DeepONet

Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators

Goal:

Learn non linear operator $G : V_1 \to V_2$ between spaces of function

- Based on **Universal Approximation Theorem for Operators**
- Designed to process input data coming from sensors(mesh-free)
- In applications the operator of interest is often of the form $\boldsymbol{\mu} \mapsto u_{\boldsymbol{\mu}}$, $\boldsymbol{\mu}$ being the parameters of a PDE and $u_{\boldsymbol{\mu}}$ its corresponding solution

Universal Approximation Theorem for Operators

Let σ be a continuous non-polynomial function, X Banach space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ two compact sets in X and $\mathbb{R}^d$, respectively, V compact set in $C(K_1)$. Let $G : V \to C(K_2)$ be a nonlinear continuous operator. Then for any $\epsilon > 0$, there are positive integers n, p and m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \ldots, n$, $k = 1, \ldots, p$ and $j = 1, \ldots, m$, such that

$$\left| G(u)(y) - \sum_{k=1}^{p} \sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \sigma(w_k \cdot y + \zeta_k) \right| < \epsilon$$

holds for all $u \in V$ and $y \in K_2$.

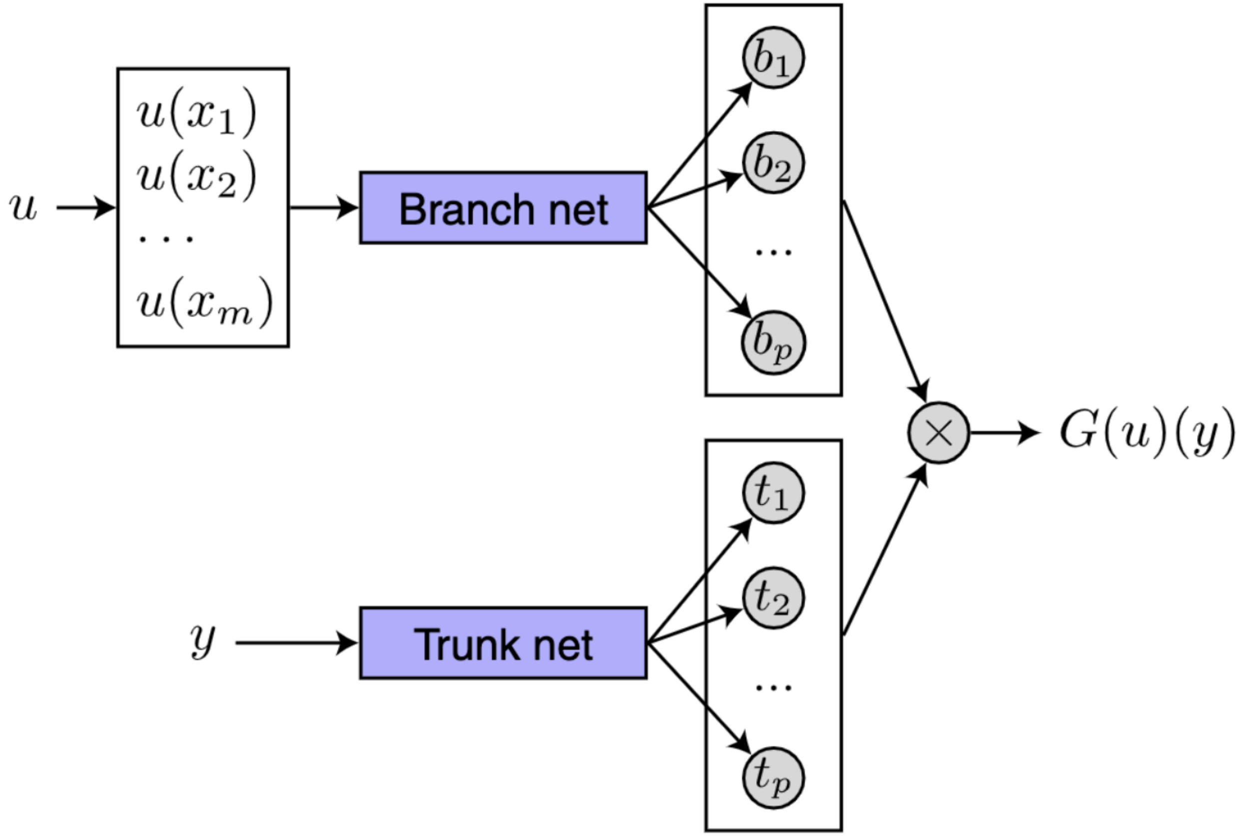
Key points from the theorem:

- An operator acting on a function space can be **reconstructed using a finite number of function values at fixed points** (cast infiinite dimensional problem to finite dimensional one).
- Sensor locations must remain consistent across all training samples
- Directly suggest us to considers **two shallow network with one hidden layer**. We can extend this with **deep networks** to increase expressivity.

DeepONet architecture:

- A **trunk network** that takes y as input and outputs a vector $[t_1, t_2, \ldots, t_p]^T \in \mathbb{R}^p$
- A **branch networks**, each taking $[u(x_1), u(x_2), \ldots, u(x_m)]^T$ as input and producing a scalar b_k for $k = 1, \ldots, p$
- The final output is computed as:

$$G(u)(y) \approx \sum_{k=1}^{p} b_k t_k.$$



Test

1. Poisson Equation (finite-dimensional parameter)

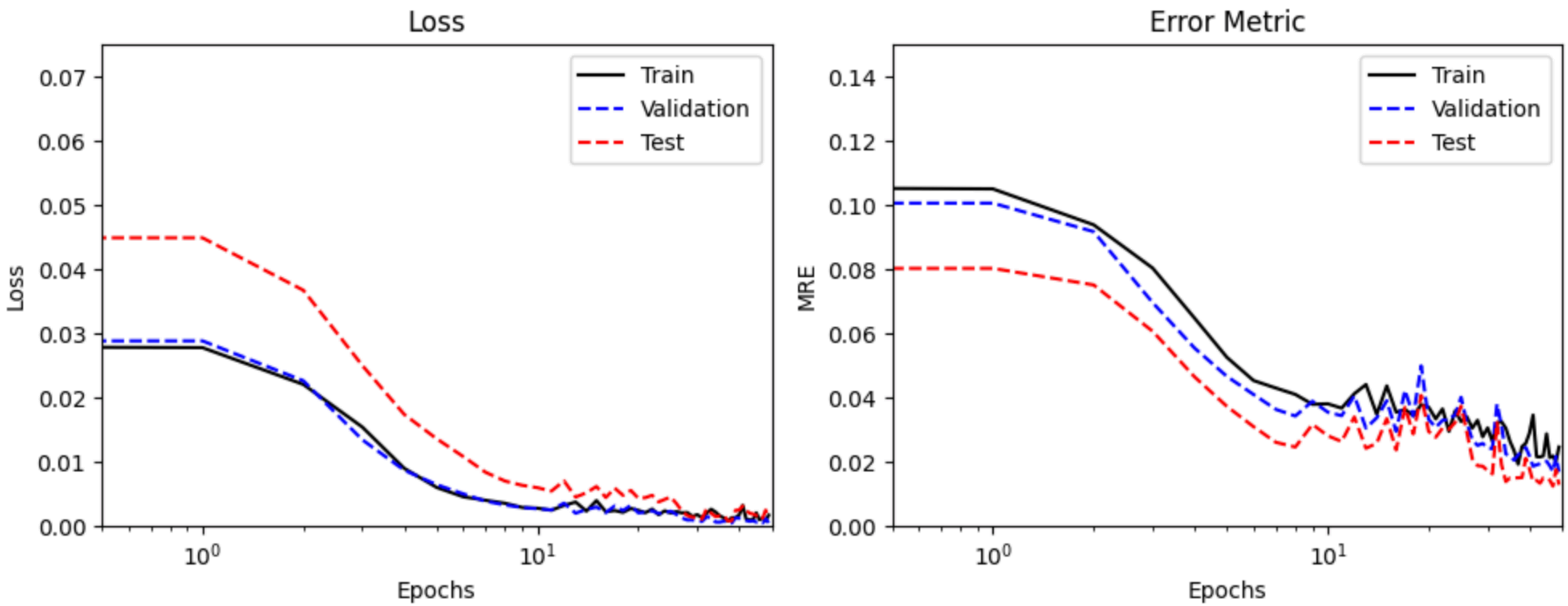
Consider the following Poisson equation

$$\begin{cases} -\sigma \Delta u(\boldsymbol{x}) = \gamma \sin(4x_1 x_2) + (1-\gamma)\cos(x_1 - 8x_2) & \boldsymbol{x} \in \Omega \\ u(\boldsymbol{x}) = b & \boldsymbol{x} \in \Gamma \\ -\nabla u(\boldsymbol{x}) \cdot \boldsymbol{n} = 0 & \boldsymbol{x} \in \partial\Omega \setminus \Gamma \end{cases}$$

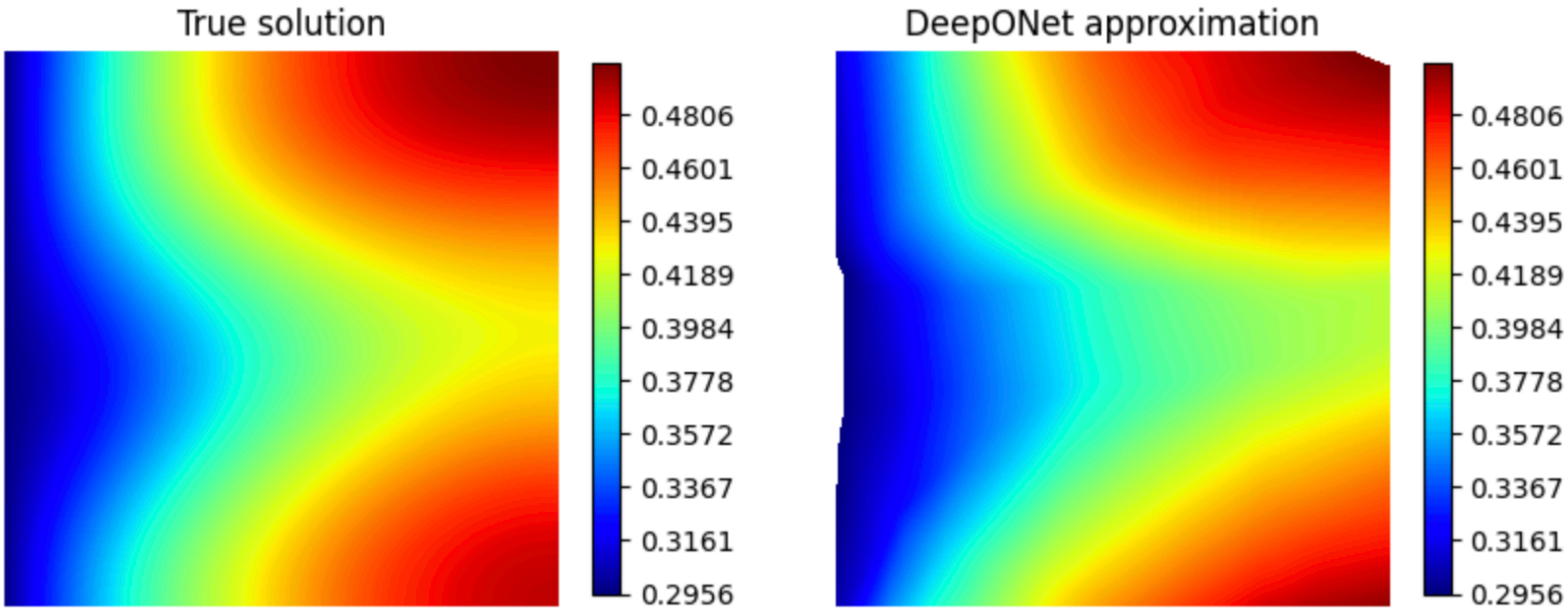
where $\Omega = (0,1)^2$ is the unit square and $\Gamma = \{(0, x_2) \ : \ 0 \leq x_2 \leq 1\}$ is the left edge.

Goal is to learn the map $\boldsymbol{\mu} \mapsto u_{\boldsymbol{\mu}}$, where $\boldsymbol{\mu} = [\sigma, b, \gamma]$,

Optimization



Result



Mean Relative error on test set: 3.03%

2. Function-to-function

Consider the following nonlinear operator G mapping 1D functions onto 1D functions acting as

$$G: \quad f(y) \mapsto \int_0^x \frac{1}{1+f(2s)^2} ds,$$

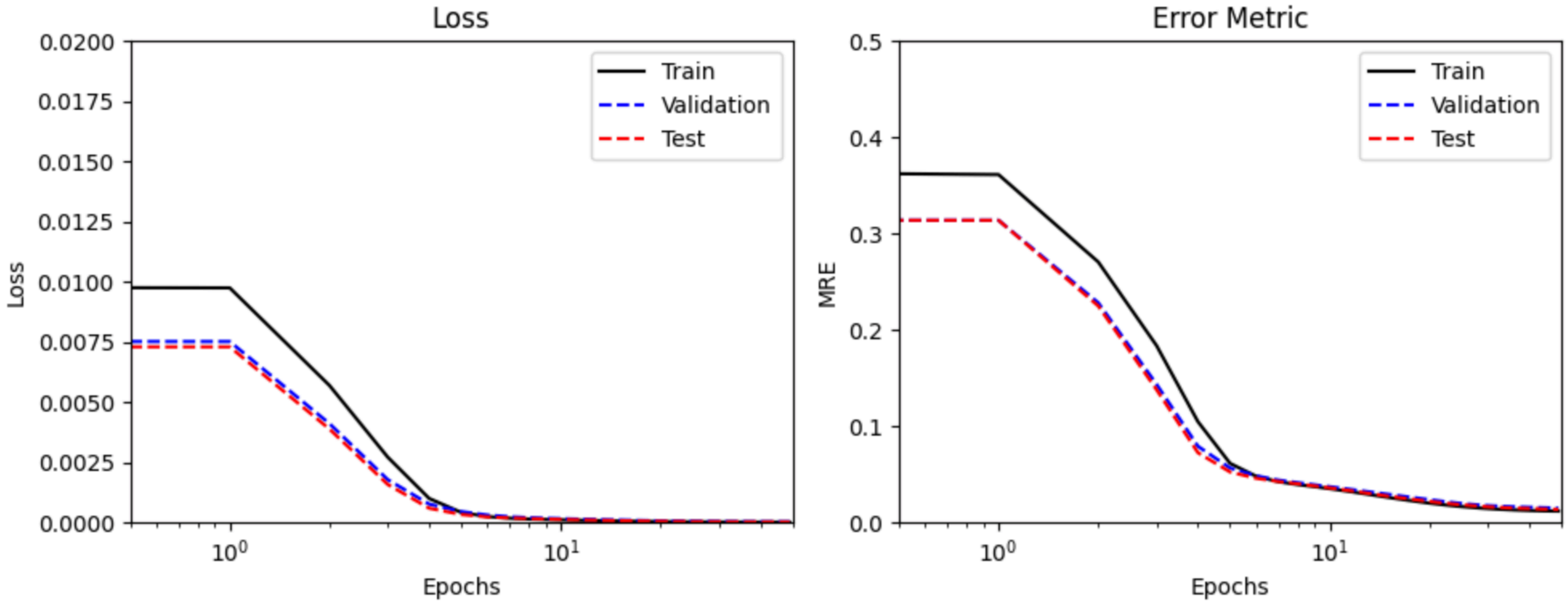
where $f = f(y)$ is defined for $y \in [0,1]$, whereas the output $u = u(x)$ is defined for $x \in [0, 1/2]$.

Input signal are sampled from Gaussian process $Z : [0,1] \to \mathbb{R}$ of the form

$$Z(y) = \sum_{j=1}^{100} e^{-j} \eta_j \sin(\pi j y),$$

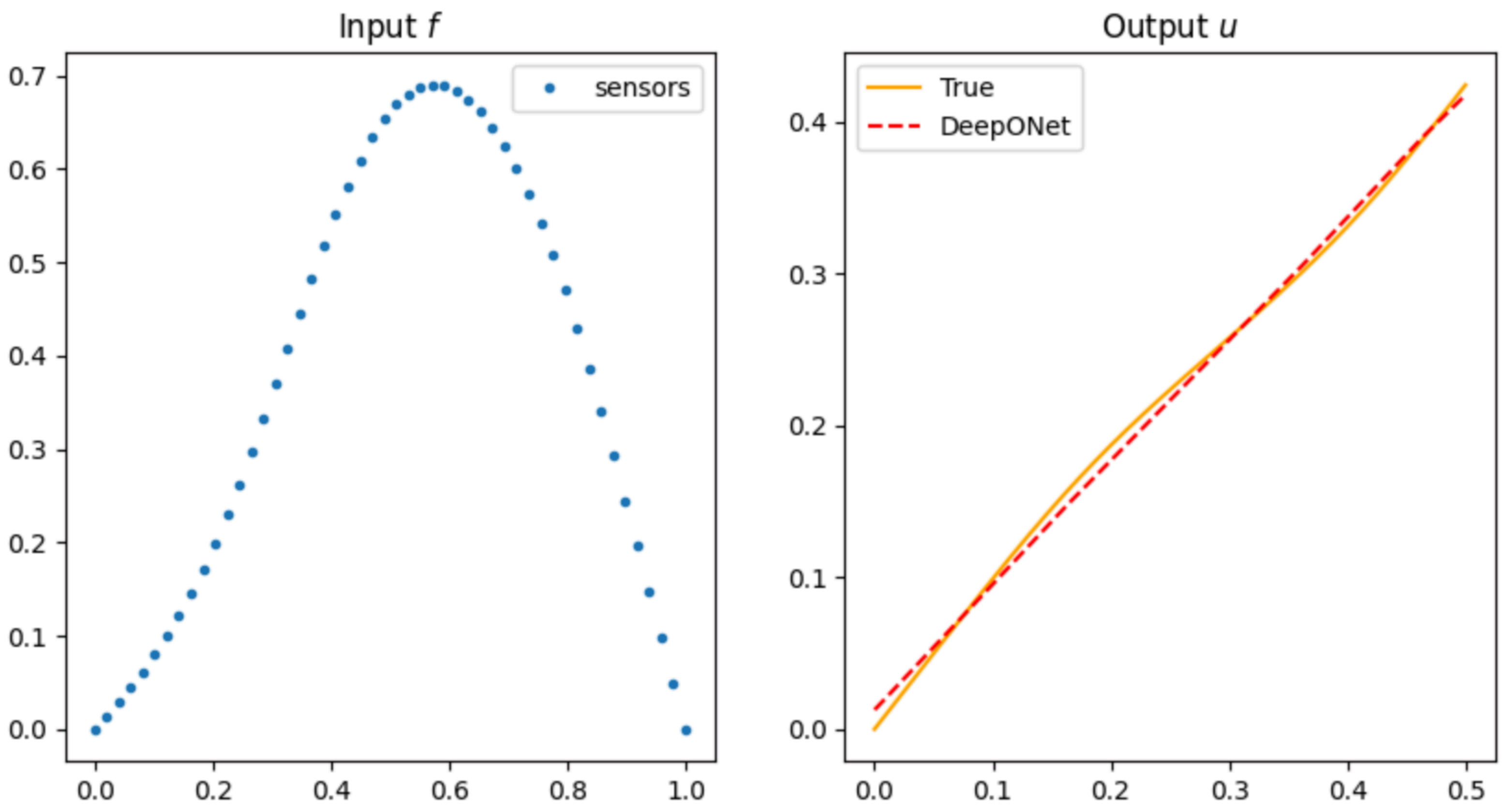
with $\eta_1, \ldots, \eta_{100}$ i.i.d. $\mathcal{N}(0,1)$.

Optimization



Sensor number: 50

Result



Mean Relative error on test set: 1.30%

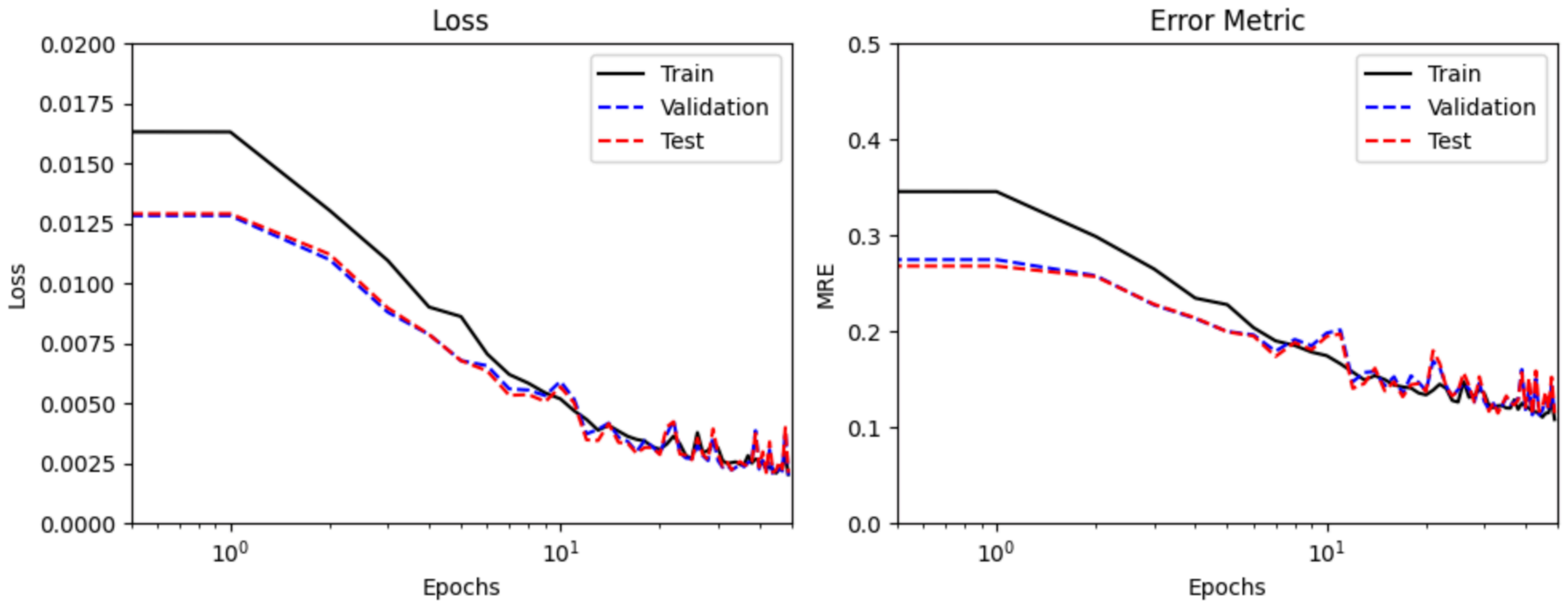
3. Darcy Flow

Consider a simplified Darcy flow model featuring a spatially distributed parameter (permeability field), namely

$$\begin{cases} -\nabla \cdot (k\nabla p) = f & \text{in } \Omega \\ -\nabla p \cdot \boldsymbol{n} \equiv 0 & \text{on } \partial\Omega \\ \int_\Omega p = 0 \end{cases}$$

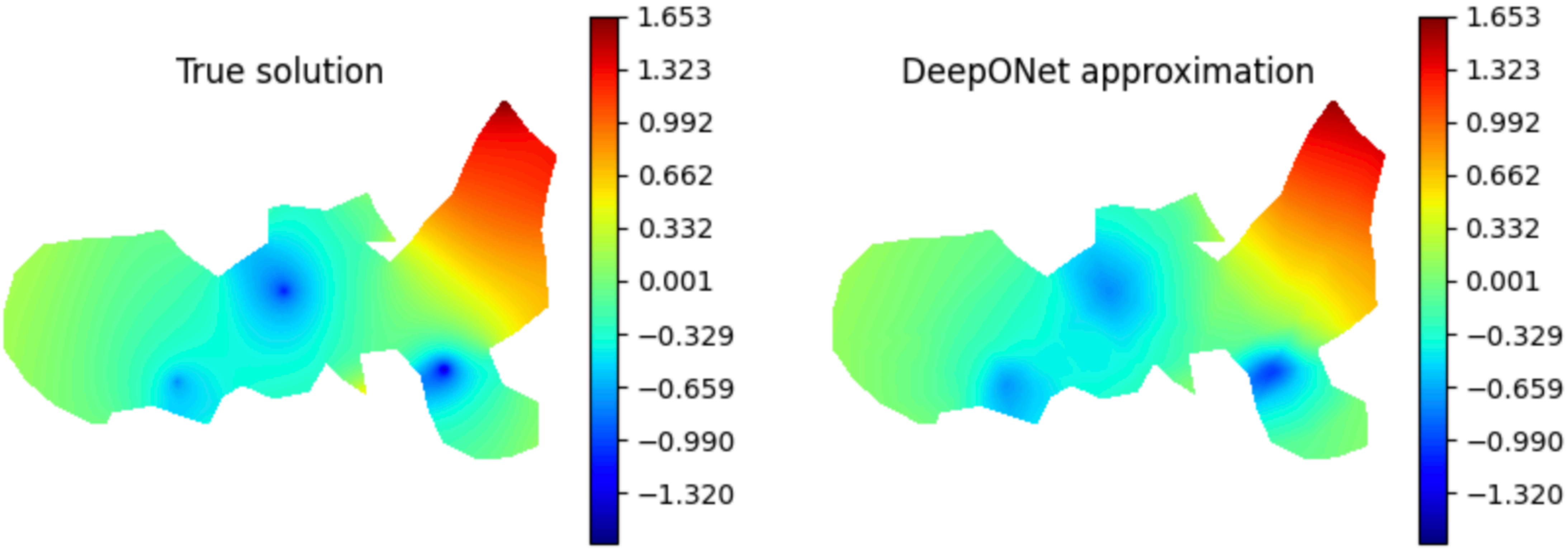
where $\Omega \subset \mathbb{R}^2$ is the spatial domain, $p : \Omega \to \mathbb{R}$ is the pressure field and $\boldsymbol{n}$ is the unit normal. The permeability field, $k : \Omega \to (0, +\infty)$ is our parameter.

Optimization



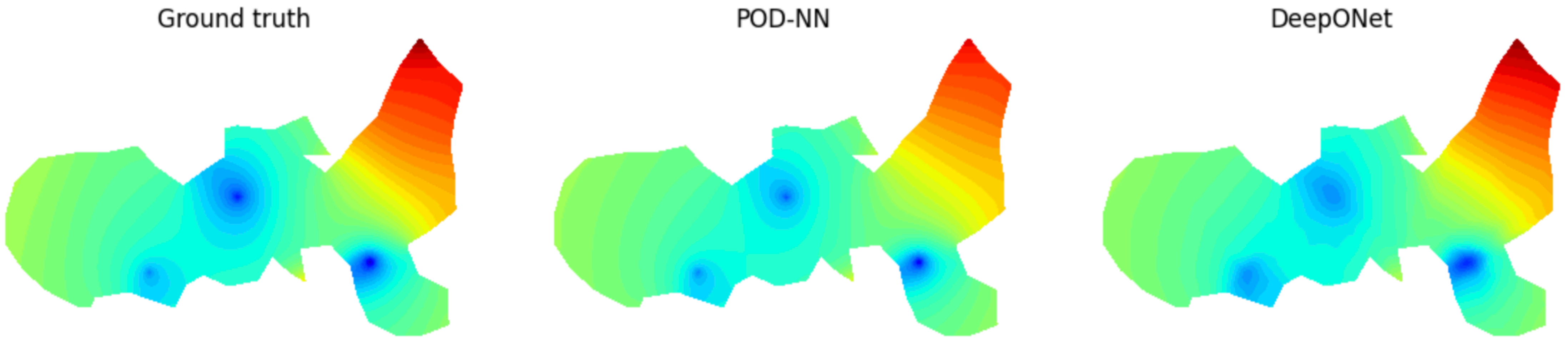
Sensor number: 131

Result



Mean Relative error on test set: 11.82%

3.1. Comparison with POD-NN

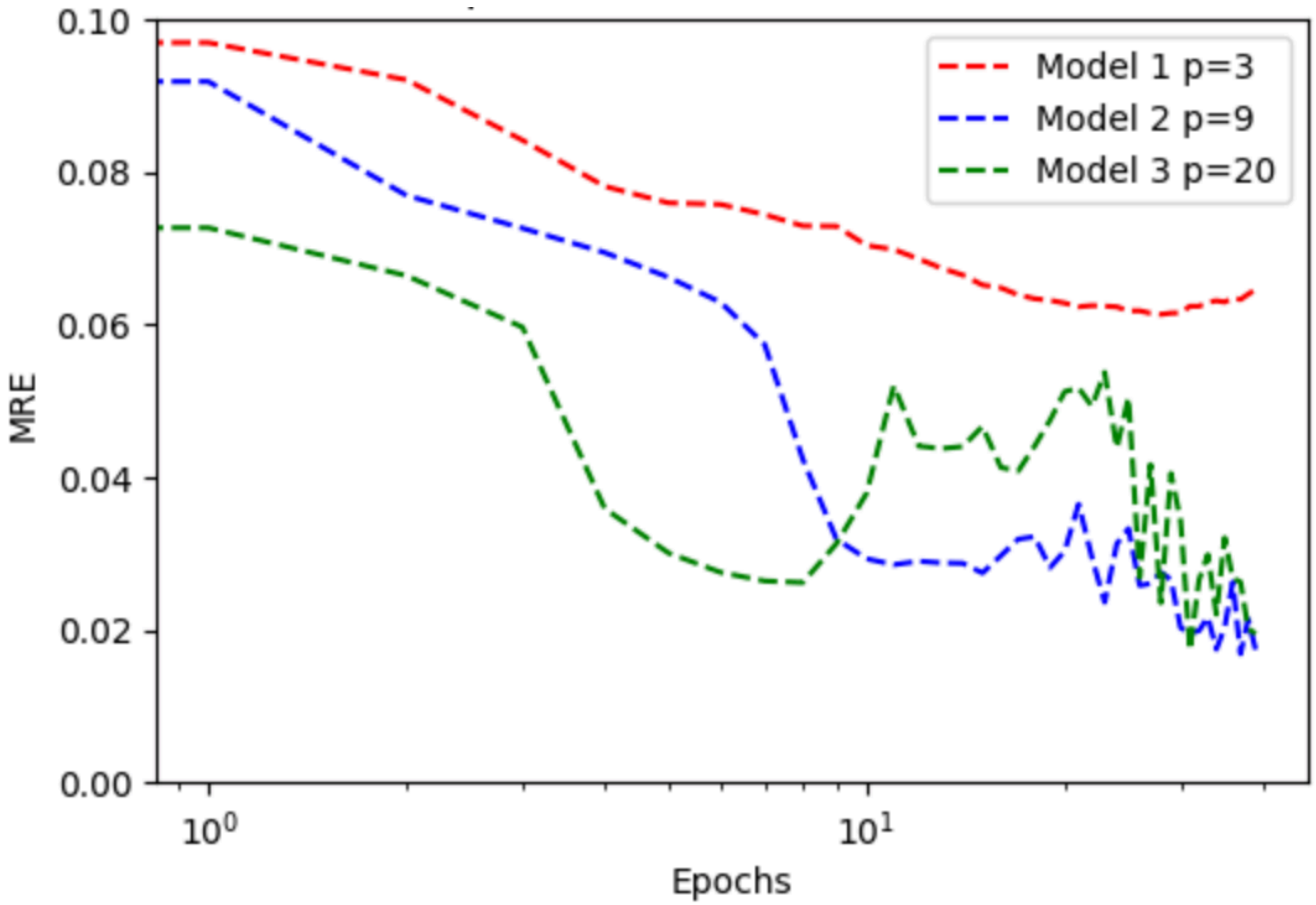


Mean Relative error POD-NN on test set: 13.71%

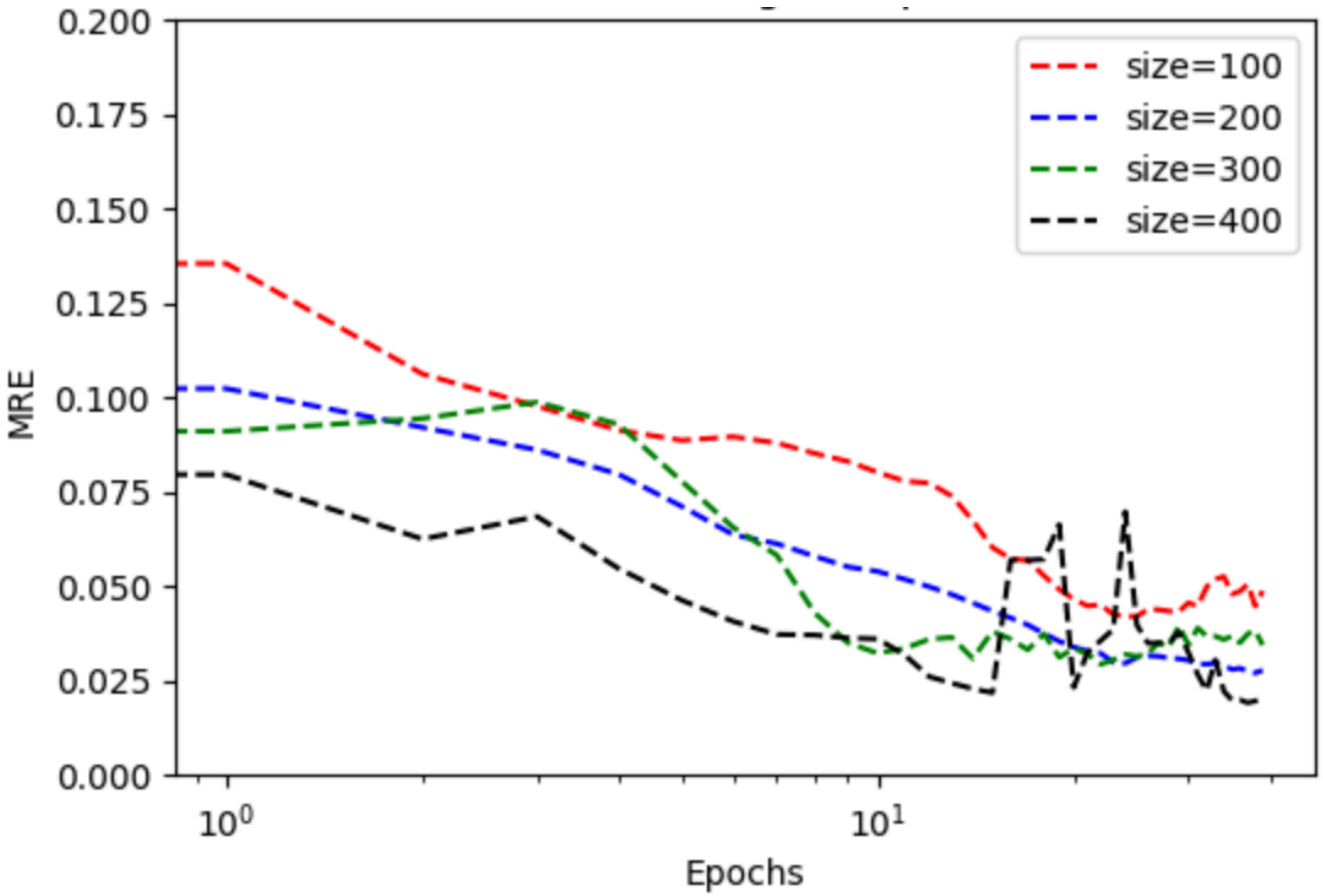
Mean Relative error DeepONet on test set: 11.82%

Other test

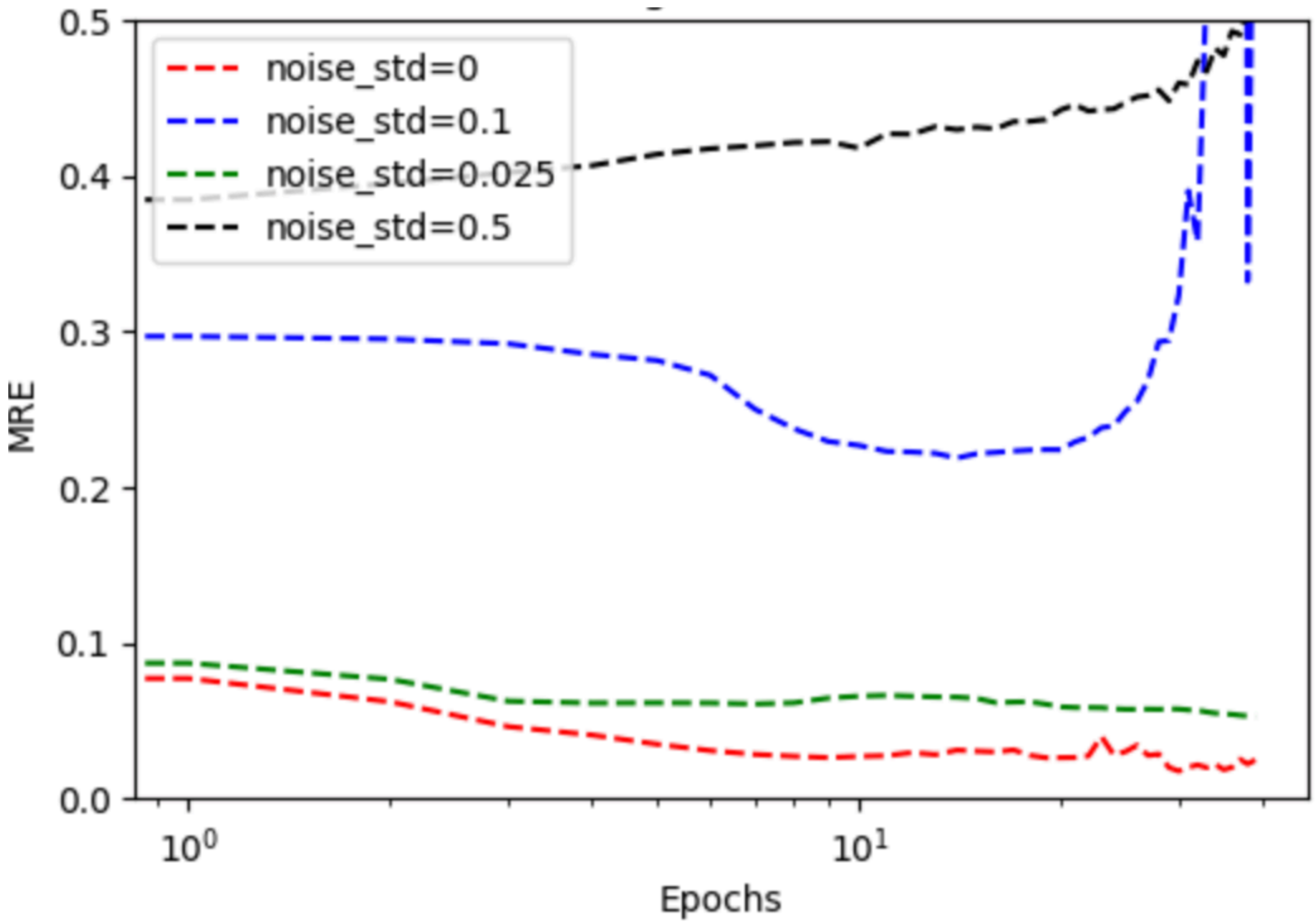
Vary the number of latent dimension (Poisson)



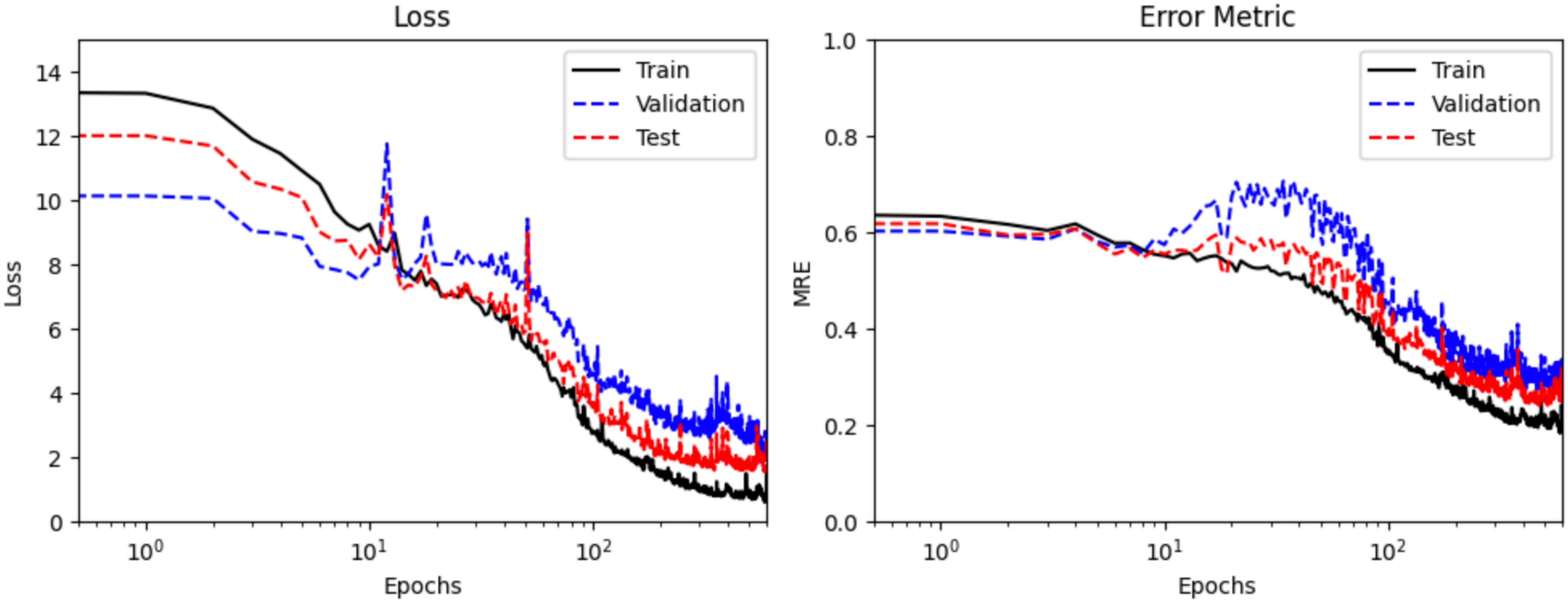
Vary the number of training data (FuncToFunc)

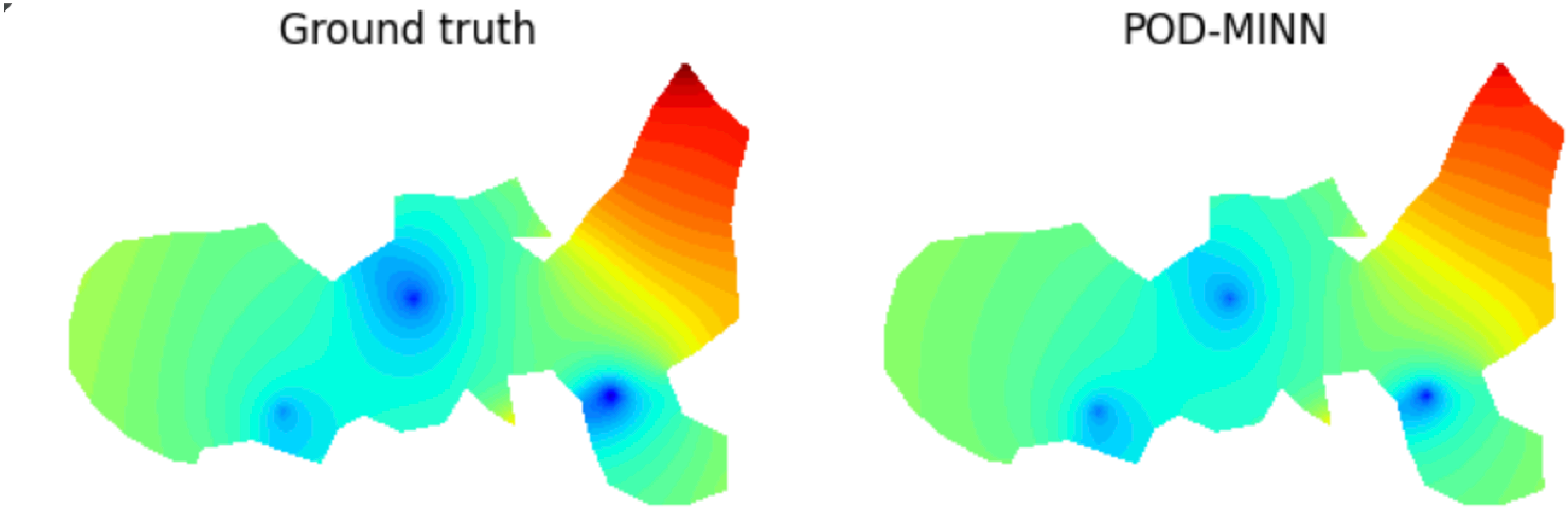


Add Gaussian noise to the measured data (Darcy)



Bonus: Mesh-Informed Neural Networks


Loss
Train
Validation
Test
Loss
Epochs
0
2
4
6
8
10
12
14
10^0
10^1
10^2
Error Metric
Train
Validation
Test
MRE
Epochs
0.0
0.2
0.4
0.6
0.8
1.0
10^0
10^1
10^2



Mean Relative error POD-MINN on test set: 13.35%