ARGONNE
ATPESC2023
EXTREME-SCALE COMPUTING

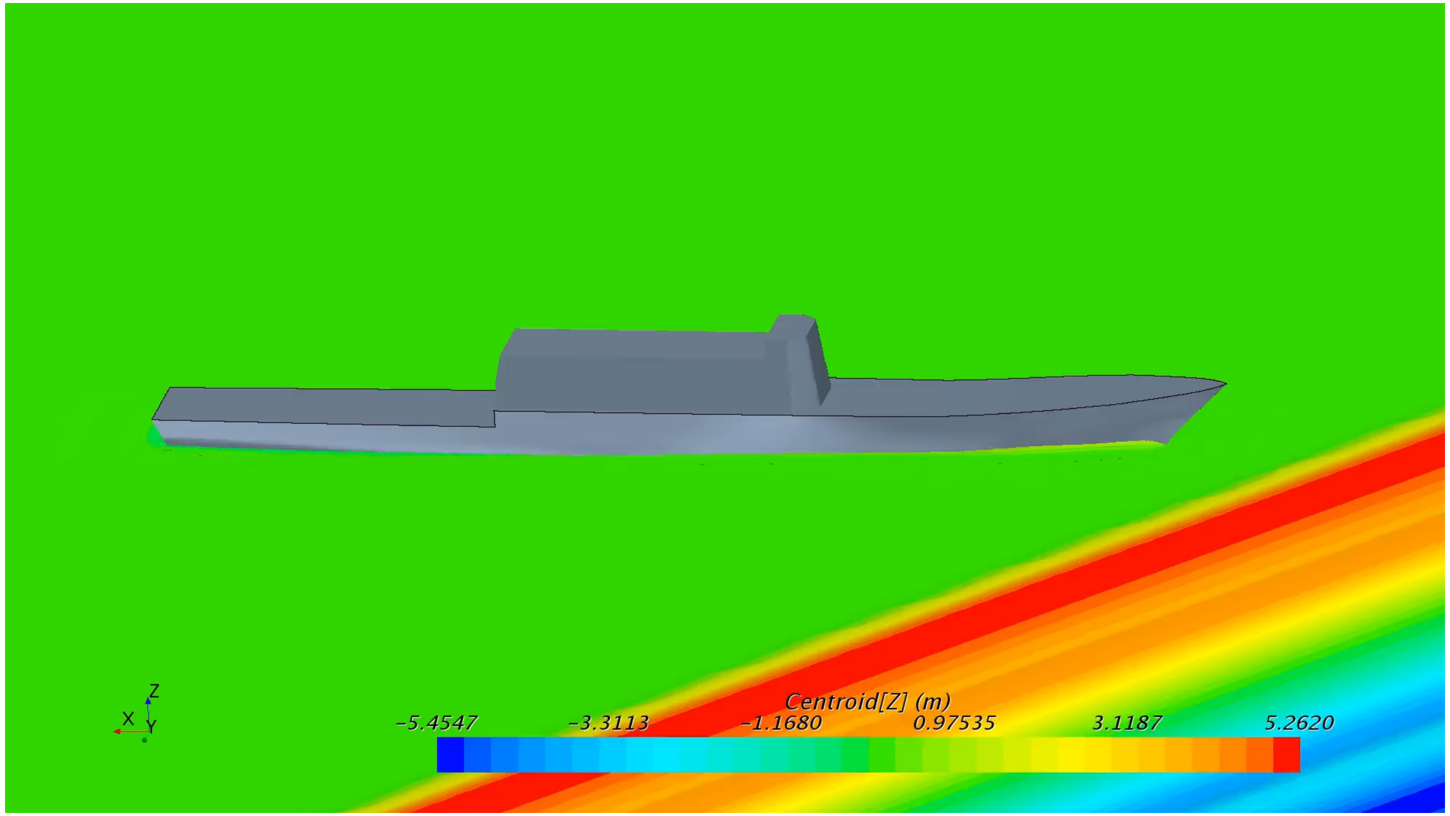
Transfer and Multi-Task Learning in Physics-Based Applications with Deep Neural Operators

Somdatta Goswami
Assistant Professor (Research)
Applied Mathematics, Brown University

CRUNCH Group

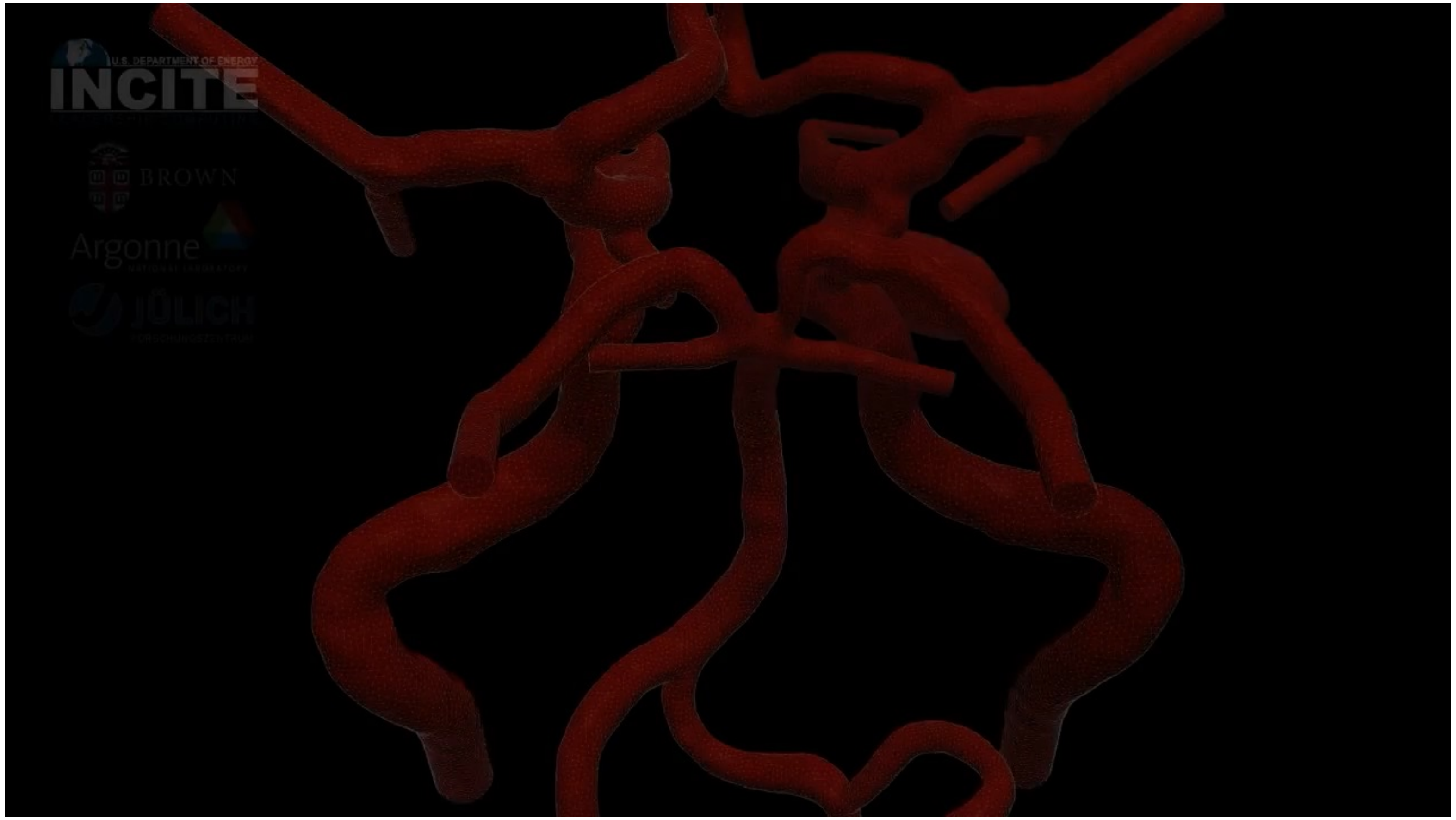
ECP EXASCALE COMPUTING PROJECT
Argonne NATIONAL LABORATORY

Can Neural Networks Predict in Real-Time (Autonomy)?



Courtesy: MIT, Michael Triantafyllou

Can we predict the rupture of aneurysm?

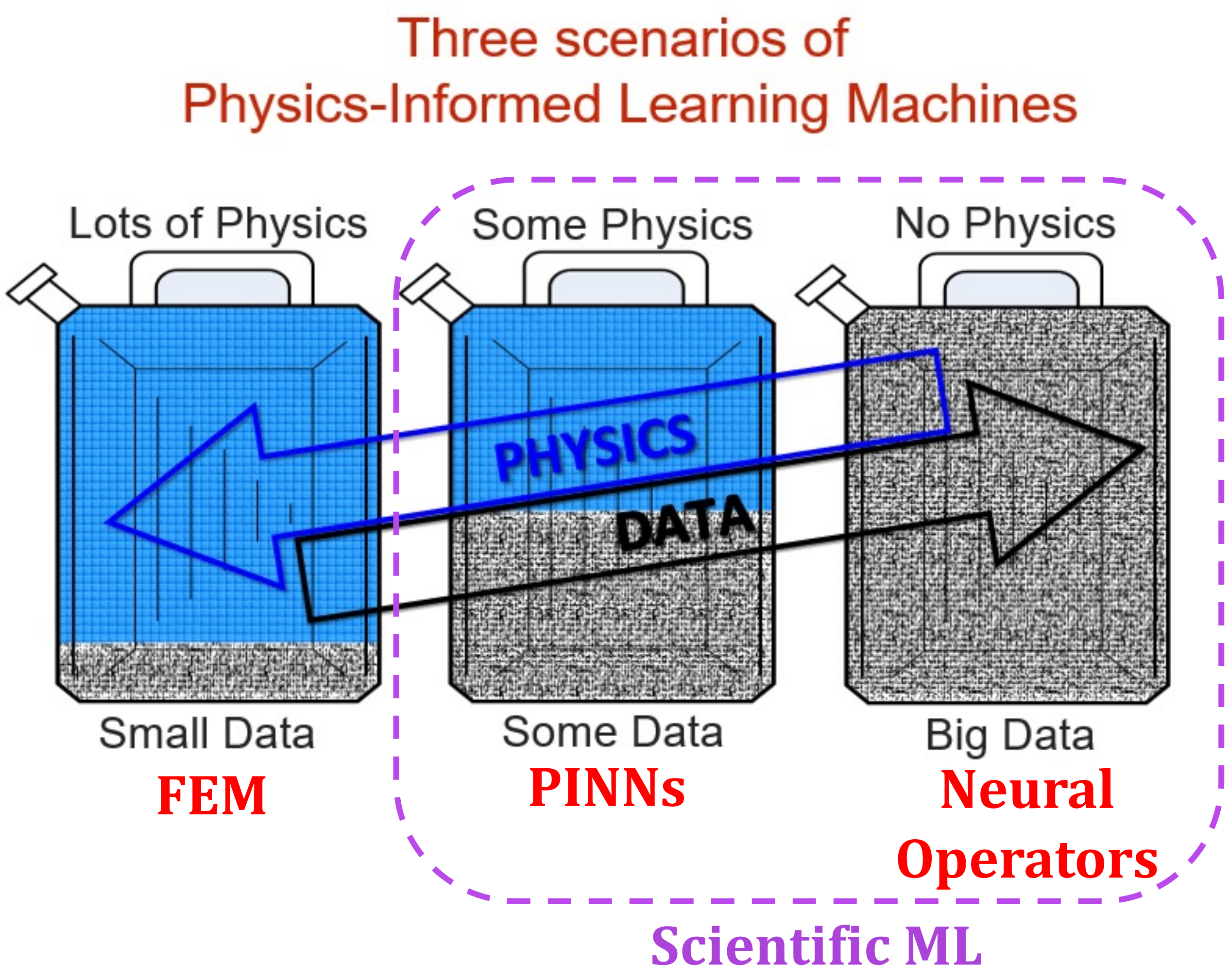


Can we predict the rupture of aneurysm?

Courtesy: Boston Children's Hospital, J. Marsden

Data + Laws of Physics

The 5D Law: Dinky, Dirty, Dynamic, Deceptive Data



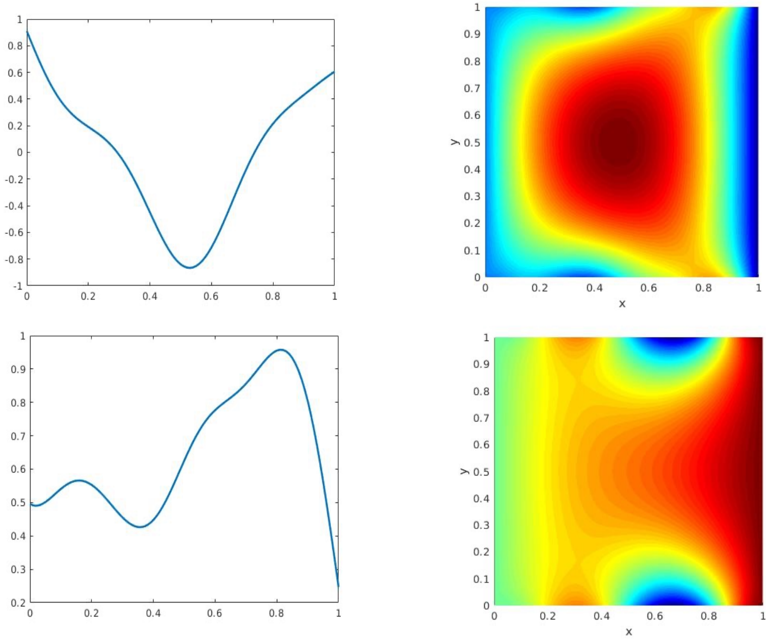
Motivation

$$k\left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2}\right) = f(x,y)$$

Domain

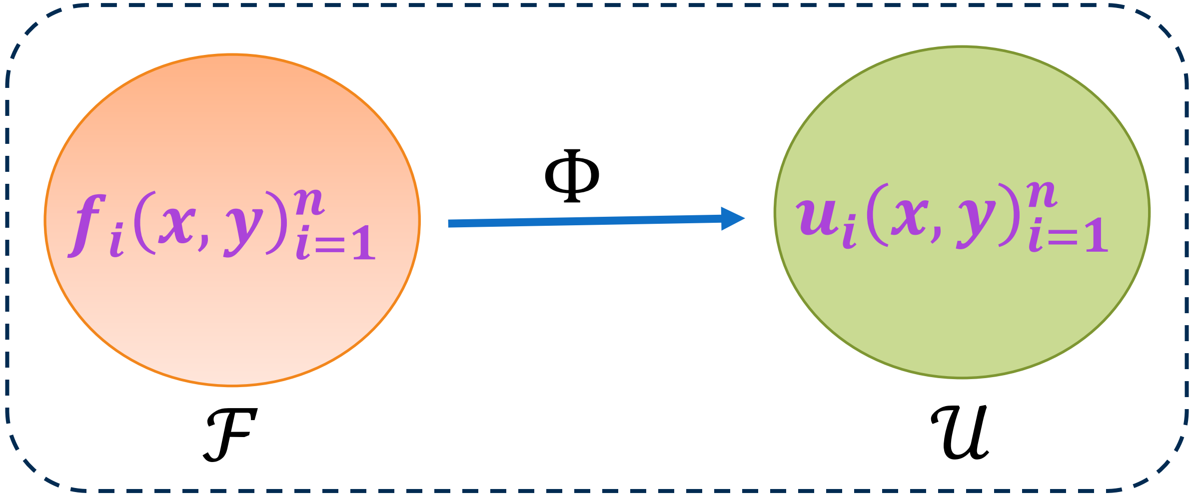
$$\Omega = [0,1]\times[0,1]$$

$\boldsymbol{f(x,y)} \rightarrow \boldsymbol{u(x,y)}$



To Generalize: Need an approach to learn to predict the solution for unseen $f(x,y)$.

Aim



Operator learning

Input-output map $\Phi\colon \mathcal{F} \to \mathcal{U}$ $\mathcal{F}, \mathcal{U}$ are infinite dimensional function space

Data $\{\mathcal{F}_i, \mathcal{U}_i\}_{i=1}^{n}$ $\mathcal{U}_i = \Phi(\mathcal{F}_i)$, $\mathcal{F}_i \sim \mu$ $i.i.d$

Operator learning $\Psi\colon \mathcal{F} \times \Theta \to \mathcal{U}$ such that $\Psi(.,\theta^*) \approx \Phi$

Training $\theta^* = \mathrm{argmin}_\theta\, l(\{\mathcal{F}_i, \Psi(\mathcal{U}_i, \theta)\})$

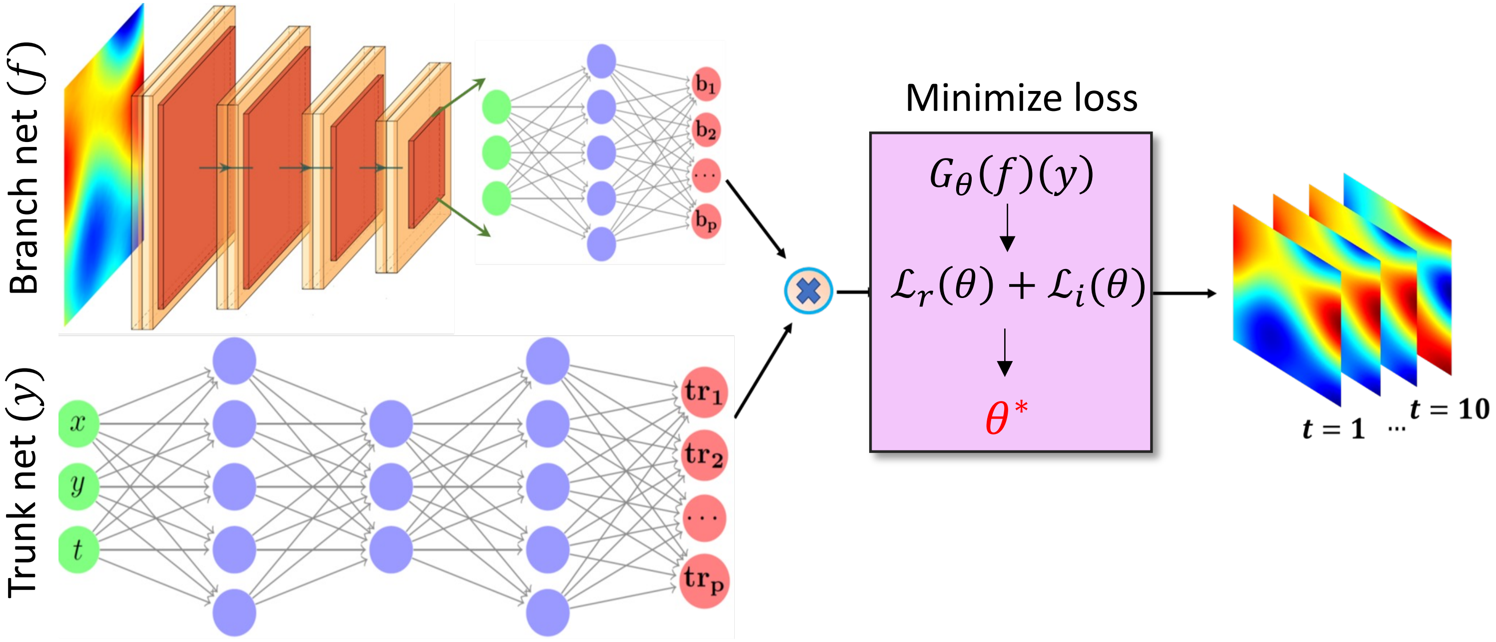
Universal Approximation Theorem for Operators

$$\mathcal{G}\colon f \to \mathcal{G}(f), \mathcal{G}(f)\colon y \in \mathbb{R}^d \to \mathbb{R}$$

Deep Operator Network (DeepONet)

- Generalized Universal Approximation Theorem for Operator [Chen '95, Lu et al. '19]
- **Branch net**: Input $\{f(x_i)\}_{i=1}^m$, output: $[b_1, b_2, .., b_p]^T \in \mathbb{R}^p$
- **Trunk net**: Input y, output: $[t_1, t_2, .., t_p]^T \in \mathbb{R}^p$
- Input f is evaluated at locations $\{y_i\}_{i=1}^m$

$$\hat{u} = G_\theta(f)(y) = \sum_{i=1}^{p} \underbrace{b_i(f(x_1), f(x_2), \ldots, f(x_m))}_{\text{branch net}} \cdot \underbrace{tr_i(y)}_{\text{trunk net}}$$



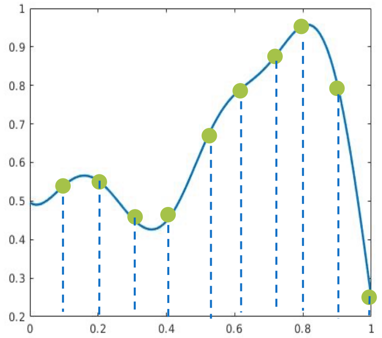
Constructing the DeepONet model

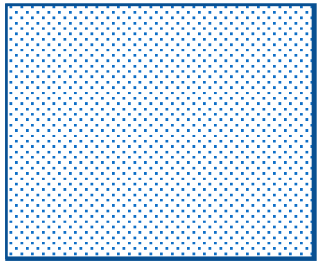
$$k\left(\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}\right)=f(x,y)$$

$$\mathcal{G}_\theta : f(x,y) \rightarrow u(x,y)$$

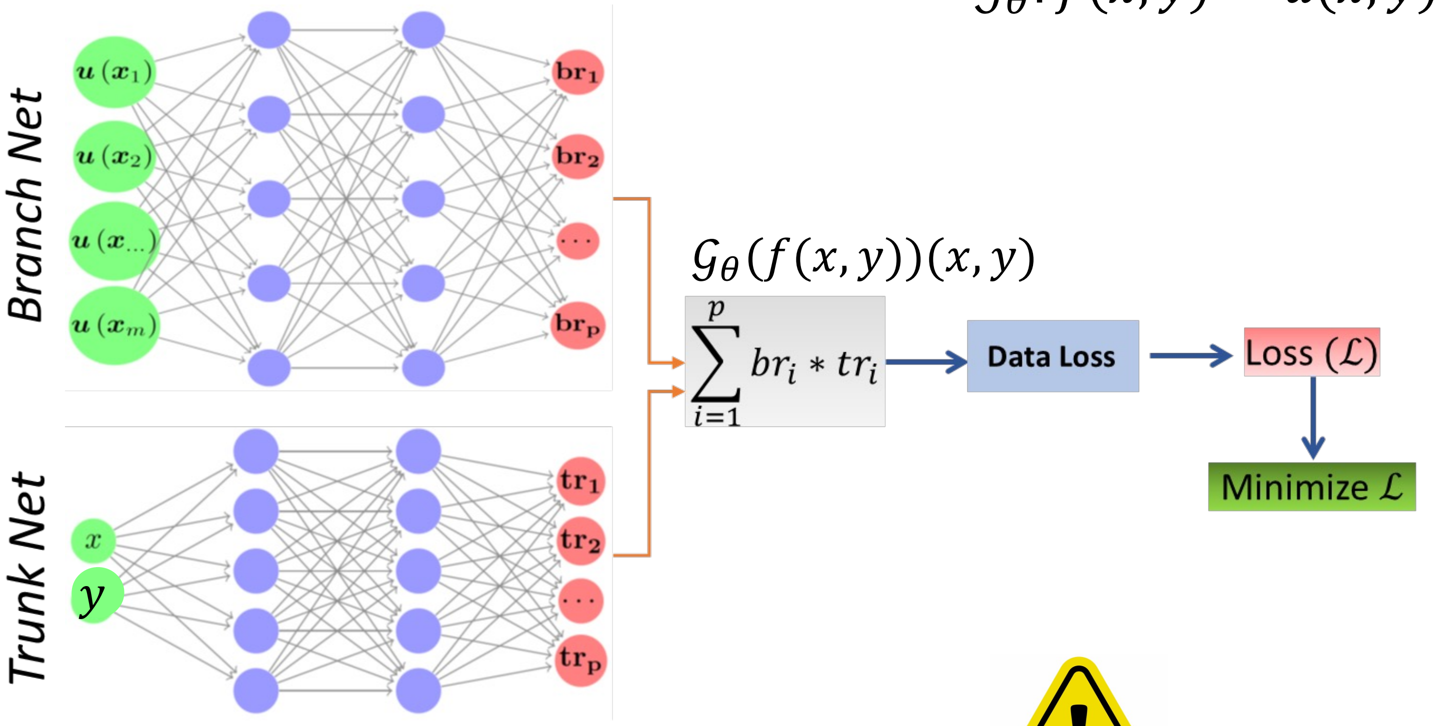
Testing Boundary conditions

$f(x,y)$





Query Points during testing



DeepONet is data hungry

Sensors $(m) = 10$

Transfer learning

- Transfer learning (TL) allows us to learn from a source data distribution a well performing model on a different but related target data distribution
- TL addresses the expense of data acquisition and labelling, potential computational power limitations and dataset distribution mismatches
- Problem setup:

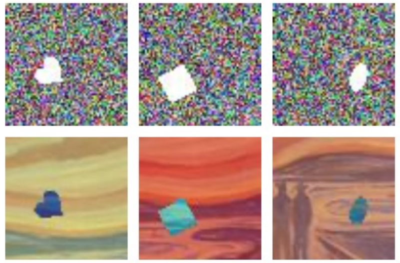
$\mathcal{D}^{\mathcal{S}} = \{(x_i{}^{\mathcal{S}}, y_i{}^{\mathcal{S}})\}$ labeled data sampled from P

$\mathcal{D}^{\mathcal{T}} = \{(x_i{}^{\mathcal{T}})\}$ (un)labeled data samples from Q

$P \neq Q$

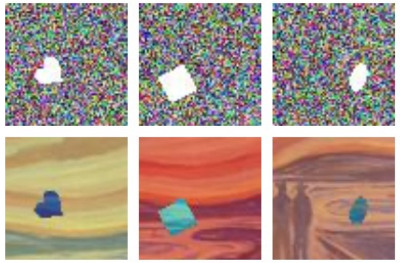
$\mathcal{S}$: source domain, $\mathcal{T}$: target domain

Covariate vs conditional shift



$$\neq \quad u_t = \left(\frac{u^2}{2}\right)_x + \nu u_{xx}.$$

Computer vision problems:



$\mathcal{D}^{\mathcal{S}} = \{(x_i{}^{\mathcal{S}}, y_i{}^{\mathcal{S}})\}$ sufficient labeled data

$\mathcal{D}^{\mathcal{T}} = \{(x_i{}^{\mathcal{T}})\}$ **unlabeled** data

$P(x_s) \neq P(x_t)$
$P(y_s|x_s) = P(y_t|x_t)$

"**covariate** shift"

Nonlinear PDE problems: $u_t = \left(\frac{u^2}{2}\right)_x + \nu u_{xx}.$

$\mathcal{D}^{\mathcal{S}} = \{(x_i^{\mathcal{S}}, y_i{}^{\mathcal{S}})\}$ sufficient labeled data

$\mathcal{D}^{\mathcal{T}} = \{(x_i^{\mathcal{T}}, y_i{}^{\mathcal{T}})\}$ few **labeled** data

$P(x_s) = P(x_t)$
$P(y_s|x_s) \neq P(y_t|x_t)$

"**conditional** shift"

$\mathcal{S}$: source domain, $\mathcal{T}$: target domain

Learning the solution of PDE on multiple domains

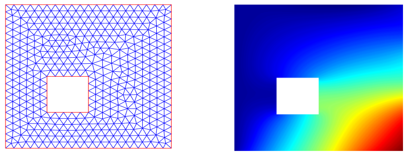
$\mathcal{X} = \mathcal{X}(\Omega; \mathbb{R}^{d_x}), \mathcal{Y} = \mathcal{Y}(\Omega; \mathbb{R}^{d_x})$

Nonlinear operator $\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y}$

Neural operator $\mathcal{G}_\theta : \mathcal{X} \rightarrow \mathcal{Y}, \quad \theta \in \Theta$

Training data $\{\mathbf{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$

$\mathcal{S}$: source domain

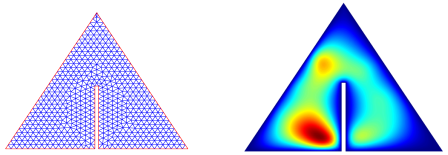


sufficient labeled data

$\mathcal{D}^{\mathcal{S}} = \{(x_i{}^{\mathcal{S}}, y_i{}^{\mathcal{S}})\}_{i=1}^{N_s}$

$N_s \gg N_t$

$\mathcal{T}$: target domain



few labeled data

$\mathcal{D}^{\mathcal{T}} = \{(x_i{}^{\mathcal{T}}, y_i{}^{\mathcal{T}})\}_{i=1}^{N_t}$

Learn surrogate 1 → Knowledge → Learn surrogate 2

Conditional shift: $P(x_s) = P(x_t)$, $P(y_s|x_s) \neq P(y_t|x_t)$

*x_s: Model inputs; y_s: Model outputs

- Learning surrogate models in *isolation* is expensive
- Training a surrogate with very few data can lead to overfitting
- Networks used for similar tasks (datasets) should be similar
- Leverage learned information between source and target models
  - ✓ Remove the need for big data for every new problem
  - ✓ Accelerate learning by fast fine-tuning of target network

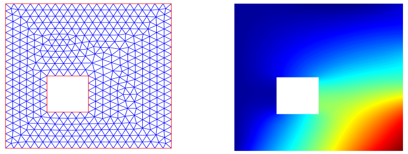
Transfer learning approach

Nonlinear operator $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$

Neural operator $\mathcal{G}_\theta : \mathcal{X} \to \mathcal{Y}, \ \theta \in \Theta$

Training data $\{\mathbf{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$

$\mathcal{S}$: source domain

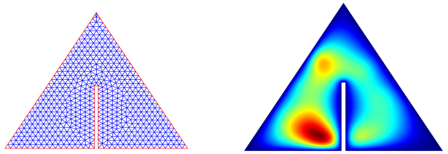


sufficient labeled data

$\mathcal{D}^{\mathcal{S}} = \{(x_i{}^{\mathcal{S}}, y_i{}^{\mathcal{S}})\}_{i=1}^{N_s}$

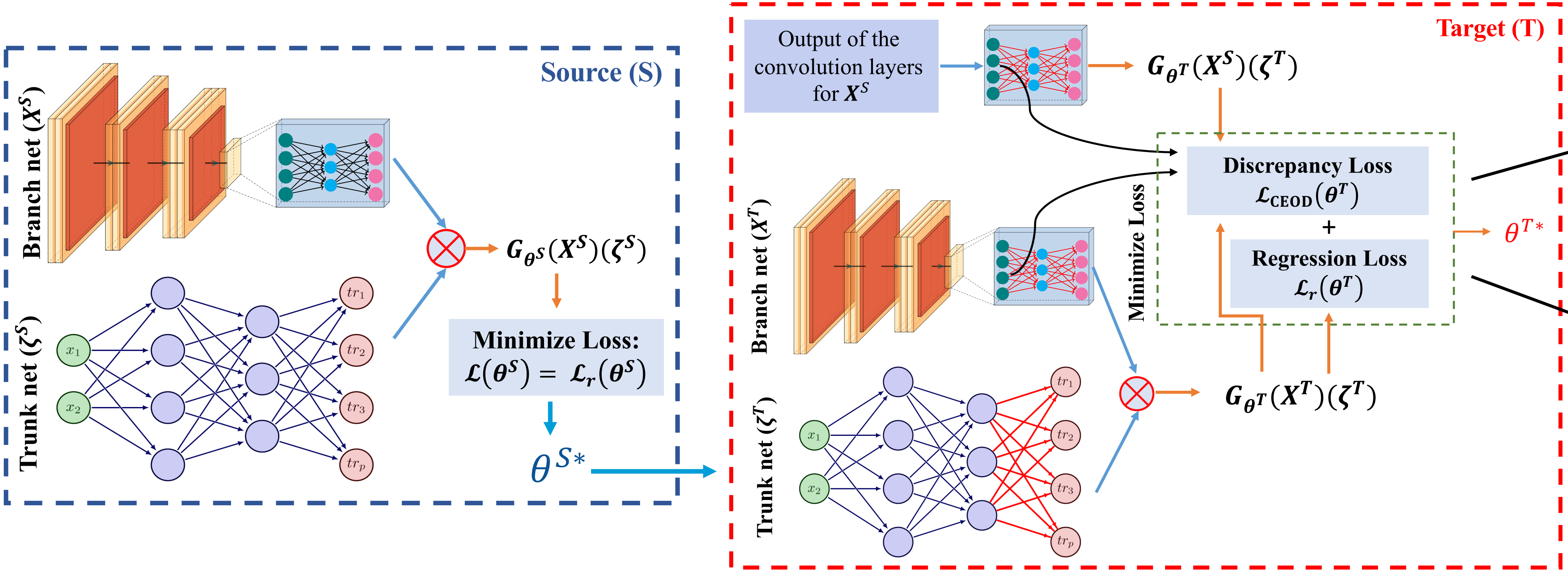
$N_s \gg N_t$

$\mathcal{T}$: target domain



few labeled data

$\mathcal{D}^{\mathcal{T}} = \{(x_i{}^{\mathcal{T}}, y_i{}^{\mathcal{T}})\}_{i=1}^{N_t}$



Preserve global properties of target distribution

Match individual samples

CEOD: Conditional embedding operator discrepancy

Given two datasets: $\mathcal{D}_p = \{(x_1, y_1), \ldots, (x_{N_1}, y_{N_1})\}$,
$\mathcal{D}_q = \{(x_1, y_1), \ldots, (x_{N_2}, y_{N_2})\}$

$$D_{\mathrm{CEOD}}(\mathcal{D}_p, \mathcal{D}_q) = \left\| \hat{\mathcal{C}}_{Y_p|X_p} - \hat{\mathcal{C}}_{Y_q|X_q} \right\|_{HS}^2$$

$$= \left\| \Phi(Y_p)\left(\mathbf{K}_{X_pX_p} + \lambda N_1 \mathbf{I}\right)^{-1} \mathcal{Y}^{\mathrm{T}}(X_p) - \Phi(Y_q)\left(\mathbf{K}_{X_qX_q} + \lambda N_2 \mathbf{I}\right)^{-1} \mathcal{Y}^{\mathrm{T}}(X_q) \right\|_{HS}^2$$

$$= \mathrm{Tr}\left\{\left(\mathbf{K}_{X_pX_p} + \lambda N_1 \mathbf{I}\right)^{-1} \mathbf{K}_{Y_pY_p} \left(\mathbf{K}_{X_pX_p} + \lambda N_1 \mathbf{I}\right)^{-1} \mathbf{K}_{X_pX_p}\right\} + \mathrm{Tr}\left\{\left(\mathbf{K}_{X_qX_q} + \lambda N_2 \mathbf{I}\right)^{-1} \mathbf{K}_{Y_qY_q} \left(\mathbf{K}_{X_qX_q} + \lambda N_2 \mathbf{I}\right)^{-1} \mathbf{K}_{X_qX_q}\right\} - 2\mathrm{Tr}\left\{\left(\mathbf{K}_{X_pX_p} + \lambda N_1 \mathbf{I}\right)^{-1} \mathbf{K}_{Y_pY_q} \left(\mathbf{K}_{X_qX_q} + \lambda N_2 \mathbf{I}\right)^{-1} \mathbf{K}_{X_qX_p}\right\}$$

- Inspired by: $D_{MMD}(\mathcal{D}_p, \mathcal{D}_q) = \left\| \hat{\mu}_{X_p} - \hat{\mu}_{X_q} \right\|_{HS}^2$
- CEOD measures the conditional distribution discrepancy in a reproducing kernel Hilbert space (RKHS)
- Constructs a Hilbert–Schmidt norm of the empirical conditional embedding operators of two distributions
- Based on the theory of kernel embeddings of conditional distributions

$\mathcal{C}_{Y|X}$: operator of inputs to outputs on a RKHS
$\Phi, \mathcal{Y}$: embed from original space to RKHS
$\mathbf{K}_{XX'}$: Gram matrix calculated with a Gaussian kernel k
λ: regularization term to avoid overfitting

Liu, X. et al. (2021). *Knowledge-Based Systems*

CEOD: Conditional embedding operator discrepancy

RVS: X, Y Original space Ω_X, Ω_Y

$$\left\{ \begin{array}{c} x \in \Omega_X \\ X \sim \mathbb{P}_X \\ Y \sim \mathbb{P}_Y \\ (X,Y) \sim \mathbb{P}_{XY} \end{array} \right\}$$

embed →

$^{*}\varphi(X): \Omega_X \to \mathcal{H}$

$\psi(Y): \Omega_Y \to \mathcal{F}$

RKHS feature spaces $\mathcal{H}, \mathcal{F}$

$$\left\{ \begin{array}{c} \varphi(x) \\ \psi(Y), \varphi(X) \\ \mu_Y, C_Y, C_{YX} \\ \mu_X, C_{XY}, C_X \end{array} \right\}$$

↓ conditioning on $X = x$

↓ conditional operator and mean embedding

$(Y|X = x) \sim \mathbb{P}_{Y|X=x}$ —embed→

$$C_{Y|X} = C_{YX} C_{XX}^{-1}$$

$$\mu_{Y|X=x} = C_{Y|X} \varphi(x)$$

$C_{Y|X}$: operator $\mathcal{H} \to \mathcal{F}$

$*\ k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}$

Given a dataset: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$:

$$\hat{C}_{Y|X} = \hat{C}_{YX} \hat{C}_{XX}^{-1} = \Phi(\mathbf{K} + \lambda N \mathbf{I})^{-1} \mathcal{Y}^{\mathrm{T}}$$

where $\Phi := (\psi(\mathbf{y}_1), \dots, \psi(\mathbf{y}_N))$

$\mathcal{Y} := (\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_N))$

$\mathbf{K} = \mathcal{Y}^{\mathrm{T}} \mathcal{Y}$ Gram matrix

λ: regularization parameter

Given two datasets $\mathcal{D}_p, \mathcal{D}_q$:

$$D_{\mathrm{CEOD}}(\mathcal{D}_p, \mathcal{D}_q) = \left\| \hat{C}_{Y_p|X_p} - \hat{C}_{Y_q|X_q} \right\|_{HS}^{2}$$

Klebanov et al. (2020) *SIAM Journal on Mathematics of Data Science*

Transfer learning DeepONet loss function

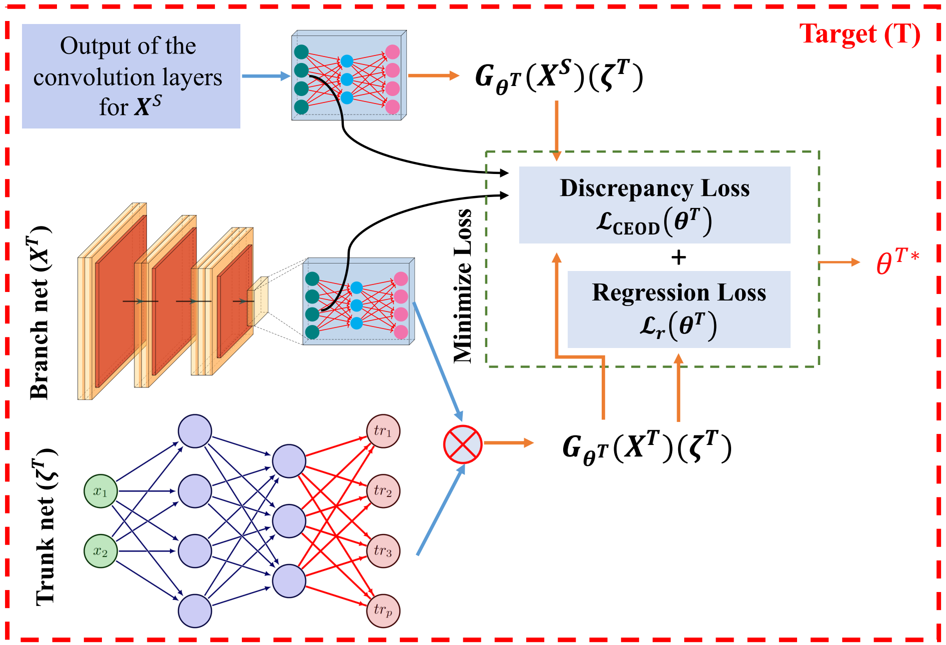
Hybrid loss function

$$\mathcal{L}(\theta^T) = \lambda_1 \mathcal{L}_r(\theta^T) + \lambda_2 \mathcal{L}_{\text{CEOD}}(\theta^T)$$

$$= \lambda_1 \frac{\|f_T(\mathbf{x}^{tL}) - \mathbf{y}^{tL}\|_2}{\|\mathbf{y}^{tL}\|_2} + \lambda_2 \left\| \hat{\mathcal{C}}_{Y_{tL}|X_{tL}} - \hat{\mathcal{C}}_{Y_{tU}|X_{tU}} \right\|_{HS}^2$$

λ_1, λ_2: Trainable coefficients updated through backpropagation[1]

Model finetuning



Regression loss

- $\mathcal{D}_t = \{(\mathbf{x}_i^{tL}, \mathbf{y}_i{}^{tL})\}_{i=1}^{N_t}$

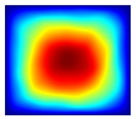
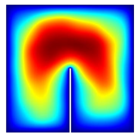
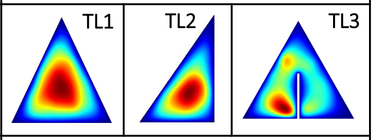
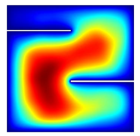
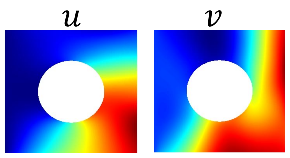
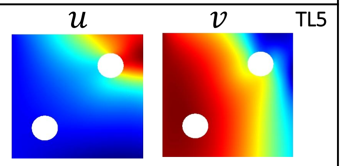
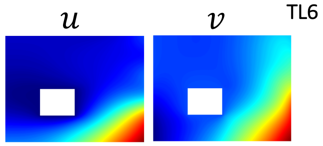
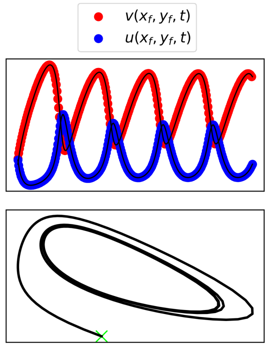
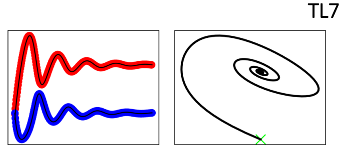
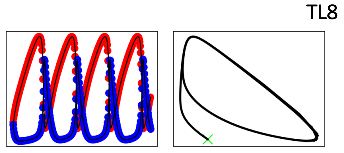
CEOD loss

- $\mathcal{D}_t^L = \{(\mathbf{x}_{b_1 i}^{tL}, \mathbf{y}_i{}^{tL})\}_{i=1}^{N_t}$
- $\mathcal{D}_t^U = \{(\mathbf{x}_{b_1 i}^{tU}, f_T(\mathbf{y}_i{}^{tU}))\}_{i=1}^{N_u}$

$\mathbf{x}_{b_1}$: output of the 1st FNN layer of branch net

[1] Kontolati, K., Goswami, S., et al. (2023). *Journal of Computational Physics*

Transfer learning applications

Application	Input Function	Model Output	Domain Visualization: Source	Domain Visualization: Target/s
Darcy Flow	Random input conductivity field $K(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$ $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{(x-x')^2}{2l^2}\right]$ $l = 0.25, \boldsymbol{x}, \boldsymbol{x}' \in [0,1]^2$	$\nabla.(K(\boldsymbol{x})\nabla h(\boldsymbol{x})) = 1$ $h(\boldsymbol{x}) = 0 \quad \forall \quad \boldsymbol{x} \in \partial\Omega$ $\mathcal{G}_\theta: K(\boldsymbol{x}) \rightarrow h(\boldsymbol{x})$	 	 
Elasticity Model	Random boundary conditions $\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$ $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}')^\wedge 2}{2l^2}\right]$ $l = 0.12, \boldsymbol{x}, \boldsymbol{x}' \in [0,1]$	$\nabla.\boldsymbol{\sigma} + \boldsymbol{f}(\boldsymbol{x}) = 0$ $(u, v) = 0 \ \forall \ x = 0$ $\mathcal{G}_\theta: f(\boldsymbol{x}) \rightarrow (u, v)$ u: X-Displacement v: Y-Displacement Material properties $E_S = 300 \cdot 10^5$, $\nu_S = 0.3$ | $E_{T_1} = 410 \cdot 10^3, \nu_{T_1} = 0.35$; $E_{T_2} = 410 \cdot 10^3, \nu_{T_2} = 0.45$		 
Brusselator Diffusion-Reaction System	Random initial condition $h_2(\boldsymbol{x}) \sim \mathcal{GP}(h_2(\boldsymbol{x}) \| \mu(\boldsymbol{x}), \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$ $v(x, y, t = 0) = h_2(x, y) \geq 0$ $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}')^\wedge 2}{2l^2}\right]$ $l_x = 0.12, l_y = 0.4, \sigma^2 = 0.15$	$\frac{\partial u}{\partial t} = D_0 \nabla^2 u + a - (1-b)u + vu^2$ $\frac{\partial v}{\partial t} = D_1 \nabla^2 v + bu - vu^2$ $x \in [0,1]^2, t \in [0,1]$ $\mathcal{G}_\theta: h_2(x, y) \rightarrow v(x, y, t)$ Model parameter $b_S = 2.2$ | $b_{T_1} = 1.7$, $b_{T_2} = 3.0$		 

geometric domain shift

geometric domain + model parameter shift

model dynamics shift

Results: Darcy Flow (TL3)

Objective:

$\mathcal{G}: K(\boldsymbol{x}) \rightarrow h(\boldsymbol{x})$

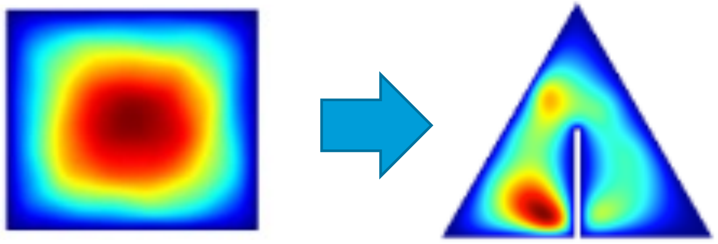
Random input conductivity field

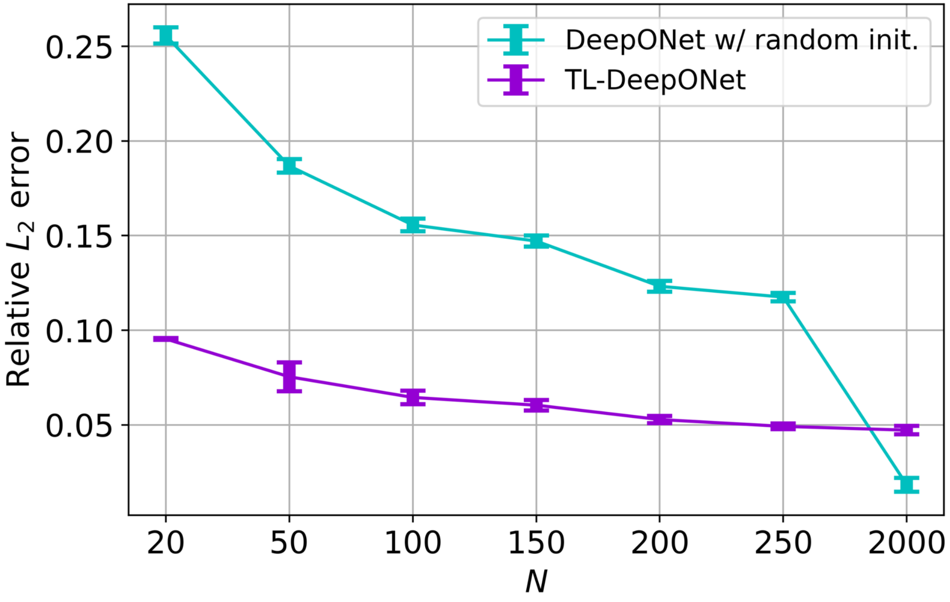
$$K(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{2l^2}\right]$$
$$l = 0.25, \boldsymbol{x}, \boldsymbol{x}' \in [0,1]^2$$

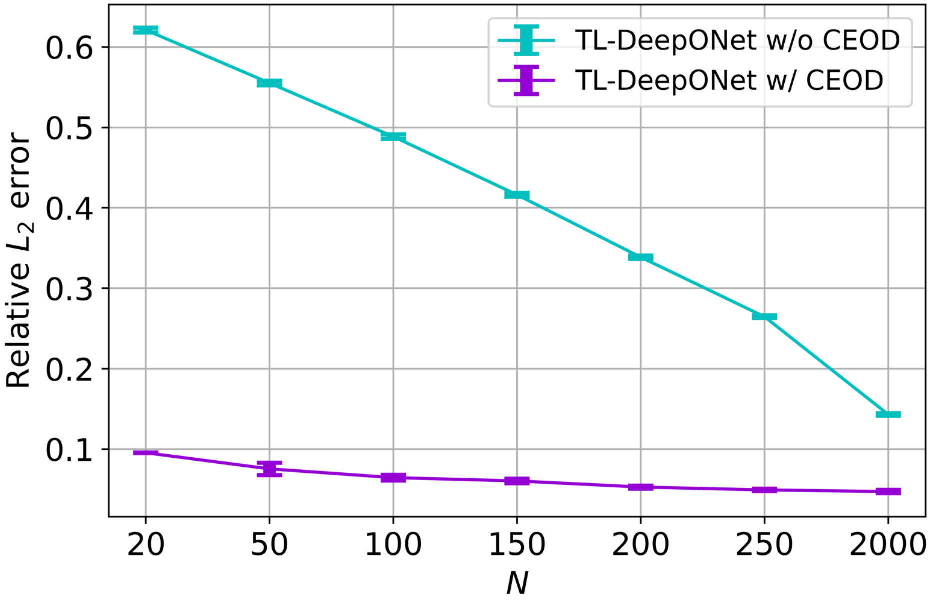
Hydraulic head

$$\nabla(K(\boldsymbol{x})\nabla h(\boldsymbol{x})) = 1$$
$$h(\boldsymbol{x}) = 0 \quad \forall \quad \boldsymbol{x} \in \partial\Omega$$

Transfer learning scenario:







Results: Darcy Flow (TL4)

Objective:

$\mathcal{G}: K(\boldsymbol{x}) \rightarrow h(\boldsymbol{x})$

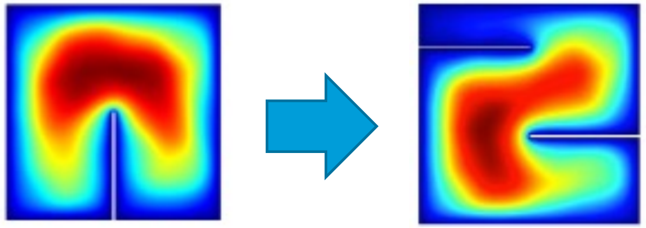
Random input conductivity field

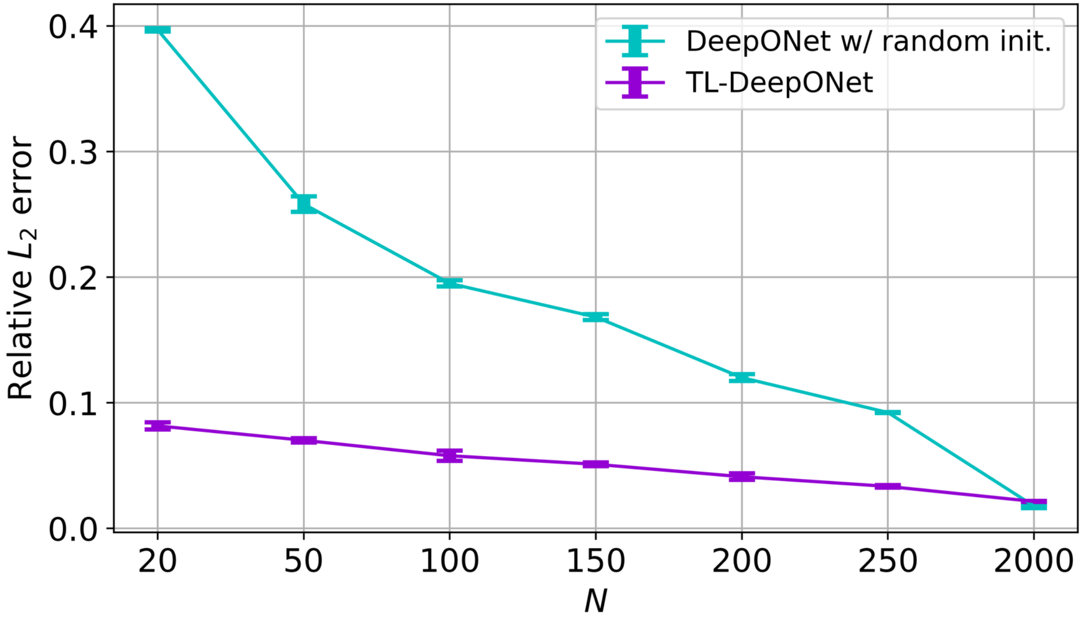
$$K(\boldsymbol{x}) \sim \mathcal{GP}\left(0, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')\right)$$
$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{2l^2}\right]$$
$$l = 0.25, \boldsymbol{x}, \boldsymbol{x}' \in [0,1]^2$$

Hydraulic head

$$\nabla.\left(K(\boldsymbol{x})\nabla h(\boldsymbol{x})\right) = 1$$
$$h(\boldsymbol{x}) = 0 \quad \forall \quad \boldsymbol{x} \in \partial\Omega$$

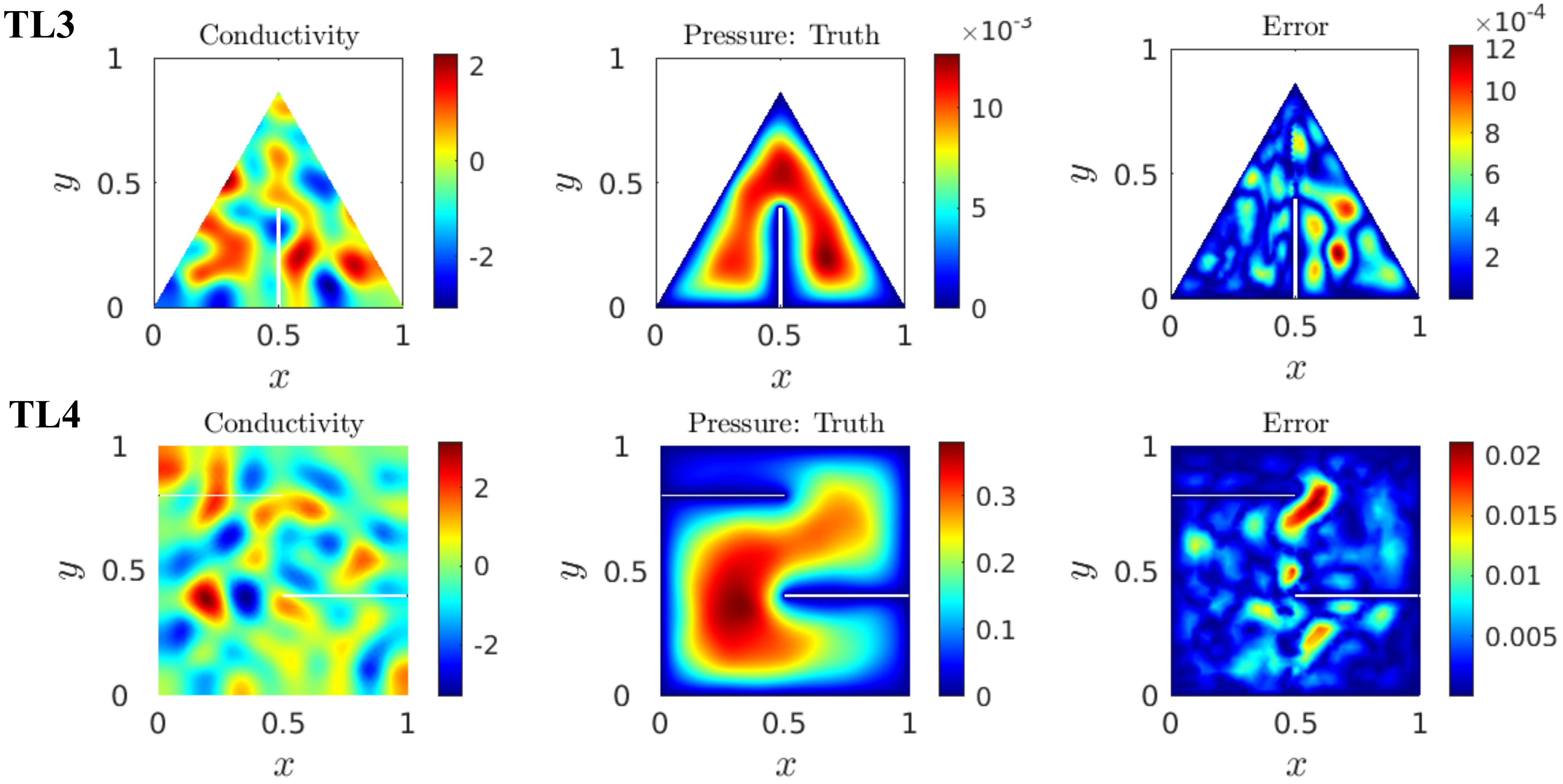
Transfer learning scenario:





Representative result: Darcy Flow

Representative results of TL-DeepONet for Darcy's problem



Representative result: Darcy Flow

Table: GPU time (s) for all Darcy flow transfer learning scenarios

	N_t	TL1	TL2	TL3	TL4
Training DeepONet (source)	2,000	15,260	15,260	15,260	2,261
Training DeepONet (target)	2,000	12,880	18,200	18,080	3,978
Training TL-DeepONet	5	11	10	10	83
	20	129	116	112	139
	50	416	399	302	289
	100	439	437	351	300
	150	459	439	378	302
	200	462	480	406	304
	250	531	528	586	305
	2,000	595	601	653	350

* Simulations performed on single NVIDIA RTX A6000 GPU

Results: Linear elasticity (TL5)

Objective:

$$\mathcal{G}: f(\boldsymbol{x}) \rightarrow [u(\boldsymbol{x}), v(\boldsymbol{x})]$$

Random RHS

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'))$$

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{(x-x')^2}{2l^2}\right]$$

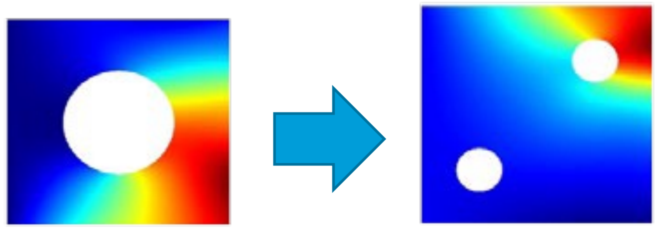
$$l = 0.12, \boldsymbol{x}, \boldsymbol{x}' \in [0,1]^2$$

Displacement

$$\nabla.\sigma + f(\boldsymbol{x}) = 0$$

$$u(\boldsymbol{x}) = v(\boldsymbol{x}) = 0 \ \ \forall\, x = 0$$

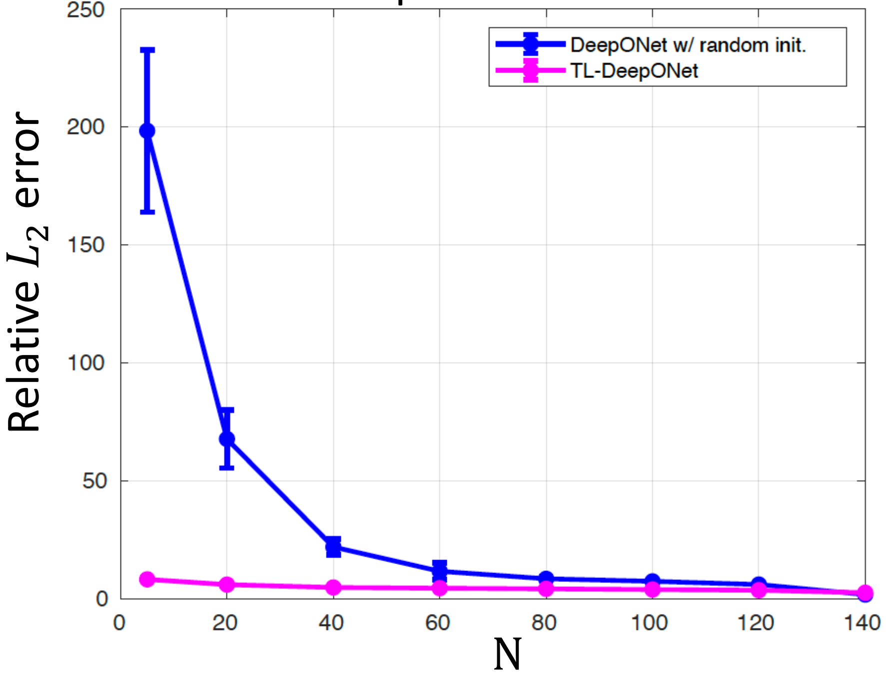
Transfer learning scenario:



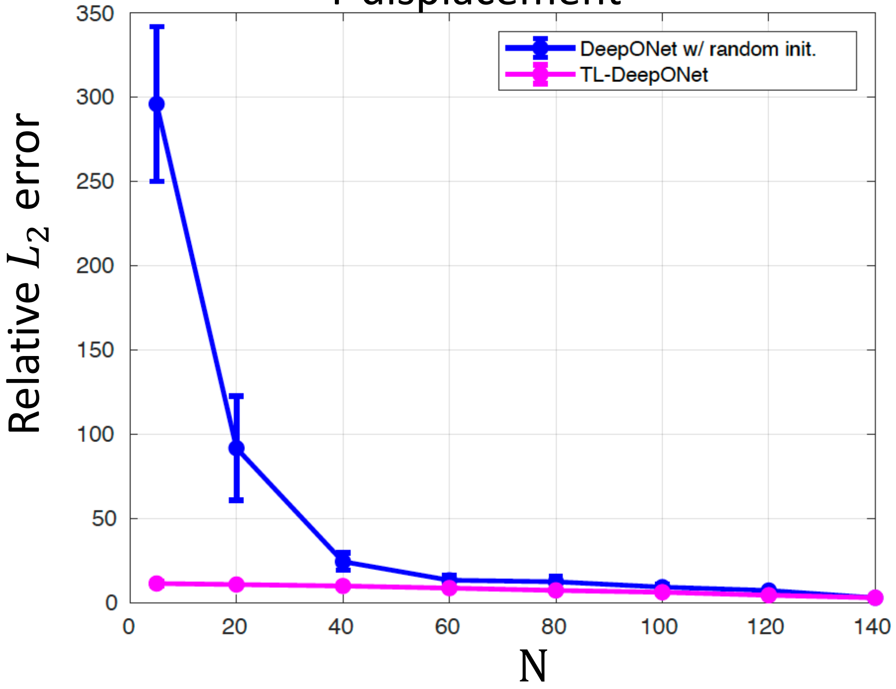
$E_s = 300e5$, $\nu_s = 0.3$

$E_T = 410e3$, $\nu_T = 0.35$

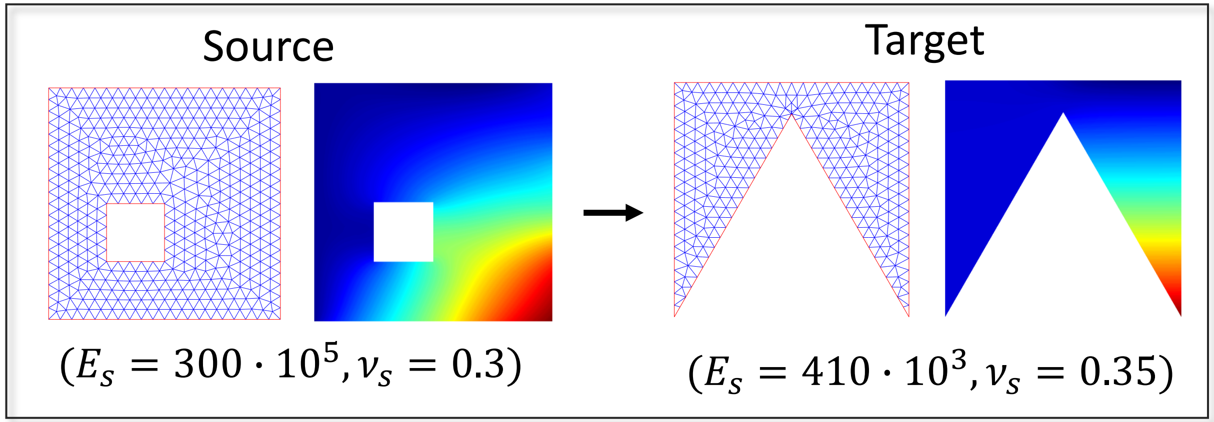
X-displacement



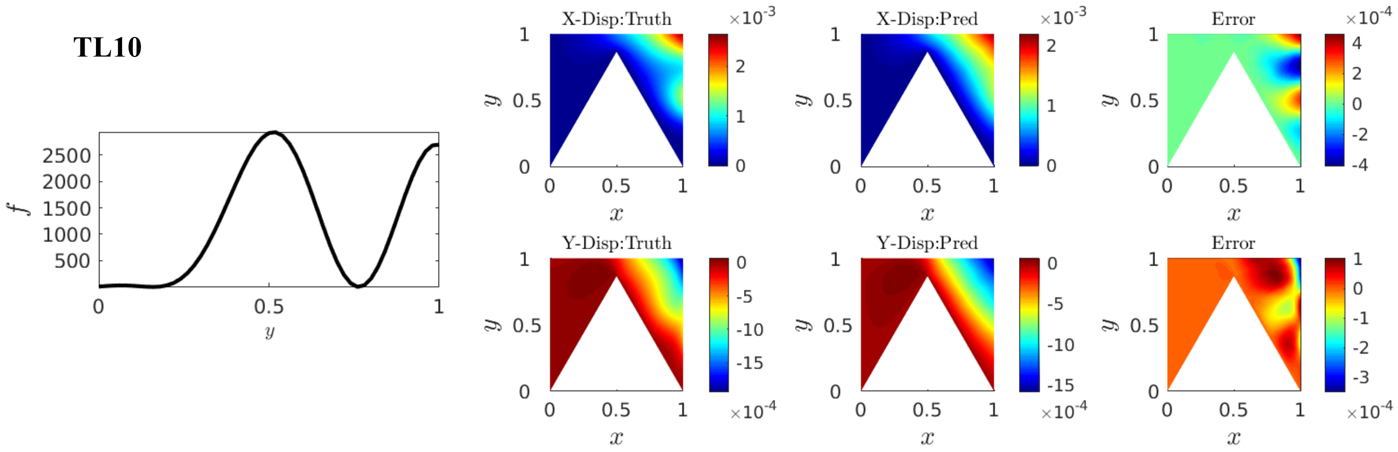
Y-displacement



Results: Limitations of TL-DeepONet



	N_t	L_2 (%) $u(\boldsymbol{x})$	L_2 (%) $v(\boldsymbol{x})$	time (s)
Training DeepONet (source)	1,900	2.30 ± 0.49	3.22 ± 0.48	10,060
Training DeepONet (target)	1,900	2.72 ± 0.26	1.92 ± 0.41	11,750
Training TL-DeepONet	5	60.28 ± 1.95	57.48 ± 2.54	18
	20	29.14 ± 0.31	21.42 ± 3.20	148
	50	16.4 ± 2.01	18.72 ± 1.18	25
	100	11.37 ± 0.34	14.15 ± 0.96	44
	150	9.66 ± 0.26	11.95 ± 0.61	116
	200	5.52 ± 0.20	10.94 ± 0.20	384
	250	4.08 ± 0.06	9.18 ± 0.14	480
	1,900	3.82 ± 0.20	7.89 ± 0.21	512



- Different internal, external boundaries and material properties;
- Predicted displacements field deviates significantly with the response predicted by training DeepONet from scratch

* Simulations performed on single NVIDIA RTX A6000 GPU

Key takeaways

- Learn operators on multiple PDE domains via transfer learning and treat trained models as reusable building blocks
- TL-DeepONet:
  - Performs well under small-data regimes
  - TL-DeepONet accelerates learning via fine-tuning pre-trained models
  - Enhances generalizability in neural operators
- Code availability: https://github.com/katiana22/TL-DeepONet.git

References

1. **Goswami, S.*, Kontolati, K.*, Shields, M. D., & Karniadakis, G. E. (2022). Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 1-10.**
2. **Kahana, A., Zhang, E., Goswami, S., Karniadakis, G., Ranade, R., & Pathak, J. (2023). On the geometry transferability of the hybrid iterative numerical solver for differential equations. Computational Mechanics, 1-14.**
3. Lu, Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. (2021) Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence* 3, no. 3: 218-229.
4. Chen, T., & Chen, H. (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4), 911-917.
5. Liu, X., Li, Y., Meng, Q., Chen, G.: Deep transfer learning for conditional shift in regression. Knowledge-Based Systems 227, 107216 (2021).

* denotes equal contribution

The team

Dr. George Karniadakis
Professor
Brown University

Dr. Somdatta Goswami
Assistant Professor (Research)
Brown University

Dr. Michael Shields
Associate Professor
Johns Hopkins University

Dr. Katiana Kontolati
Johns Hopkins University

Computing support:

CCV
@Brown

RockFish
@JHU

Positions available

Ph.D. -2 (starting Spring)

Postdocs – 1 (immediately)

in the Department of Civil and Systems Engineering at Johns Hopkins University

Topics: SciML for materials and mechanics

Email: somdatta89@gmail.com

Thank you!