# Sampling strategies for DeepONets

Group 8: Laurits and Markus

EEMCS

2025

# Schedule

- Research question
- Methodology 1D case
- Numerical results 1D case
- Methodology 2D case
- Numerical results 2D case
- Discussion and conclusion

# Research question

Consider a general differential equation

$$\begin{aligned} Lu &= f \qquad \text{in } \Omega \\ u &= g \qquad \text{on } \overline{\Omega} \end{aligned}$$

for some differential operator $L : C^k(\Omega) \mapsto C(\Omega)$. With a DeepONet (DON), the aim is to estimate an operator $G : C(\Omega) \mapsto C^k(\Omega)$ which given $f$ estimates $u$, i.e. $G(f) = u$.

1. How does input function distribution influence accuracy of models?
2. How does the accuracy depend of the number of sensor points? How does this relate to the complexity of the input functions used in training vs testing?
3. Can we improve the accuracy and/or generalizability of the model by sampling the training data at each iteration during the training process?

# Research question: Agenda

We firstly investigate the three sub-questions separately on a very simple 1D problem

$$\frac{\partial u(x)}{\partial x} = w\cos(wx) = f(x), \quad u(0) = 0.$$

Then we try to combine the results from the simple test problem on a more complex problem; a 1D time-dependent convection diffusion equation (CDE)

$$\frac{\partial u}{\partial t} + \mu\frac{\partial u}{\partial x} - \epsilon\frac{\partial^2 u}{\partial x^2} = f(x),\ u(x,0) = \sin(\pi x),\ u(0,t) = u(1,t) = 0$$

where $x, t \in [0,1], \mu = 1, \epsilon = 0.1$. This is effectively a 2D problem.

# Methodology: DeepONet idea

Consider a general differential equation

$$\begin{aligned} Lu &= f \qquad \text{in } \Omega \\ u &= g \qquad \text{on } \overline{\Omega} \end{aligned}$$

for some differential operator $L : C^k(\Omega) \mapsto C(\Omega)$. With a DeepONet (DON), the aim is to estimate an operator $G : C(\Omega) \mapsto C^k(\Omega)$ which given $f$ estimates $u$, i.e. $G(f) = u$.
The DON has two components:

- Branch network that takes forcing term $f$ as input, usually in the form sensor points $f = (f(y_1), f(y_2), ..., f(y_m)) \in \Omega$.
- Trunk network that takes coordinates as input $x = (x_1, x_2, ..., x_n) \in \Omega$.

# Methodology: DeepONet architecture

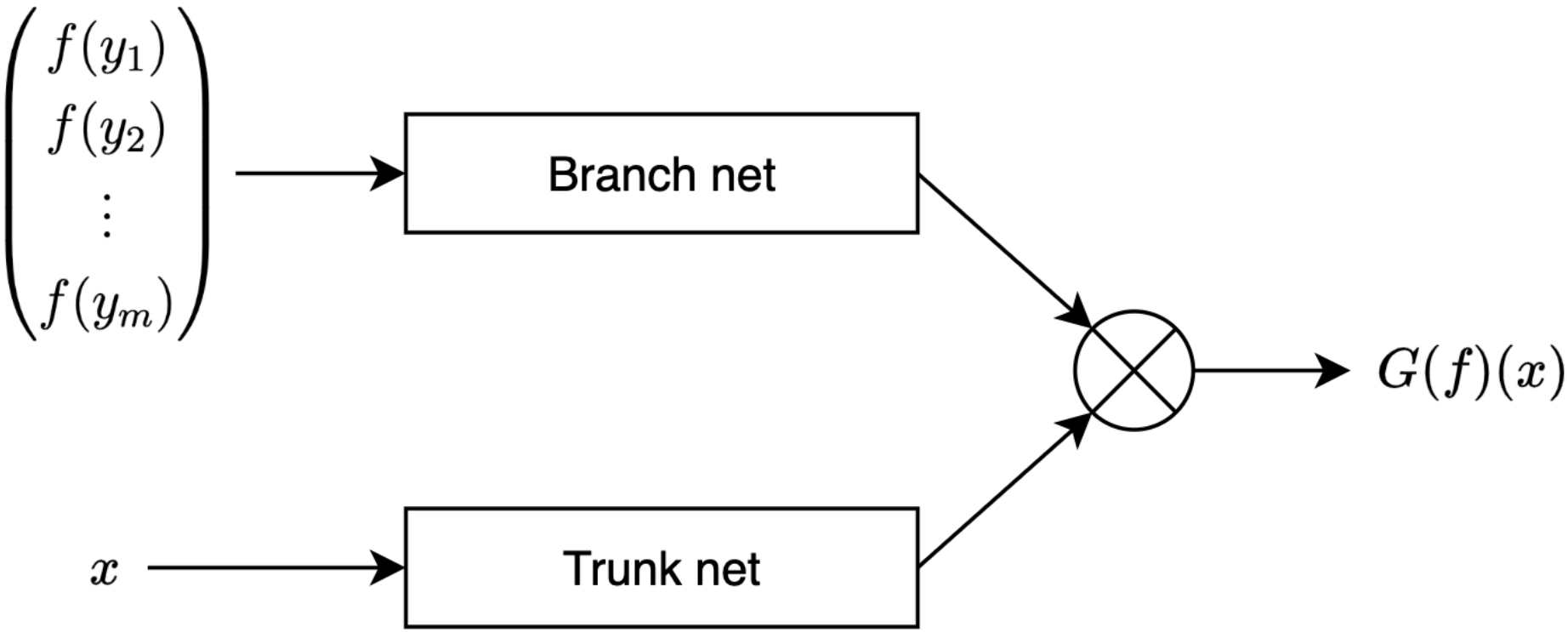



Figure: Illustration of the structure of a DeepONet. The output $G(f)(x)$ is an approximation of $u(x)$.

# Methodology: Data generation 1D case

The 1D problem considered in the first part of the report

$$u_x = \omega \cos(\omega x) = f(x), \quad u(0) = 0, \qquad \text{for } x \in [-1, 1],$$

has the explicit solution $u(x) = \sin(\omega x)$. In this way, the data points for the loss function

$$(f, x, u(x))_i = (\omega_i \cos(x_i), x_i, \sin(\omega_i x_i))$$

can be directly computed at each training step by simply sampling $\omega$. Define the training and test input function spaces

$$\Omega := \{f(x) = \omega \cos(\omega x), \quad \omega \in [a, b]\},$$
$$\Omega_{test} := \{f(x) = \omega \cos(\omega x), \quad \omega \in [a - \delta, b + \delta]\}.$$

# Methodology: DeepONet training 1D case

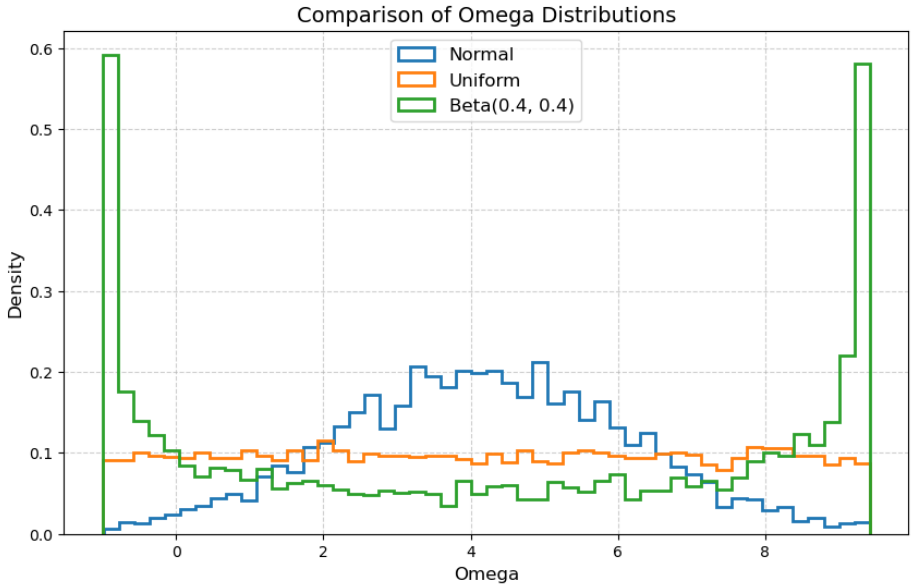
Use adaptive moments to optimize with a learning rate schedule in both 1D case and 2D case.
The loss function for the 1D case incorporates data loss and physics loss by

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \mathcal{L}_{phy}(\theta)$$

$$\mathcal{L}_{data}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(G(f_i)(x_i) - \sin(\omega_i x_i))^2$$

$$\mathcal{L}_{phy}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\partial_x G(f_i)(x_i) - \omega_i \cos(\omega_i x_i))^2.$$

# Numerical results: Parameter distribution 1D case



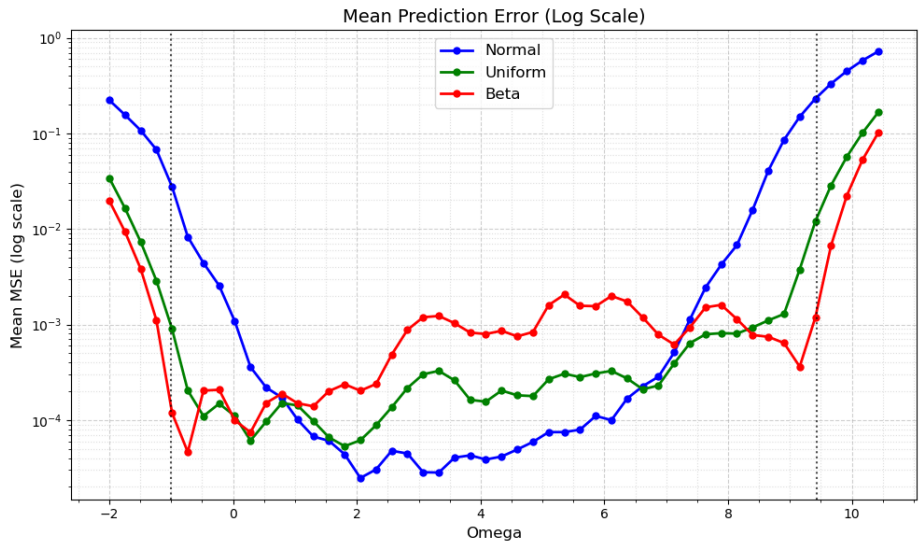




Figure: Mean MSE of the DON trained on different $\omega$ distributions.

DONs struggle to generalize beyond the training input function space. Even within the training space, accuracy correlates with the frequency of similar input functions during training.

# Numerical results: Number of input sensors 1D case

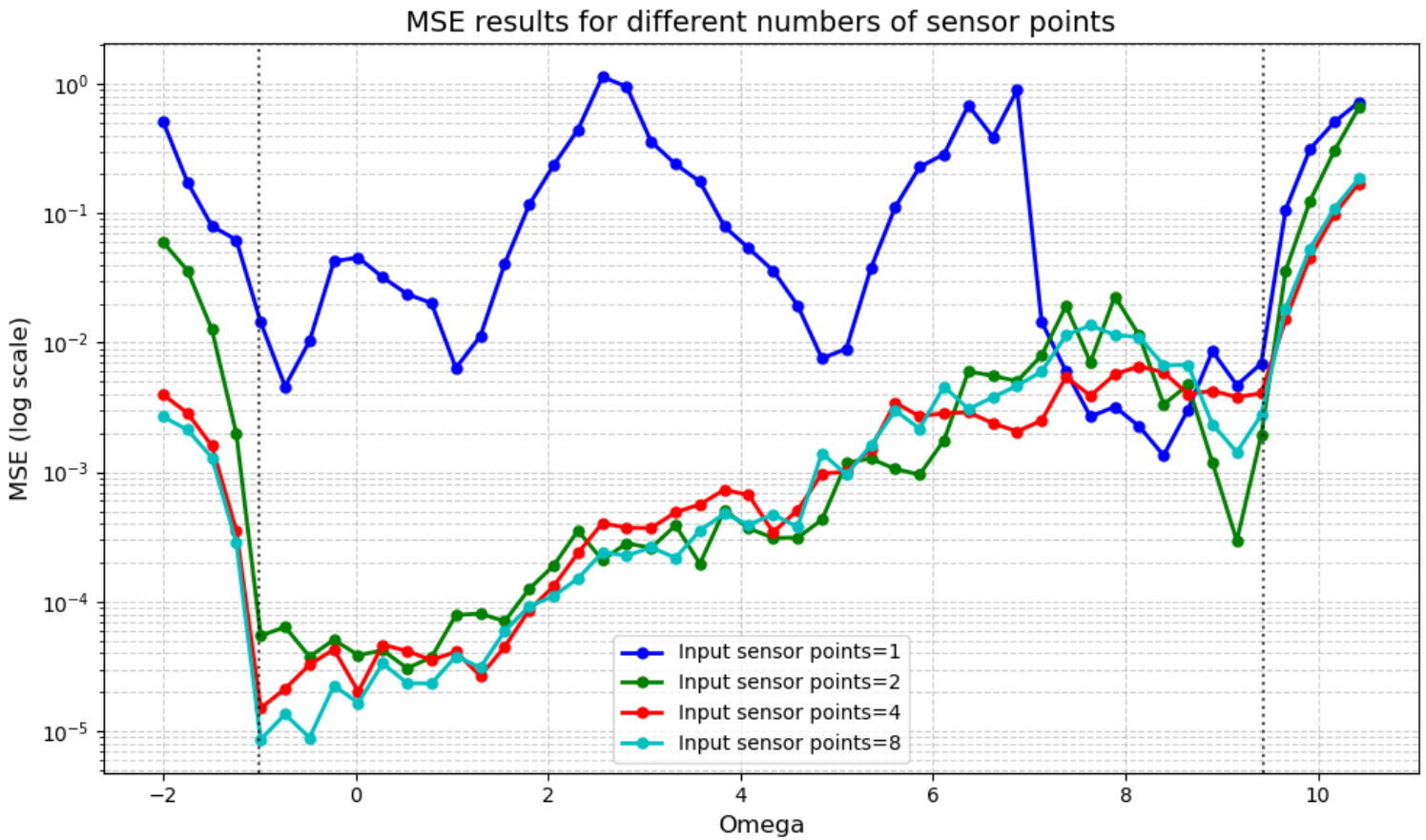



Figure: Mean MSE with different numbers of sensor points $y_i, i = 1, ..., m$. We chose here $y_1 = -0.7, y_2 = 0.9$.

# Numerical results: Number of input sensors 1D case

To explain this, consider the system of equations

$$f(y_1) = \omega \cos(\omega y_1)$$
$$f(y_2) = \omega \cos(\omega y_2).$$

Given $f(y_1), f(y_2)$, can we estimate $\omega$ uniquely? Consider the residual

$$R(\omega) = (\omega_{true} \cos(\omega_{true} y_1) - \omega \cos(\omega y_1))^2 + (\omega_{true} \cos(\omega_{true} y_2) - \omega \cos(\omega y_2))^2$$

and plot for different choices of $\omega_{true}, \omega$.

# Numerical results: Number of input sensors 1D case

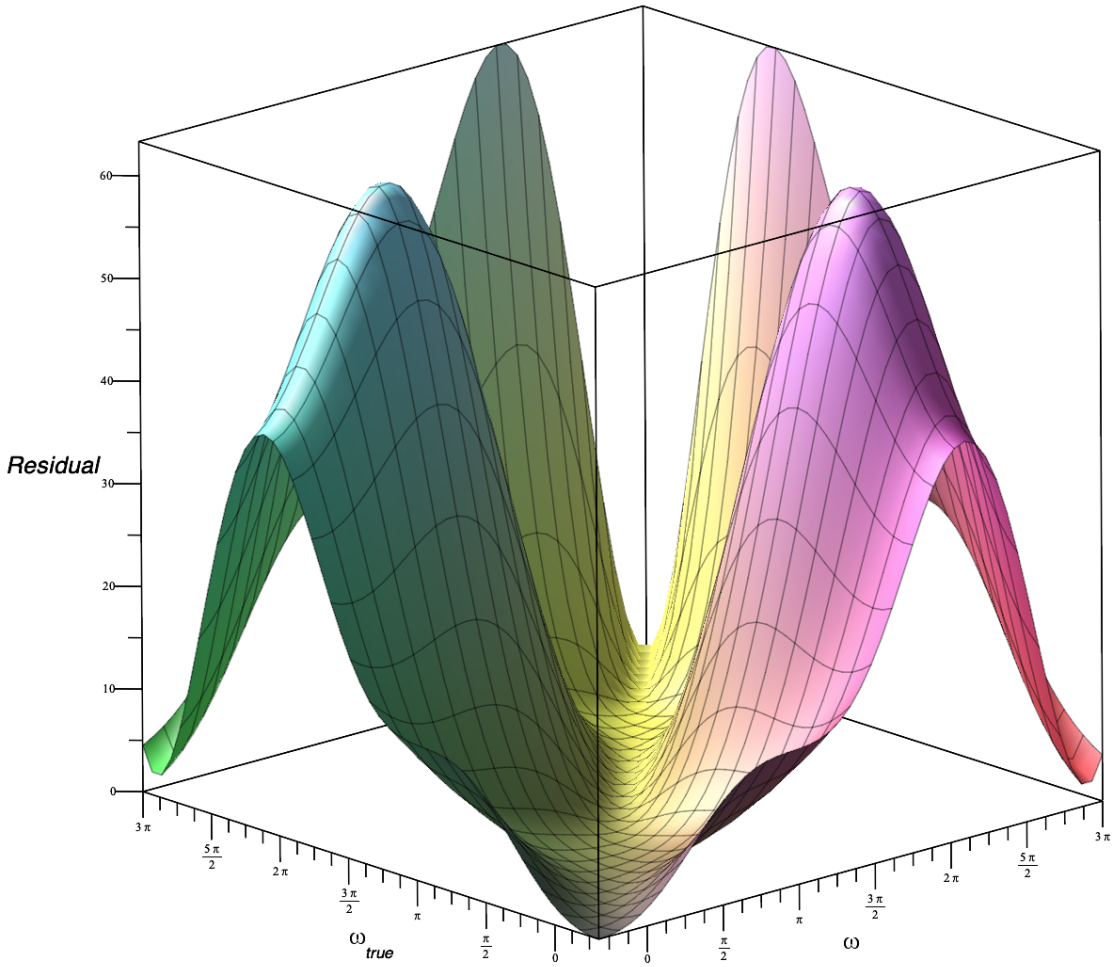



Figure: Plotting $R(\omega)$ for $y_1 = 0.5, y_2 = 0.2$. We see that $R(\omega) = 0$ for only one $\omega$ for each $\omega_{true}$.

# Numerical results: Batch sampling 1D case

There is a strong relationship between the necessary number of input sensor points and the complexity of the input function space. This threshold may relate to the point at which an input function can be uniquely reconstructed from the sensor data. Adding more sensors beyond the threshold may help generalization outside the training domain.

# Numerical results: Batch sampling 1D case

When training, the loss function is computed form all the data points $(f, x, u(x))_i, i = 1, ..., N$ at each training step. We now instead sample a random batch of $M < N$ data points uniformly from the training data set without replacement such that

$$\{(f, x, u(x))_i\}_{i=1}^{N} \mapsto \{(f, x, u(x))_i\}_{i=1}^{M}.$$

Letting $0 < b_r < 1$ denote the batch ratio, we have $M = \lfloor b_r N \rfloor$.
The idea behind this batch sampling strategy is two-fold:

- Faster training with smaller effective batch size.
- Avoid local minima in loss-landscape.

# Numerical results: Batch sampling 1D case

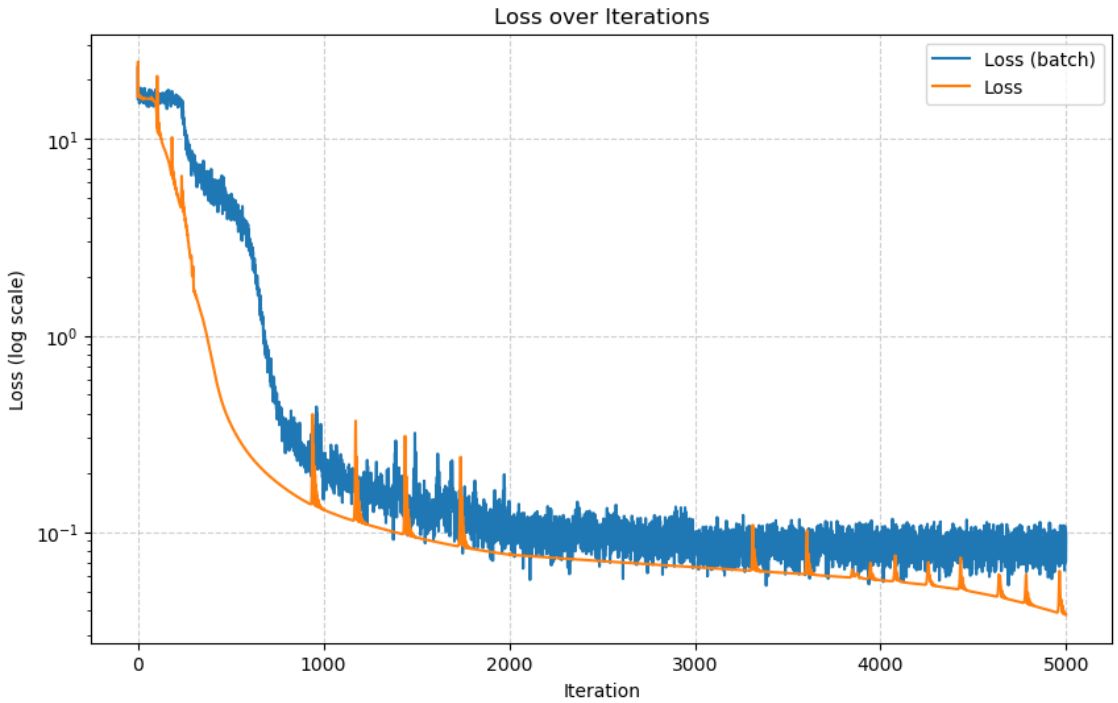



Figure: Loss function vs training iteration with regular training and batch sampling.

# Numerical results: Batch sampling 1D case

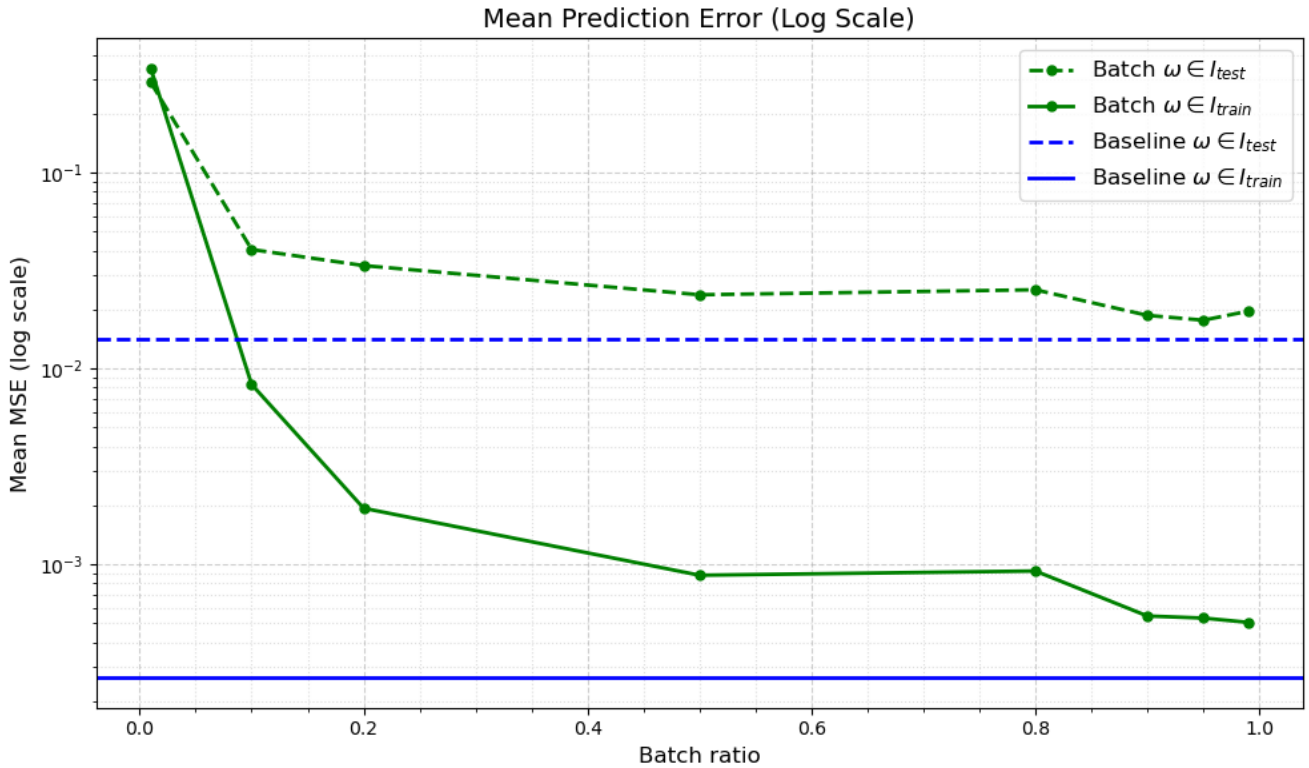



Figure: Train MSE of DON on input functions within the training domain $\Omega$ where $\omega \in [a, b] = I_{train}$ and of DON on input functions outside training domain $\Omega_{train} \setminus \Omega$ where $\omega \in [a - \delta, a] \cup [b, b + \delta] = I_{test}$.

# Numerical results: Batch sampling 1D case

Batch sampling does not improve generalization performance, but significantly increases computational efficiency. Perhaps the simplicity of the 1D problem gives a loss-landspace with few significant local minima, in which the regular training can get stuck.

# Methodology: Data generation convection diffusion equation

For

$$\frac{\partial u}{\partial t} + \mu \frac{\partial u}{\partial x} - \epsilon \frac{\partial^2 u}{\partial x^2} = f(x),\ u(x,0) = \sin(\pi x),\ u(0,t) = u(1,t) = 0$$

where $x, t \in [0,1], \mu = 1, \epsilon = 0.1$ Implemented a implicit finite element solver to generate data with right hand sides chosen as

$$f \in \Omega_4 := \{f(x) = \sum_{n=1}^{4} a_n \cdot \sin(n\pi x), \quad |a_n| \leq 1\}.$$

Solve for N different random right hand sides $f \in \Omega_4$ and sample P random points from Finite element solution. In this way, the same input function $f$ appears in multiple data triplets

$$(f, (x,t), u(x,t))_{i,j} = (f_i, (x_{ij}, t_{ij}), u_i(x_{ij}, t_{ij})) \text{ for } i = 1, ..., N, j = 1, ..., P.$$

For example, with $N, P = 100$ the data set consists of 10.000 data triplets.

# Methodology: DeepONet training convection diffusion equation

The loss function balances between data loss, physics loss, initial value loss and boundary loss by

$$\mathcal{L}_{combined} = \mathcal{L}_{data} + 10 \cdot (\mathcal{L}_{physics} + \mathcal{L}_{initial} + \mathcal{L}_{boundary})$$

$$\mathcal{L}_{data}(\theta) = \frac{1}{NP} \sum_{i,j=1}^{N,P} (G(f_i)(x_{ij}, t_{ij}) - u_i(x_{ij}, t_{ij}))^2$$

$$\mathcal{L}_{physics}(\theta) = \frac{1}{NP} \sum_{i,j=1}^{N,P} (\partial_t G(f_i)(x_{ij}, t_{ij}) + \mu \partial_x G(f_i)(x_{ij}, t_{ij}) - \epsilon \partial_{xx} G(f_i)(x_{ij}, t_{ij}) - f_i(x_{ij}, t_{ij}))^2$$

$$\mathcal{L}_{boundary}(\theta) = \frac{1}{NP} \sum_{i,j=1}^{N,P} ((G(f_i)(0, t_{ij}) - 0)^2 + (G(f_i)(1, t_{ij}) - 0)^2)$$

$$\mathcal{L}_{initial}(\theta) = \frac{1}{NP} \sum_{i,j=1}^{N,P} (G(f_i)(x_{ij}, 0) - u_0(x))^2$$

# Numerical results: Batch sampling convection diffusion equation

Training and validation data is generated as described earlier. Trained DeepONets with batch sampling of 20%, 50% and 100% of the training data. Both train- and validation loss is measured in $\mathcal{L}_{combined}$.

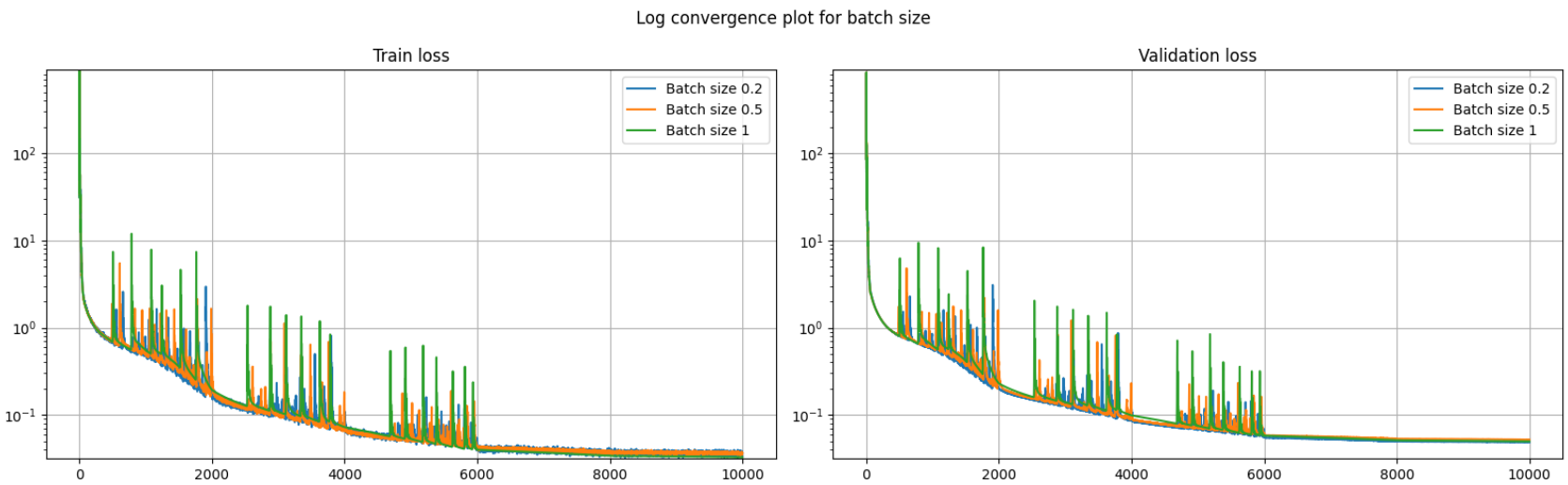



Figure: Train and validation plot for 20%, 50% and 100% batch sizes

# Numerical results: Batch sampling convection diffusion equation

The resulting DON's are tested on an input function space similar to that used for training, and also a larger one.

$$\Omega_4 := \{f(x) = \sum_{n=1}^{4} a_n \cdot \sin(n\pi x), \quad |a_n| \leq 1\}$$

$$\Omega_{test} := \{f(x) = \sum_{n=1}^{4} a_n \cdot \sin(n\pi x), \quad |a_n| \leq 1.5\}$$

| Batch size | Training time | Test loss ($\Omega_4$) | Test loss ($\Omega_{\text{test}}$) |
|---|---|---|---|
| 20% | $\approx 25min$ | $7.33\ 10^{-4}$ | $5.21\ 10^{-3}$ |
| 50% | $\approx 43min$ | $8.41\ 10^{-4}$ | $5.02\ 10^{-3}$ |
| 100% | $\approx 54min$ | $1.07\ 10^{-3}$ | $5.19\ 10^{-3}$ |

Table: Training time and test loss for $\Omega_4$ and $\Omega_{\text{test}}$ measured in $\mathcal{L}_{data}$

# Numerical results: Batch sampling convection diffusion equation

Batch sampling do not generalize the operator learning better than regular training
However, batch sampling generalize as good as regular training and reduces computation time

# Numerical results: Number of input sensors convection diffusion equation

Again we compare how the number of input sensors effect accuracy and generalization of the DeepONets. Data generation as stated earlier where right hand sides are generated as

$$\Omega_4 := \{f(x) = \sum_{n=1}^{4} a_n \cdot \sin(n\pi x), \quad |a_n| \leq 1\}$$

We expect 5 input sensors to be sufficient to get accurate solutions, any more input sensors will be redundant. Input sensors are equally distributed along $x \in [0, 1]$.

# Numerical results: Number of input sensors convection diffusion equation

From the convergence plot below we observe a clear threshold between 4 and 5 input sensors. However, there are no significant increase in accuracy for number of input sensors larger than 5.

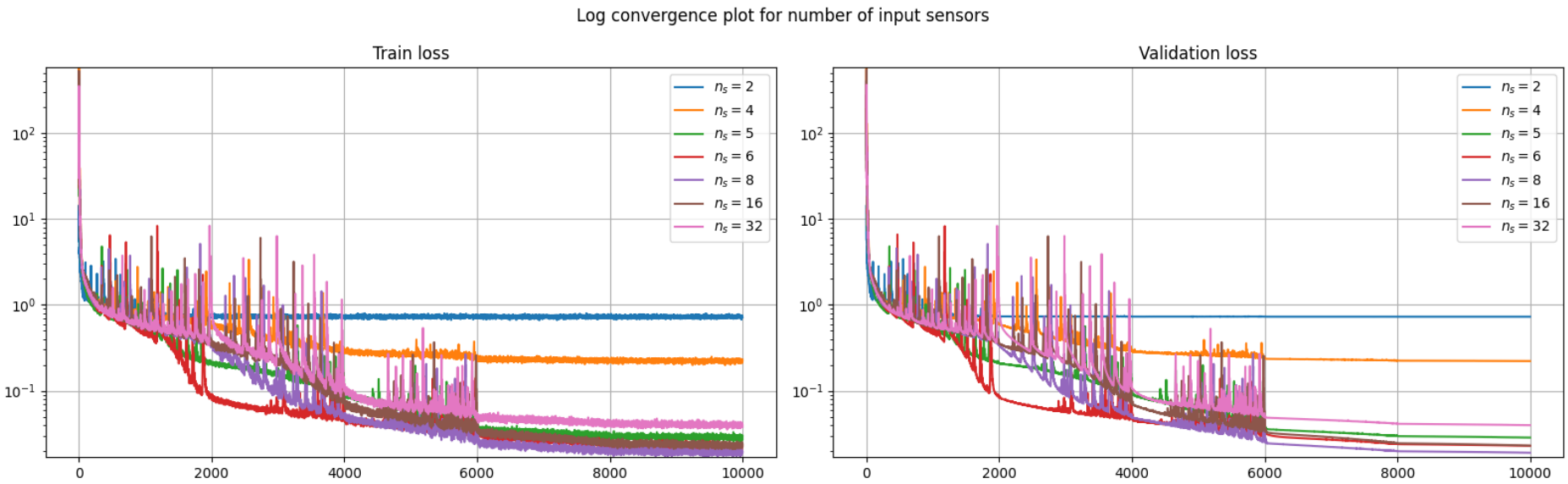



Figure: Log convergence plot for number of input sensors $= \{2,4,5,6,8,16,32\}$, measured in $\mathcal{L}_{combined}$

# Numerical results: Number of input sensors convection diffusion equation

Comparison for different test loss

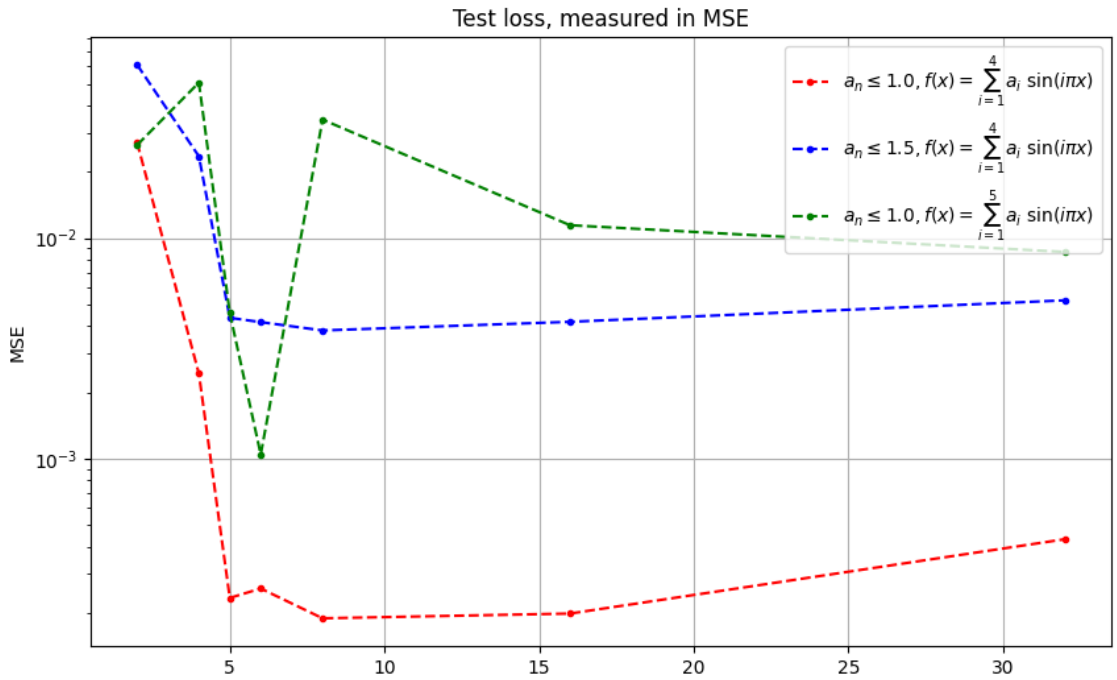



Figure: Comparison of test loss for different right hand side generations for number of input sensors $2, 4, 5, 6, 8, 16, 32$. All measured as Mean Squared Error

# Numerical results: Number of input sensors convection diffusion equation

There is a clear threshold between number of input sensors and number of basis functions in right hand side
Over this threshold the number of input sensors become redundant and one should aim to design DeepONets with a minimal number of sensor points

# Discussion and conclusion

- **Generalization limitations:** DONs struggle to generalize beyond the training input function space. Even within the training space, accuracy correlates with the frequency of similar input functions during training.
- **Sensor point threshold:** There is a strong relationship between the necessary number of input sensor points and the complexity of the input function space. This threshold may relate to the point at which an input function can be uniquely reconstructed from the sensor data. Adding more sensors beyond the threshold may help generalization outside the training domain.
- **Effect of batch sampling:** Batch sampling does not improve generalization performance, but significantly increases computational efficiency. This allows for potentially more training iterations in the same time budget, which could indirectly improve performance.

# Discussion and conclusion

- **Future work - sampling strategies:** Explore alternative batch sampling methods, such as those that sample only spatio-temporal points (as in [SK24]) instead of full data triplets. Their method might enable better learning since all input functions are presented at each iteration.
- **Future work - complex problems:** Test batch sampling strategies on more complex problems with potentially irregular loss landscapes to better understand their effectiveness and general applicability.
- **Future work - sensor distribution:** Investigate the effect of non-uniform input sensor distributions. However, in real-world applications, sensor placement may be constrained by physical or logistical limitations.

# References I

 Somdatta Goswami Sharmila Karumuria, Lori Graham-Bradya, *Efficient training of deep neural operator networks via randomized sampling*, 2024.