Universität Bremen

Center for Industrial Mathematics (ZeTeM)

**Faculty 03**

Mathematics / Computer science

# Outlook: DeepONets

Janek Gödeke, Tom Freudenberg

Bremen, 21.07.2023

# What did we do so far?

Learned PDEs for single parameter

a) For fixed $D \in \mathbb{R}$ solve parachute example:

$$\begin{cases} \partial_t^2 u(t) &= D\big(\partial_t u(t)\big)^2 - g \quad \text{for} \quad t \in [0,3] \\ u(0) &= H \\ \partial_t u(0) &= 0 \end{cases}$$

- Learned function $u_\theta \approx u : [0,3] \to \mathbb{R}$

# What did we do so far?

Learned PDEs for single parameter

a) For fixed $D \in \mathbb{R}$ solve parachute example:

$$\begin{cases} \partial_t^2 u(t) &= D\big(\partial_t u(t)\big)^2 - g \quad \text{for} \quad t \in [0,3] \\ u(0) &= H \\ \partial_t u(0) &= 0 \end{cases}$$

- Learned function $u_\theta \approx u : [0,3] \to \mathbb{R}$

b) For fixed $D \in \mathbb{R}$ solve heat equation:

$$\Delta \partial_t u = D \Delta_x u \quad \text{in} \quad \Omega \times I_t$$

- Learned function $u_\theta \approx u : \Omega \times I_t \to \mathbb{R}$

# What did we do so far?

Learned PDEs for single parameter

a) For fixed $D \in \mathbb{R}$ solve parachute example:

$$\begin{cases} \partial_t^2 u(t) &= D\big(\partial_t u(t)\big)^2 - g \quad \text{for} \quad t \in [0,3] \\ u(0) &= H \\ \partial_t u(0) &= 0 \end{cases}$$

- Learned function $u_\theta \approx u : [0,3] \to \mathbb{R}$

b) For fixed $D \in \mathbb{R}$ solve heat equation:

$$\Delta \partial_t u = D \Delta_x u \quad \text{in} \quad \Omega \times I_t$$

- Learned function $u_\theta \approx u : \Omega \times I_t \to \mathbb{R}$

!! Bonus exercises: PINNs can solve for multiple $D \in [D_{min}, D_{max}]$

$$u_\theta(D, x, t) \approx u_D(x, t)$$

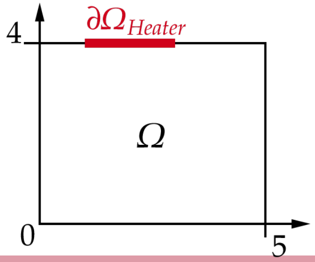
# What did we do so far?

## Learned PDE for single parameter function

- Room-with-heater example

$$
\begin{aligned}
\partial_t u(x,t) &= D\Delta_x u(x,t) && \text{for} \quad (x,t) \in \Omega \times I_t \\
u(x,t) &= h(t) && \text{for} \quad (x,t) \in \partial\Omega_{Heater} \times I_t
\end{aligned}
$$

- Fixed temperature function $h(t)$ of the heater



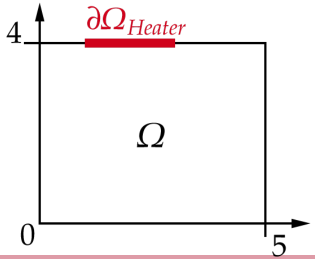
# What did we do so far?

Learned PDE for single parameter function

- Room-with-heater example

$$\begin{aligned} \partial_t u(x,t) &= D\Delta_x u(x,t) && \text{for} \quad (x,t) \in \Omega \times I_t \\ u(x,t) &= h(t) && \text{for} \quad (x,t) \in \partial\Omega_{Heater} \times I_t \end{aligned}$$

- Fixed temperature function $h(t)$ of the heater
- If solution for different $\tilde{h}(t)$ is desirable ...
- ... start new training with PINNs :(



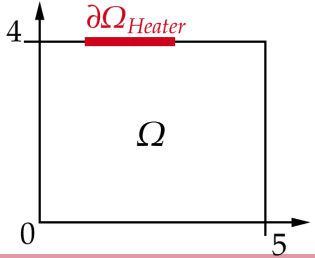
# What did we do so far?

## Learned PDE for single parameter function

- Room-with-heater example

$$
\begin{aligned}
\partial_t u(x,t) &= D\Delta_x u(x,t) && \text{for} \quad (x,t) \in \Omega \times I_t \\
u(x,t) &= h(t) && \text{for} \quad (x,t) \in \partial\Omega_{Heater} \times I_t
\end{aligned}
$$

- Fixed temperature function $h(t)$ of the heater
- If solution for different $\tilde{h}(t)$ is desirable ...
- ... start new training with PINNs :(
- **Goal:** Solve PDE for many $\tilde{h}(t)$ with ONE network



# Simpler Example for Illustration

- **Goal:** For **multiple** functions $f : [0,1] \to \mathbb{R}$ use **single** NN to solve

$$\begin{cases} u^{(1)}(x) & = f(x) \quad \text{for} \quad x \in [0,1] \\ u(0) & = 0 \end{cases}$$

# Simpler Example for Illustration

- **Goal:** For **multiple** functions $f : [0, 1] \to \mathbb{R}$ use **single** NN to solve

$$\begin{cases} u^{(1)}(x) &= f(x) \quad \text{for} \quad x \in [0, 1] \\ u(0) &= 0 \end{cases}$$

- **Second goal:** After training on $f_1, f_2 ...$, want good performance on test set

# Simpler Example for Illustration

- **Goal:** For **multiple** functions $f : [0,1] \to \mathbb{R}$ use **single** NN to solve

$$\begin{cases} u^{(1)}(x) &= f(x) \quad \text{for} \quad x \in [0,1] \\ u(0) &= 0 \end{cases}$$

- **Second goal:** After training on $f_1, f_2 \ldots$, want good performance on test set

$\rightarrow$ Want NN that learns the mapping

$$\mathcal{A} : f \mapsto u_f \qquad \text{"}f \text{ is mapped to corr. solution } u_f\text{"}$$

**Problem:** Inputs of NNs can only be vectors in $\mathbb{R}^m$, but $\mathcal{A}$ receives a function!

# Operator Learning

- Want to learn *solution operator* $\mathcal{A} : f \mapsto u_f$ for

$$\begin{cases} u^{(1)}(x) &= f(x) \quad \text{for} \quad x \in [0, 1] \\ u(0) &= 0 \end{cases}$$

# Operator Learning

- Want to learn *solution operator* $\mathcal{A} : f \mapsto u_f$ for

$$\begin{cases} u^{(1)}(x) & = f(x) \quad \text{for} \quad x \in [0, 1] \\ u(0) & = 0 \end{cases}$$

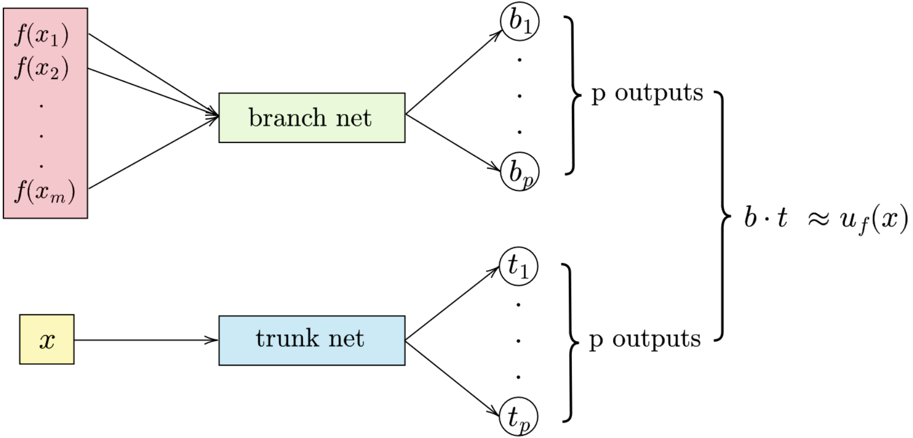
**Idea:**

- Fix points $x_1, \dots, x_m$ in domain of $f$
- Consider NNs of the form

$$u_\theta : \mathbb{R}^m \times [0, 1] \longrightarrow \mathbb{R}$$
$$\big(f(x_1), f(x_2), \dots, f(x_m), x\big) \longmapsto u(x)$$

# DeepONets - Divide and Conquer

(Lu, Jin, Karniadakis; 2019)



# DeepONets - Construction Steps

## Construction of DeepONet

$$u_\theta : \mathbb{R}^m \times [0,1] \longrightarrow \mathbb{R}$$
$$\Big(f(x_1), f(x_2), ..., f(x_m), x\Big) \longmapsto b \cdot t \approx u(x)$$

- Choose and **fix** sample points $x_1, ..., x_m$ for $f$

# DeepONets - Construction Steps

## Construction of DeepONet

$$u_\theta : \mathbb{R}^m \times [0,1] \longrightarrow \mathbb{R}$$
$$\big(f(x_1), f(x_2), ..., f(x_m), x\big) \longmapsto b \cdot t \approx u(x)$$

- Choose and **fix** sample points $x_1, ..., x_m$ for $f$
- Set number $p$ of output neurons of branch/trunk
  $\rightarrow$ Outputs $b, t \in \mathbb{R}^p$ of branch/trunk nets must have same size!

# DeepONets - Construction Steps

## Construction of DeepONet

$$u_\theta : \mathbb{R}^m \times [0,1] \longrightarrow \mathbb{R}$$
$$\Big(f(x_1), f(x_2), ..., f(x_m), x\Big) \longmapsto b \cdot t \approx u(x)$$

- Choose and **fix** sample points $x_1, ..., x_m$ for $f$
- Set number $p$ of output neurons of branch/trunk
  $\rightarrow$ Outputs $b, t \in \mathbb{R}^p$ of branch/trunk nets must have same size!
- Build architecture of branch/trunk net, e.g. fully-connected

# DeepONets - Train with Data

## Train the DeepONet

$$u_\theta : \mathbb{R}^m \times [0,1] \longrightarrow \mathbb{R}$$
$$\Big(f(x_1), f(x_2), ..., f(x_m), x\Big) \longmapsto b \cdot t \approx u(x)$$

- Data: Consider Parameter-solution tuples $(f, u_f)$

# DeepONets - Train with Data

## Train the DeepONet

$$u_\theta : \mathbb{R}^m \times [0,1] \longrightarrow \mathbb{R}$$
$$\Big(f(x_1), f(x_2), \dots, f(x_m), x\Big) \longmapsto b \cdot t \approx u(x)$$

- Data: Consider Parameter-solution tuples $(f, u_f)$
- Data points look like $f(x_1), \dots, f(x_m)$ and $u_f(\tilde{x}_1), \dots, u_f(\tilde{x}_k)$

# DeepONets - Train with Data

## Train the DeepONet

$$u_\theta : \mathbb{R}^m \times [0,1] \longrightarrow \mathbb{R}$$
$$\Big(f(x_1), f(x_2), \ldots, f(x_m), x\Big) \longmapsto b \cdot t \approx u(x)$$

- Data: Consider Parameter-solution tuples $(f, u_f)$
- Data points look like $f(x_1), \ldots, f(x_m)$ and $u_f(\tilde{x}_1), \ldots, u_f(\tilde{x}_k)$
- Loss function for single tuple $(f, u_f)$:

$$\frac{1}{k}\sum_{j=1}^{k}\Big\|u_\theta\big(f(x_1), \ldots, f(x_m), \tilde{x}_j\big) - u_f(\tilde{x}_j)\Big\|^2$$

# DeepONet - Parachute Example

- Learn solution operator $\mathcal{A} : D \mapsto u$ of the ODE

$$\begin{cases} \partial_t^2 u(t) &= D(t)\big(\partial_t u(t)\big)^2 - g \quad \text{for} \quad t \in [0,3] \\ u(0) &= H \\ \partial_t u(0) &= 0 \end{cases}$$

# DeepONet - Parachute Example

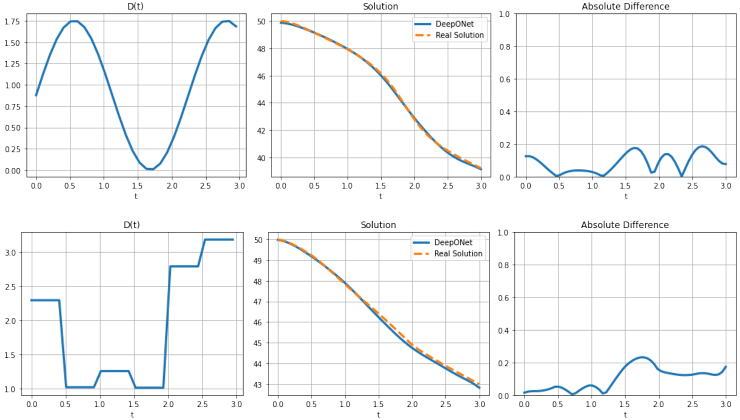
- Learn solution operator $\mathcal{A} : D \mapsto u$ of the ODE

$$\begin{cases} \partial_t^2 u(t) &= D(t)\big(\partial_t u(t)\big)^2 - g \quad \text{for} \quad t \in [0,3] \\ u(0) &= H \\ \partial_t u(0) &= 0 \end{cases}$$

- Time dependent $D$ "=" parachute is opening and closing while falling
- Analytic solution is unknown $\rightarrow$ employ classical schemes to create training data
- Trained with TorchPhysics

# DeepONet - Parachute Example

## Evaluation on test set



# DeepONet Properties

- Mesh-independent
- Once trained: evaluation for given parameters fast
- Good generalization for unseen data
- Multiple parameter functions $\rightarrow$ multiple branch net

# DeepONet Properties

- Mesh-independent
- Once trained: evaluation for given parameters fast
- Good generalization for unseen data
- Multiple parameter functions $\rightarrow$ multiple branch net

- Needs lot of data for training:
  - Expensive and time consuming to obtain
  - Physics-informed DeepONets
- May need problem specific adaptions
- Often trial and error for finding good parameters

# The End

Thank you for participating! :)

Thanks to Tom for his support!