VarMiON: **Var**iationally **Mi**metic **O**perator **N**etwork

Deep Ray
Department of Mathematics
University of Maryland, College Park

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

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

Supported by ARO grant W911NF2010050.

ECCOMAS
Lisboa, June 3-7, 2024

UNIVERSITY OF MARYLAND

Operator learning

Consider the generic PDE:

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

Interested in approximating the solution operator

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

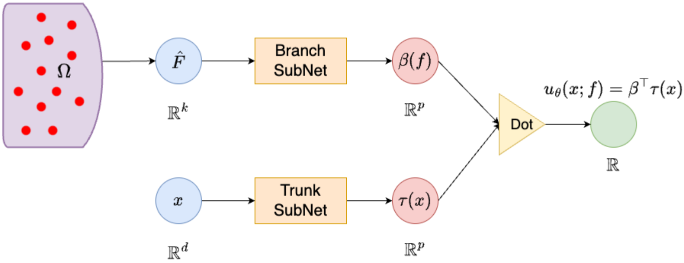
Consider the generic PDE:

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

Interested in approximating the solution operator

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

- Deep Operator Networks (DeepONets) [Lu et al., 2021]
- Comprise two subnetworks: the **trunk** (basis) and the **branch** (coefficients).



Consider the generic PDE:

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

Interested in approximating the solution operator

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

- Deep Operator Networks (DeepONets) [Lu et al., 2021]
- Comprise two subnetworks: the **trunk** (basis) and the **branch** (coefficients).
- Neural operators – an alternate strategy to approximate $\mathcal{S}$
- Come in many flavors: Fourier Neural Operators [Li et al., 2020], Graph Kernel Net [Li et al., 2020], PCA-NET [Bhattacharya et al., 2021], ...

Consider the generic PDE:

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

Interested in approximating the solution operator

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

- Deep Operator Networks (DeepONets) [Lu et al., 2021]
- Comprise two subnetworks: the **trunk** (basis) and the **branch** (coefficients).
- Neural operators – an alternate strategy to approximate $\mathcal{S}$
- Come in many flavors: Fourier Neural Operators [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
CMAME, 2024

Problem Statement

- Consider a generic linear elliptic PDE:

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

  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

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

- 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

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

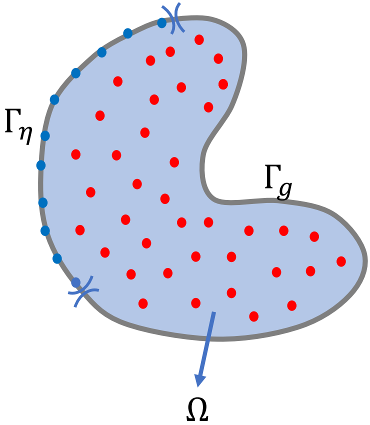
where

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

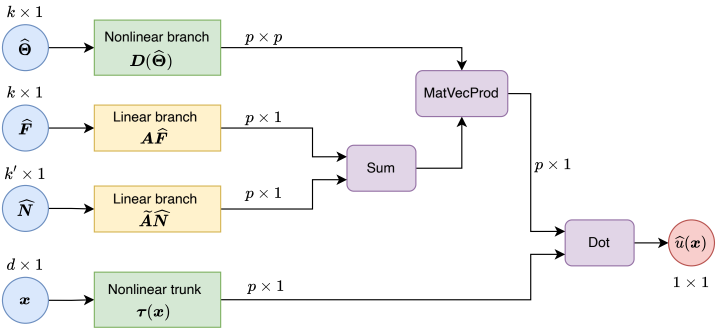
VarMiON will mimic this structure!

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

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

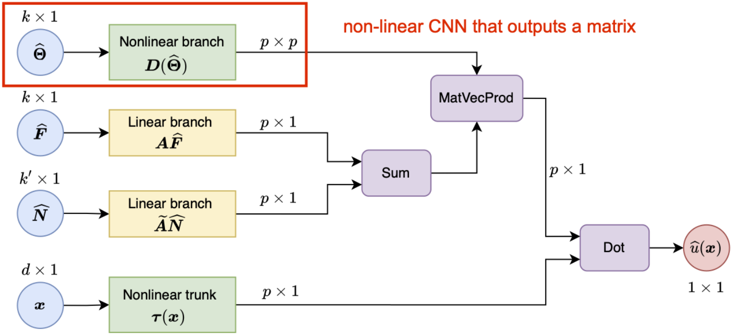


The network is comprises several sub-networks with latent dimension p:



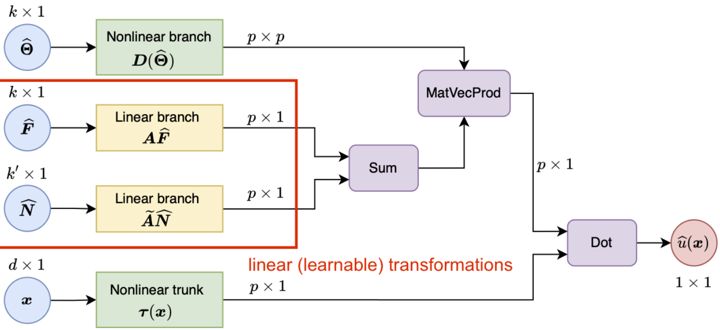
The network is comprises several sub-networks with latent dimension p:

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



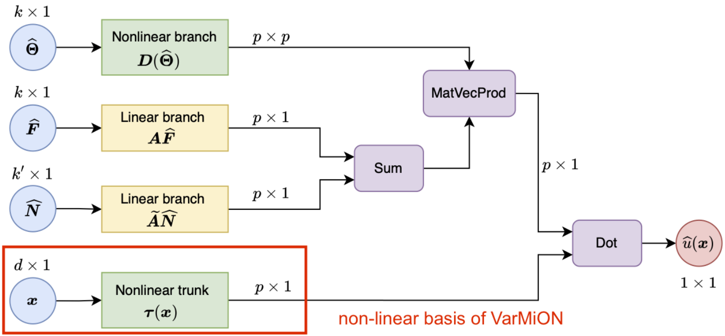
The network is comprises several sub-networks with latent dimension p:

- 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): $\boldsymbol{x} \mapsto \boldsymbol{\tau}(\boldsymbol{x}) = (\tau_1(\boldsymbol{x}), \cdots \tau_p(\boldsymbol{x}))^\top$

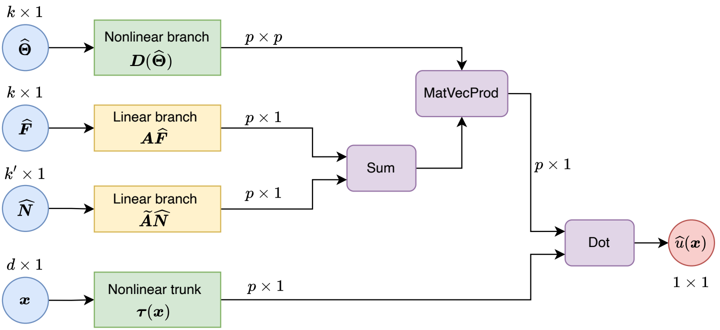


VarMiON operator is

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

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

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

where

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

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

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

- Can prove $\boldsymbol{D}$ is the reduced order counterpart of $\boldsymbol{K}^{-1}$ $(p \ll q)$
- DeepONet typically has a single nonlinear branch for inputs.
- VarMiON explicitly constructs matrix operators. DeepONet does not.

Training

- Use supervised learning.
- Dataset generated using another numerical method, such as FEM
- Find network parameters to minimize a least squares mismatch loss.

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

$$\mathcal{E}(f, \theta, \eta) := \|\mathcal{S}(f, \theta, \eta) - \widehat{\mathcal{S}}(\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{N}})\|_{L^2}.$$

Generalization error estimate (Patel et al., 2024)

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

$\epsilon_h \rightarrow$ numerical error in training data (FEM error)

$\epsilon_s \rightarrow$ covering estimate

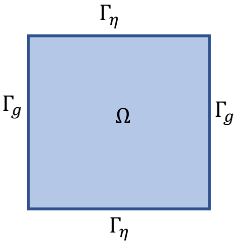
$\epsilon_t \rightarrow$ training error

$\alpha, \alpha', \gamma \rightarrow$ quadrature convergence rates

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

Steady-state heat conduction

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

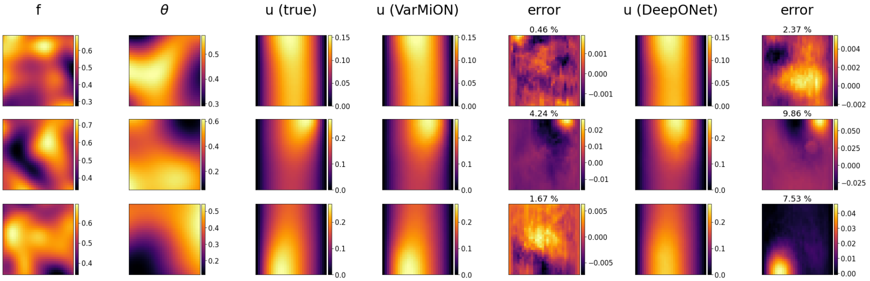


- Input: thermal conductivity θ, heat sources f, and heat flux η. Output: temperature u.
- Inputs: Gaussian Random Fields.
- Networks with two inputs (θ, f) and three inputs (θ, f, η).
- Compare VarMiON and vanilla DeepONet with same p
- Similar number of network parameters. Identical trunk architecture.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

- 10,000 samples generated using Fenics.
- 9,000 for training/validation and 1,000 for testing.
- Latent space $p = 64$ for DeepONet and VarMiON.
- 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	111,248	1.07 $\pm$ 0.39 %
	VarMiON	109,077	**0.96** $\pm$ **0.25 %**
Uniformly sampled input with ReLU Trunk	DeepONet	49,928	1.98 $\pm$ 0.79 %
	VarMiON	46,345	**1.01** $\pm$ **0.39 %**
Uniformly sampled input with RBF Trunk	DeepONet	17,911	1.39 $\pm$ 0.60 %
	VarMiON	17,409	**0.84** $\pm$ **0.40 %**

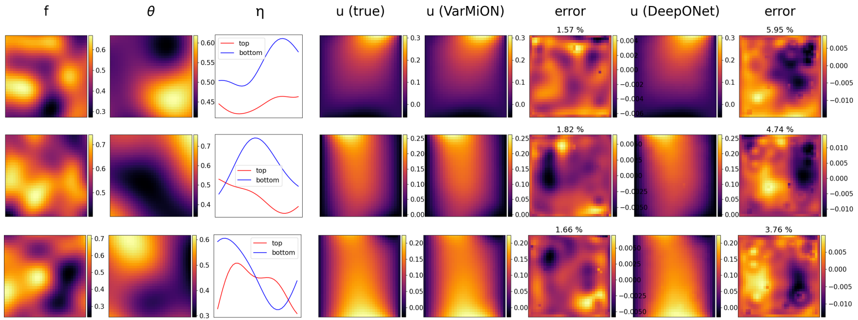
Predictions by VarMiON and DeepONet.



- Input functions f, θ and η.
- 10,000 samples generated using Fenics.
- 9,000 for training/validation and 1,000 for testing $\rightarrow$ same as two input case!
- Latent space $p = 72$ for DeepONet and VarMiON.
- Average value of relative L_2 error reported below.

Case	Model	Number of parameters	Relative L_2 error
Uniformly sampled input with RBF Trunk	DeepONet	31,143	6.29 $\pm$ 3.43%
	VarMiON	31,849	**2.36 $\pm$ 1.13 %**

Predictions by VarMiON and DeepONet.



MIONet has been proposed [Jin et al., 2022] to handle multiple input functions.

Comparison as the number of training samples n is varied

Training dataset size	Model	Relative L_2 error
$n = 1000$	DeepONet	10.23 ± 5.20
	MIONet	88.07 ± 69.68
	VarMiON	$\mathbf{4.27 \pm 2.23}$
$n = 2000$	DeepONet	9.00 ± 5.63
	MIONet	85.03 ± 123.77
	VarMiON	$\mathbf{4.04 \pm 1.86}$
$n = 4000$	DeepONet	7.19 ± 3.75
	MIONet	88.39 ± 62.10
	VarMiON	$\mathbf{2.90 \pm 1.50}$
$n = 6000$	DeepONet	6.28 ± 3.55
	MIONet	82.89 ± 34.19
	VarMiON	$\mathbf{2.74 \pm 1.31}$

A word on non-linear PDEs

- VarMiON needs to be carefully constructed due to the dependence on the structure of the variational form.
- Have constructed a VarMiON for the regularized Eikonal equation.
- We believe the VarMiON architecture needs to be customized depending on the type of PDE (ongoing work)

Concluding remarks

- VarMiON: an operator network that mimics variational formulation.
- Handle multiple inputs – precise specification of branch nets based on weak form.
- Error analysis reveals important components.
- Numerical results point to better and more robust performance.
- Possible extension to nonlinear advection-diffusion-reaction.
- Need a custom architecture based on the weak form.
- Next? Petrov-Galerkin variant, other nonlinear operators, physics-informed residuals, time-dependent problems, hyperbolic systems, and specification of geometry, etc.

Concluding remarks

- VarMiON: an operator network that mimics variational formulation.
- Handle multiple inputs – precise specification of branch nets based on weak form.
- Error analysis reveals important components.
- Numerical results point to better and more robust performance.
- Possible extension to nonlinear advection-diffusion-reaction.
- Need a custom architecture based on the weak form.
- Next? Petrov-Galerkin variant, other nonlinear operators, physics-informed residuals, time-dependent problems, hyperbolic systems, and specification of geometry, etc.

Questions?

Parameters for linear PDE experiments

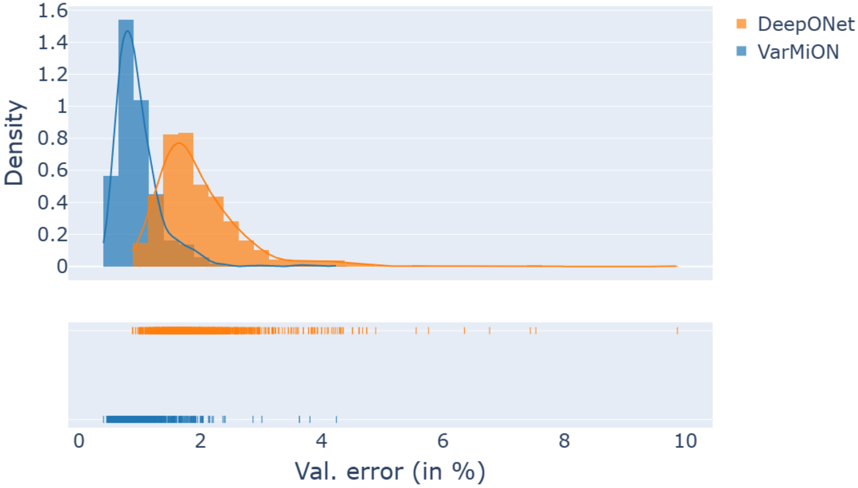
2 input case:

$$\widehat{\boldsymbol{F}} \in \mathbb{R}^{100}, \quad \widehat{\boldsymbol{\Theta}} \in \mathbb{R}^{100}, \quad p = 64$$

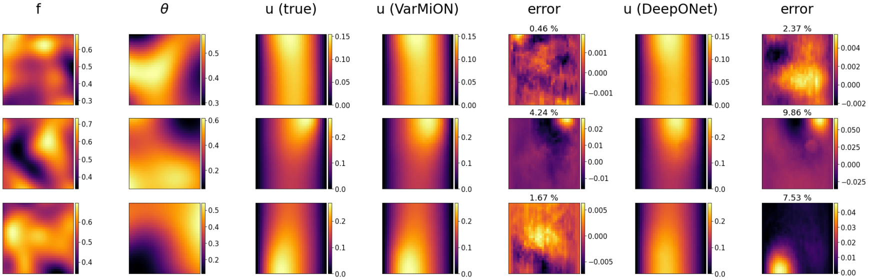
3 input case:

$$\widehat{\boldsymbol{F}} \in \mathbb{R}^{144}, \quad \widehat{\boldsymbol{\Theta}} \in \mathbb{R}^{144}, \quad \widehat{\boldsymbol{N}} \in \mathbb{R}^{24}, \quad p = 72$$

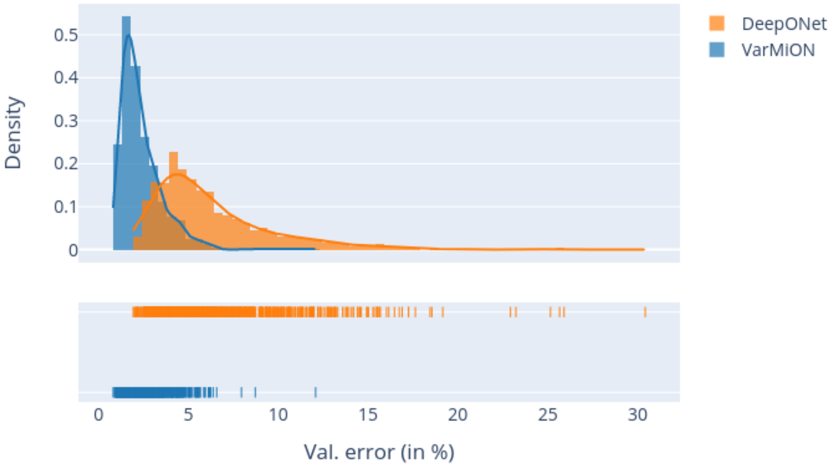
Density of scaled L_2 error (ReLU trunk and uniform spatial sampling).



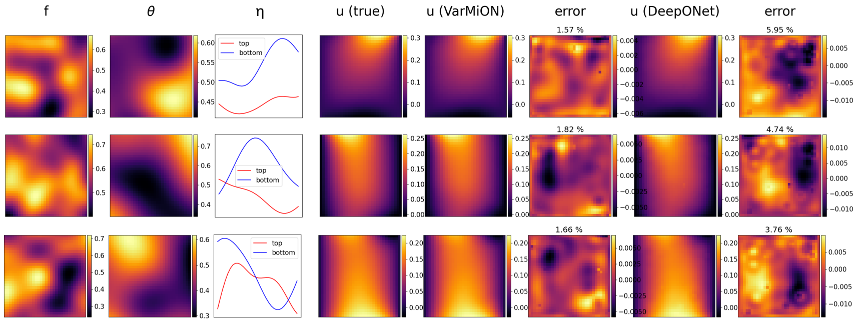
Predictions by VarMiON and DeepONet.



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



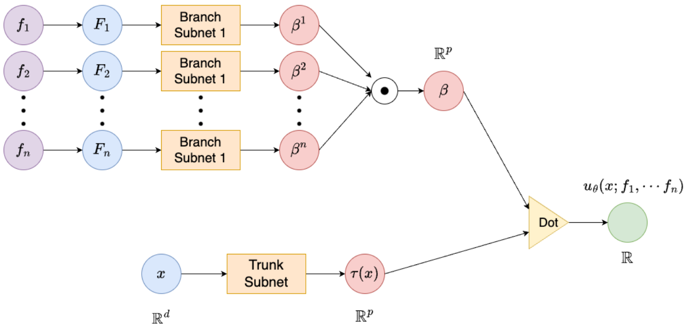
Predictions by VarMiON and DeepONet.



MIONet has been proposed [Jin et al., 2022] to handle multiple input functions.

Separate branch for each f_i, followed by a Hadamard product

$$\beta = (\beta^1 \odot \beta^2 \odot \cdots \odot \beta^n), \quad u_\theta(x; f_1, \cdots, f_n) = \beta^\top \tau(x)$$



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.