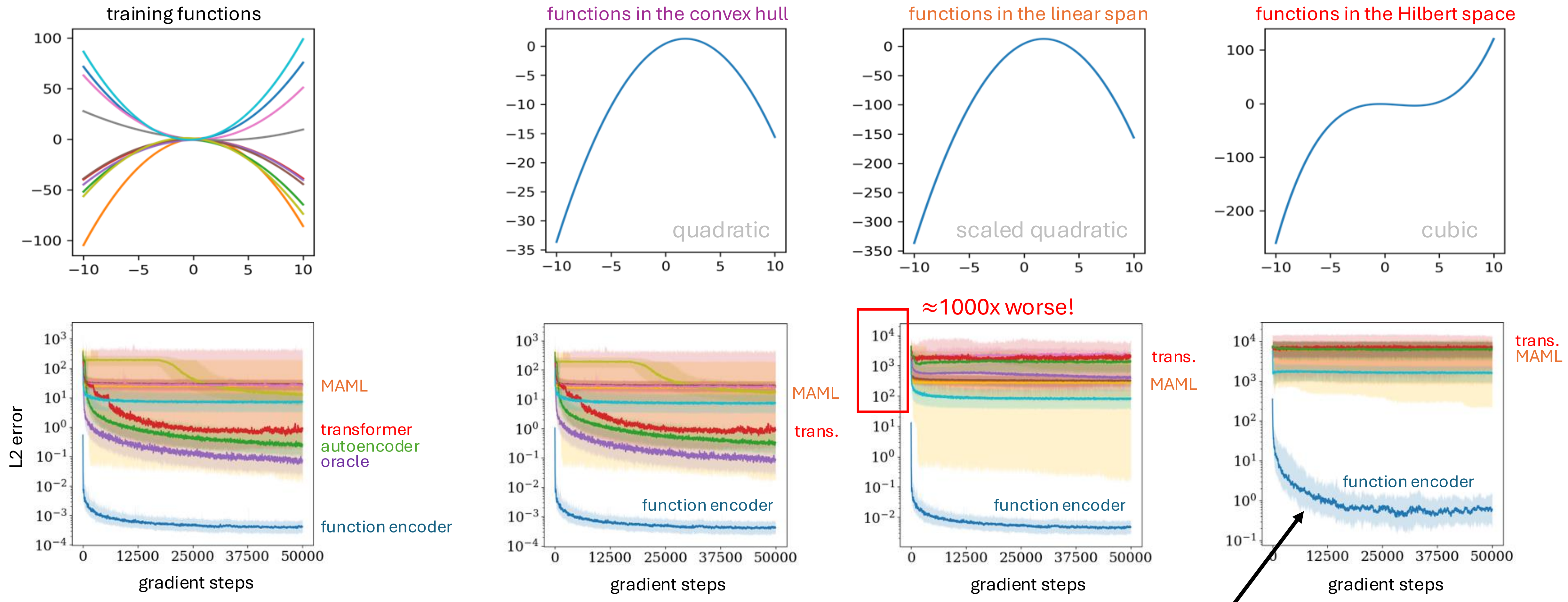
extrapolation to the Hilbert space

interpolation in the convex hull

training data

A very simple transfer test

How well do existing approaches transfer?



training functions

functions in the convex hull

functions in the linear span

functions in the Hilbert space

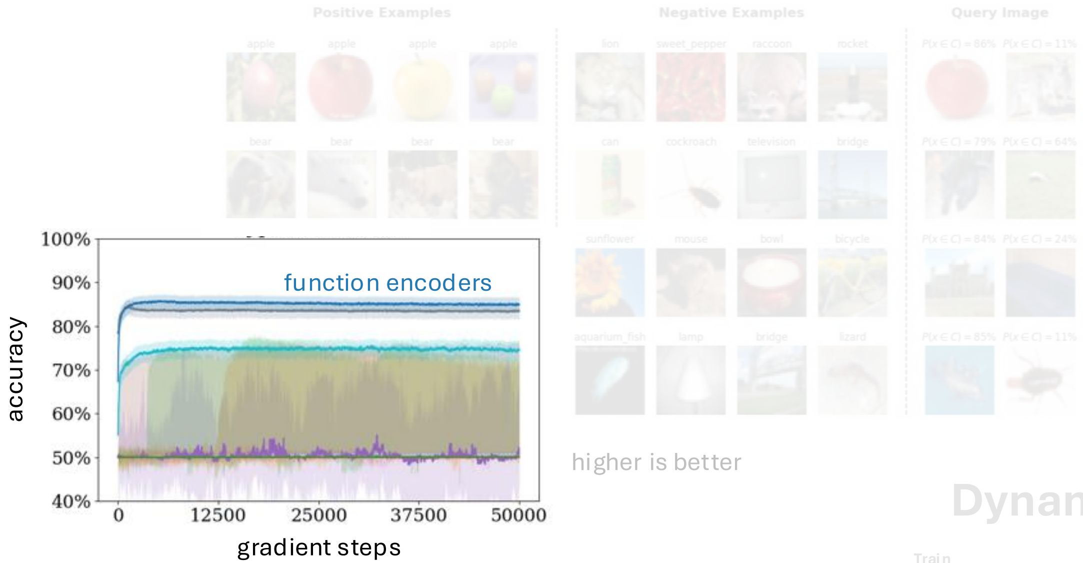
≈1000x worse!

indicates partial transfer

Ingebrand, T., Thorpe, A. J., & Topcu, U. (2025). Function Encoders: A Principled Approach to Transfer Learning in Hilbert Spaces. ICML

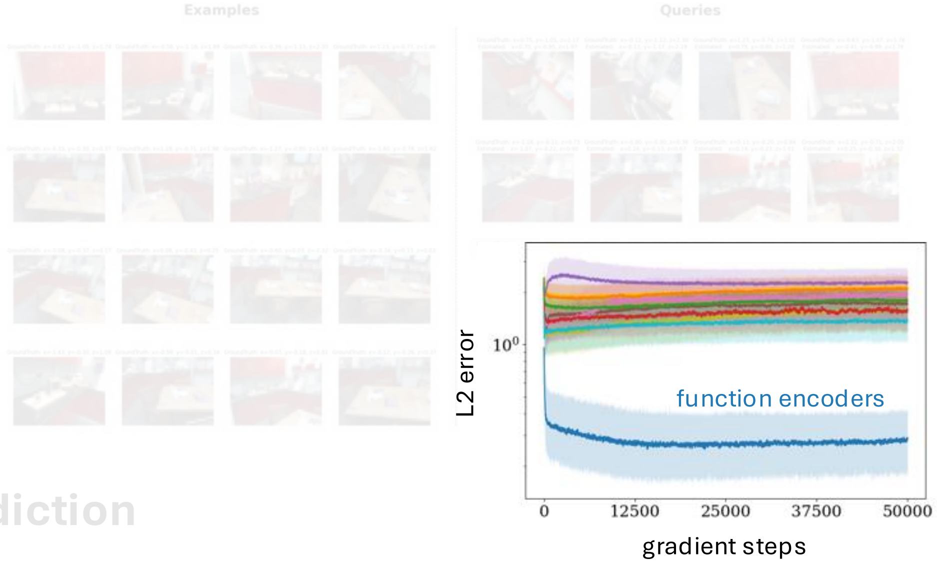
Different transfer applications

Image Classification



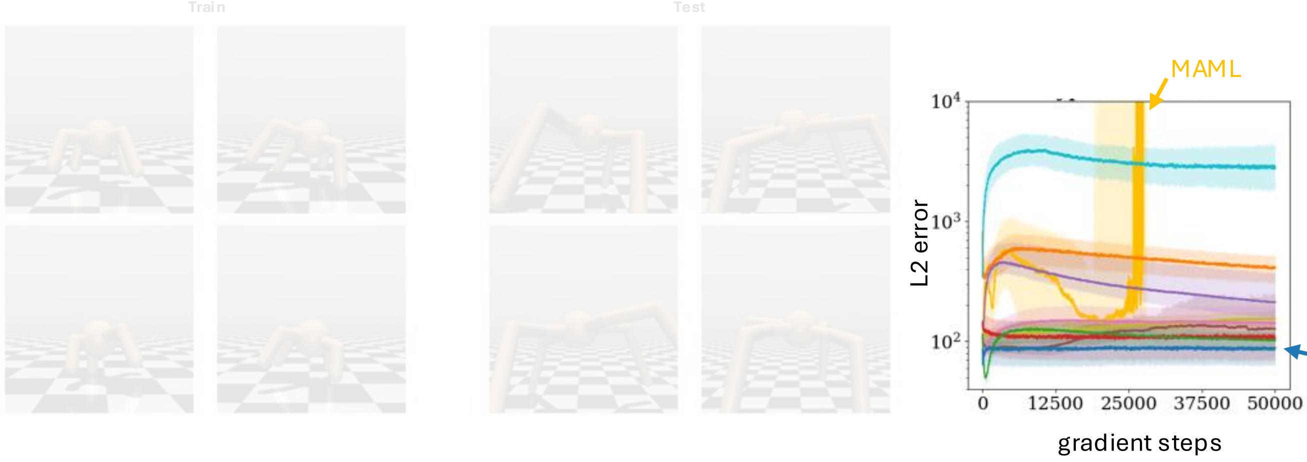
higher is better

Pose Estimation



lower is better

Dynamics Prediction



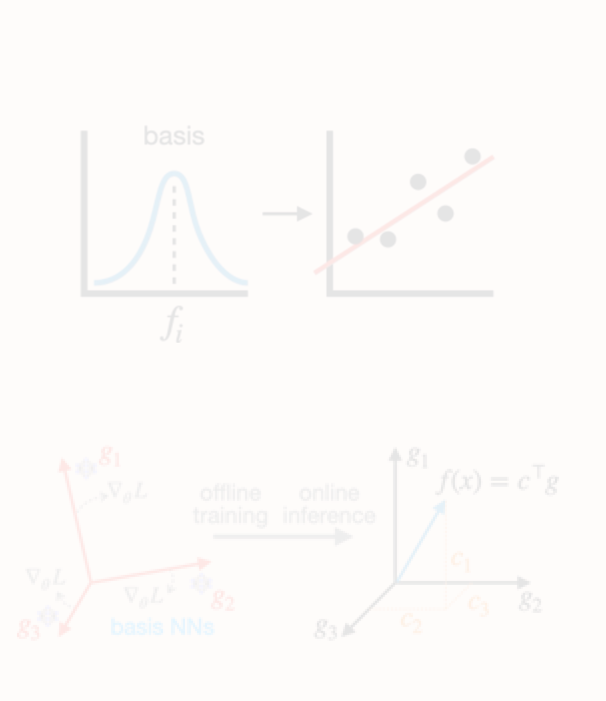
lower is better

function encoders

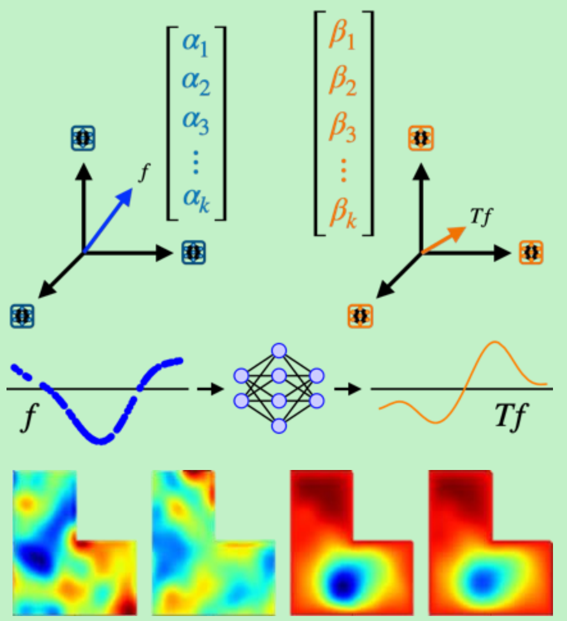
Ingebrand, T., Thorpe, A. J., & Topcu, U. (2025). Function Encoders: A Principled Approach to Transfer Learning in Hilbert Spaces. ICML

Basis-to-basis operator learning

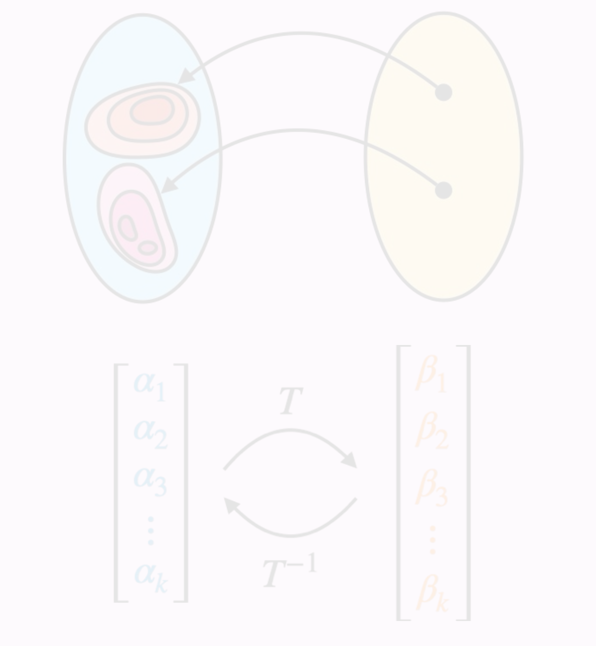
Function Encoders



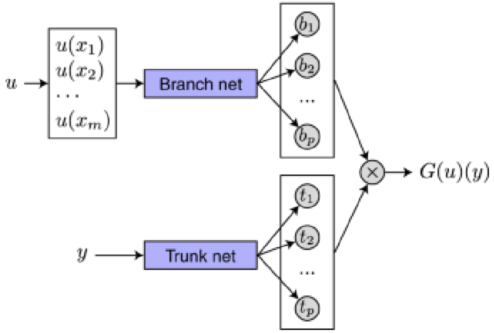
Basis-to-Basis Operator Learning

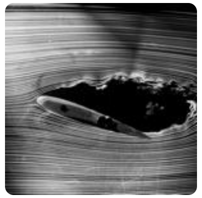


Inverse Neural Operators



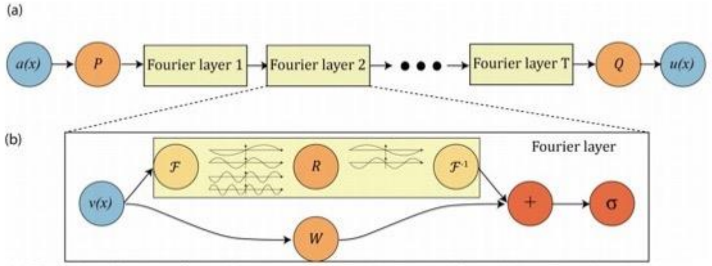
Neural operator learning: function to function maps

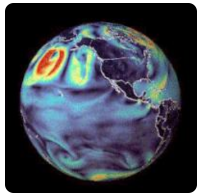




DeepONet

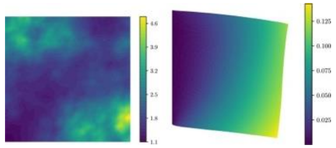
Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators.





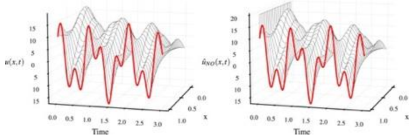
Fourier Neural Operators

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations.



DINO

O'Leary-Roseberry, T., Chen, P., Villa, U., & Ghattas, O. (2024). Derivative-informed neural operator: an efficient framework for high-dimensional parametric derivative learning

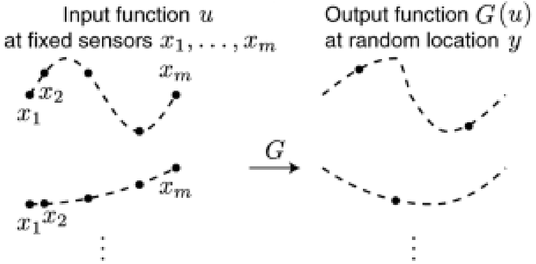


PDE Control

Bhan, L., Shi, Y., & Krstic, M. (2023). Neural operators for bypassing gain and control computations in PDE backstepping

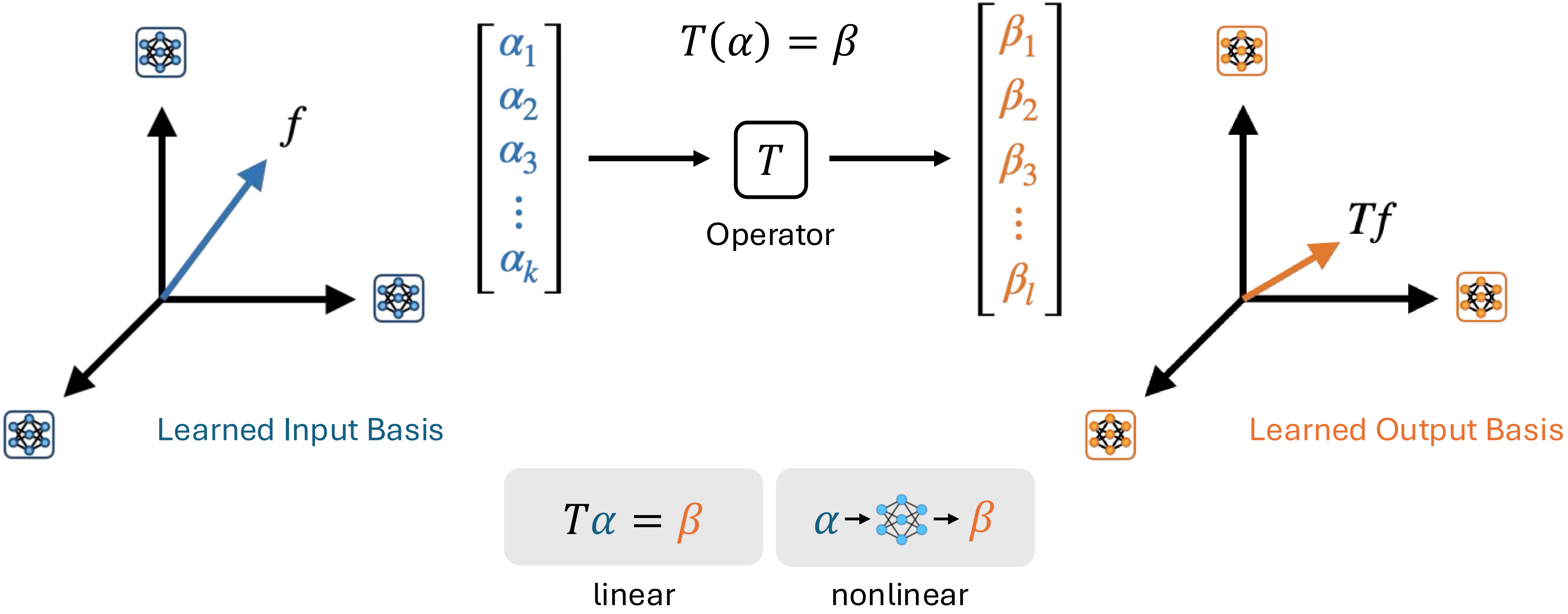
Main challenge:

DeepONet & FNO require the input data to be on a **fixed grid** or **mesh**



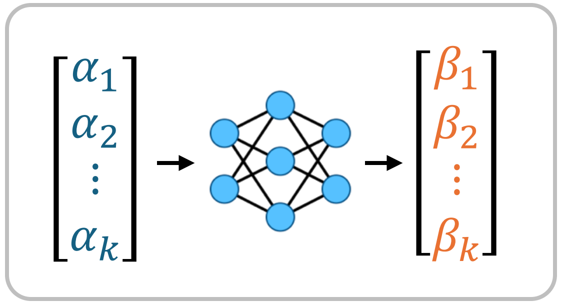
Basis-to-basis operator learning (B2B)

Given: input-output pairs of transformations (f, Tf)
Goal: approximate $T: \mathcal{F} \to \mathcal{H}$

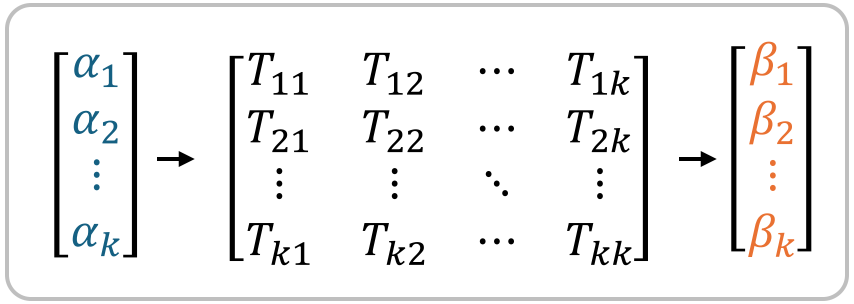

f
$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_k \end{bmatrix}$
$T(\alpha) = \beta$
T
Operator
$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_l \end{bmatrix}$
Tf
Learned Input Basis
Learned Output Basis
$T\alpha = \beta$
linear
$\alpha \rightarrow \rightarrow \beta$
nonlinear

Ingebrand, T., Thorpe, A. J., Goswami, S., Kumar, K., & Topcu, U. (2025). Basis-to-basis operator learning using function encoders. CMAME

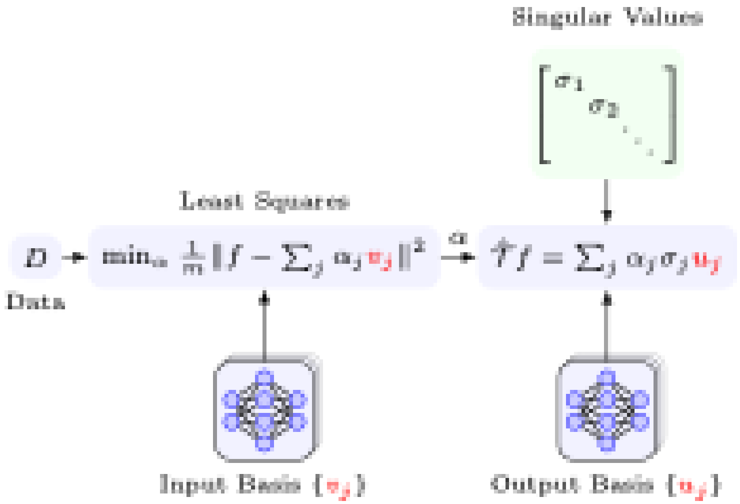
Basis-to-basis variants



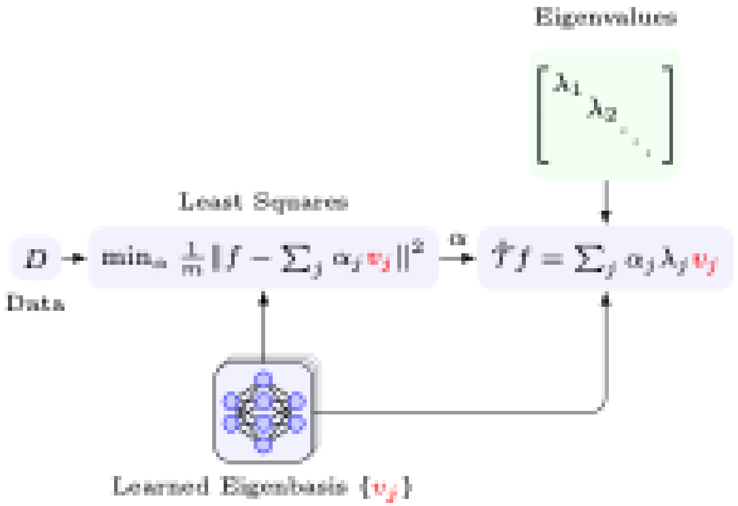
B2B (nonlinear)


$\begin{bmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_k\end{bmatrix} \to \begin{bmatrix}T_{11} & T_{12} & \cdots & T_{1k}\\ T_{21} & T_{22} & \cdots & T_{2k}\\ \vdots & \vdots & \ddots & \vdots\\ T_{k1} & T_{k2} & \cdots & T_{kk}\end{bmatrix} \to \begin{bmatrix}\beta_1\\ \beta_2\\ \vdots\\ \beta_k\end{bmatrix}$

B2B (linear)



Singular Value Decomposition


Eigenvalues
λ_1
λ_2
Least Squares
D
Data
$\min_\alpha \frac{1}{m}\|f - \sum_j \alpha_j v_j\|^2$
α
$\hat{T}f = \sum_j \alpha_j \lambda_j v_j$
Learned Eigenbasis $\{v_j\}$

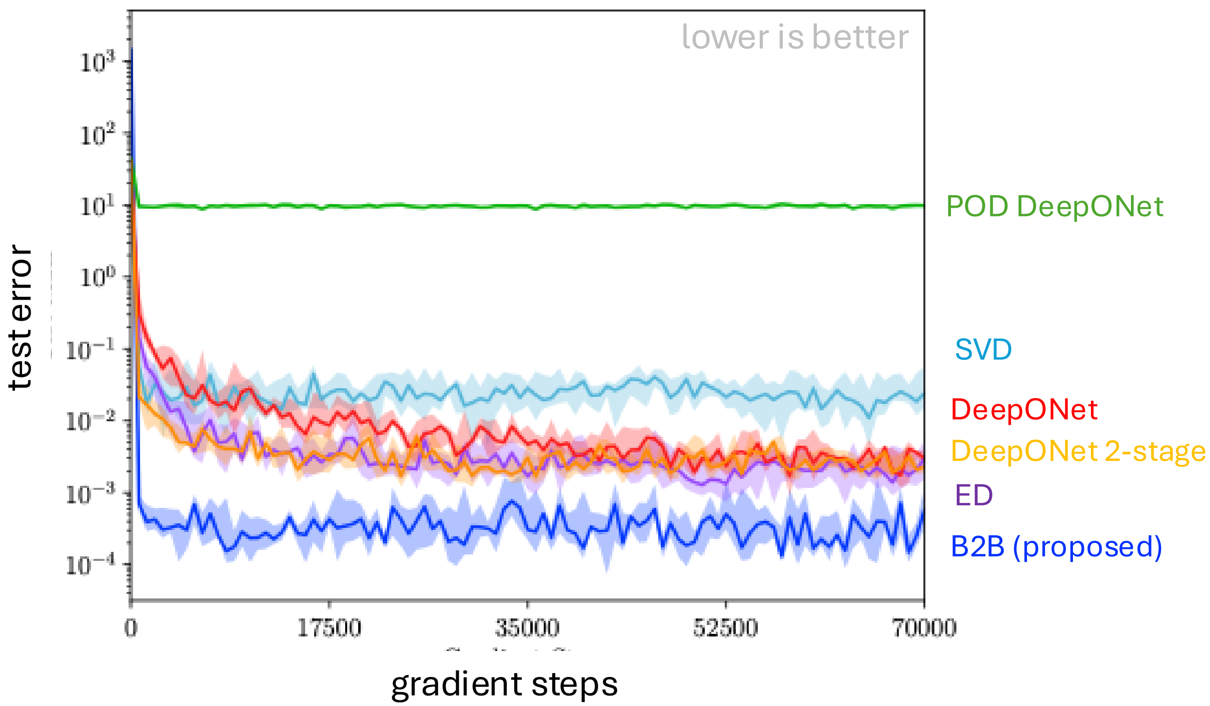
Eigen-decomposition

Ingebrand, T., Thorpe, A. J., Goswami, S., Kumar, K., & Topcu, U. (2025). Basis-to-basis operator learning using function encoders. CMAME

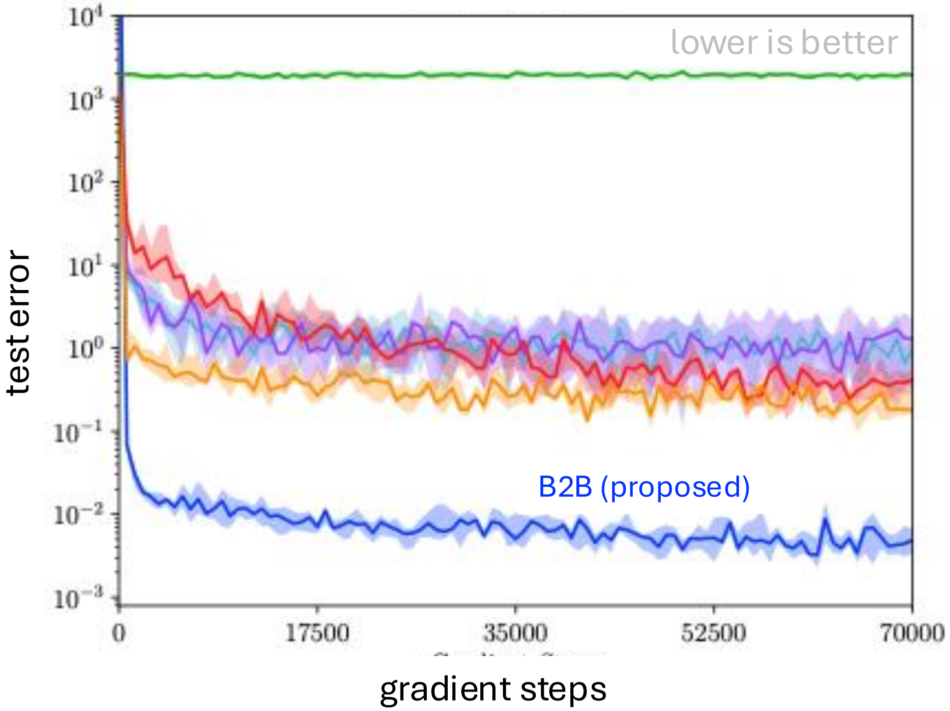
An illustrative linear example: derivative & antiderivative

$$\frac{ds(x)}{dx} = u(x), \qquad s(0) = 0, \qquad Tu(x) = s(x = 0) + \int_0^x u(t)\,dt$$

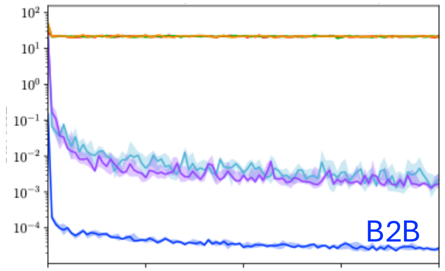
derivative operator



antiderivative operator



varying sensor locations:



- **extrapolates** to the linear span
- **maintains accuracy,** even when the measurement locations **change**

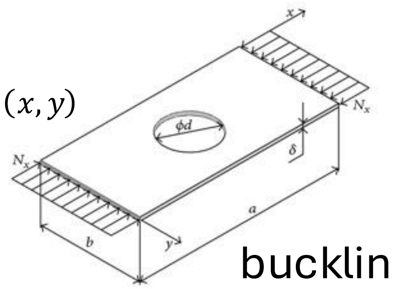
A nonlinear example of basis-to-basis for PDE modeling

Modeling the solution of partial differential equations

input function

↓

$$\nabla \cdot \sigma + f(\boldsymbol{x}) = 0, \boldsymbol{x} = (x, y)$$
$$(u, v) = 0, \forall x = 0$$



buckling

loading function

(input function)

displacement

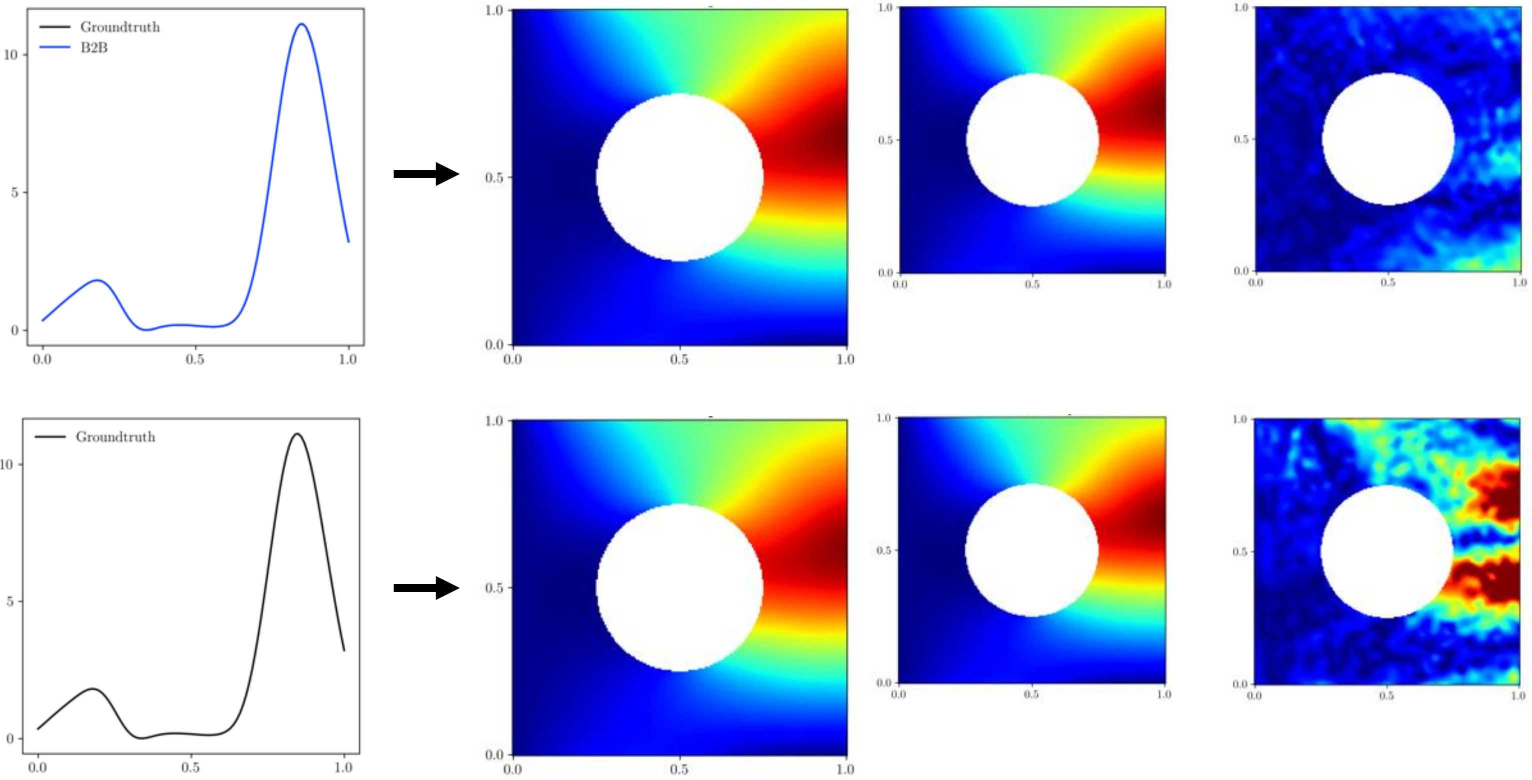
(output function)

ground truth

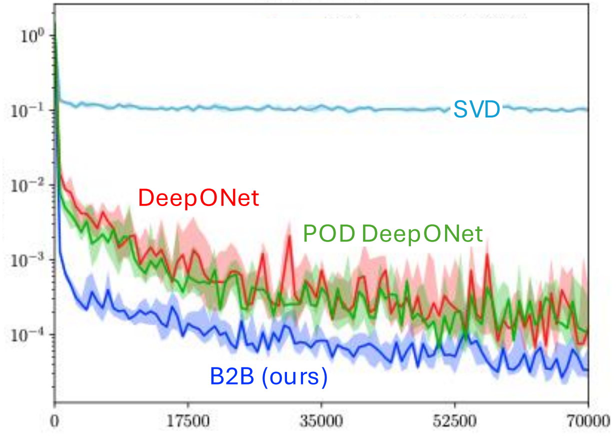
error

B2B

DeepONet



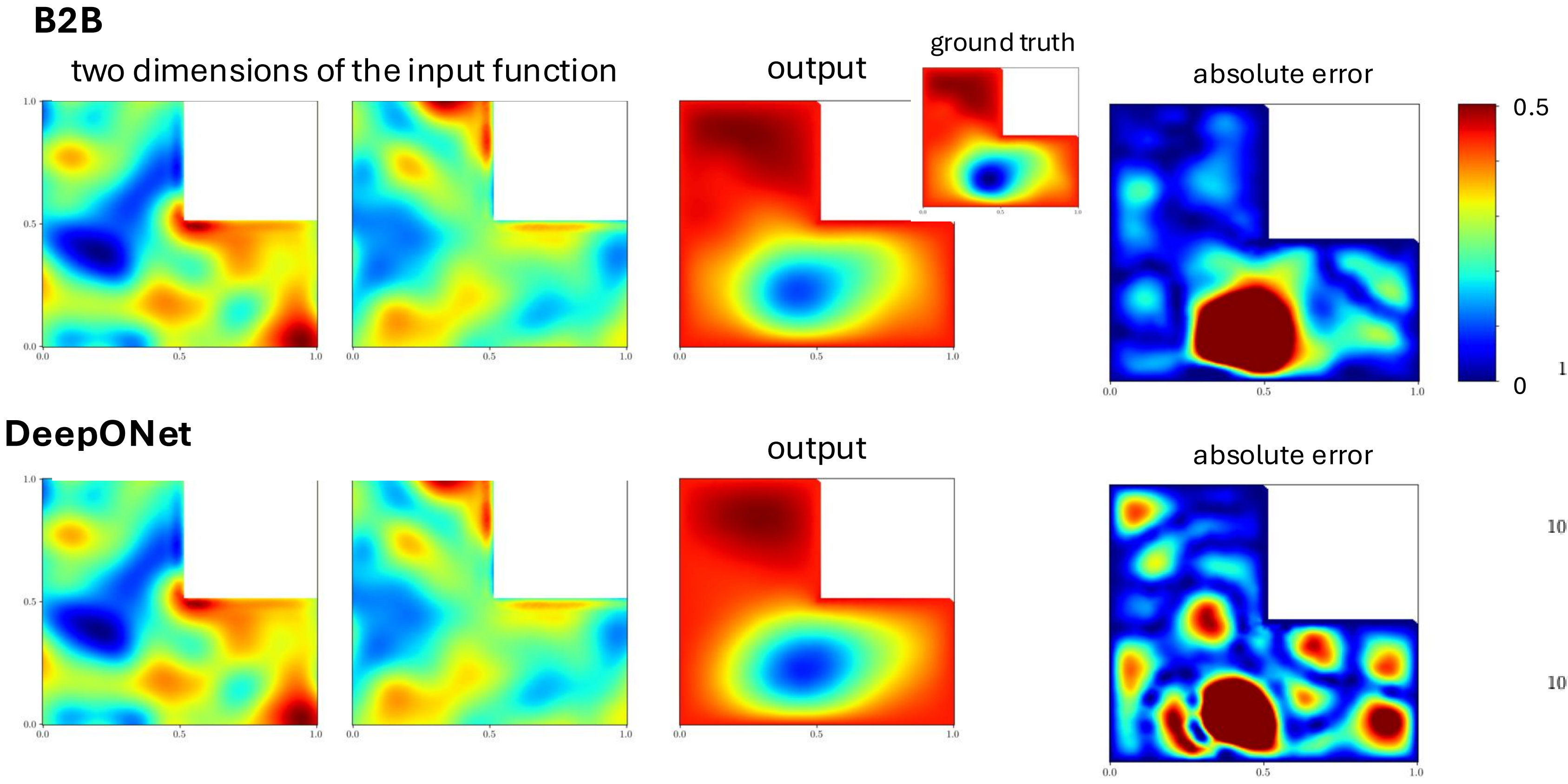
B2B has lower error, and **doesn't rely on a fixed grid or mesh**.



Neural operators model the **entire** solution, not just one instance!

L-shaped 2D Darcy flow

$$\nabla \cdot \big(k(x)\nabla u(x)\big) + f(x) = 0, \quad x = (x, y) \in \Omega \coloneqq (0,1)^2 \times [0.5,1)^2,$$
$$u(x) = 0, \quad x \in \partial\Omega$$



B2B

two dimensions of the input function

output

ground truth

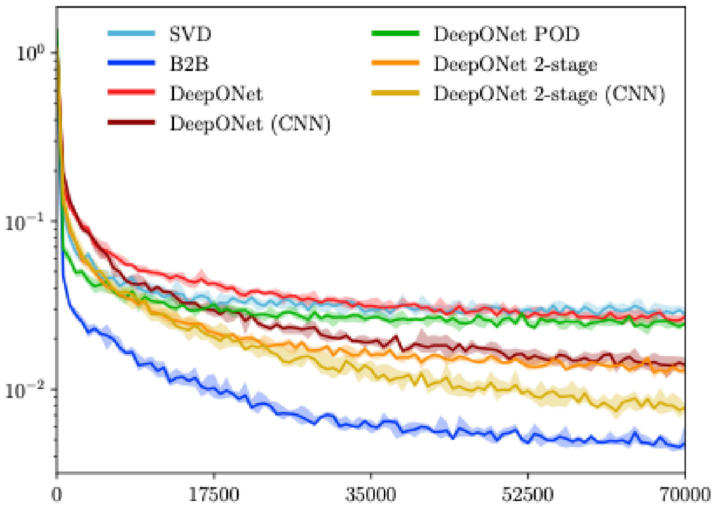
absolute error

DeepONet

output

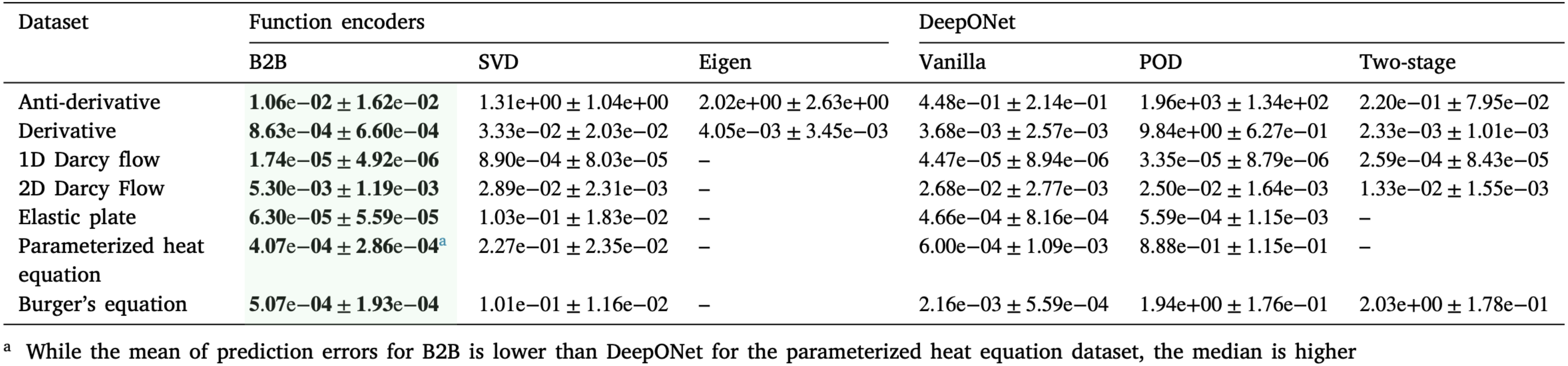
absolute error

B2B demonstrates better accuracy and lower variance



Quantitative results

our proposed approaches



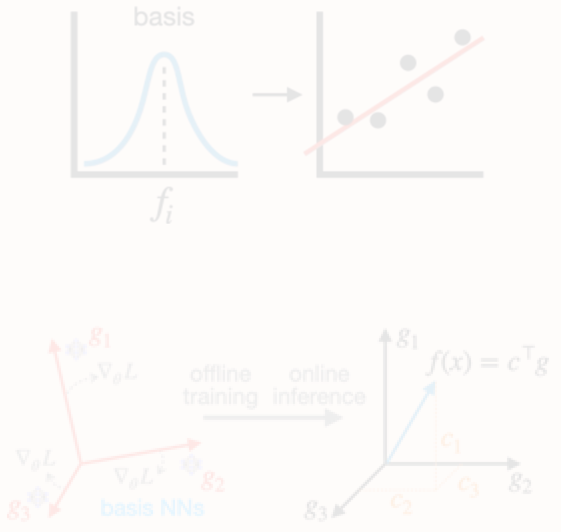
Dataset	Function encoders			DeepONet		
	B2B	SVD	Eigen	Vanilla	POD	Two-stage
Anti-derivative	**1.06e−02 ± 1.62e−02**	1.31e+00 ± 1.04e+00	2.02e+00 ± 2.63e+00	4.48e−01 ± 2.14e−01	1.96e+03 ± 1.34e+02	2.20e−01 ± 7.95e−02
Derivative	**8.63e−04 ± 6.60e−04**	3.33e−02 ± 2.03e−02	4.05e−03 ± 3.45e−03	3.68e−03 ± 2.57e−03	9.84e+00 ± 6.27e−01	2.33e−03 ± 1.01e−03
1D Darcy flow	**1.74e−05 ± 4.92e−06**	8.90e−04 ± 8.03e−05	–	4.47e−05 ± 8.94e−06	3.35e−05 ± 8.79e−06	2.59e−04 ± 8.43e−05
2D Darcy Flow	**5.30e−03 ± 1.19e−03**	2.89e−02 ± 2.31e−03	–	2.68e−02 ± 2.77e−03	2.50e−02 ± 1.64e−03	1.33e−02 ± 1.55e−03
Elastic plate	**6.30e−05 ± 5.59e−05**	1.03e−01 ± 1.83e−02	–	4.66e−04 ± 8.16e−04	5.59e−04 ± 1.15e−03	–
Parameterized heat equation	**4.07e−04 ± 2.86e−04**[a]	2.27e−01 ± 2.35e−02	–	6.00e−04 ± 1.09e−03	8.88e−01 ± 1.15e−01	–
Burger's equation	**5.07e−04 ± 1.93e−04**	1.01e−01 ± 1.16e−02	–	2.16e−03 ± 5.59e−04	1.94e+00 ± 1.76e−01	2.03e+00 ± 1.78e−01

[a] While the mean of prediction errors for B2B is lower than DeepONet for the parameterized heat equation dataset, the median is higher

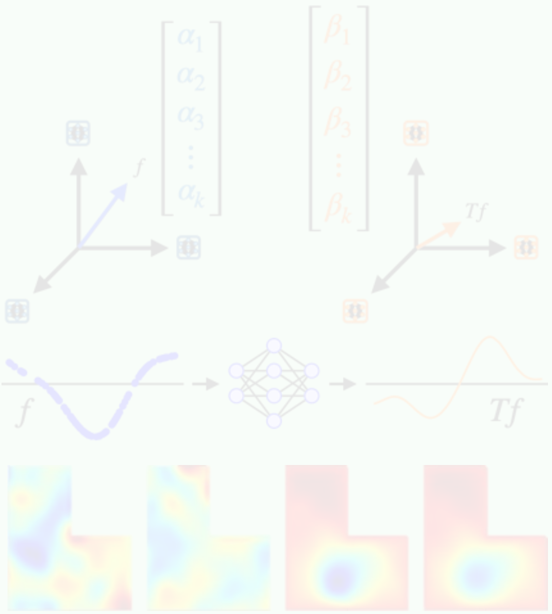
B2B outperforms DeepONet on several PDE benchmarks

Basis-to-basis operator learning

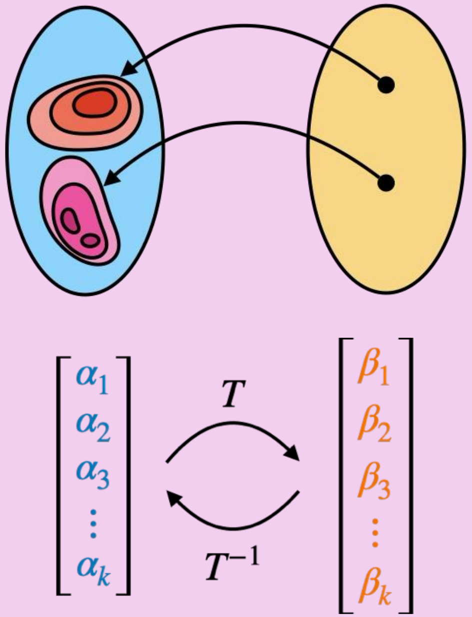
Function Encoders



Basis-to-Basis Operator Learning



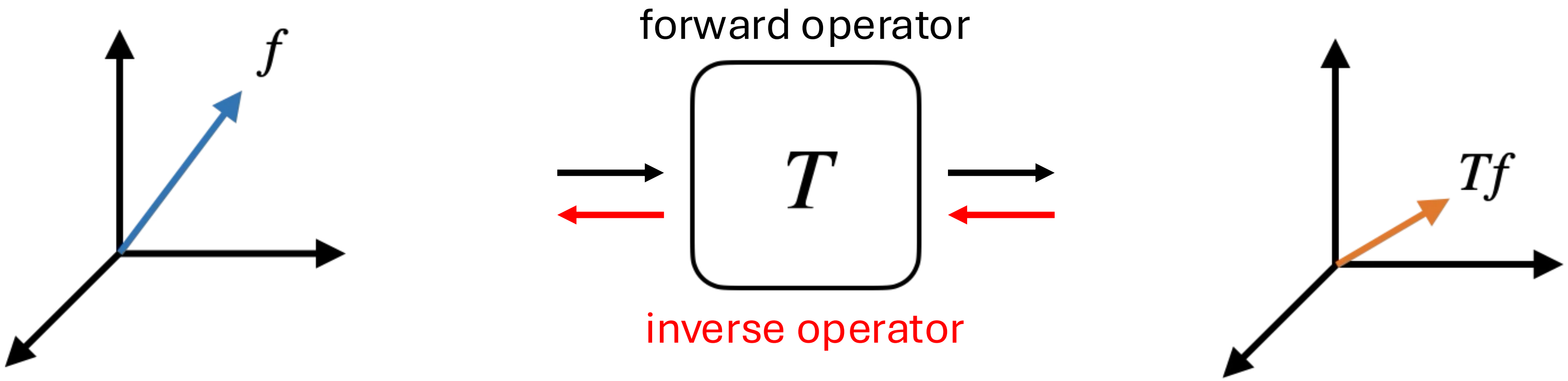
Inverse Neural Operators



Inverse neural operators

Given: input-output pairs of transformations (f, Tf)
Goal: approximate $T: \mathcal{F} \to \mathcal{H}$

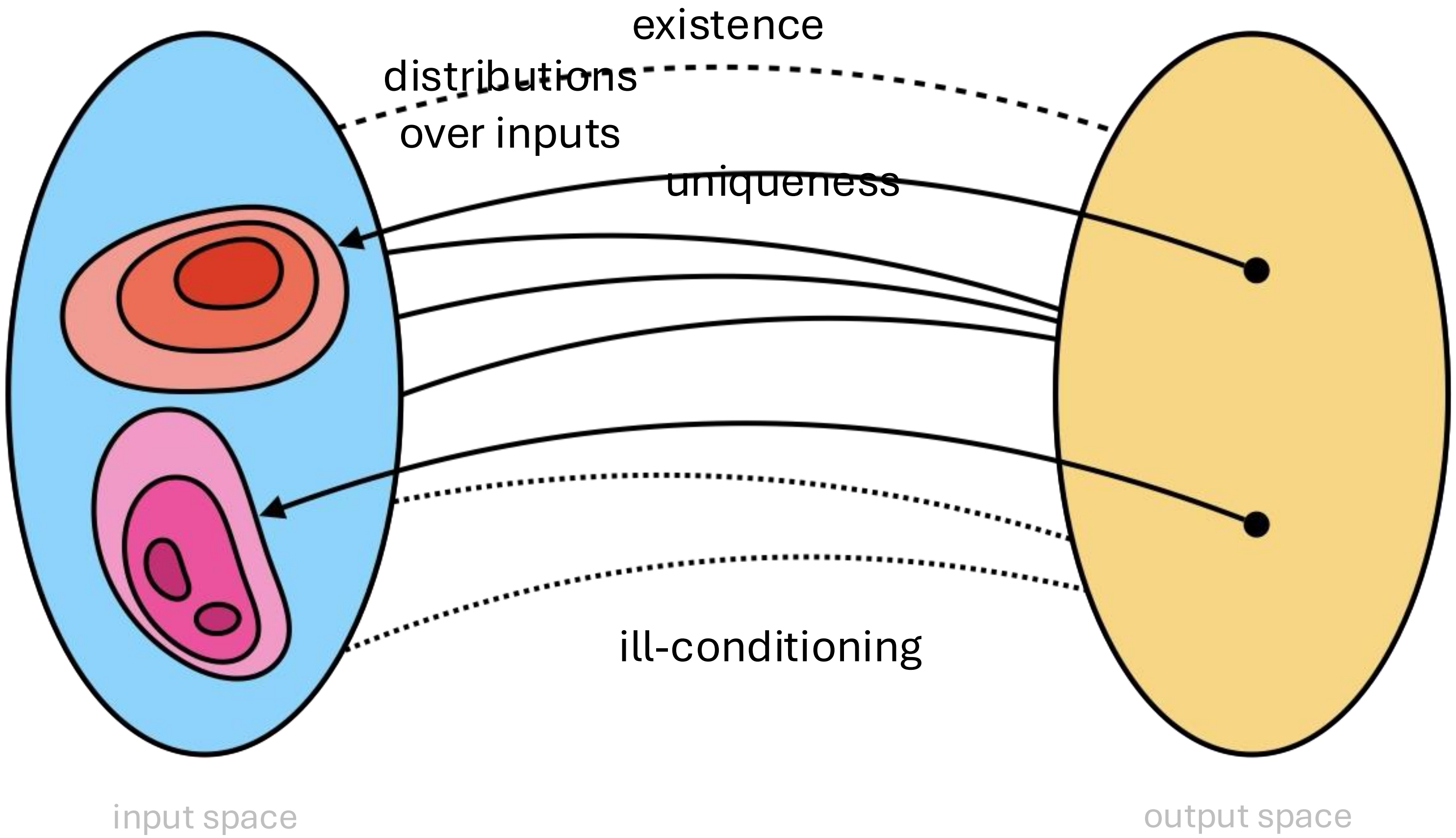
Inverse problem: given Tf, estimate f



so what's the problem?

Inverse maps are **ill-posed**

The probabilistic approach:

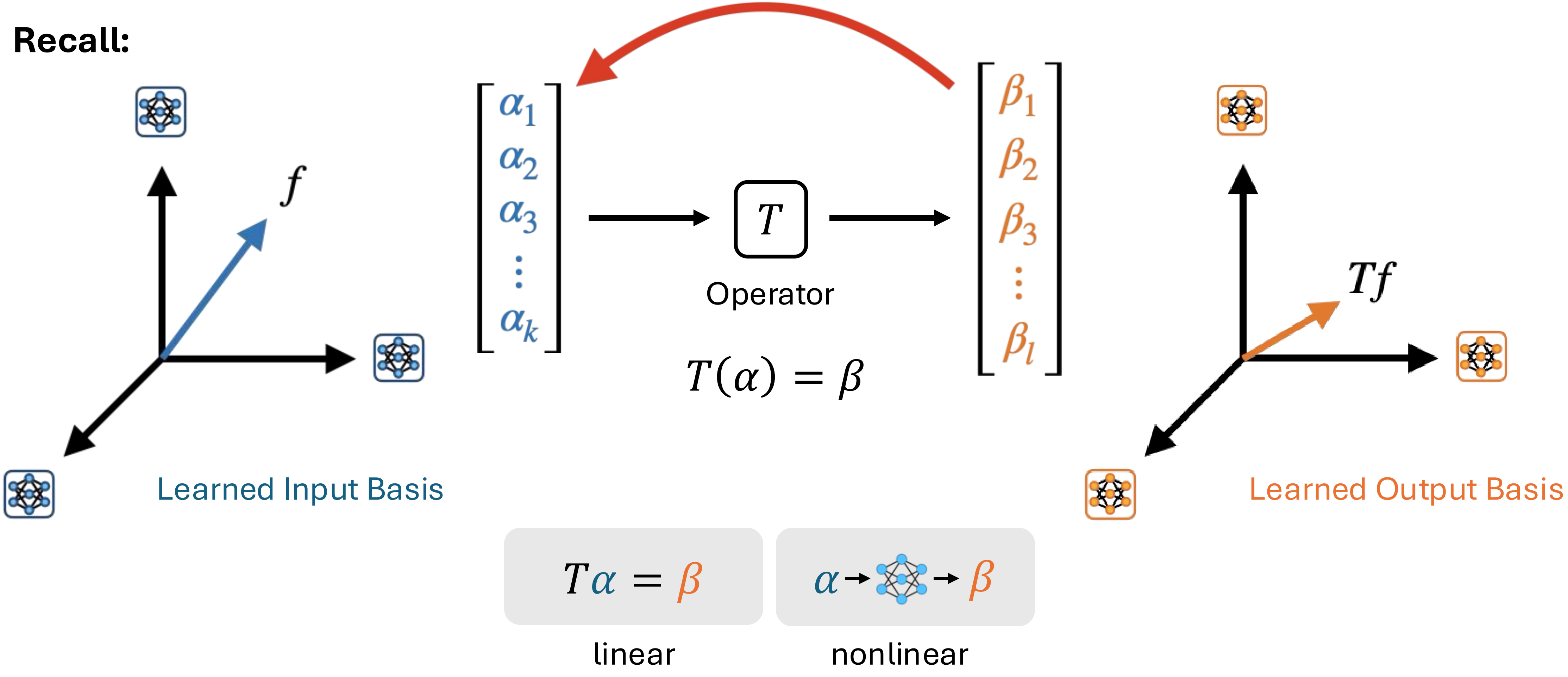


requires us to reason over probability distributions

Inverse neural operators (using B2B)

Problem: compute T^{-1} such that $\alpha \sim T^{-1}(\beta)$

Recall:



f

$$\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ \vdots\\ \alpha_k\end{bmatrix} \longrightarrow \boxed{T} \longrightarrow \begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\\ \vdots\\ \beta_l\end{bmatrix}$$

Operator

$$T(\alpha) = \beta$$

Tf

Learned Input Basis

Learned Output Basis

$T\alpha = \beta$

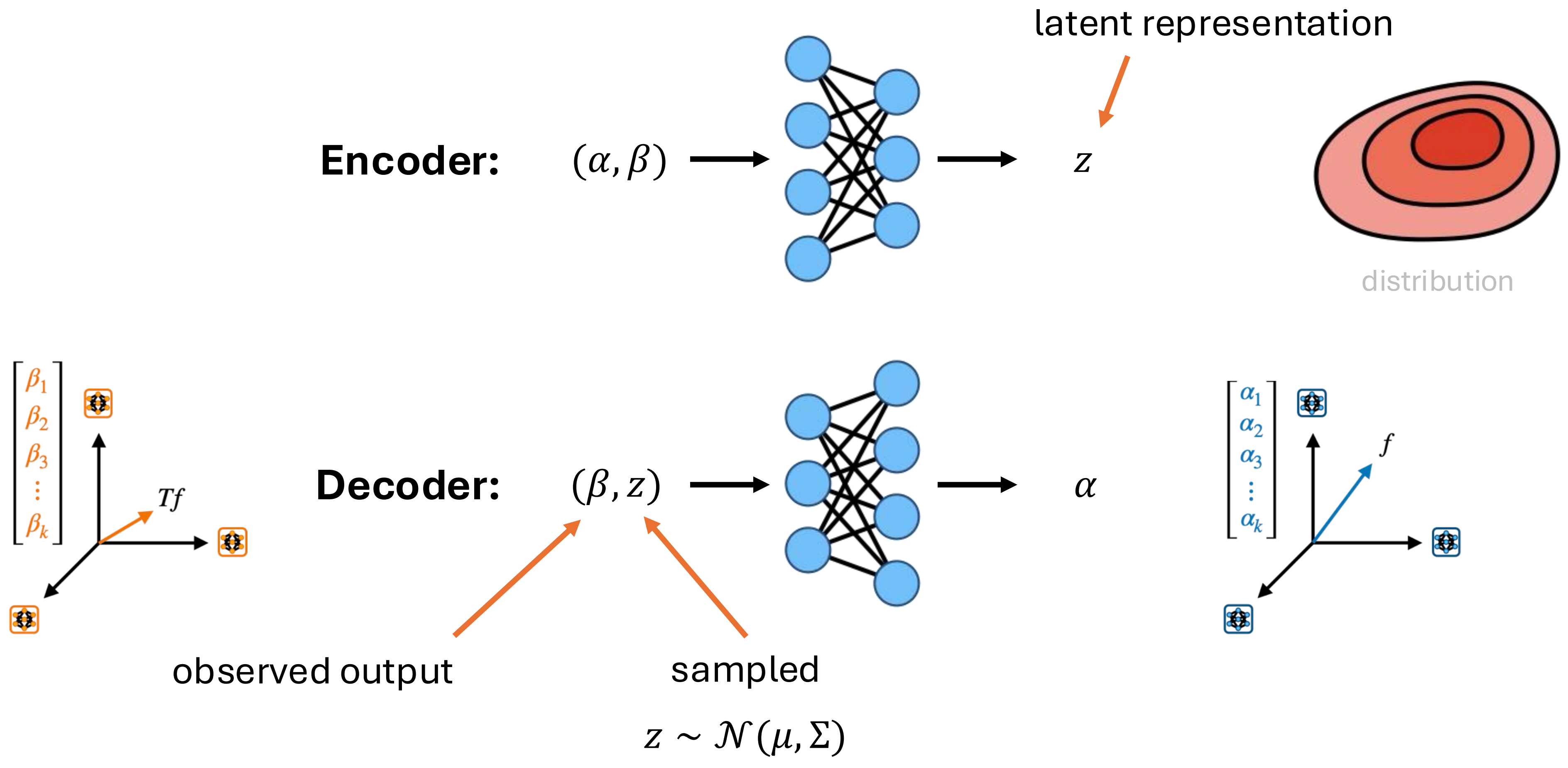
linear

$\alpha \rightarrow$ $\rightarrow \beta$

nonlinear

Invertible networks

Using **conditional variational autoencoders** to model the inverse map



latent representation

Encoder: $(\alpha, \beta) \longrightarrow$ $\longrightarrow z$

Decoder: $(\beta, z) \longrightarrow$ $\longrightarrow \alpha$

observed output

sampled

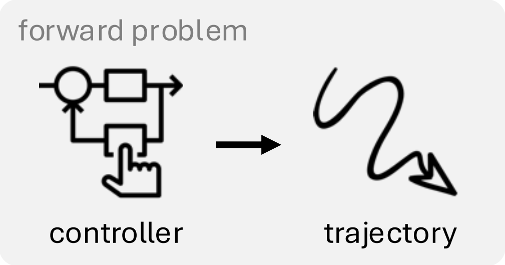
$z \sim \mathcal{N}(\mu, \Sigma)$

distribution
$\begin{bmatrix}\beta_1\\\beta_2\\\beta_3\\\vdots\\\beta_k\end{bmatrix}$
Tf
$\begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\\\vdots\\\alpha_k\end{bmatrix}$
f

Operator learning & inverse problems

Neural operator learning represents a **new frontier in learning and autonomy**

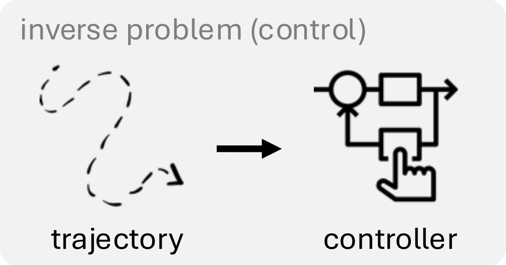
We can't focus on solving single instances, we need **global** solutions for adaptation & transfer



forward problem

controller

trajectory



inverse problem (control)

trajectory

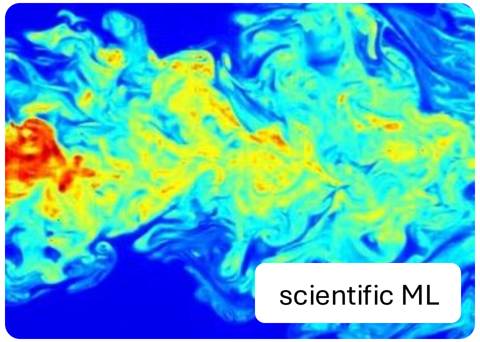
controller

Bellman operator: $Tf(x,u) = R(x,u) + \gamma\langle \mathbb{P}(\cdot \mid x,u), V_f \rangle$

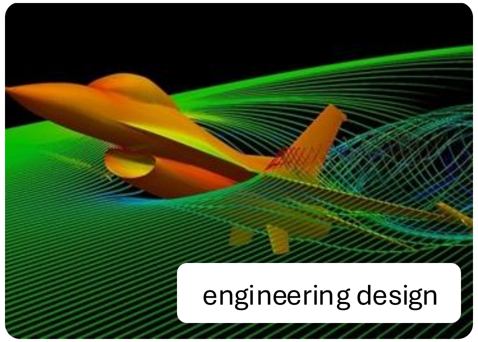
basis-to-basis inverse operator learning

$$\begin{tikzcd} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_k \end{bmatrix} \arrow[r, shift left, "Tf"] & \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_k \end{bmatrix} \arrow[l, shift left, "T^{-1}f"] \end{tikzcd}$$

"inferring causes from effects"



scientific ML



engineering design

modeling complex systems

Takeaways

- We provide a novel operator learning approach that combines **basis learning** with **neural operator learning**
- Avoids a key limitation, which is the need for a **fixed grid or mesh**
- **Outperforms** existing approaches on PDE benchmarks

Questions?

adam.thorpe@austin.utexas.edu