A variationally mimetic operator network

Deep Ray
Department of Aerospace & Mechanical Engineering
University of Southern California

Email: deepray@usc.edu
Website: deepray.github.io

Jointly with Dhruv Patel, Michael Abdelmalik, Assad A. Oberai & Thomas J. R. Hughes

Supported by ARO grant W911NF2010050, NRL grant N00173-19-P-1119, AIER-USC, CARC-USC

USC Viterbi
School of Engineering

CMX Seminar, Caltech
November 16, 2022

Motivation

- PDEs used to model physical processes.
- Typically solved using FDM, FVM, FEM, ...
- Many-query applications need many calls to the solver: e.g.
  - Uncertainty quantification
  - PDE-constrained optimization
  - Inverse problems
- Single solve expensive for large-scale applications $\rightarrow$ multiple solves $\mathcal{O}(10^5)$ prohibitively expensive!
- Need to design efficient, robust surrogates.
- Recent interest in deep learning-based surrogates $\rightarrow$ focus of this talk!

- Operator learning using networks
- **Var**iationally **Mi**metic **O**perator **N**etwork (VarMiON)
- Error estimates
- Numerical results
- Conclusion

Consider the generic PDE:

$$\mathcal{L}(u(\boldsymbol{x})) = f(\boldsymbol{x}), \qquad \forall\, \boldsymbol{x} \in \Omega$$

- Solution approximated by a network $\widehat{u}(\boldsymbol{x};\boldsymbol{\psi})$ with parameters $\boldsymbol{\psi}$.
- Minimize PDE residual at collocation points (Lagaris et al., 2000): Solve

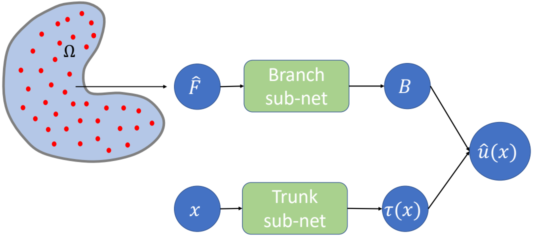
$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \Pi(\boldsymbol{\psi}), \quad \Pi(\boldsymbol{\psi}) = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{L}(\widehat{u}(\boldsymbol{x}_i;\boldsymbol{\psi})) - f(\boldsymbol{x}_i)\|^2$$

- Rediscovered as Physics Informed Neural Nets (PINNs) with deeper structures (Raissi et al., 2019).
- However, solve one instance of the PDE – needs to be retrained if f changes.

We are interested in approximating the solution operator

$$\mathcal{S} : \mathcal{F} \longrightarrow \mathcal{V}$$
$$f \mapsto u(.; f)$$

- Sample input functions at sensor nodes and generate solution at any $\boldsymbol{x}$ (Chen & Chen, 1995).
- Extended using deep networks as DeepONets (Lu et al., 2021).
- Sub-nets for learned basis functions (trunk), & coefficients (branch).
- Supervised learning algorithm.



We are interested in approximating the solution operator

$$\begin{aligned} \mathcal{S} : \mathcal{F} &\longrightarrow \mathcal{V} \\ f &\mapsto u(.;f) \end{aligned}$$

- Neural operators – an alternate strategy to approximate $\mathcal{S}$
- Extension of NNs to functions.
- Philosophy – "formulate the algorithm in the infinite dimensional setting and then discretize".
- Several variants: FNO (Li et al., 2020), Graph Kernel Net (Li et al., 2020), PCA-NET (Bhattacharya et al., 2021), ...

This talk: Operator Networks that mimic (approximate) variational form* of the PDE.

* Variationally Mimetic Operator Networks; Patel, R, Abdelmalik, Hughes, Oberai; 2022 (arXiv:2209.12871)

Problem Statement

- Consider a generic linear elliptic PDE:

$$\begin{aligned} \mathcal{L}(u(\boldsymbol{x});\theta(\boldsymbol{x})) &= f(\boldsymbol{x}), \qquad && \forall\, \boldsymbol{x} \in \Omega, \\ \mathcal{B}(u(\boldsymbol{x});\theta(\boldsymbol{x})) &= \eta(\boldsymbol{x}), && \forall\, \boldsymbol{x} \in \Gamma_\eta, \\ u(\boldsymbol{x}) &= 0, && \forall\, \boldsymbol{x} \in \Gamma_g, \end{aligned}$$

  where $f \in \mathcal{F} \subset L^2(\Omega)$, $\eta \in \mathcal{N} \subset L^2(\Gamma_\eta)$, $\theta \in \mathcal{T} \subset L^\infty(\Omega)$.

- The variational formulation: find $u \in \mathcal{V} \subset H^1_g$ such that $\forall\, w \in \mathcal{V}$,

$$a(w,u;\theta) = (w,f) + (w,\eta)_{\Gamma_\eta}.$$

- The solution operator is

$$\begin{aligned} \mathcal{S} : \mathcal{X} = \mathcal{F} \times \mathcal{T} \times \mathcal{N} &\longrightarrow \mathcal{V} \subset H^1_g \\ (f,\theta,\eta) &\mapsto u(.;f,\theta,\eta) \end{aligned}$$

  This mapping can be non-linear in θ.

Discrete weak formulation

- Evaluate approximate solution in finite-dimensional space $\mathcal{V}^h = \text{span}\{\phi_i(\boldsymbol{x}) : 1 \leq i \leq q\}$.

- Discrete weak formulation: find $u^h \in \mathcal{V}^h$ such that $\forall\ w^h \in \mathcal{V}^h$,

$$a(w^h, u^h; \theta^h) = (w^h, f^h) + (w, \eta^h)_{\Gamma_\eta}.$$

- Any function $v^h \in \mathcal{V}^h$ can be written as

$$v^h(\boldsymbol{x}) = \boldsymbol{V}^\top \boldsymbol{\Phi}(\boldsymbol{x}), \quad \boldsymbol{V} = (v_1, \cdots, v_q)^\top, \quad \boldsymbol{\Phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \cdots, \phi_q(\boldsymbol{x}))^\top.$$

  Plugging this into discrete weak form gives...

Discrete weak form

- Linear system of equations,

$$\boldsymbol{K}(\theta^h)\boldsymbol{U} = \boldsymbol{M}\boldsymbol{F} + \widetilde{\boldsymbol{M}}\boldsymbol{N}$$

  where the matrices are given by

$$K_{ij}(\theta^h) = a(\phi_i, \phi_j; \theta^h), \quad M_{ij} = (\phi_i, \phi_j), \quad \widetilde{M}_{ij} = (\phi_i, \phi_j)_{\Gamma_\eta} \quad 1 \leq i, j \leq q.$$

- Discrete solution operator is

$$\mathcal{S}^h : \mathcal{X}^h = \mathcal{F}^h \times \mathcal{T}^h \times \mathcal{N}^h \longrightarrow \mathcal{V}^h$$
$$(f^h, \theta^h, \eta^h) \mapsto u^h(.; f^h, \theta^h, \eta^h) = \boldsymbol{B}(f^h, \eta^h, \theta^h)^\top \boldsymbol{\Phi}$$

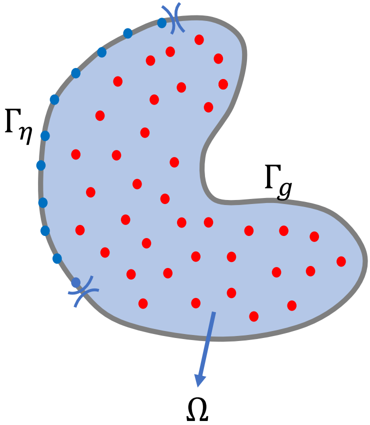
  where

$$\boldsymbol{B}(f^h, \theta^h, \eta^h) = \boldsymbol{U} = \boldsymbol{K}^{-1}(\theta^h)(\boldsymbol{M}\boldsymbol{F} + \widetilde{\boldsymbol{M}}\boldsymbol{N}).$$

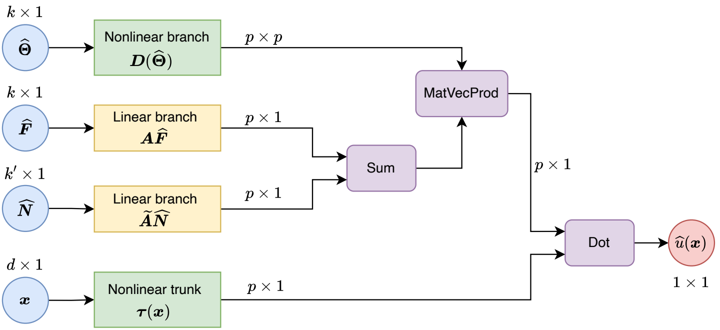
VarMiON will mimic this structure!

Evaluate (f, θ, η) at some fixed **sensor nodes** to get the discrete sample vectors

$$\widehat{\boldsymbol{F}} = (f(\widehat{\boldsymbol{x}}_1), \cdots, f(\widehat{\boldsymbol{x}}_k))^\top, \quad \widehat{\boldsymbol{\Theta}} = (\theta(\widehat{\boldsymbol{x}}_1), \cdots, \theta(\widehat{\boldsymbol{x}}_k))^\top, \quad \widehat{\boldsymbol{N}} = (\eta(\widehat{\boldsymbol{x}}^b_1), \cdots, \eta(\widehat{\boldsymbol{x}}^b_{k'}))^\top$$

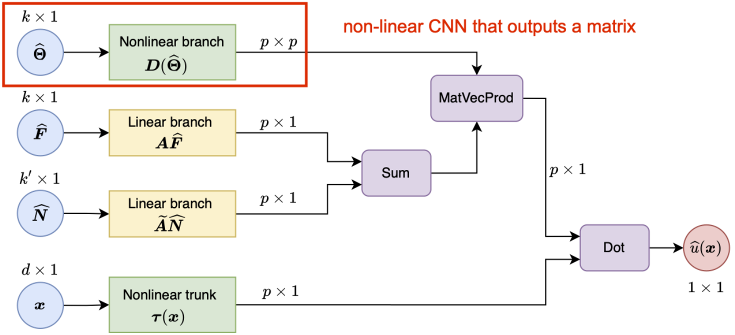


The network is comprises several sub-networks with latent dimension p:



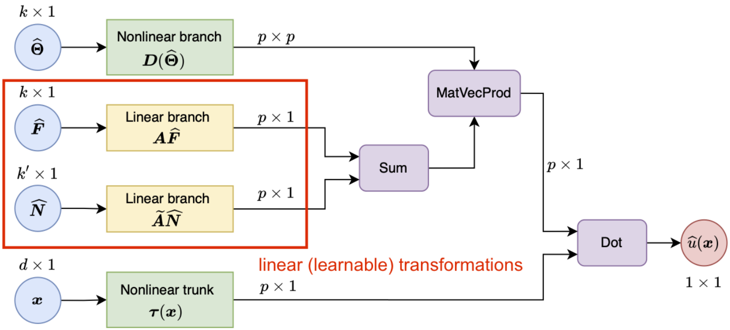
The network is comprises several sub-networks with latent dimension p:

- Non-linear branch net: $\widehat{\boldsymbol{\Theta}} \mapsto \boldsymbol{D}(\widehat{\boldsymbol{\Theta}}) \in \mathbb{R}^{p\times p}$



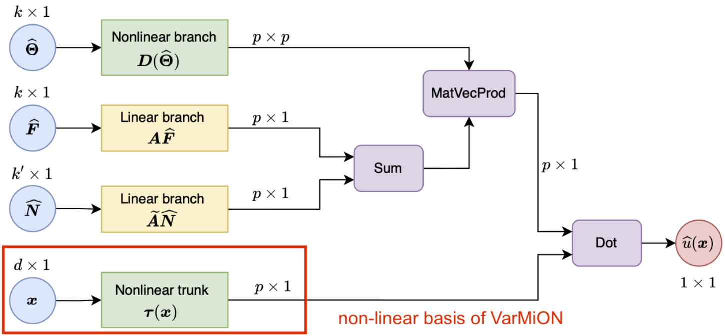
The network is comprises several sub-networks with latent dimension p:

- Non-linear branch net: $\widehat{\boldsymbol{\Theta}} \mapsto \boldsymbol{D}(\widehat{\boldsymbol{\Theta}}) \in \mathbb{R}^{p\times p}$
- Two linear branches with learnable matrices $\boldsymbol{A}, \widetilde{\boldsymbol{A}}$



The network is comprises several sub-networks with latent dimension p:

- Non-linear branch net: $\widehat{\boldsymbol{\Theta}} \mapsto \boldsymbol{D}(\widehat{\boldsymbol{\Theta}}) \in \mathbb{R}^{p\times p}$
- Two linear branches with learnable matrices $\boldsymbol{A}, \widetilde{\boldsymbol{A}}$
- Non-linear trunk (basis of VarMiON): $\boldsymbol{x} \mapsto \boldsymbol{\tau}(\boldsymbol{x}) = (\tau_1(\boldsymbol{x}), \cdots \tau_p(\boldsymbol{x}))^\top$

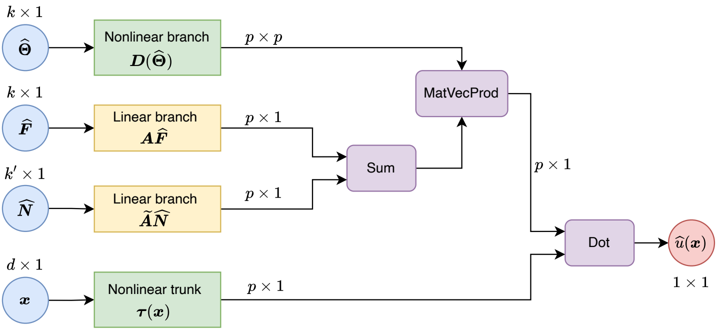


VarMiON operator is

$$\widehat{\mathcal{S}} : \mathbb{R}^k \times \mathbb{R}^k \times \mathbb{R}^{k'} \longrightarrow \mathcal{V}^{\tau} = \text{span}\{\tau_i(\boldsymbol{x}) : 1 \leq i \leq p\}$$
$$(\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{N}}) \mapsto \widehat{u}(.; \widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{N}}) = \boldsymbol{\beta}(f^h, \eta^h, \theta^h)^{\top} \boldsymbol{\tau}$$

where

$$\boldsymbol{\beta}(\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{H}}) = \boldsymbol{D}(\widehat{\boldsymbol{\Theta}})(\boldsymbol{A}\widehat{\boldsymbol{F}} + \widetilde{\boldsymbol{A}}\widehat{\boldsymbol{N}}).$$



VarMiON operator is

$$\widehat{\mathcal{S}} : \mathbb{R}^k \times \mathbb{R}^k \times \mathbb{R}^{k'} \longrightarrow \mathcal{V}^\tau = \text{span}\{\tau_i(\boldsymbol{x}) : 1 \le i \le p\}$$

$$(\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{N}}) \mapsto \widehat{u}(.; \widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{N}}) = \boldsymbol{\beta}(f^h, \eta^h, \theta^h)^\top \boldsymbol{\tau}$$

where

$$\boldsymbol{\beta}(\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{H}}) = \boldsymbol{D}(\widehat{\boldsymbol{\Theta}})(\boldsymbol{A}\widehat{\boldsymbol{F}} + \widetilde{\boldsymbol{A}}\widehat{\boldsymbol{N}}).$$

Compare this to the discrete solution operator:

$$\mathcal{S}^h : \mathcal{F}^h \times \mathcal{T}^h \times \mathcal{N}^h \longrightarrow \mathcal{V}^h$$

$$(f^h, \theta^h, \eta^h) \mapsto u^h(.; f^h, \theta^h, \eta^h) = \boldsymbol{B}(f^h, \eta^h, \theta^h)^\top \boldsymbol{\Phi}$$

where

$$\boldsymbol{B}(f^h, \theta^h, \eta^h) = \boldsymbol{K}^{-1}(\theta^h)(\boldsymbol{M}\boldsymbol{F} + \widetilde{\boldsymbol{M}}\boldsymbol{N}).$$

In comparison with the variational formulation

- Can prove $\boldsymbol{D}$ is the reduced order counterpart of $\boldsymbol{K}^{-1}$ $(p \ll q)$
- While the basis $\boldsymbol{\Phi}$ are fixed, $\boldsymbol{\tau}$ are learned from training data.

In comparison with a vanilla DeepONet

- DeepONet typically has a single nonlinear branch for inputs.
- VarMiON explicitly constructs matrix operators. DeepONet does not.

1. For $1 \leq j \leq J$, consider distinct samples $(f_j, \theta_j, \eta_j) \in \mathcal{X}$.
2. Obtain the discrete approximations $(f_j^h, \theta_j^h, \eta_j^h) \in \mathcal{X}^h$.
3. Find the discrete numerical solution $u_j^h = \mathcal{S}^h(f_j^h, \theta_j^h, \eta_j^h)$.
4. Choose output nodes $\{\boldsymbol{x}_l\}_{l=1}^{L}$ to sample the numerical solution $u_{jl}^h = u_j^h(\boldsymbol{x}_l)$.
5. Generate the input vectors $(\widehat{\boldsymbol{F}}_j, \widehat{\boldsymbol{\Theta}}_j, \widehat{\boldsymbol{N}}_j)$.
6. Collect input & output to form training set with $J \times L$ samples

$$\mathbb{S} = \{(\widehat{\boldsymbol{F}}_j, \widehat{\boldsymbol{\Theta}}_j, \widehat{\boldsymbol{N}}_j, \boldsymbol{x}_l, u_{jl}^h) : 1 \leq j \leq J,\ 1 \leq l \leq L\},$$

Find the network weights that minimize the loss function

$$\Pi(\boldsymbol{\psi}) = \frac{1}{J} \sum_{j=1}^{J} \Pi_j(\boldsymbol{\psi}), \qquad \Pi_j(\boldsymbol{\psi}) = \sum_{l=1}^{L} w_l \left(u_{jl}^h - \widehat{\mathcal{S}}_{\boldsymbol{\psi}}(\widehat{\boldsymbol{F}}_j, \widehat{\boldsymbol{\Theta}}_j, \widehat{\boldsymbol{N}}_j)[\boldsymbol{x}_l] \right)^2.$$

where ψ are all the trainable parameters of the VarMiON.

Error Estimates

The generalization error for any $(f, \theta, \eta) \in \mathcal{X}$

$$\mathcal{E}(f, \theta, \eta) := \|\mathcal{S}(f, \theta, \eta) - \widehat{\mathcal{S}}(\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{N}})\|.$$

Split into four errors:

$$\begin{aligned} \mathcal{E}(f, \theta, \eta) \quad \leq \quad & \|\mathcal{S}(f, \theta, \eta) - \mathcal{S}(f_j, \theta_j, \eta_j)\| \longrightarrow \text{Stability of } \mathcal{S} \\ & + \|\mathcal{S}(f_j, \theta_j, \eta_j) - \mathcal{S}^h(f_j^h, \theta_j^h, \eta_j^h)\| \longrightarrow \text{Numerical error in generating data} \\ & + \|\mathcal{S}^h(f_j^h, \theta_j^h, \eta_j^h) - \widehat{\mathcal{S}}(\widehat{\boldsymbol{F}}_j, \widehat{\boldsymbol{\Theta}}_j, \widehat{\boldsymbol{N}}_j)\| \longrightarrow \text{Training error of VarMiON} \\ & + \|\widehat{\mathcal{S}}(\widehat{\boldsymbol{F}}_j, \widehat{\boldsymbol{\Theta}}_j, \widehat{\boldsymbol{N}}_j) - \widehat{\mathcal{S}}(\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{N}})\| \longrightarrow \text{Stability of VarMiON } \widehat{\mathcal{S}} \end{aligned}$$

Generalization error estimate (Patel et al., 2022)

If $\mathcal{X} := \mathcal{F} \times \mathcal{T} \times \mathcal{N}$ is compact, the non-linear branch is Lipschitz, i.e.,

$$\|\boldsymbol{D}(\widehat{\boldsymbol{\Theta}}) - \boldsymbol{D}(\widehat{\boldsymbol{\Theta}}')\|_2 \leq L_D \|\widehat{\boldsymbol{\Theta}} - \widehat{\boldsymbol{\Theta}}'\|_2.$$

Then, the generalization error can be bounded as

$$\mathcal{E}(f, \theta, \eta) \leq \mathcal{C} \left(\epsilon_h + \epsilon_s + \sqrt{\epsilon_t} + \frac{1}{k^{\alpha/2}} + \frac{1}{(k')^{\alpha'/2}} + \frac{1}{L^{\gamma/2}} \right)$$

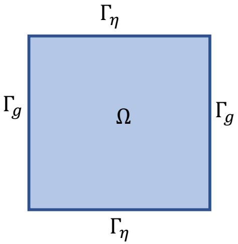
where

$$\begin{aligned} \epsilon_h &\rightarrow \text{numerical error in training data} \\ \epsilon_s &\rightarrow \text{covering estimate} \\ \epsilon_t &\rightarrow \text{training error} \\ \alpha, \alpha', \gamma &\rightarrow \text{quadrature convergence rates} \end{aligned}$$

The constant C depends on the stability constants of $\mathcal{S}$ and $\widehat{\mathcal{S}}$.

Steady-state heat conduction

$$\begin{aligned} -\nabla\cdot(\theta(\boldsymbol{x})\nabla u(\boldsymbol{x})) &= f(\boldsymbol{x}), && \forall\, \boldsymbol{x}\in\Omega,\\ \theta(\boldsymbol{x})\nabla u(\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{x}) &= \eta(\boldsymbol{x}), && \forall\, \boldsymbol{x}\in\Gamma_\eta,\\ u(\boldsymbol{x}) &= 0, && \forall\, \boldsymbol{x}\in\Gamma_g. \end{aligned}$$



- Input: thermal conductivity θ, heat sources f, and heat flux η. Output: temperature u.
- Inputs: Gaussian Random Fields.
- Networks with two inputs (θ, f) and three inputs (θ, f, η).
- Compare VarMiON ($p = 100$) and vanilla DeepONet.
- Similar number of network parameters. Identical trunk architecture.
- Robustness: sampling (spatially uniform or random) and trunk functions (ReLU or RBF).

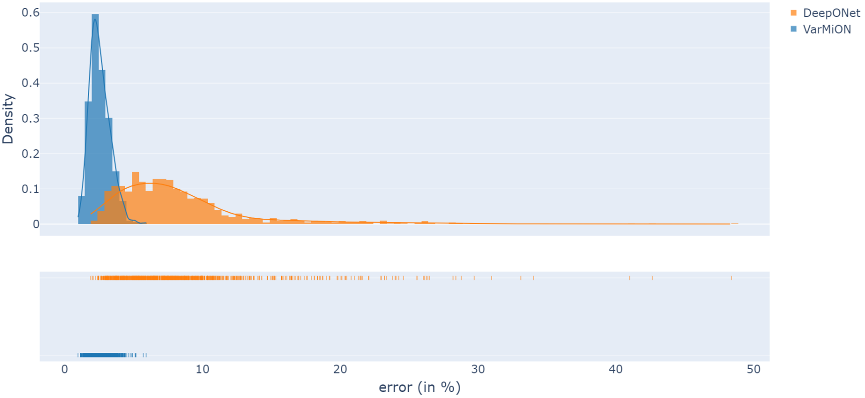
Numerical Example: Two Inputs

- 10,000 samples generated using Fenics.
- 9,000 for training/validation and 1,000 for testing.
- Average value of relative L_2 error reported below.

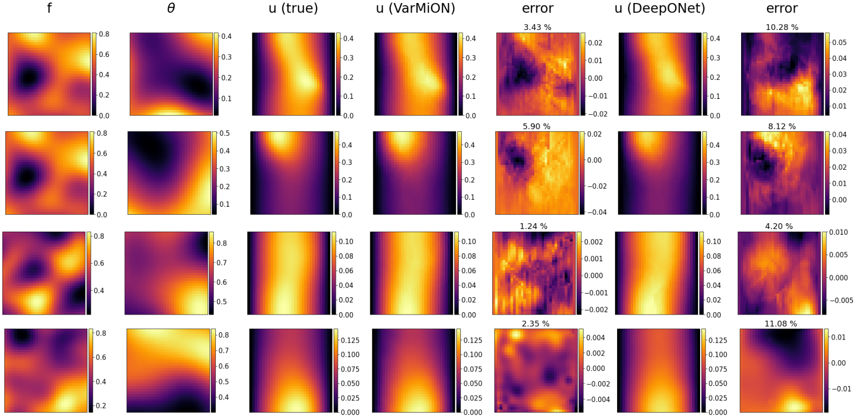
Case	Model	Number of parameters	Relative L_2 error
Randomly sampled input with ReLU Trunk	DeepONet VarMiON	111,248 109,077	3.22 % **2.61 %**
Uniformly sampled input with ReLU Trunk	DeepONet VarMiON	49,928 46,345	8.17 % **2.53 %**
Uniformly sampled input with RBF Trunk	DeepONet VarMiON	17,911 17,409	3.83 % **2.28 %**

Numerical Example: Two inputs

Density of scaled L_2 error (ReLU trunk and uniform spatial sampling).



Predictions by VarMiON and DeepONet.



Numerical Example: Three Inputs

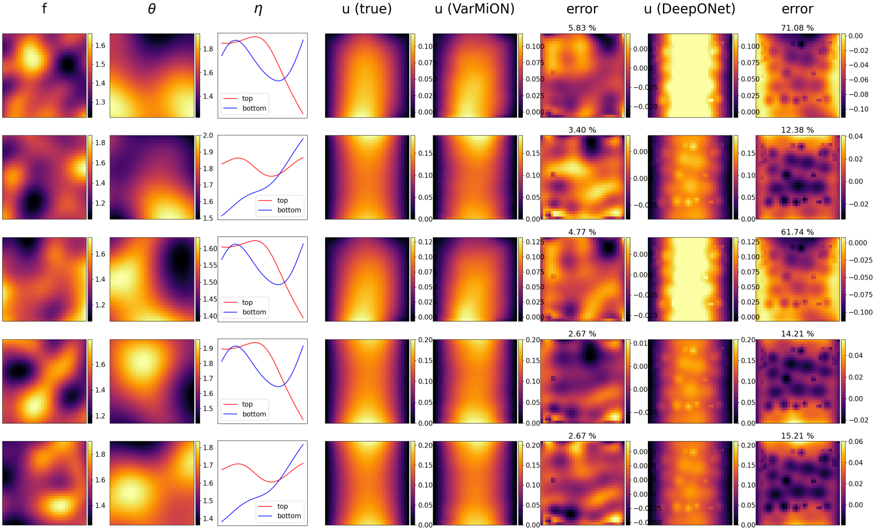
- Input functions f, θ and η.
- 15,625 samples generated using Fenics.
- 14,625 for training/validation and 1,000 for testing.
- Average value of relative L_2 error reported below.

Case	Model	Number of parameters	Relative L_2 error
Uniformly sampled input	DeepONet	28,090	31.41 %
with RBF Trunk	VarMiON	31,849	**4.02 %**

Density of scaled L_2 error (RBF trunk and uniform spatial sampling).

Numerical Example: Three inputs

Predictions by VarMiON and DeepONet.



Concluding remarks

- VarMiON: an operator network that mimics variational formulation.
- Takes the form of a reduced order model.
- Precise specification of the branch network – depends on weak form of PDE.
- Error analysis reveals important components.
- Numerical results point to better and more robust performance.
- Several extensions: nonlinear operators, physics-informed residuals, time-dependent problems, hyperbolic systems, and specification of geometry.