Neural Operators

Nick Dexter[†]

[†]Assistant Professor, Department of Scientific Computing, Florida State University, Tallahassee, FL, USA

FLORIDA STATE UNIVERSITY

FSU SC Machine Learning Seminar

Friday, October 13th, 2023 – Tallahassee, FL, USA

Motivation

Surrogate modeling: development of computationally efficient approximations for expensive simulations, e.g. those arising from numerical solution of partial differential equations (PDEs).

Often times the cost of solving the PDE dominates the costs of modeling. The idea is to use a cheap surrogate to replace the PDE solver to enable efficient evaluation in "many-query" computations in science and engineering applications.

Operator approximation: PDEs are often formulated in terms of operators acting on infinite-dimensional function spaces. The goal is to replace the operator equations with the surrogate neural operator model.

Note: This talk is based on the paper "The Cost-Accuracy Trade-Off in Operator Learning with Neural Networks" by Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart.

Motivation

Surrogate modeling: development of computationally efficient approximations for expensive simulations, e.g. those arising from numerical solution of partial differential equations (PDEs).

Often times the cost of solving the PDE dominates the costs of modeling. The idea is to use a cheap surrogate to replace the PDE solver to enable efficient evaluation in "many-query" computations in science and engineering applications.

Operator approximation: PDEs are often formulated in terms of operators acting on infinite-dimensional function spaces. The goal is to replace the operator equations with the surrogate neural operator model.

Note: This talk is based on the paper "The Cost-Accuracy Trade-Off in Operator Learning with Neural Networks" by Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart.

Motivation

Surrogate modeling: development of computationally efficient approximations for expensive simulations, e.g. those arising from numerical solution of partial differential equations (PDEs).

Often times the cost of solving the PDE dominates the costs of modeling. The idea is to use a cheap surrogate to replace the PDE solver to enable efficient evaluation in "many-query" computations in science and engineering applications.

Operator approximation: PDEs are often formulated in terms of operators acting on infinite-dimensional function spaces. The goal is to replace the operator equations with the surrogate neural operator model.

Note: This talk is based on the paper "The Cost-Accuracy Trade-Off in Operator Learning with Neural Networks" by Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart.

Numerical analysis for PDE modeling & error sources for operator learning

Numerical analysis for PDE modeling:

- error as a function of the resolution of the finite dimensional approximation
- cost of running the model, at a given level of finite-dimensional resolution, often dominated any of the following
  - matrix inversion
  - matrix-vector multiplies
  - time-stepping and iteration count for nonlinear solvers

Errors for operator learning:

- discretization of the input/output spaces
- parameterization of the operator approximations
- finite data volume
- the optimizer used in training

Numerical analysis for PDE modeling & error sources for operator learning

Numerical analysis for PDE modeling:

- error as a function of the resolution of the finite dimensional approximation
- cost of running the model, at a given level of finite-dimensional resolution, often dominated any of the following
  - matrix inversion
  - matrix-vector multiplies
  - time-stepping and iteration count for nonlinear solvers

Errors for operator learning:

- discretization of the input/output spaces
- parameterization of the operator approximations
- finite data volume
- the optimizer used in training

Analysis preliminaries

Cauchy sequence: $x_1, x_2, x_3, \ldots$ is called a Cauchy sequence if for every $\varepsilon > 0$, there is an $N \in \mathbb{N}$ such that for all $m, n \in \mathbb{N}$ with $m, n > N$, we have $\|x_n - x_m\| < \varepsilon$.

Complete space: a metric space $(\mathcal{V}, d)$ (typically with metric $d(x, y) = \|x - y\|_{\mathcal{V}}$ for $x, y \in \mathcal{V}$) is called complete if every Cauchy sequence converges to an element of $\mathcal{V}$.

- **Hilbert space:** complete inner product space $(\mathcal{V}, \langle \cdot, \cdot \rangle_{\mathcal{V}})$,
- **Banach space:** complete normed space $(\mathcal{V}, \|\cdot\|_{\mathcal{V}})$.

σ-algebra: Let $\mathcal{V}$ be a set, and $P(\mathcal{V})$ be its power set (set of all subsets of $\mathcal{V}$ including the empty set $\emptyset$ and $\mathcal{V}$ itself). Then $\Sigma \subseteq P(\mathcal{V})$ is called a σ-algebra if and only if

- $\mathcal{V} \in \Sigma$
- Σ is closed under complements: if $A \in \Sigma$, then $A^c = \mathcal{V} \setminus A \in \Sigma$
- Σ is closed under countable unions: if $A_1, A_2, A_3, \ldots \in \Sigma$, then $\cup_i A_i \in \Sigma$.

If Σ is a σ-algebra, then Σ is also closed under countable intersections, and $\emptyset \in \Sigma$. Elements of Σ are called measurable sets, and $(\mathcal{V}, \Sigma)$ is called a **measurable space**.

Analysis preliminaries

Cauchy sequence: $x_1, x_2, x_3, \ldots$ is called a Cauchy sequence if for every $\varepsilon > 0$, there is an $N \in \mathbb{N}$ such that for all $m, n \in \mathbb{N}$ with $m, n > N$, we have $\|x_n - x_m\| < \varepsilon$.

Complete space: a metric space $(\mathcal{V}, d)$ (typically with metric $d(x, y) = \|x - y\|_{\mathcal{V}}$ for $x, y \in \mathcal{V}$) is called complete if every Cauchy sequence converges to an element of $\mathcal{V}$.

- **Hilbert space:** complete inner product space $(\mathcal{V}, \langle \cdot, \cdot \rangle_{\mathcal{V}})$,
- **Banach space:** complete normed space $(\mathcal{V}, \|\cdot\|_{\mathcal{V}})$.

σ-algebra: Let $\mathcal{V}$ be a set, and $P(\mathcal{V})$ be its power set (set of all subsets of $\mathcal{V}$ including the empty set $\emptyset$ and $\mathcal{V}$ itself). Then $\Sigma \subseteq P(\mathcal{V})$ is called a σ-algebra if and only if

- $\mathcal{V} \in \Sigma$
- Σ is closed under complements: if $A \in \Sigma$, then $A^c = \mathcal{V} \setminus A \in \Sigma$
- Σ is closed under countable unions: if $A_1, A_2, A_3, \ldots \in \Sigma$, then $\cup_i A_i \in \Sigma$.

If Σ is a σ-algebra, then Σ is also closed under countable intersections, and $\emptyset \in \Sigma$. Elements of Σ are called measurable sets, and $(\mathcal{V}, \Sigma)$ is called a **measurable space**.

Analysis preliminaries

Let $(\mathcal{U}, \Gamma)$ and $(\mathcal{V}, \Sigma)$ be measurable spaces. A function $f : \mathcal{U} \to \mathcal{V}$ is said to be **measurable** if for every $E \in \Sigma$ the pre-image of E under f is in Γ; that is, for all $E \in \Sigma$

$$f^{-1}(E) := \{x \in \mathcal{U} : f(x) \in E\} \in \Gamma.$$

Given measurable spaces $(\mathcal{U}, \Gamma)$ and $(\mathcal{V}, \Sigma)$, a measurable mapping $f : \mathcal{U} \to \mathcal{V}$ and a measure $\mu : \Gamma \to [0, +\infty]$, the **push-forward** of μ is defined to be the measure $f^{\#}(\mu) : \Sigma \to [0, +\infty]$ given by

$$f^{\#}(\mu)(B) = \mu(f^{-1}(B)) \qquad \text{for } B \in \Sigma.$$

The **Borel** σ**-algebra** on $\mathcal{V}$ is the smallest σ-algebra containing all open sets (or, equivalently, all closed sets).

Formal problem statement

Consider the following Banach spaces of functions $\mathcal{U}$, $\mathcal{V}$:

- $\mathcal{U} = \{u : D_u \to \mathbb{R}^{d_i}\}$, where $D_u \subseteq \mathbb{R}^{d_x}$,
- $\mathcal{V} = \{v : D_v \to \mathbb{R}^{d_o}\}$, where $D_v \subseteq \mathbb{R}^{d_y}$.

Goal: determine operators $\Psi^\dagger : \mathcal{U} \to \mathcal{V}$ from (samples from) the probability measure $(\mathrm{Id}, \Psi^\dagger)^{\#}(\mu)$. Here

- μ is supported on $\mathcal{U}$ and equipped with the Borel σ-algebra,
- push-forward $(\mathrm{Id}, \Psi^\dagger)^{\#}(\mu)$ is supported on $\mathcal{U} \times \mathcal{V}$, equipped with Borel σ-algebra.

For simplicity assume D_u and D_v are bounded.

Here we seek to approximate $\Psi^\dagger$ by different classes of parametric operators, each of which characterizes a different neural network architecture. Consider:

- $\Theta \subseteq \mathbb{R}^p$ – the space of parameters of the neural network,
- family of functions from $\mathcal{U}$ to $\mathcal{V}$ defined by $\Psi : \mathcal{U} \times \Theta \to \mathcal{V}$.

Formal problem statement

Consider the following Banach spaces of functions $\mathcal{U}$, $\mathcal{V}$:

- $\mathcal{U} = \{u : D_u \to \mathbb{R}^{d_i}\}$, where $D_u \subseteq \mathbb{R}^{d_x}$,
- $\mathcal{V} = \{v : D_v \to \mathbb{R}^{d_o}\}$, where $D_v \subseteq \mathbb{R}^{d_y}$.

Goal: determine operators $\Psi^\dagger : \mathcal{U} \to \mathcal{V}$ from (samples from) the probability measure $(\mathrm{Id}, \Psi^\dagger)^{\#}(\mu)$. Here

- μ is supported on $\mathcal{U}$ and equipped with the Borel σ-algebra,
- push-forward $(\mathrm{Id}, \Psi^\dagger)^{\#}(\mu)$ is supported on $\mathcal{U} \times \mathcal{V}$, equipped with Borel σ-algebra.

For simplicity assume D_u and D_v are bounded.

Here we seek to approximate $\Psi^\dagger$ by different classes of parametric operators, each of which characterizes a different neural network architecture. Consider:

- $\Theta \subseteq \mathbb{R}^p$ – the space of parameters of the neural network,
- family of functions from $\mathcal{U}$ to $\mathcal{V}$ defined by $\Psi : \mathcal{U} \times \Theta \to \mathcal{V}$.

Formal problem statement

Idealized case of infinite data: the parameters $\theta \in \Theta$ are taken as θ^*, the minimizer of the risk defined by

$$\text{Risk}: \ \mathcal{R}_\infty(\theta) := \mathbb{E}^{u\sim\mu}\|\Psi^\dagger(u) - \Psi(u;\theta)\|_{\mathcal{V}}^2 = \|\Psi^\dagger - \Psi\|_{L^2_\mu(\mathcal{U},\mathcal{V})}^2,$$

Here the Bochner space $L^2_\mu(\mathcal{U},\mathcal{V})$ is defined with integration over $\mathcal{U}$, and we say $u \in L^2_\mu(\mathcal{U},\mathcal{V})$ if

$$\|u\|_{L^2_\mu(\mathcal{U},\mathcal{V})} := \left(\int_{\mathcal{U}} \|v(u)\|_{\mathcal{V}}^2 \, \mathrm{d}\mu(u)\right)^{1/2} < \infty.$$

Practical setting: we only have access to the samples $\{u_n\}_{n=1}^N$ from μ, assumed to be independent and identically distributed and define the empirical measure μ^N, together with $\{\Psi^\dagger(u_n)\}_{n=1}^N$, resulting in the empirical risk:

$$\begin{aligned}\text{Empirical risk}: \ \mathcal{R}_N(\infty)(\theta) &:= \mathbb{E}^{u\sim\mu^N}\|\Psi^\dagger(u) - \Psi(u;\theta)\|_{\mathcal{V}}^2 \\ &= \frac{1}{N}\sum_{n=1}^N \|\Psi^\dagger(u_n) - \Psi(u_n;\theta)\|_{\mathcal{V}}^2.\end{aligned}$$

The parameter $\theta^{*,N}$ is the value of θ which minimizes $\mathcal{R}_N(\infty)(\theta)$, and non-convexity of the optimization over θ means we generally only have access to an approximation of $\theta^{*,N}$.

Some types of neural operators...

- PCA-based neural operators (PCA-Net)
- Deep operator networks (DeepONet)
- Pointwise evaluation (PARA-Net)
- Physics-informed neural operator (PINO)
- Fourier neural operator (FNO)
- Convolutional neural operator (CNO)
- Graph neural operator (GNO) and Graph kernel network (GKN)
- Geometry-informed neural operator (GINO)
- Wavelet neural operator (WNO) and multi-wavelet neural operator
- Laplace neural operator (LNO)
- Basis enhanced neural operator (Belnet)
- and many more... (Green's functions, attention mechanism, autoencoders, generative adversarial, kernel integral operators, spectral, U-NO (U-shaped), FNO-transformer, multipole graph, Galerkin transformer, ...)

Some types of neural operators... we'll focus on these

- PCA-based neural operators (PCA-Net)
- Deep operator networks (DeepONet)
- Pointwise evaluation (PARA-Net)
- Physics-informed neural operator (PINO)
- Fourier neural operator (FNO)
- Convolutional neural operator (CNO)
- Graph neural operator (GNO) and Graph kernel network (GKN)
- Geometry-informed neural operator (GINO)
- Wavelet neural operator (WNO) and multi-wavelet neural operator
- Laplace neural operator (LNO)
- Basis enhanced neural operator (Belnet)
- and many more... (Green's functions, attention mechanism, autoencoders, generative adversarial, kernel integral operators, spectral, U-NO (U-shaped), FNO-transformer, multipole graph, Galerkin transformer, ...)

Further preliminaries

For two normed spaces $\mathcal{U}$ and $\mathcal{V}$ with norms $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_{\mathcal{V}}$, respectively, if the inclusion map

$$i : \mathcal{U} \hookrightarrow \mathcal{V} : x \mapsto x$$

is continuous, i.e., there exists a constant $C > 0$ with

$$\|x\|_{\mathcal{V}} \leq C\|x\|_{\mathcal{U}}$$

for every x in $\mathcal{U}$, then we say $\mathcal{U}$ is **continuously embedded** in $\mathcal{V}$.

Let $\mathcal{H}$ denote a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and induced norm $\|\cdot\|_{\mathcal{H}}$.

We then consider the Gelfand triple $\mathcal{U} \subseteq \mathcal{H} \subseteq \mathcal{U}^*$ where the embedding of $\mathcal{U}$ in $\mathcal{H}$ is continuous and $\mathcal{U}^*$ denotes the dual space of $\mathcal{U}$ – the space of continuous linear functionals on $\mathcal{U}$.

Given $L_k \in \mathcal{U}^*$, $k = 1, \ldots, d_u$, the function u in $\mathcal{U}$ is partially characterized by the finite dimensional vector $Lu = \{L_k u\}_{k=1}^{d_u}$ (this will be used to characterize the inputs to Ψ in the neural network architectures to follow.

Further preliminaries

For two normed spaces $\mathcal{U}$ and $\mathcal{V}$ with norms $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_{\mathcal{V}}$, respectively, if the inclusion map

$$i : \mathcal{U} \hookrightarrow \mathcal{V} : x \mapsto x$$

is continuous, i.e., there exists a constant $C > 0$ with

$$\|x\|_{\mathcal{V}} \leq C\|x\|_{\mathcal{U}}$$

for every x in $\mathcal{U}$, then we say $\mathcal{U}$ is **continuously embedded** in $\mathcal{V}$.

Let $\mathcal{H}$ denote a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and induced norm $\|\cdot\|_{\mathcal{H}}$.

We then consider the Gelfand triple $\mathcal{U} \subseteq \mathcal{H} \subseteq \mathcal{U}^*$ where the embedding of $\mathcal{U}$ in $\mathcal{H}$ is continuous and $\mathcal{U}^*$ denotes the dual space of $\mathcal{U}$ – the space of continuous linear functionals on $\mathcal{U}$.

Given $L_k \in \mathcal{U}^*$, $k = 1, \ldots, d_u$, the function u in $\mathcal{U}$ is partially characterized by the finite dimensional vector $Lu = \{L_k u\}_{k=1}^{d_u}$ (this will be used to characterize the inputs to Ψ in the neural network architectures to follow.

Further preliminaries

For two normed spaces $\mathcal{U}$ and $\mathcal{V}$ with norms $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_{\mathcal{V}}$, respectively, if the inclusion map

$$i : \mathcal{U} \hookrightarrow \mathcal{V} : x \mapsto x$$

is continuous, i.e., there exists a constant $C > 0$ with

$$\|x\|_{\mathcal{V}} \leq C\|x\|_{\mathcal{U}}$$

for every x in $\mathcal{U}$, then we say $\mathcal{U}$ is **continuously embedded** in $\mathcal{V}$.

Let $\mathcal{H}$ denote a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and induced norm $\|\cdot\|_{\mathcal{H}}$.

We then consider the Gelfand triple $\mathcal{U} \subseteq \mathcal{H} \subseteq \mathcal{U}^*$ where the embedding of $\mathcal{U}$ in $\mathcal{H}$ is continuous and $\mathcal{U}^*$ denotes the dual space of $\mathcal{U}$ – the space of continuous linear functionals on $\mathcal{U}$.

Given $L_k \in \mathcal{U}^*$, $k = 1, \ldots, d_u$, the function u in $\mathcal{U}$ is partially characterized by the finite dimensional vector $Lu = \{L_k u\}_{k=1}^{d_u}$ (this will be used to characterize the inputs to Ψ in the neural network architectures to follow.

Further preliminaries

When $\mathcal{U}$ and $\mathcal{V}$ are themselves Hilbert spaces, we may compute the mean and convariance operator with respect to μ and $(\Psi^\dagger)^\# \mu$, respectively, assuming the first and second moments exist.

The eigen-decomposition of the covariance operators (PCA) gives rise to orthonormal bases $\{\phi_j\}_{j\in\mathbb{N}}$ in $\mathcal{U}$ and $\{\psi_j\}_{j\in\mathbb{N}}$ in $\mathcal{V}$. Note that

$$\phi_j : D_u \to \mathbb{R}^{d_i}, \quad j \in \mathbb{N}; \qquad \psi_j : D_v \to \mathbb{R}^{d_o}, \quad j \in \mathbb{N}.$$

PCA-Net

Consider $\mathcal{H} = \mathcal{U} = L^2(D_u; \mathbb{R}^{d_i})$. Truncate the PCA basis of μ to the first d_u modes, $\{\psi_j\}_{j=1}^{d_u}$, and truncate the PCA basis of $(\Psi^\dagger)^\# \mu$ to the first d_v modes, $\{\psi_j\}_{j=1}^{d_v}$.

Introduce the function $\alpha : \mathbb{R}^{d_u} \times \Theta \to \mathbb{R}^{d_v}$ defined componentwise by

$$\alpha_j : \mathbb{R}^{d_u} \times \Theta \to \mathbb{R}, \quad j = 1, 2, \ldots, d_v$$

which maps from the PCA coefficients $\{L_k u\}_{k=1}^{d_u}$ of μ to the PCA coefficients $\{\alpha_j\}_{j=1}^{d_v}$ of $(\Psi^\dagger)^\# \mu$, and is parameterized by $\theta \in \Theta$. Here α denotes a finite dimensional NN.

The approximating operator between $\mathcal{U}$ and $\mathcal{V}$ is then defined by

$$\Psi_{\mathrm{PCA}}(u; \theta)(y) = \sum_{j=1}^{d_v} \alpha_j(Lu; \theta)\psi_j(y) = \alpha(Lu; \theta)^\top \psi(y) \qquad \forall u \in \mathcal{U} \quad y \in D_v$$

where $Lu = \{L_k u\}_{k=1}^{d_u}$ and $\psi(y) = \{\psi_j(y)\}_{k=1}^{d_v}$. Note that each α_j is $\mathbb{R}$-valued, so α may be viewed as a column vector in $\mathbb{R}^{d_v}$, whilst each ψ_j is $\mathbb{R}^{d_o}$-valued, so ψ may be viewed as a matrix in $\mathbb{R}^{d_v} \times \mathbb{R}^{d_o}$. Thus $\Psi_{\mathrm{PCA}}(u; \theta)$ is $\mathbb{R}^{d_o}$-valued.

PCA-Net

Consider $\mathcal{H} = \mathcal{U} = L^2(D_u; \mathbb{R}^{d_i})$. Truncate the PCA basis of μ to the first d_u modes, $\{\psi_j\}_{j=1}^{d_u}$, and truncate the PCA basis of $(\Psi^\dagger)^\# \mu$ to the first d_v modes, $\{\psi_j\}_{j=1}^{d_v}$.

Introduce the function $\alpha : \mathbb{R}^{d_u} \times \Theta \to \mathbb{R}^{d_v}$ defined componentwise by

$$\alpha_j : \mathbb{R}^{d_u} \times \Theta \to \mathbb{R}, \quad j = 1, 2, \ldots, d_v$$

which maps from the PCA coefficients $\{L_k u\}_{k=1}^{d_u}$ of μ to the PCA coefficients $\{\alpha_j\}_{j=1}^{d_v}$ of $(\Psi^\dagger)^\# \mu$, and is parameterized by $\theta \in \Theta$. Here α denotes a finite dimensional NN.

The approximating operator between $\mathcal{U}$ and $\mathcal{V}$ is then defined by

$$\Psi_{\mathrm{PCA}}(u; \theta)(y) = \sum_{j=1}^{d_v} \alpha_j(Lu; \theta)\psi_j(y) = \alpha(Lu; \theta)^\top \psi(y) \qquad \forall u \in \mathcal{U} \quad y \in D_v$$

where $Lu = \{L_k u\}_{k=1}^{d_u}$ and $\psi(y) = \{\psi_j(y)\}_{k=1}^{d_v}$. Note that each α_j is $\mathbb{R}$-valued, so α may be viewed as a column vector in $\mathbb{R}^{d_v}$, whilst each ψ_j is $\mathbb{R}^{d_o}$-valued, so ψ may be viewed as a matrix in $\mathbb{R}^{d_v} \times \mathbb{R}^{d_o}$. Thus $\Psi_{\mathrm{PCA}}(u; \theta)$ is $\mathbb{R}^{d_o}$-valued.

Remarks:

- PCA is defined on a Hilbert space, given measure μ; in practice this is implemented on finite-dimensional approximations of a Hilbert space, using samples from μ.

- The linear functionals $\{L_k\}$ in $\mathcal{U}^*$ and functions $\{\psi_j\}$ in $\mathcal{V}$ are precomputed using PCA and decoupled from the network training. Preliminary calculations using PCA on the dataset are conducted to define the risk or empirical risk.

DeepONet

The operator between $\mathcal{U}$ and $\mathcal{V}$ is defined as follows:

$$\begin{aligned}\Psi_{\text{DeepONet}}(u;\theta)(y) &= \sum_{j=1}^{d_v} \alpha_j(Lu;\theta_\alpha)\psi_j(y;\theta_\psi) \\ &= \alpha(Lu;\theta_\alpha)^\top \psi(y;\theta_\psi), \qquad \forall u \in \mathcal{U}, \quad y \in D_v.\end{aligned}$$

Here

- the functions $\alpha_j : \mathbb{R}^{d_u} \times \Theta \to \mathbb{R}$ have the same structure as in PCA-Net, and are collectively referred to as the "branch" of the DeepONet architecture.
- the functions $\{\psi_j\}$ are collectively referred to as the "trunk" of the DeepONet architecture and are defined by networks of the form $\psi_j : D_v \times \Theta \to \mathbb{R}^{d_o}$, $j = 1, 2, \ldots, d_v$.
- θ encapsulates both θ_α and θ_ψ appearing in α and ψ.

Remarks:

- For DeepONet both the branch and the trunk are neural networks, whereas for PCA-Net only the branch is. For DeepONet, ψ is selected via optimization during training, while for PCA-Net ψ is defined by the precomputing step directly from the data.

- In DeepONets, L may not depend on μ if pointwise evaluations are used.

- For both network architectures, the norm $\|\cdot\|_{\mathcal{V}}$ is approximated using pointwise evaluations at a set of points $\{y_l\}$.

PARA-Net

As with the PCA-Net approach, first truncate the PCA basis of μ, $\{\phi_j\}_{j=1}^{d_u}$, using the first d_u modes, and define $L_k \in \mathcal{U}^*$ by $L_k u = \langle \phi_k, u \rangle_{\mathcal{U}}$, $k = 1, 2, \ldots, d_u$.

Introduce the real-valued network

$$\psi : \mathbb{R}^{d_u} \times D_v \times \Theta \to \mathbb{R}^{d_o}$$

and then define $\Psi_{\mathrm{PARA}} : \mathcal{U} \times \Theta \to \mathcal{V}$ by

$$\Psi_{\mathrm{PARA}}(u; \theta)(y) = \psi(Lu, y; \theta).$$

Remarks:

- ψ is a standard neural network between Euclidean spaces.
- By defining a collection of linear functions L in $\mathcal{U}^*$, evaluating this finite dimensional neural network at input (Lu, y), and varying over u in $\mathcal{U}$ and y in D_v we create an operator.

Fourier neural operator (FNO-Net)

Let R, Q denote standard finite dimensional neural networks

$$R : \mathbb{R}^{d_i} \times \Theta \to \mathbb{R}^{d_f}$$
$$Q : \mathbb{R}^{d_f} \times \Theta \to \mathbb{R}^{d_o},$$

where d_f is the number of channels used in FNO-Net, typically larger than d_i or d_o. We use R and Q to define operators $\mathcal{R}$ and $\mathcal{Q}$ by pointwise evaluation:

$$(\mathcal{R}u)(x, \theta_R) = R(u(x), \theta_R)$$
$$(\mathcal{Q}u)(x, \theta_Q) = Q(u(x), \theta_Q).$$

Here the operator $\mathcal{R}$ *lifts* the inputs to the channels and the operator $\mathcal{Q}$ *projects* the channels to the output.

Note that $\mathcal{R}u$ is well-defined in $C_{\text{per}}(D; \mathbb{R}^{d_f})$, the space of periodic continuous functions of dimension d_f, if $u \in \mathcal{U}$ and $\mathcal{U}$ is continuously embedded into $C_{\text{per}}(D; \mathbb{R}^{d_i})$.

Fourier neural operator (FNO)

We now define the l-th Fourier neural layer (FNL):

$$\mathcal{L}_l(v)(x,\theta) = \sigma\left(W_l v(x) + (\mathcal{K}v)(x;\gamma_l)\right),$$

where σ is an activation function, applied pointwise to $x \in D$; $W_l \in \mathbb{R}^{d_f \times d_f}$ is a matrix applied pointwise to $v(x)$; $\mathcal{K}$ is a parameterized non-local operator (variety of forms commonly used). One such is the FNO form:

$$\text{FNO}: \qquad (\mathcal{K}v)(x;\gamma) = \mathcal{F}^{-1}(P(\gamma)(\mathcal{F}v))(x).$$

Here $\mathcal{F}$ denotes the Fourier transform of a periodic function $v : D \to \mathbb{R}^{d_f}$ so that $\mathcal{F}v : \mathbb{Z}^d \to \mathbb{C}^{d_f}$, and $\mathcal{F}^{-1}$ denotes its inverse.

Remarks:

- For each point in the Fourier domain $\mathbb{Z}^d$, the action of $P(\gamma)$ is an element of $\mathbb{C}^{d_f \times d_f}$.
- FNO can be implemented using a FFT on a uniform lattice of pointwise evaluations of the functions v and $P(\gamma)\mathcal{F}(v)$. Only the first $k_{\max}$ modes are kept when evaluating $\mathcal{F}$.

Fourier neural operator (FNO)

We now define the l-th Fourier neural layer (FNL):

$$\mathcal{L}_l(v)(x,\theta) = \sigma\left(W_l v(x) + (\mathcal{K}v)(x;\gamma_l)\right),$$

where σ is an activation function, applied pointwise to $x \in D$; $W_l \in \mathbb{R}^{d_f \times d_f}$ is a matrix applied pointwise to $v(x)$; $\mathcal{K}$ is a parameterized non-local operator (variety of forms commonly used). One such is the FNO form:

$$\text{FNO}: \qquad (\mathcal{K}v)(x;\gamma) = \mathcal{F}^{-1}(P(\gamma)(\mathcal{F}v))(x).$$

Here $\mathcal{F}$ denotes the Fourier transform of a periodic function $v : D \to \mathbb{R}^{d_f}$ so that $\mathcal{F}v : \mathbb{Z}^d \to \mathbb{C}^{d_f}$, and $\mathcal{F}^{-1}$ denotes its inverse.

Remarks:

- For each point in the Fourier domain $\mathbb{Z}^d$, the action of $P(\gamma)$ is an element of $\mathbb{C}^{d_f \times d_f}$.
- FNO can be implemented using a FFT on a uniform lattice of pointwise evaluations of the functions v and $P(\gamma)\mathcal{F}(v)$. Only the first $k_{\max}$ modes are kept when evaluating $\mathcal{F}$.

Fourier neural operator (FNO)

Remarks (contd.):

- Here P reduces to a complex-valued $(k_{\max} \times d_f \times d_f)$-tensor, which is applied as an element of $\mathbb{C}^{d_f \times d_f}$ on each of the $k_{\max}$ Fourier modes.

Then the FNO-Net is defined as

$$\Psi_{\mathrm{FNO}}(u;\theta) = \mathcal{Q} \circ \mathcal{L}_L \circ \cdots \circ \mathcal{L}_2 \circ \mathcal{L}_1 \circ \mathcal{R}(u).$$

Here θ denotes the collection of hyperparameters θ_R and θ_Q along with $\{W_l, \gamma_l\}$ for each Fourier neural layer l. None of these are shared, but their choice is coupled through empirical risk minimization.

For simplicity, the layer width d_f is fixed, but this is not necessary.

Also, here Q and R are affine, but they could be taken as linear or nonlinear.

Finally, in this implementation the number of retained Fourier modes scales exponentially with the spatial dimension of the problem whereas PCA-Net and DeepONets do not have parameters that explicitly depend on this dimension.

Numerical studies

- Navier-Stokes equation: the map between forcing and vorticity field at a later time is learned.
- Helmholtz equation: the map between the (inhomogeneous) wavespeed field and the disturbance field (solution) is learned.
- Structural mechanics equation: the map between an applied boundary load and the interior von Mises stress field is learned.
- Advection equation: the solution operator from initial condition to solution at a later time is learned.

Numerical studies

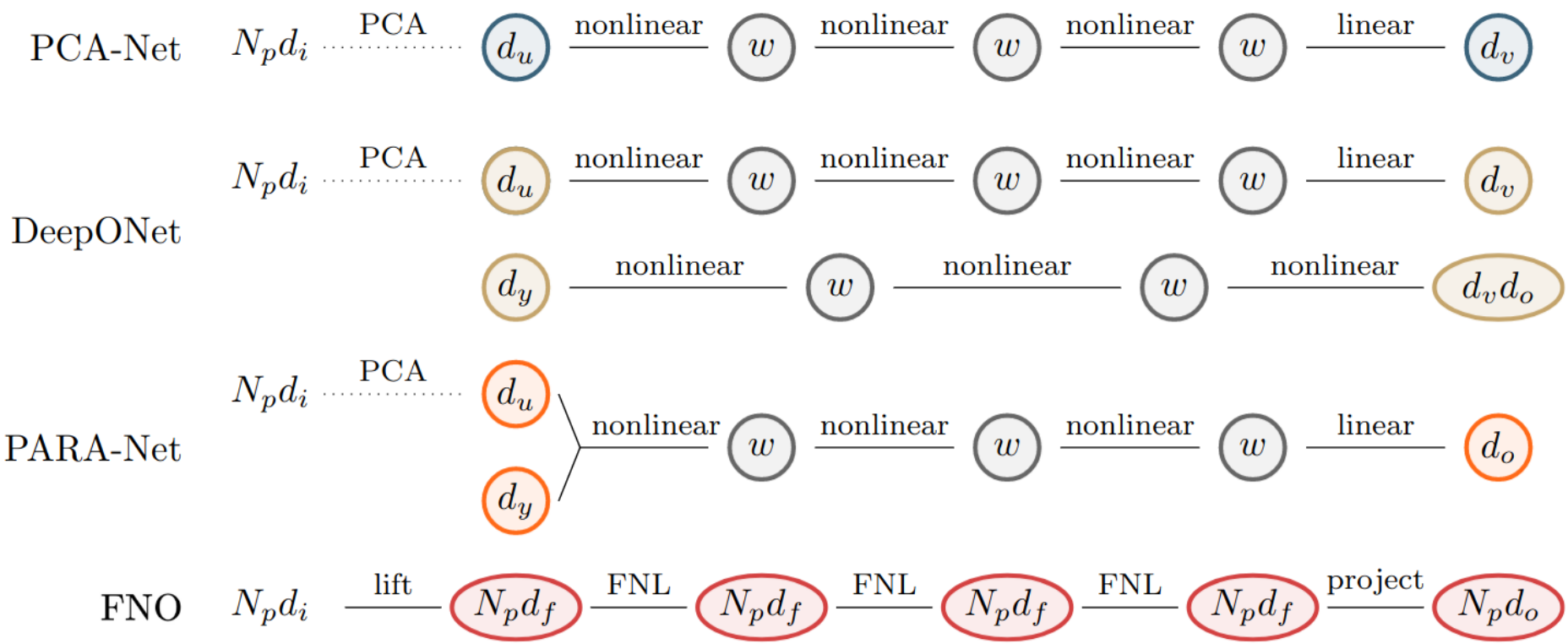

Figure 4.1: Schematic of neural network architectures in numerical experiments. Circles represent layers; the width of each layer is given in the circle. Edges represent transformations between layers; the type of transformation between each layer is noted above each edge. Nonlinear and linear transformations are standard fully-connected layers; the lift and project layers are defined in (3.1); the Fourier Neural Layer (FNL) is defined in (3.2).

Numerical studies

Architecture	Evaluation cost	Cost scaling
PCA-Net	$d_u(2N_p d_i - 1) + 2d_u w + 4w^2 + 2d_v w + 3w + (2d_v - 1)N_p d_o$	$\mathcal{O}(N_p + w^2)$
DeepONet	$d_u(2N_p d_i - 1) + 2d_u w + 4w^2 + 2d_v w + 3w + (2d_v - 1)N_p d_o$	$\mathcal{O}(N_p + w^2)$
PARA-Net	$d_u(2N_p d_i - 1) + [2(d_u + d_y)w + 4w^2 + 2wd_o + 3w]N_p$	$\mathcal{O}(N_p w^2)$
FNO	$2N_p d_f(d_i + d_o) + 3(10d_f N_p log(N_p) + k_{max}(2d_f^2 - d_f) + 2d_f^2 N_p)$	$\mathcal{O}(d_f N_p log(N_p) + N_p d_f^2)$

Table 4.1: Evaluation cost of the four neural network architectures considered in this work.

Architecture	Parameter complexity
PCA-Net	$2w^2 + w(d_u + d_v) + 3w + d_v$
DeepONet	$4w^2 + w(d_u + d_v + d_y + d_v d_o) + 6w + d_v + d_v d_o$
PARA-Net	$2w^2 + w(d_o + d_u + d_y) + 3w + d_o$
FNO	$d_f d_i + d_f + d_f d_o + d_o + 3(d_f^2 + d_f^2 k_{max})$

Table 4.2: Parameter complexity of the four neural network architectures considered in this work in terms of input and output space dimensions as well as network size parameter w or d_f.

Navier-Stokes equation

Formulation: Consider the vorticity-stream $(\omega - \psi)$ formulation of the incompressible Navier-Stokes equations on a two-dimensional periodic domain, $D = D_u = D_v = [0, 2\pi]^2$:

$$\frac{\partial \omega}{\partial t} + (v \cdot \nabla)\omega - \nu \Delta \omega = f',$$
$$\omega = -\Delta \psi \qquad \int_D \psi = 0,$$
$$v = \left(\frac{\partial \psi}{\partial x_2}, -\frac{\partial \psi}{\partial x_1} \right).$$

Again, the map from forcing f' to vorticity field ω at time $t = T$ is the target. Also, here f' is assumed to be a centered Gaussian with covariance $C = (-\Delta + \tau^2)^{-d}$.

- $-\Delta$ is the Laplacian on D subject to periodic boundary conditions on the space of spatial-mean zero functions,
- $\tau = 3$ is the inverse length scale of the random field
- $d = 4$ determines its regularity.
- $T = 10$ and $\nu = 0.025$

Other details: solved using pseudo-spectral method (64×64 grid), Orszag 2/3-Rule to eliminate aliasing error, 42^2 Fourier modes, and time-integration is done using Crank-Nicolson with $\Delta t = 10^{-3}$.

Navier-Stokes equation

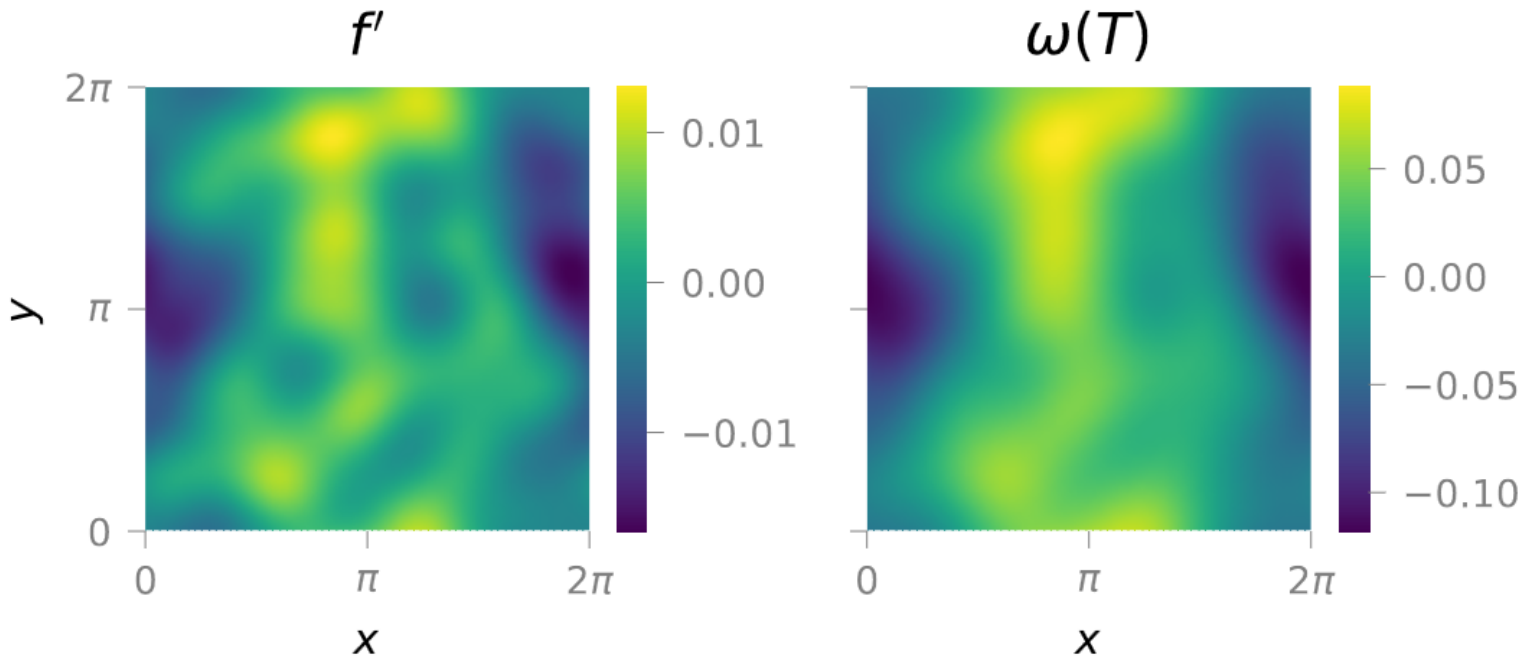

Figure 4.2: Navier-Stokes problem: sample input and output functions (left and right, respectively).

Navier-Stokes equation

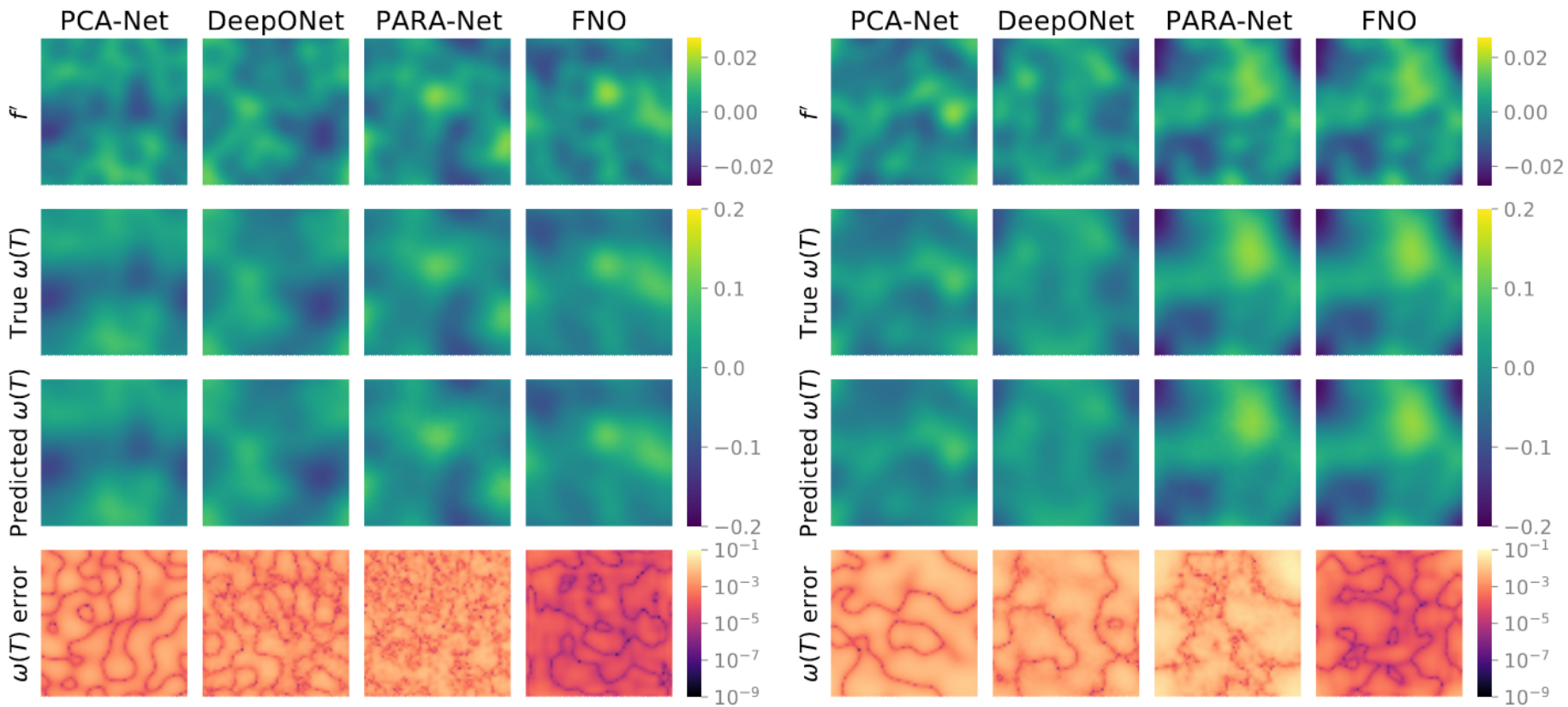

Figure 4.3: Navier-Stokes test problem: learned model vorticity predictions for inputs resulting in **median (left)** and **largest (right)** test errors for networks of size $w = 128$ / $d_f = 16$ trained on $N = 10000$ data.

Helmholtz equation

Formulation: Consider domain $\Omega = D_u = D_v = [0,1]^2$. Given frequency $\omega = 10^3$ and wavespeed field $c : \Omega \to \mathbb{R}$, the excitation field $u : \Omega \to \mathbb{R}$ solves

$$\begin{aligned} \left(-\Delta - \frac{\omega^2}{c^2(x)}\right) u &= 0 && \text{in } \Omega \\ \frac{\partial u}{\partial n} &= 0 && \text{on } \partial\Omega_1, \partial\Omega_2, \partial\Omega_4 \\ \frac{\partial u}{\partial n} &= u_N && \text{on } \partial\Omega_3. \end{aligned}$$

Note: the Neumann boundary condition imposed on $\partial\Omega$ is non-zero only on the top edge. Throughout, u_N is fixed at $\mathbf{1}_{\{0.35 \le x \le 0.65\}}$. The wavespeed field is assumed to be $c(x) = 20 + \tanh(\tilde{c}(x))$, where $\tilde{c}$ is a centered Gaussian

$$\tilde{c} \sim \mathcal{N}(0, C) \qquad \text{and} \qquad C = (-\Delta + \tau^2)^{-d}.$$

Here $-\Delta$ denotes the Laplacian on D_u subject to homogeneous Neumann boundary conditions on the space of spatial-mean zero functions, $d = 2$ and $\tau = 3$.

Recall we're interested in the map from the wavespeed field c to the solution u.

Also, this is solved using the finite element method on a 100×100 grid.

Helmholtz equation

Formulation: Consider domain $\Omega = D_u = D_v = [0,1]^2$. Given frequency $\omega = 10^3$ and wavespeed field $c : \Omega \to \mathbb{R}$, the excitation field $u : \Omega \to \mathbb{R}$ solves

$$\left(-\Delta - \frac{\omega^2}{c^2(x)}\right) u = 0 \qquad \text{in } \Omega$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega_1, \partial\Omega_2, \partial\Omega_4$$

$$\frac{\partial u}{\partial n} = u_N \qquad \text{on } \partial\Omega_3.$$

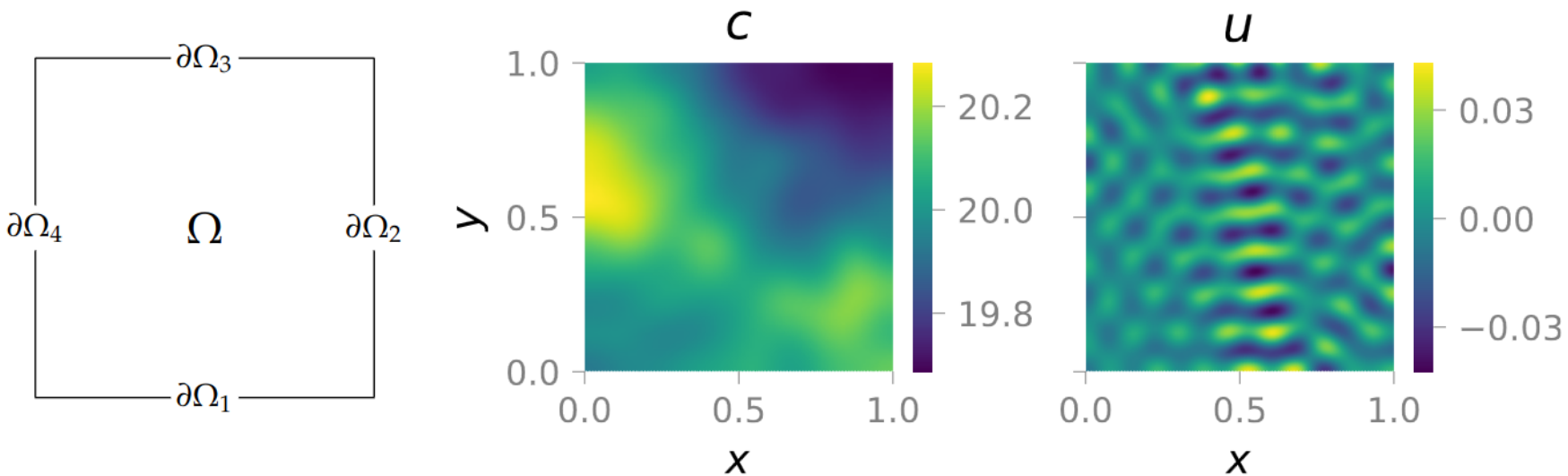

Figure 4.4: Helmholtz test problem: **Left:** schematic of unit domain with labeled boundaries. **Center:** Sample input wave speed field. **Right:** Sample output disturbance field.

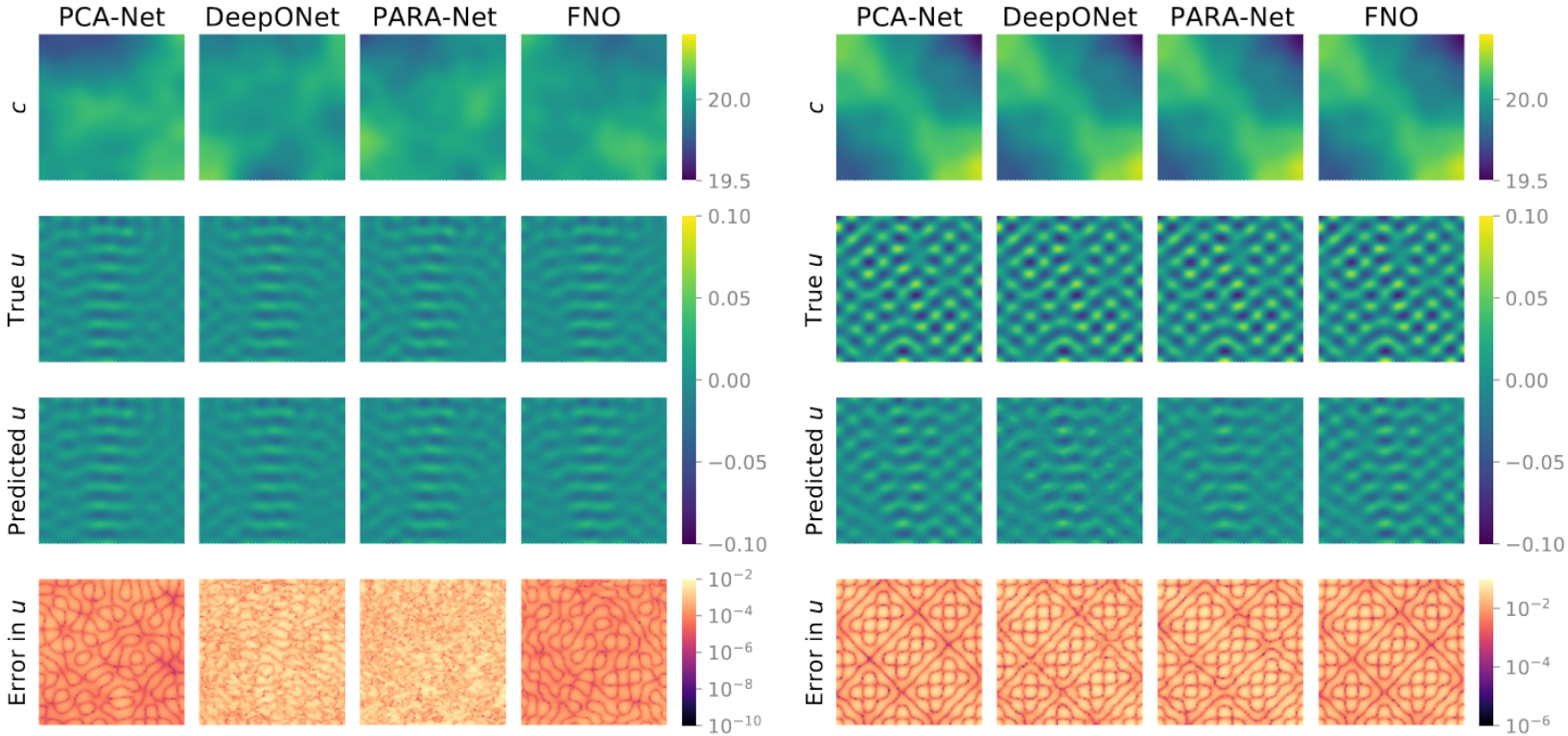

Figure 4.5: Helmholtz test problem: learned model predictions for inputs resulting in **median (left)** and **largest (right)** test errors for networks of size $w = 128$ / $d_f = 16$ trained on $N = 10000$ data.

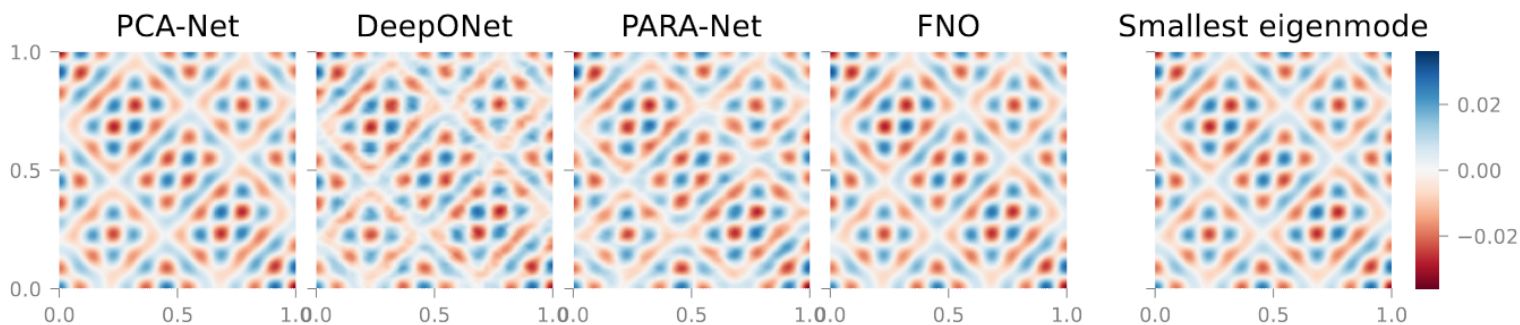

Figure 4.6: Helmholtz test problem largest test error case: normalized differences between learned model predictions and true solution (left four panels) and comparison to normalized smallest eigenmode of the Helmholtz operator for this test case.

Structural mechanics equation

Formulation: Consider the governing equation of an elastic solid undergoing infinitesimal deformations:

$$\begin{aligned} \Delta \cdot \sigma &= 0 && \text{in } \Omega, \\ u &= \bar{u} && \text{on } \Gamma_u, \\ \sigma \cdot n &= \bar{t} && \text{on } \Gamma_t, \end{aligned}$$

u is the displacement vector, σ is the (Cauchy) stress tensor, and Ω is the domain.

The prescribed displacement $\bar{u}$ and the surface traction $\bar{t}$ are imposed on Γ_u and Γ_t, respectively, with the outward unit normal n, where $\Gamma_u \cap \Gamma_t = \emptyset$ and $\Gamma_u \cup \Gamma_t = \partial\Omega$.

To solve for displacement u, also need constitutive model, mapping deformation gradient to stress. The matrix is made of incompressible Rivlin-Saunders material with density $\rho = 0.8$ and energy density function parameters $C_1 = 1.863 \times 10^5$, $C_2 = 9.79 \times 10^3$ and the cylindrical fiber at the center is made of linear elastic material with density $\rho = 3.2$, Young's modulus $E = 4 \times 10^6$ and Poisson ratio $\nu = 0.35$.

Structural mechanics equation

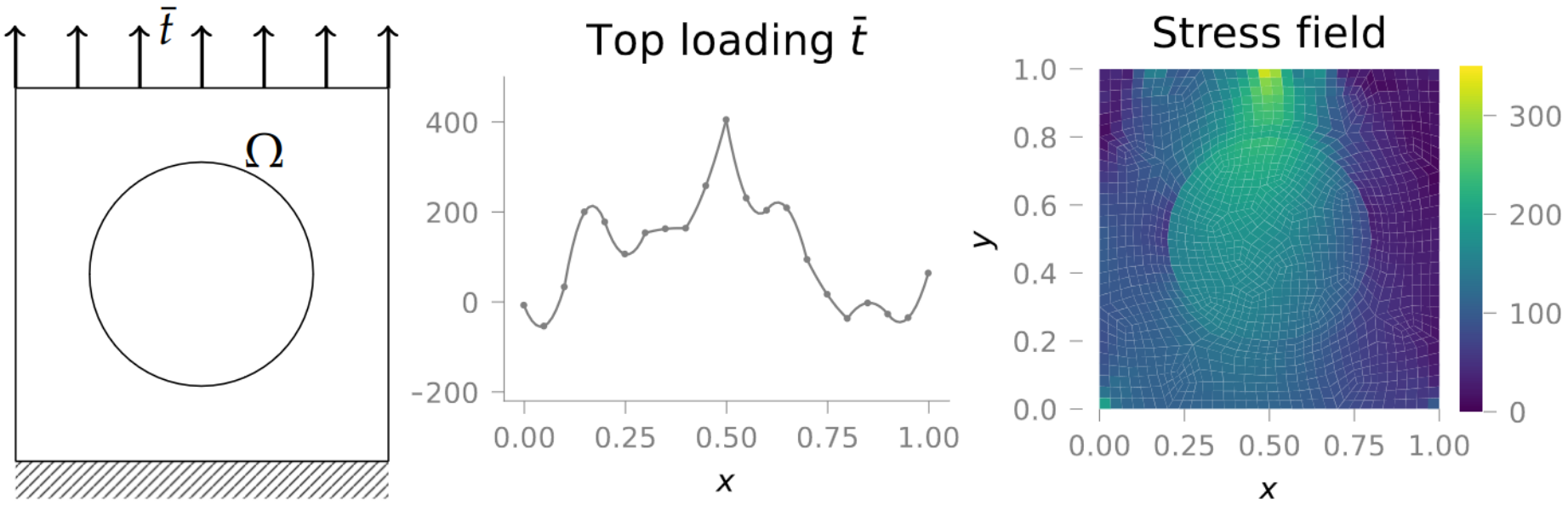

Figure 4.7: Solid mechanics test problem. Schematic of fiber-reinforced material and loading (**left**), sample load drawn from Gaussian distribution (**center**), and corresponding von Mises stress field (**right**).

Other details: data generated using `NNFEM` library, mesh consists of 189 quadratic quadrilateral elements, top edge is discretized by 10 quadratic elements and hence 21 points. Inputs are the load on the 21 points, outputs are the stress field on Gaussian quadrature points (9×189).

Since FNO operates on uniform grids, and requires input and output data to have the same dimensions, the stress field is interpolated on a 41×41 grid via radial basis function interpolation, and the load is interpolated on a 41 grid and extruded in the y direction.

Structural mechanics equation

Recall we are interested in the map from the one-dimensional load $\bar{t}$ to the von Mises stress field τ_{vM} on the two dimensional domain Ω. The load $\bar{t}$ is drawn from a Gaussian random field with mean 100 and covariance $400^2 C$ with

$$\bar{t} \sim \mathcal{N}(100, 400^2 C) \qquad \text{and} \qquad C = (-\Delta + \tau^2)^{-d}.$$

Here $-\Delta$ denotes the Laplacian on Ω subject to homogeneous Neumann boundary conditions on the space of spatial-mean zero functions, $\tau = 3$ denotes the inverse length scale of the random field and $d = 1$ determines its regularity.

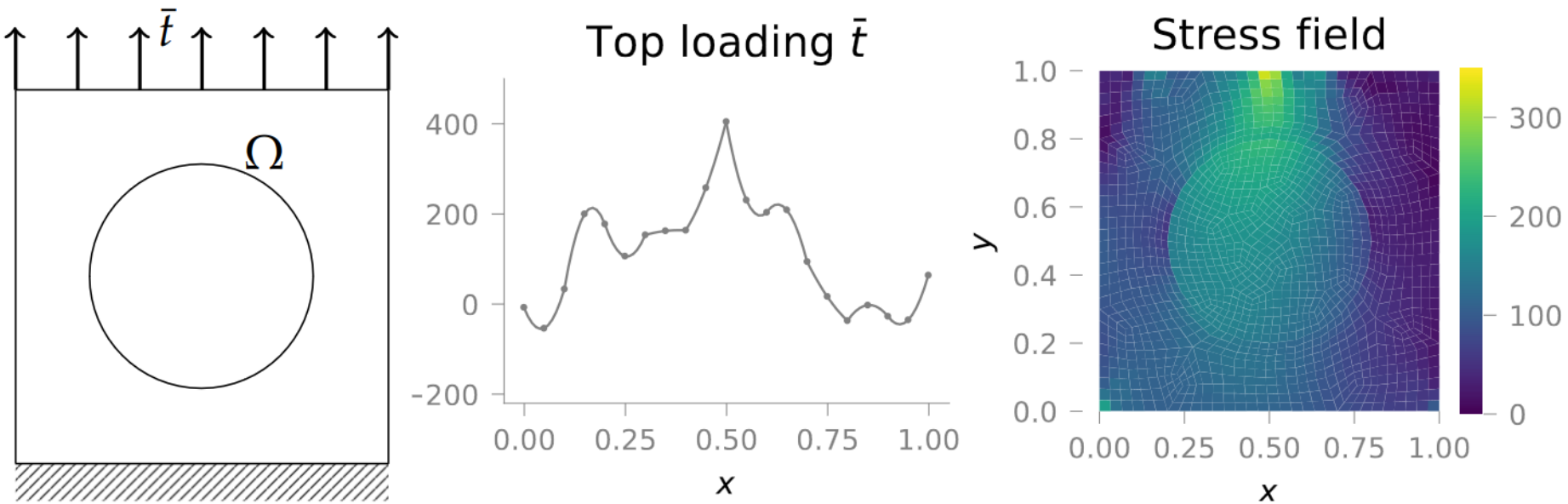

Figure 4.7: Solid mechanics test problem. Schematic of fiber-reinforced material and loading (**left**), sample load drawn from Gaussian distribution (**center**), and corresponding von Mises stress field (**right**).

Structural mechanics equation

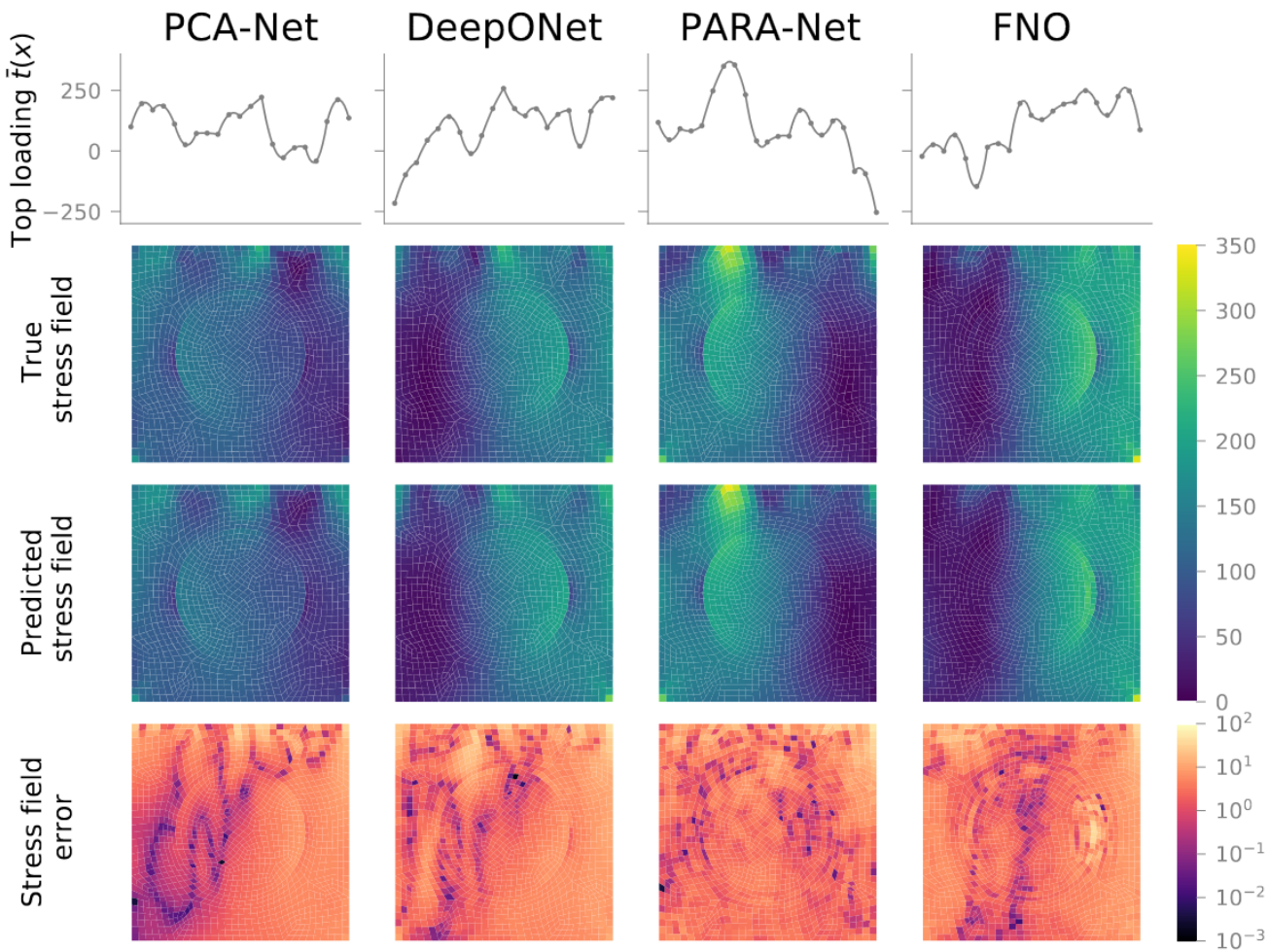

Figure 4.8: Structural mechanics test problem: learned model stress field predictions for inputs resulting in **median** test errors for networks of size $w = 128$ / $d_f = 16$ trained on $N = 10000$ data.

Structural mechanics equation

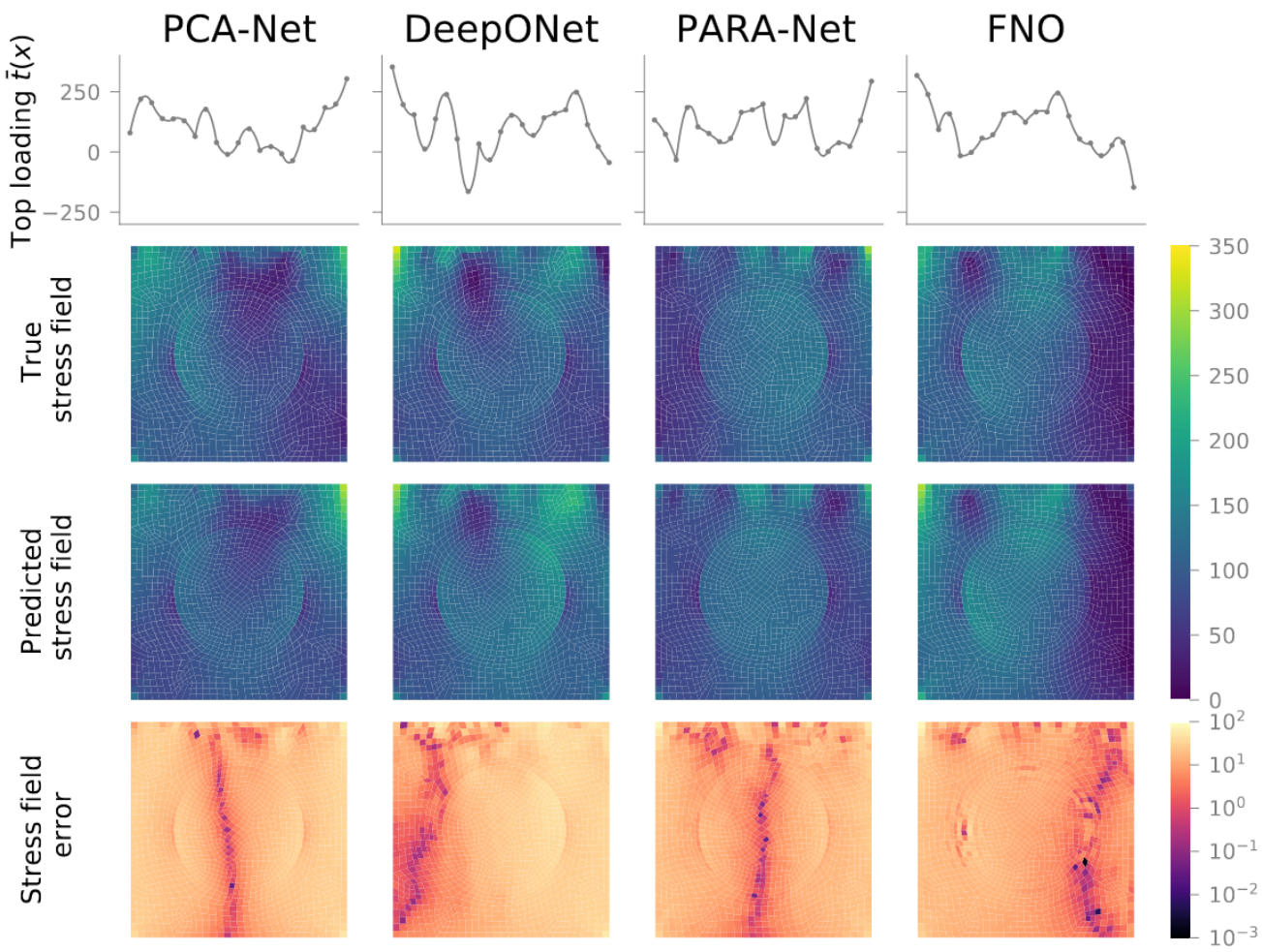

Figure 4.9: Structural mechanics test problem: learned model stress field predictions for inputs resulting in **largest** test errors for networks of size $w = 128$ / $d_f = 16$ trained on $N = 10000$ data.

Advection equation

Formulation: The 1D advection equation in $\Omega = [0, 1)$ is

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \qquad x \in \Omega,$$
$$u(0) = u_0,$$

where $c = 1$ is the constant advection speed, and periodic boundary conditions are imposed.

Recall, we are interested in the map from the initial condition u_0 to the solution $u(\,\cdot\,, T)$ at time $T = 0.5$. The initial condition is $u_0 = -1 + 2\mathbf{1}_{\{\tilde{u}_0 \geq 0\}}$, where $\tilde{u}_0$ is a centered Gaussian

$$\tilde{u}_0 \sim \mathcal{N}(0, C) \qquad \text{and} \qquad C = (-\Delta + \tau^2)^{-d}.$$

Here $-\Delta$ denotes the Laplacian on Ω subject to periodic conditions on the space of spatial-mean zero functions, $\tau = 3$ denotes the inverse length scale of the random field and $d = 2$ determines the regularity of $\tilde{u}_0$.

Advection equation

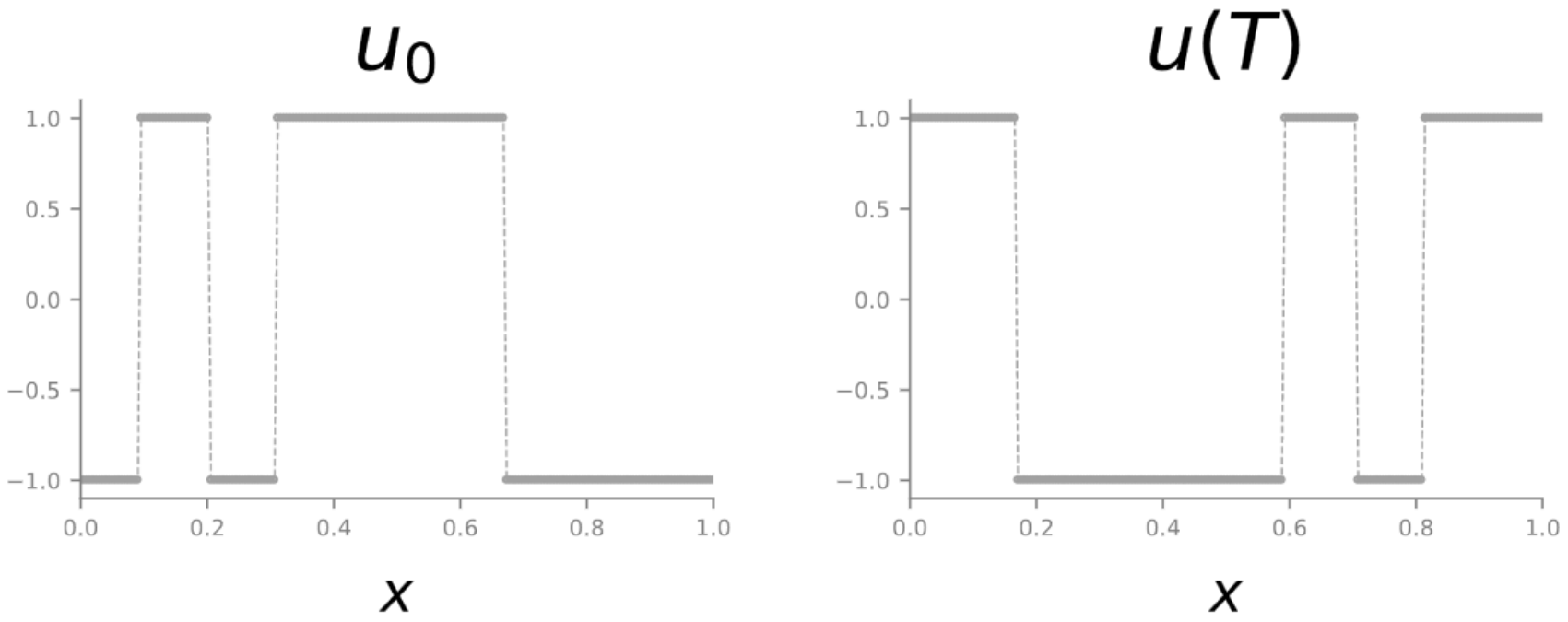

Figure 4.10: Sample input/output functions for the advection equation, left: u_0, right: $u(T)$.

Advection equation

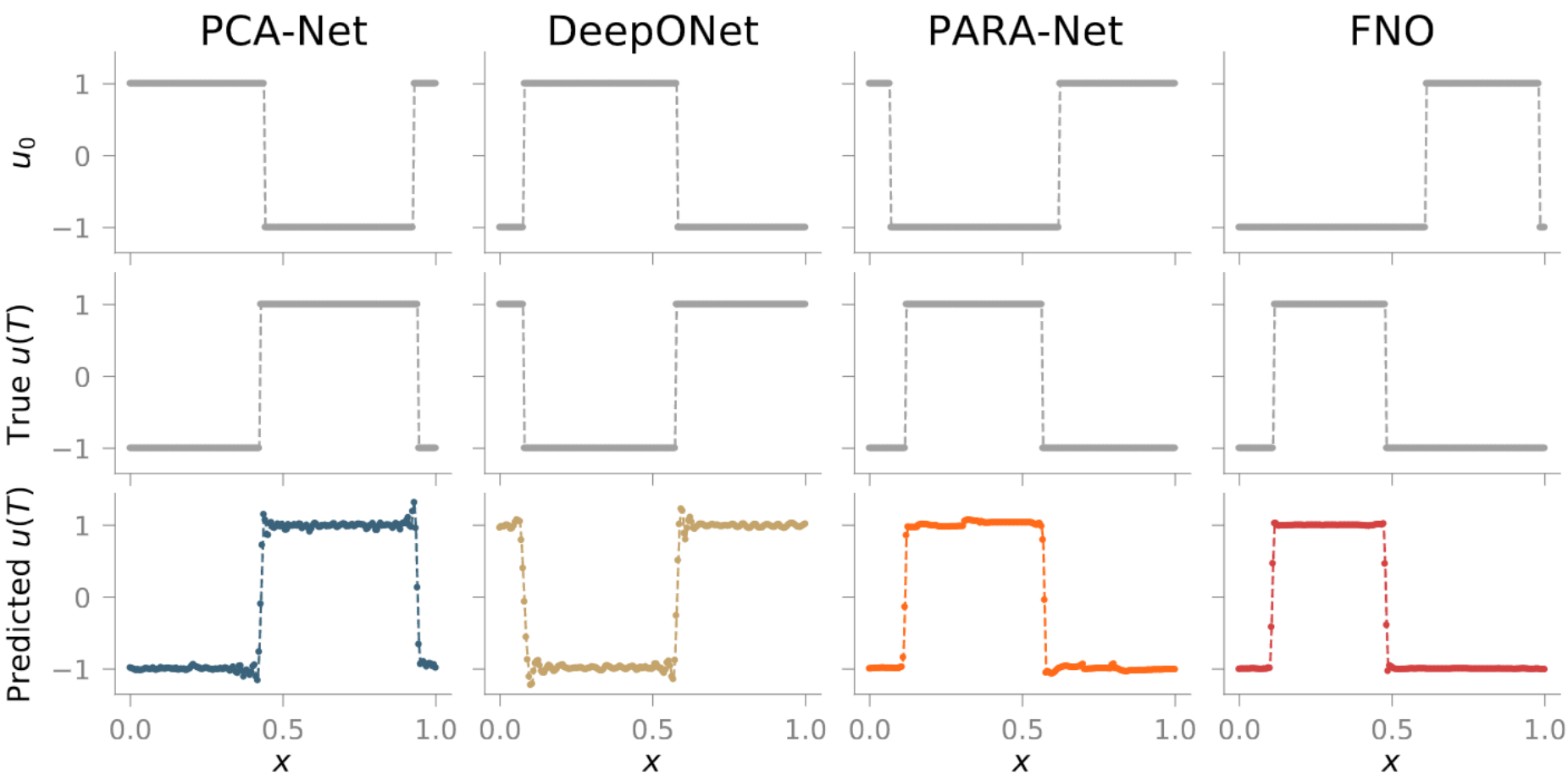

Figure 4.11: Advection test problem: learned model output predictions for inputs resulting in **median (top)** and **largest (bottom)** test errors for networks of size $w = 128$ / $d_f = 16$ trained on $N = 10000$ data.

Advection equation

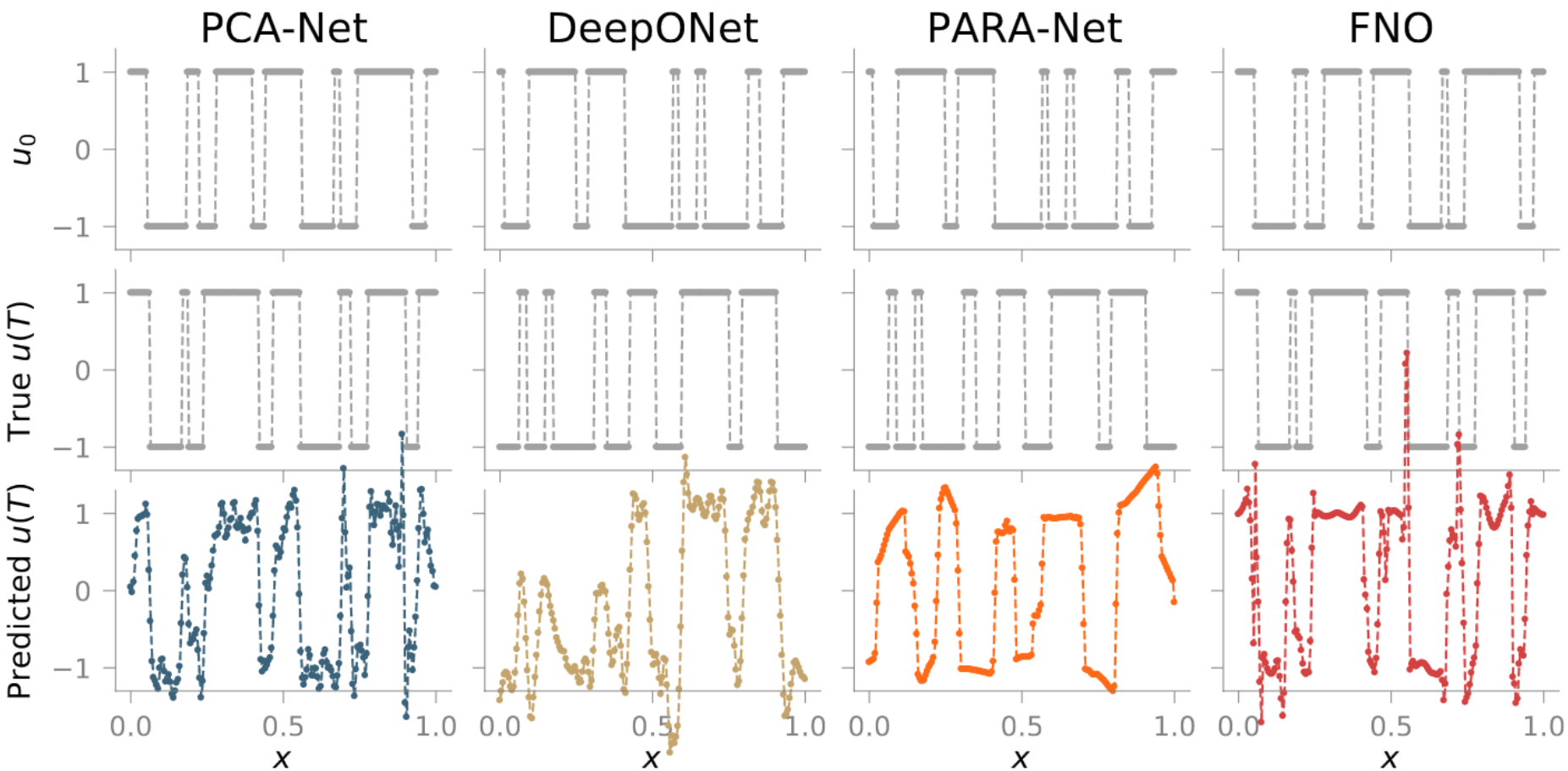

Figure 4.11: Advection test problem: learned model output predictions for inputs resulting in **median (top)** and **largest (bottom)** test errors for networks of size $w = 128$ / $d_f = 16$ trained on $N = 10000$ data.

Accuracy vs. training cost

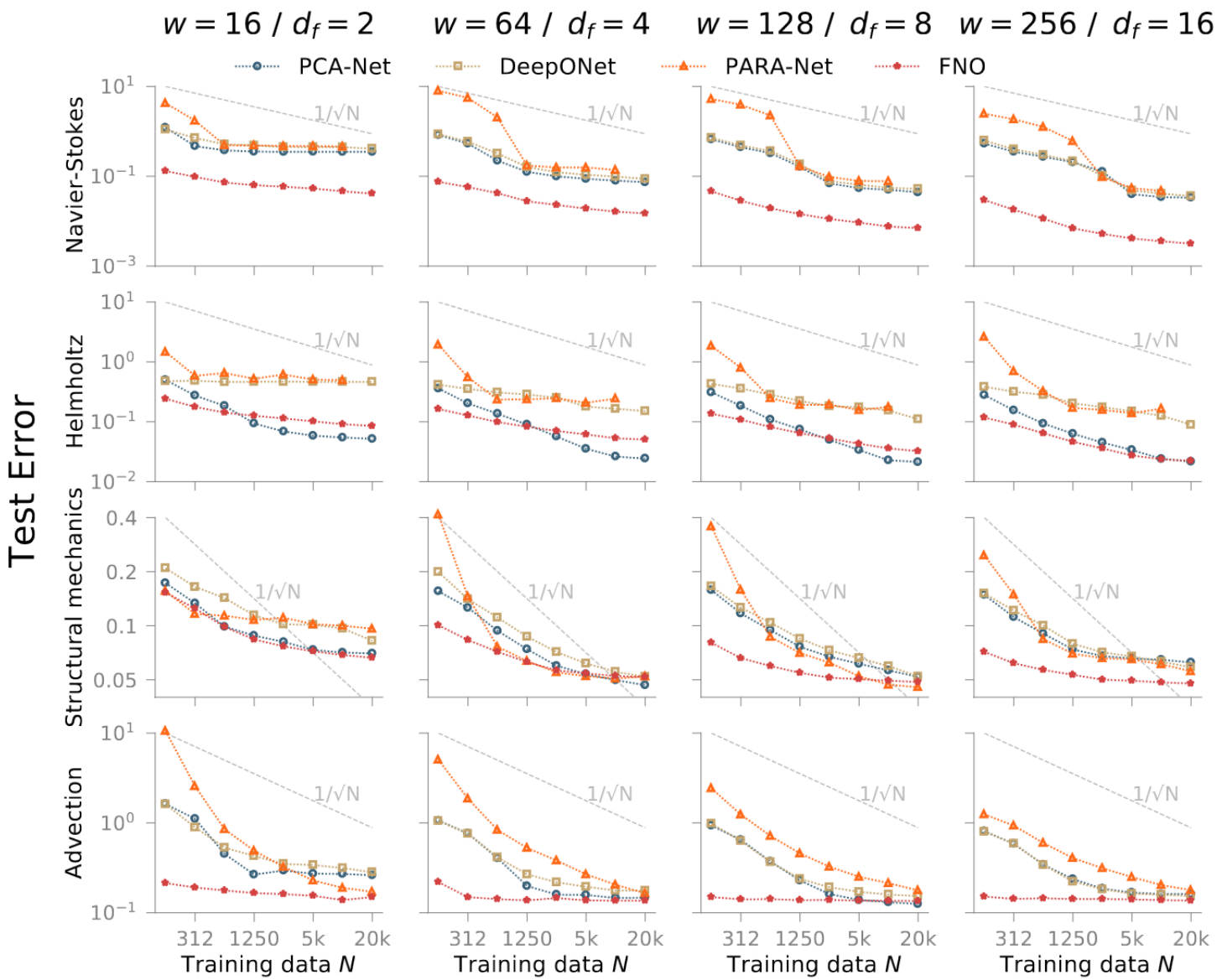

Figure 4.12: Test error vs. training data amount N for Navier-Stokes, Helmholtz, structural mechanical, and advection problems (top to bottom). Network size increases left to right.

Accuracy vs. network size

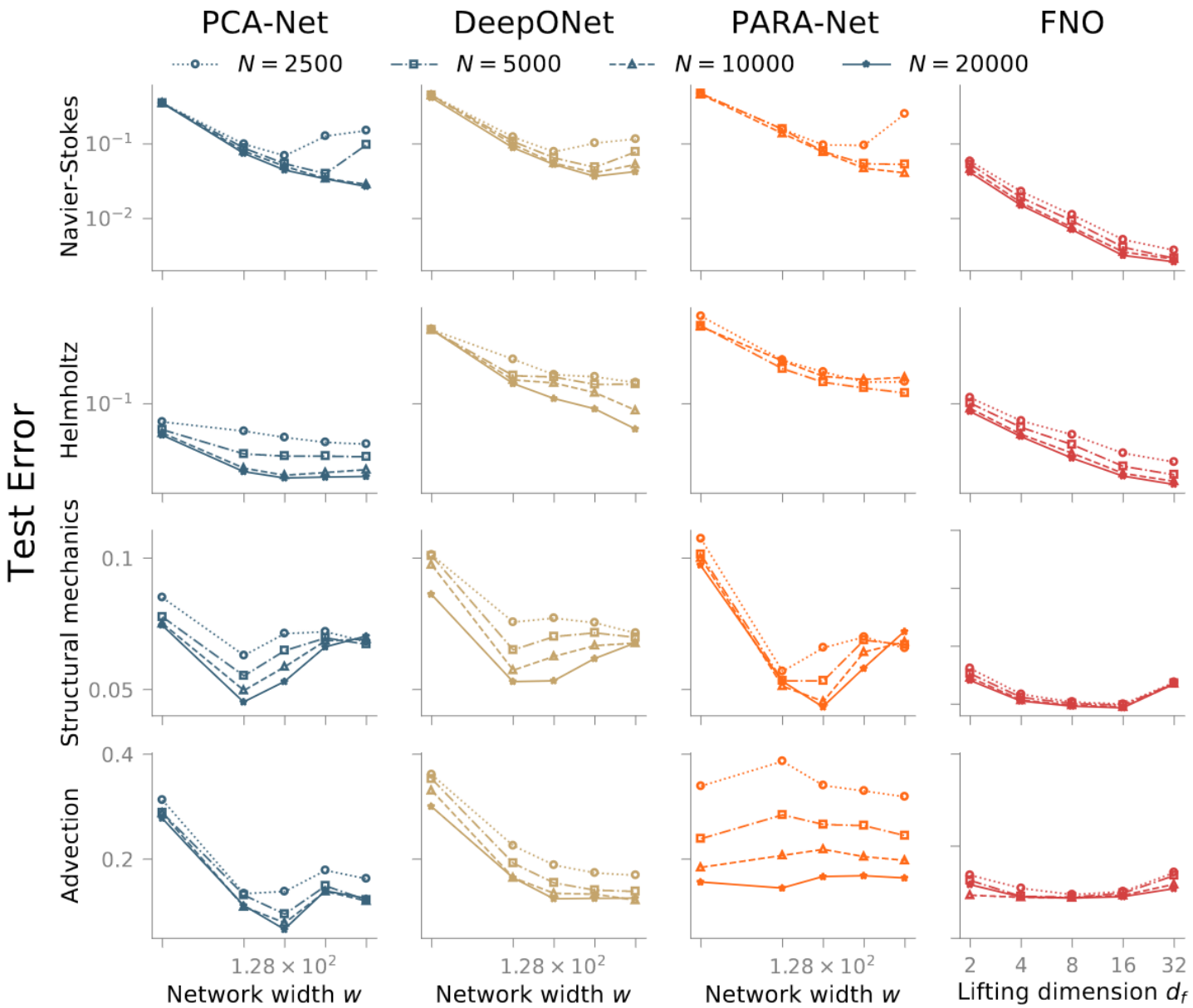

Figure 4.13: Test error vs. network size, as measured by internal layer width w for PCA-Net, DeepONet, and PARA-Net, and as measured by number of channels d_f for FNO. Different lines correspond to different training

Further comparisons

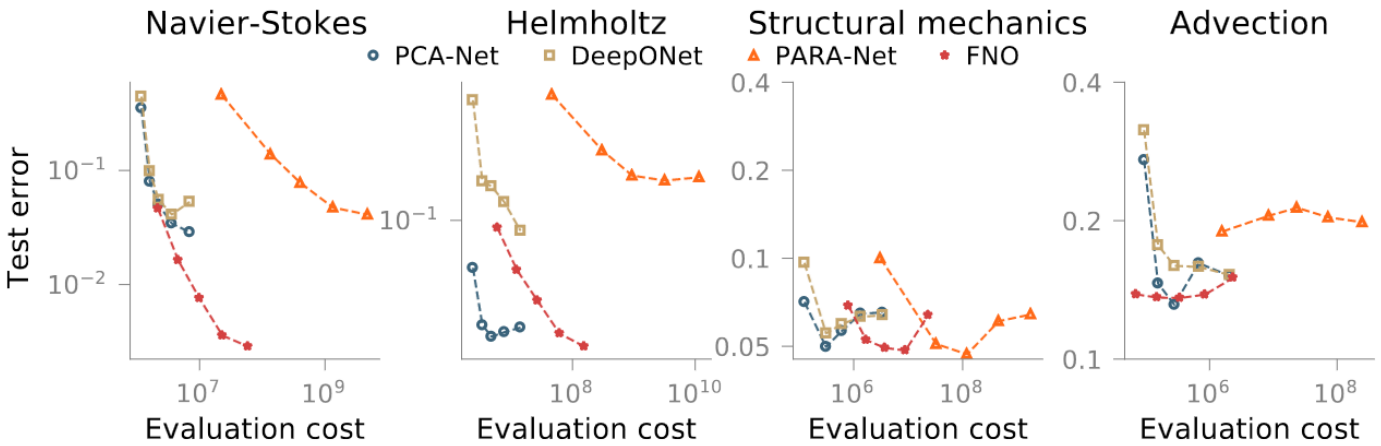

Figure 4.14: Test error vs. evaluation cost for all four test problems for fixed training data volume $N = 10000$. See Figure C.1 for error vs. evaluation cost curves for all training data volumes tested.

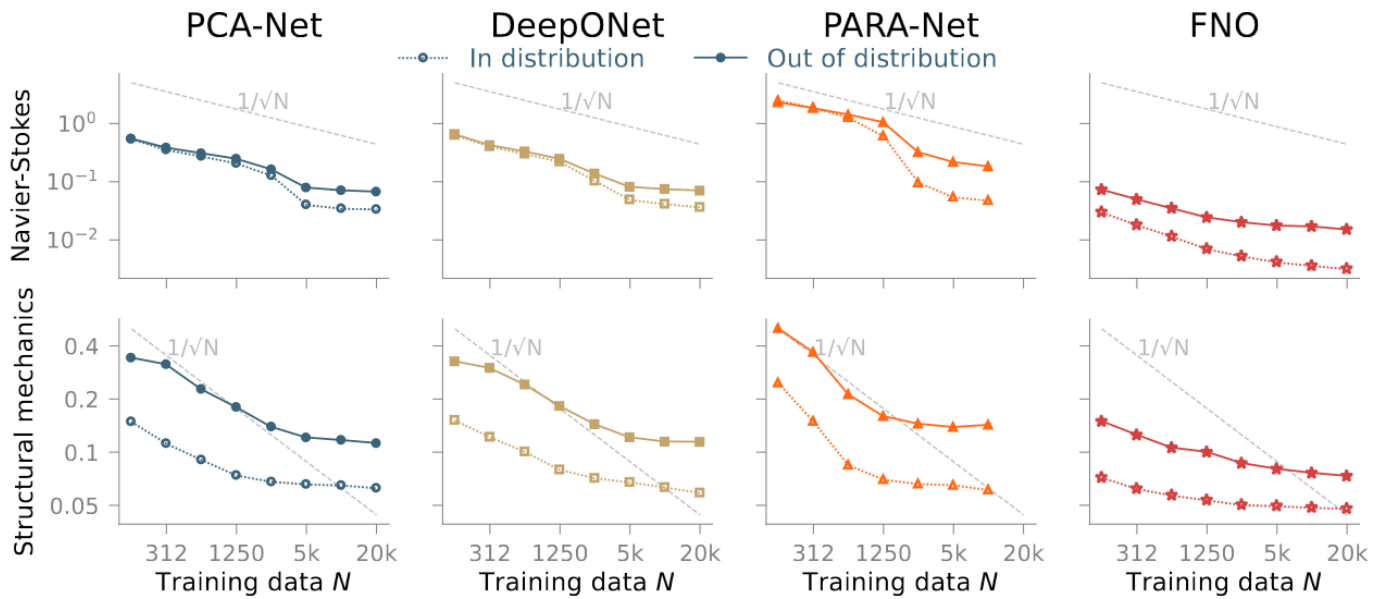

Figure 4.15: In- vs. out-of-distribution test error comparison for Navier-Stokes (top) and structural mechanics (bottom) test problems for largest networks tested ($w = 256$, $d_f = 16$).

Demonstration

https://colab.research.google.com/drive/18gbQ9z-4ZTSb6LYcf393vEXyRrkxkXK5

References

"The Cost-Accuracy Trade-Off in Operator Learning with Neural Networks" by Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart.

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