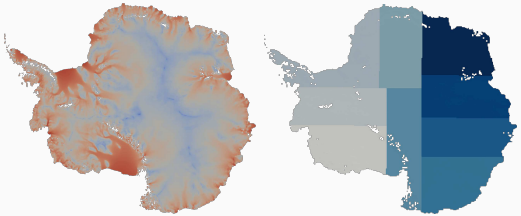
Domain decomposition for neural networks

Alexander Heinlein[1]

Shenzhen Institute of Advanced Technology, Shenzhen, China, July 31, 2024

[1]Delft University of Technology

Domain Decomposition Methods



Images based on **Heinlein, Perego, Rajamanickam (2022)**

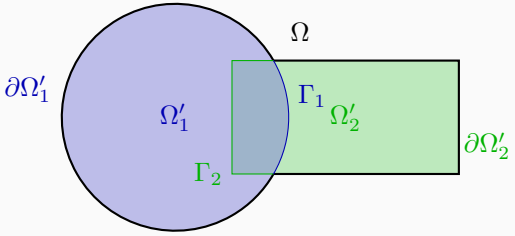
Idea

Decomposing a large **global problem** into smaller **local problems**:

- **Better robustness** and **scalability** of numerical solvers
- **Improved computational efficiency**
- Introduce **parallelism**

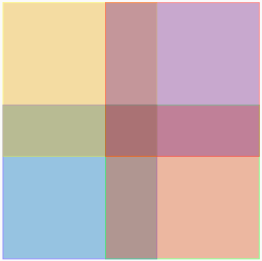
Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

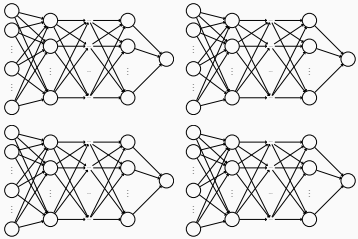
- Schwarz used the algorithm to establish the **existence of harmonic functions** with prescribed boundary values on **regions with non-smooth boundaries**.

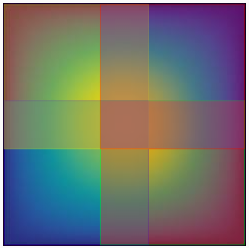


Domain Decomposition for Neural Networks

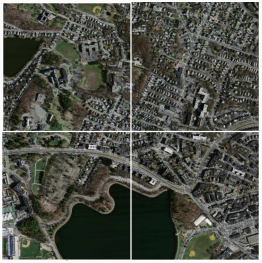
I)







II)



Domain Decomposition Methods and Machine Learning – Literature

A **non-exhaustive literature overview**:

- **Machine Learning for adaptive BDDC, FETI–DP, and AGDSW**: **Heinlein, Klawonn, Lanser, Weber (2019, 2020, 2021, 2021, 2021, 2022); Klawonn, Lanser, Weber (2024)**
- **cPINNs, XPINNs**: **Jagtap, Kharazmi, Karniadakis (2020); Jagtap, Karniadakis (2020)**
- **Classical Schwarz iteration for PINNs or DeepRitz (D3M, DeepDDM, etc)**:: **Li, Tang, Wu, and Liao (2019); Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2022, arXiv 2023); Kim, Yang (2022, arXiv 2023)**
- **FBPINNs, FBKANs**: **Moseley, Markham, and Nissen-Meyer (2023); Dolean, Heinlein, Mishra, Moseley (2024, 2024); Heinlein, Howard, Beecroft, Stinis (acc. 2024 / arXiv:2401.07888); Howard, Jacob, Murphy, Heinlein, Stinis (arXiv:2406.19662)**
- **DDMs for CNNs**: **Gu, Zhang, Liu, Cai (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (acc. 2024); Verburg, Heinlein, Cyr (subm. 2024)**

An overview of the state-of-the-art in early 2021:

A. Heinlein, A. Klawonn, M. Lanser, J. Weber

Combining machine learning and domain decomposition methods for the solution of partial differential equations — A review

GAMM-Mitteilungen. 2021.

An overview of the state-of-the-art in late 2023:

A. Klawonn, M. Lanser, J. Weber

Machine learning and domain decomposition methods – a survey

arXiv:2312.14050. 2023

Outline

1. Multilevel domain decomposition-based architectures for physics-informed neural networks

   Based on joint work with

   Victorita Dolean (University of Strathclyde, University Côte d'Azur)

   Ben Moseley and **Siddhartha Mishra** (ETH Zürich)

2. Multifidelity domain decomposition-based physics-informed neural networks for time-dependent problems

   Based on joint work with

   Damien Beecroft (University of Washington)

   Amanda A. Howard and Panos Stinis (Pacific Northwest National Laboratory)

3. Domain Decomposition for Convolutional Neural Networks

   Based on joint work with

   Eric Cyr (Sandia National Laboratories)

   Corné Verburg (Delft University of Technology)

Multilevel domain decomposition-based architectures for physics-informed neural networks

Artificial Neural Networks for Solving Ordinary and Partial Differential Equations

Isaac Elias Lagaris, Aristidis Likas, *Member, IEEE,* and Dimitrios I. Fotiadis

Published in IEEE Transactions on Neural Networks, Vol. 9, No. 5, 1998.

Approach

Solve a general differential equation subject to boundary conditions

$$G(\boldsymbol{x}, \Psi(\boldsymbol{x}), \nabla\Psi(\boldsymbol{x}), \nabla^2\Psi(\boldsymbol{x})) = 0 \quad \text{in } \Omega$$

by solving an **optimization problem**

$$\min_{\theta} \sum_{\boldsymbol{x}_i} G(\boldsymbol{x}_i, \Psi_t(\boldsymbol{x}_i, \theta), \nabla\Psi_t(\boldsymbol{x}_i, \theta), \nabla^2\Psi_t(\boldsymbol{x}_i, \theta))^2$$

where $\Psi_t\big(\boldsymbol{x}, \theta\big)$ is a **trial function**, $\boldsymbol{x}_i$ **sampling points inside the domain** Ω and θ are **adjustable parameters**.

Construction of the trial functions

The trial functions **satisfy the boundary conditions explicitly**:

$$\Psi_t(\boldsymbol{x}, \theta) = A(\boldsymbol{x}) + F(\boldsymbol{x}, \mathrm{NN}(\boldsymbol{x}, \theta))$$

- NN is a **feedforward neural network** with **trainable parameters** θ and input $x \in \mathbb{R}^n$
- A and F are **fixed functions**, chosen s.t.:
  - A **satisfies the boundary conditions**
  - F **does not contribute to the boundary conditions**

Earlier related work: **Dissanayake & Phan-Thien (1994)**

Neural Networks for Solving Differential Equations

Approach

Solve a general differential equation subject to boundary conditions

$$G(\boldsymbol{x}, \Psi(\boldsymbol{x}), \nabla\Psi(\boldsymbol{x}), \nabla^2\Psi(\boldsymbol{x})) = 0 \quad \text{in } \Omega$$

by solving an **optimization problem**

$$\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}_i} G(\boldsymbol{x}_i, \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}), \nabla\Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}), \nabla^2\Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}))^2$$

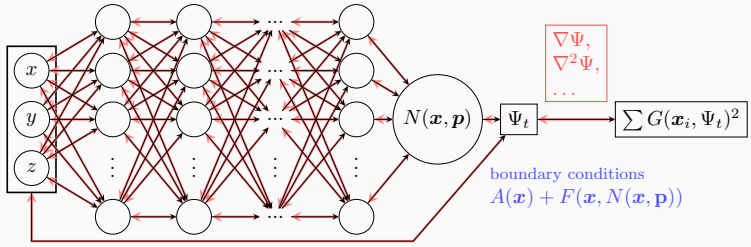
where $\Psi_t(\boldsymbol{x}, \boldsymbol{\theta})$ is a **trial function**, $\boldsymbol{x}_i$ **sampling points inside the domain** Ω and $\boldsymbol{\theta}$ are **adjustable parameters**.

Construction of the trial functions

The trial functions **satisfy the boundary conditions explicitly**:

$$\Psi_t(\boldsymbol{x}, \boldsymbol{\theta}) = A(\boldsymbol{x}) + F(\boldsymbol{x}, \mathrm{NN}(\boldsymbol{x}, \boldsymbol{\theta}))$$

- NN is a **feedforward neural network** with **trainable parameters** $\boldsymbol{\theta}$ and input $x \in \mathbb{R}^n$
- A and F are **fixed functions**, chosen s.t.:
  - A **satisfies the boundary conditions**
  - F **does not contribute to the boundary conditions**



Lagaris et. al's Method – Motivation

Solve the **boundary value problem**

$$\Delta\Psi_t(\boldsymbol{x},\boldsymbol{\theta}) + 1 = 0 \text{ on } [0,1],$$
$$\Psi_t(0,\boldsymbol{\theta}) = \Psi_t(1,\boldsymbol{\theta}) = 0,$$

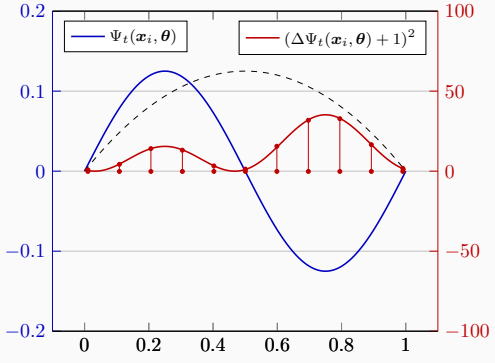
via a **collocation approach**:

$$\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}_i} \left(1 - \Delta\Psi_t(\boldsymbol{x}_i,\boldsymbol{\theta})\right)^2$$

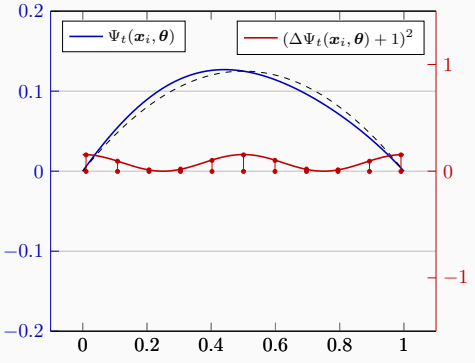
Boundary conditions

The boundary conditions can be **enforced explicitly**, for instance, via the ansatz:

$$\Psi_t(\boldsymbol{x},\boldsymbol{\theta}) = \sin(\pi\boldsymbol{x}) \cdot F(\boldsymbol{x}, \mathrm{NN}(\boldsymbol{x},\boldsymbol{\theta}))$$



$(\Delta\Psi_t(\boldsymbol{x}_i,\boldsymbol{\theta}) + 1)^2 >> 0$



$(\Delta\Psi_t(\boldsymbol{x}_i,\boldsymbol{\theta}) + 1)^2 \approx 0$

Physics-Informed Neural Networks (PINNs)

In the **physics-informed neural network (PINN)** approach introduced by **Raissi et al. (2019)**, a **neural network** is employed to **discretize a partial differential equation**

$$\mathcal{N}[u] = f, \quad \text{in } \Omega.$$

PINNs use a **hybrid loss function**:

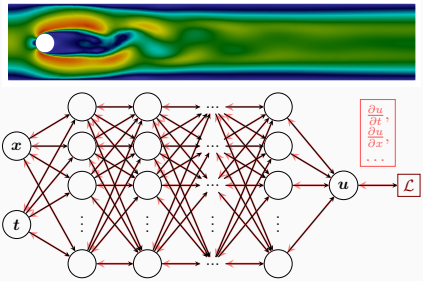
$$\mathcal{L}(\boldsymbol{\theta}) = \omega_{\text{data}}\mathcal{L}_{\text{data}}(\boldsymbol{\theta}) + \omega_{\text{PDE}}\mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}),$$

where ω_{data} and ω_{PDE} are **weights** and

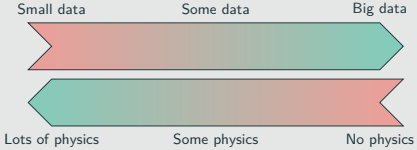
$$\mathcal{L}_{\text{data}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(u(\hat{\boldsymbol{x}}_i, \boldsymbol{\theta}) - u_i\right)^2,$$

$$\mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \left(\mathcal{N}[u](\boldsymbol{x}_i, \boldsymbol{\theta}) - f(\boldsymbol{x}_i)\right)^2.$$

See also **Dissanayake and Phan-Thien (1994); Lagaris et al. (1998)**.



Hybrid loss



- **Known solution values** can be included in $\mathcal{L}_{\text{data}}$
- **Initial and boundary conditions** are also included in $\mathcal{L}_{\text{data}}$

Advantages

- **"Meshfree"**
- **Small data**
- **Generalization properties**
- **High-dimensional problems**
- **Inverse** and **parameterized problems**

Drawbacks

- **Training cost** and **robustness**
- **Convergence not well-understood**
- **Difficulties with scalability** and **multi-scale problems**

Available Theoretical Results for PINNs – An Example

Mishra and Molinaro. *Estimates on the generalisation error of PINNs*, **2022**

Estimate of the generalization error

The generalization error (or total error) satisfies

$$\mathcal{E}_G \leq C_{\text{PDE}} \mathcal{E}_T + C_{\text{PDE}} C_{\text{quad}}^{1/p} N^{-\alpha/p}$$

where

- $\mathcal{E}_G = \mathcal{E}_G(\boldsymbol{X}, \theta) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **general. error** (V Sobolev space, $\boldsymbol{X}$ training data set)
- $\mathcal{E}_T$ **training error** (l^p **loss** of the residual of the PDE)
- N **number of the training points** and α **convergence rate of the quadrature**
- C_{PDE} and C_{quad} **constants** depending on the **PDE** respectively the **quadrature** as well as on the **neural network**

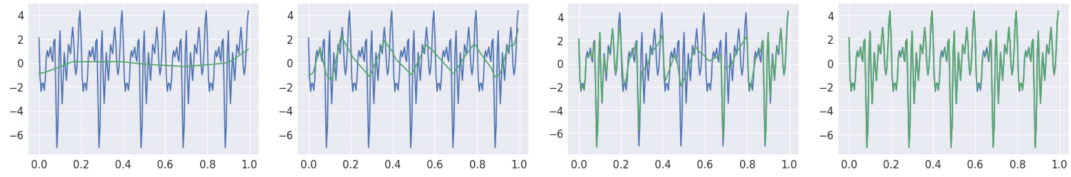
Rule of thumb:

"As long as the PINN is trained well, it also generalizes well"

Scaling Issues in Neural Network Training

Spectral bias

Neural networks **prioritize learning lower frequency functions** first irrespective of their amplitude.



Rahaman et al., *On the spectral bias of neural networks*, **ICML (2019)**

- Solving solutions on **large domains and/or with multiscale features** potentially requires **very large neural networks**.
- Training may **not sufficiently reduce the loss** or take **large numbers of iterations**.
- Significant **increase on the computational work**

Dependence on the choice of **activation functions**: **Hong et al. (arXiv 2022)**

Convergence analysis of PINNs via the **neural tangent kernel**: **Wang, Yu, Perdikaris,** *When and why PINNs fail to train: A neural tangent kernel perspective*, **JCP (2022)**

Motivation – Some Observations on the Performance of PINNs

Solve

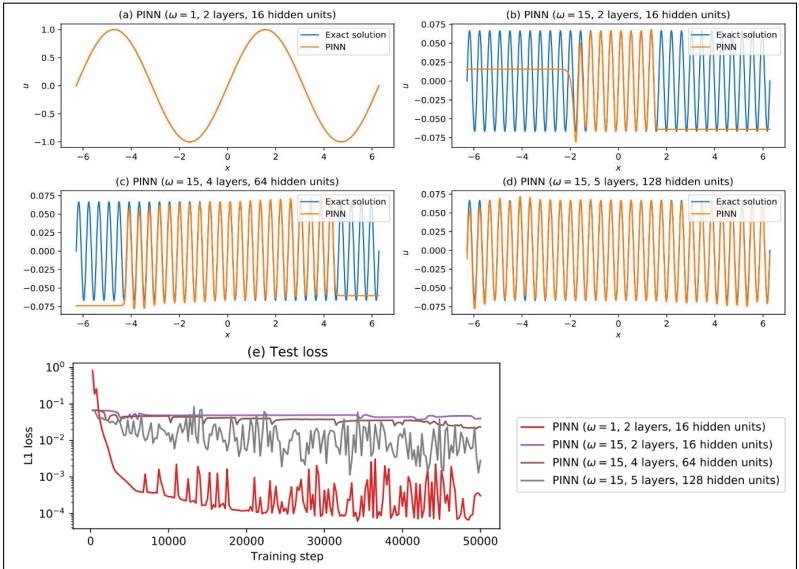
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

for different values of ω using **PINNs with varying network capacities**.

Scaling issues

- Large computational domains
- Small frequencies

Cf. **Moseley, Markham, and Nissen-Meyer (2023)**



(a) 321 free parameters (d) 66 433 free parameters

Motivation – Some Observations on the Performance of PINNs

Solve

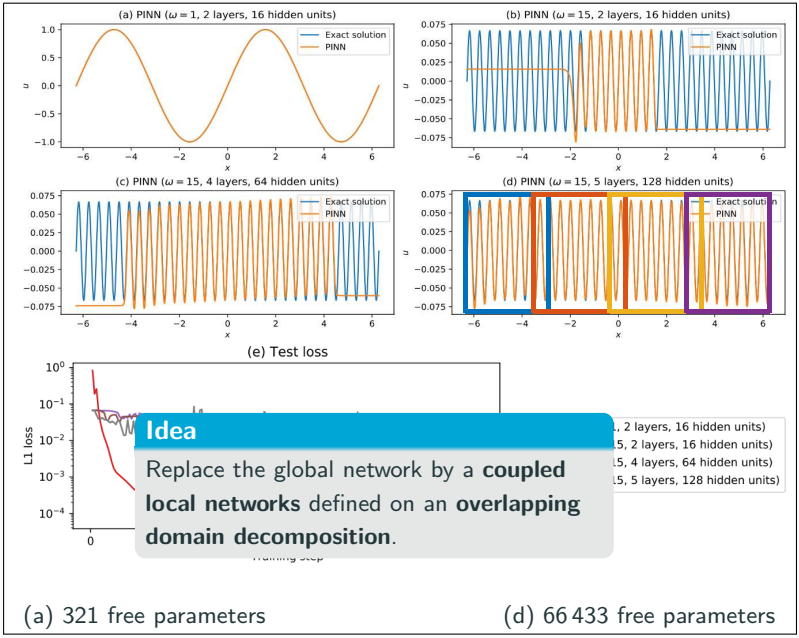
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

for different values of ω using **PINNs with varying network capacities**.

Scaling issues

- Large computational domains
- Small frequencies

Cf. **Moseley, Markham, and Nissen-Meyer (2023)**



Idea

Replace the global network by a **coupled local networks** defined on an **overlapping domain decomposition**.

(a) 321 free parameters (d) 66 433 free parameters

Finite Basis Physics-Informed Neural Networks (FBPINNs)

In the **finite basis physics informed neural network (FBPINNs) method** introduced in **Moseley, Markham, and Nissen-Meyer (2023)**, we employ the **PINN** approach and **hard enforcement of the boundary conditions**; cf. **Lagaris et al. (1998)**.

FBPINNs use the **network architecture**

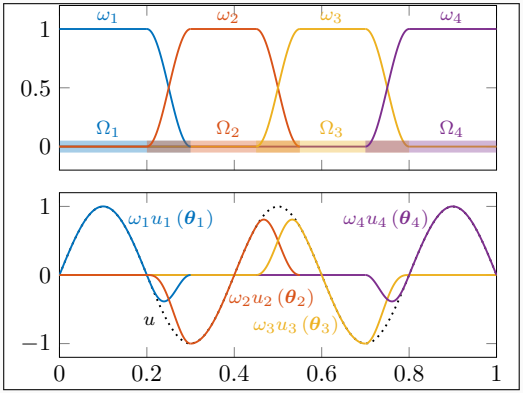
$$u(\theta_1, \dots, \theta_J) = \mathcal{C} \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

$$\mathcal{L}(\theta_1, \dots, \theta_J) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{N}[\mathcal{C} \sum_{x_i \in \Omega_j} \omega_j u_j](x_i, \theta_j) - f(x_i) \right)^2.$$

Here:

- **Overlapping DD**: $\Omega = \bigcup_{l=1}^{J} \Omega_j$
- **Partition of unity** ω_j with $\text{supp}(\omega_j) \subset \Omega_j$ and $\sum_{j=1}^{J} \omega_j \equiv 1$ on Ω



Hard enf. of boundary conditions

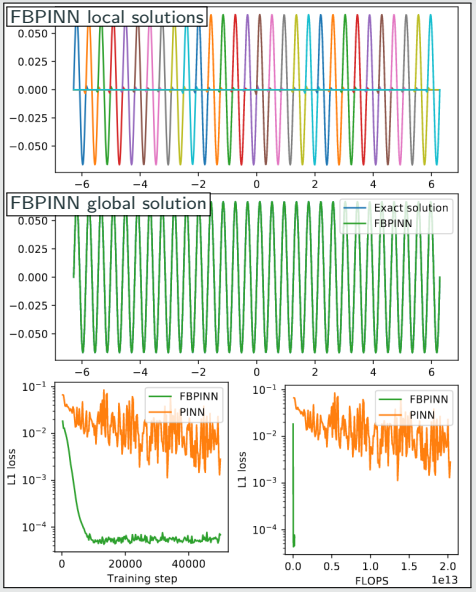
Loss function

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{N}[\mathcal{C}u](x_i, \theta) - f(x_i))^2,$$

with constraining operator $\mathcal{C}$, which **explicitly enforces the boundary conditions**.

Numerical Results for FBPINNs

PINN vs FBPINN (Moseley et al. (2023))



Scalability of FBPINNs

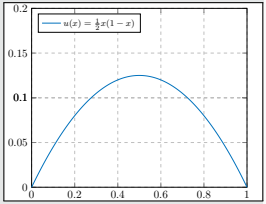
Consider the **simple boundary value problem**

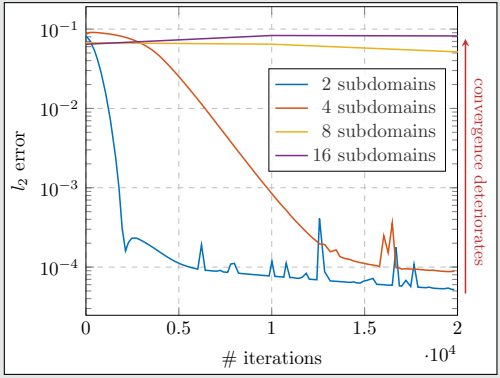
$$-u'' = 1 \text{ in } [0,1],$$

$$u(0) = u(1) = 0,$$

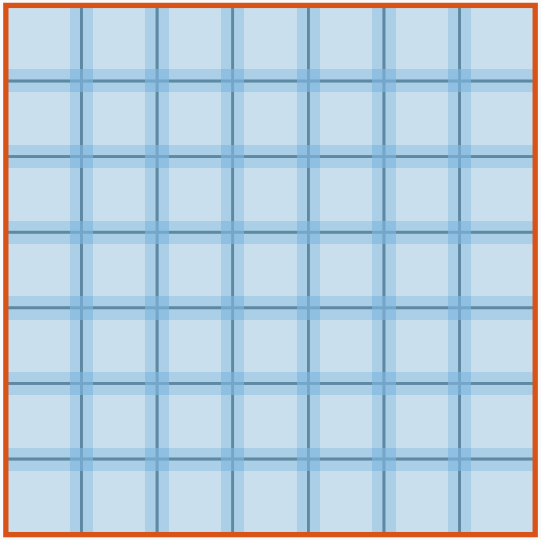
which has the **solution**

$$u(x) = 1/2 x(1-x).$$

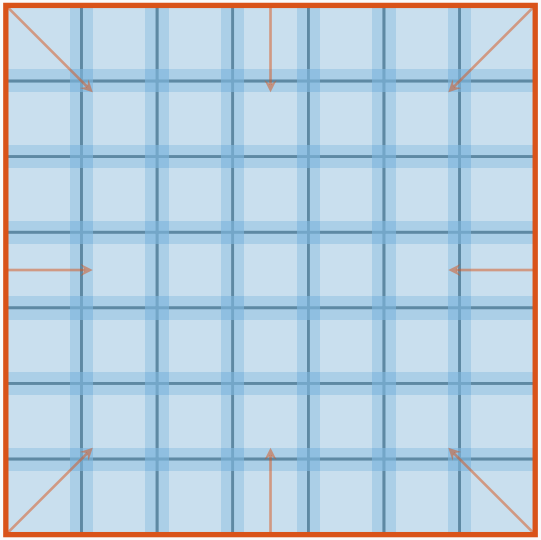




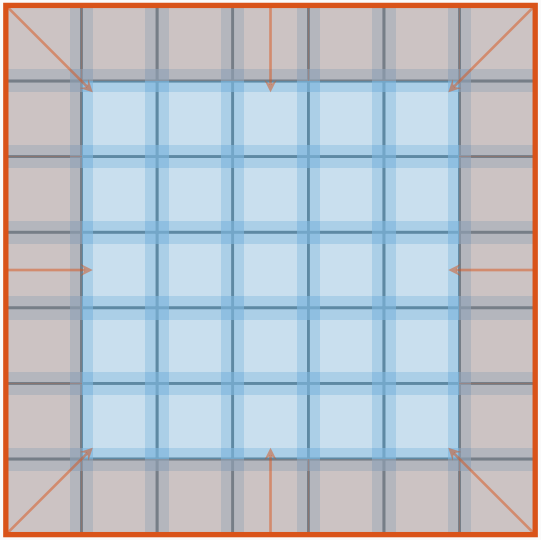
Transport of Information – One-Level Overlapping Schwarz Methods



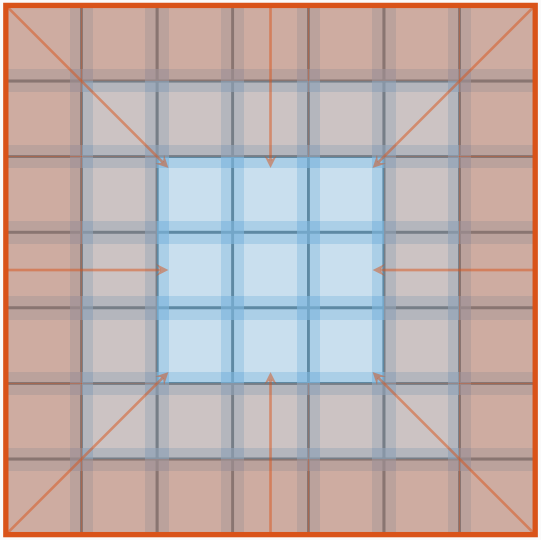
Transport of Information – One-Level Overlapping Schwarz Methods



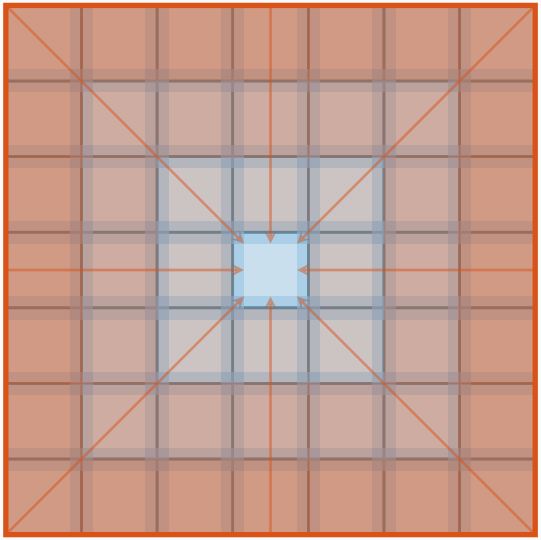
Transport of Information – One-Level Overlapping Schwarz Methods



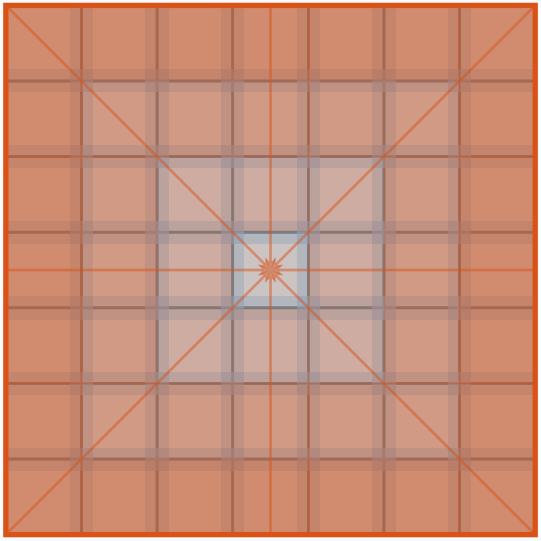
Transport of Information – One-Level Overlapping Schwarz Methods



Transport of Information – One-Level Overlapping Schwarz Methods



Transport of Information – One-Level Overlapping Schwarz Methods



Information (in particular, boundary data) is **only exchanged via the overlapping regions**, leading to **slow convergence** $\rightarrow$ establish a **faster / global transport of information**.

Multi-Level FBPINN Algorithm

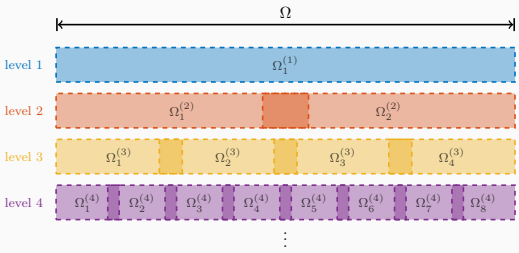
We introduce a **hierarchy of L overlapping domain decompositions**

$$\Omega = \bigcup_{j=1}^{J^{(l)}} \Omega_j^{(l)}$$

and corresponding window functions $\omega_j^{(l)}$ with

$$\text{supp}(\omega_j^{(l)}) \subset \Omega_j^{(l)} \text{ and } \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} \equiv 1 \text{ on } \Omega.$$

This yields the ***L*-level FBPINN algorithm**:

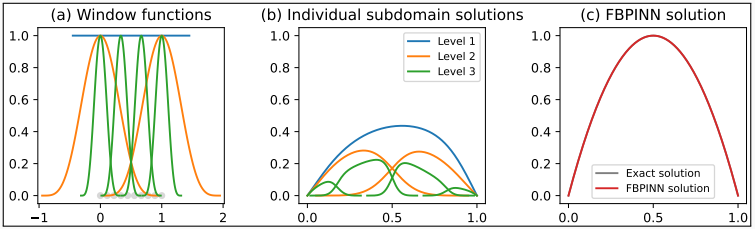


L-level network architecture

$$u\big(\theta_1^{(1)}, \ldots, \theta_{J^{(L)}}^{(L)}\big) = \mathcal{C}\Big(\sum_{l=1}^{L} \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} u_j^{(l)}(\theta_j^{(l)})\Big)$$

Loss function

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \Big(\mathcal{H}[\mathcal{C} \sum_{x_i \in \Omega_j^{(l)}} \omega_j^{(l)} u_j^{(l)}](x_i, \theta_j^{(l)}) - f(x_i)\Big)^2$$

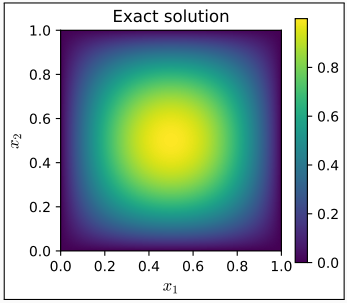


Multilevel FBPINNs – 2D Laplace

Let us consider the **simple two-dimensional boundary value problem**

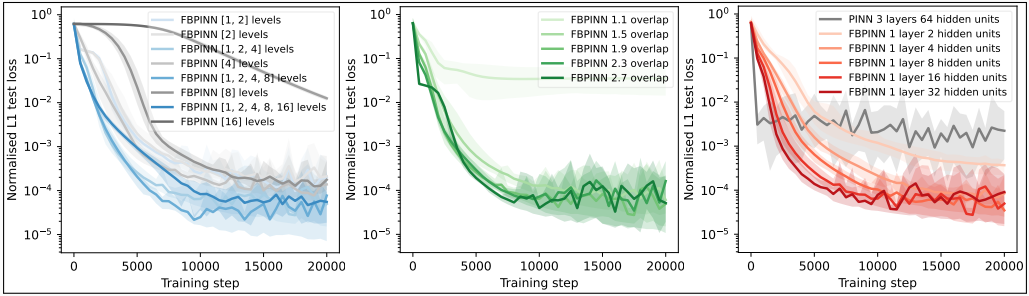
$$-\Delta u = 32(x(1-x)+y(1-y)) \quad \text{in } \Omega = [0,1]^2,$$
$$u = 0 \quad \text{on } \partial\Omega,$$

which has the **solution** $u(x,y) = 16(x(1-x)y(1-y))$.



Baseline model:

# levels	# hidden units	overlap δ
3	16	1.9



Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.

Implementation using JAX

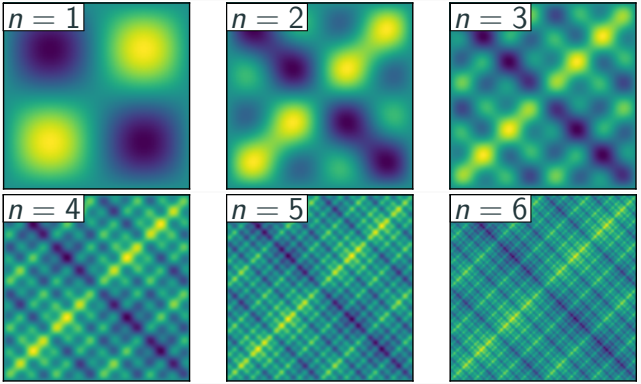
Multi-Frequency Problem

Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

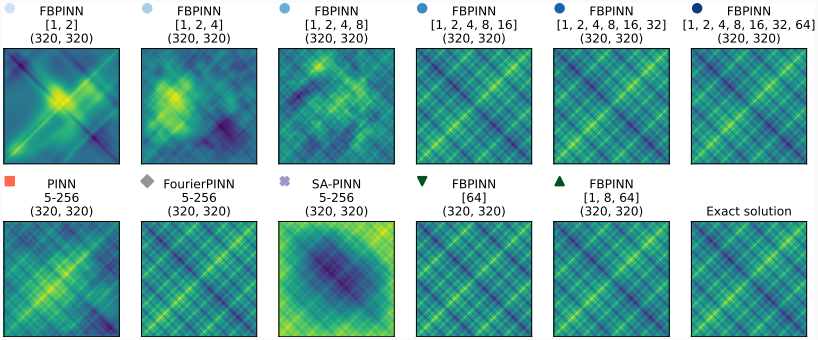
$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega = [0,1]^2, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

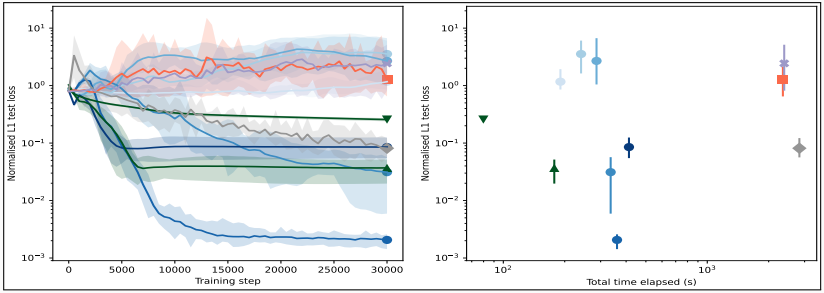
with $\omega_i = 2^i$.

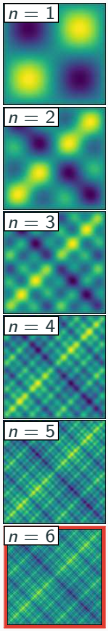
For increasing values of n, we obtain the **analytical solutions**:



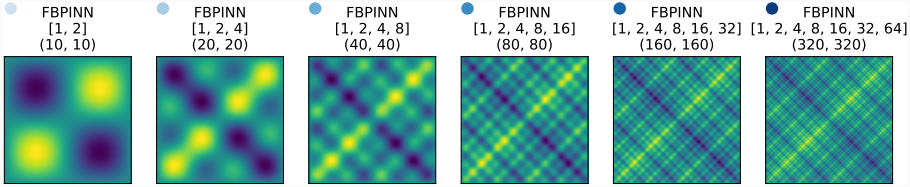
Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling

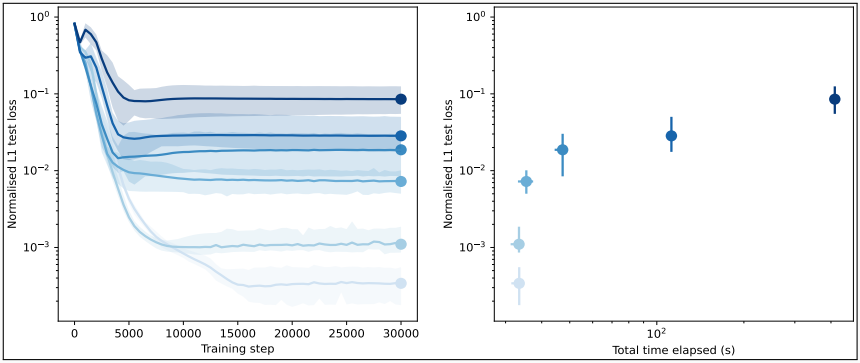


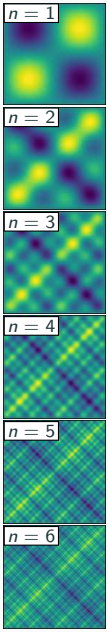




Multi-Level FBPINNs for a Multi-Frequency Problem – Weak Scaling







- Ongoing: **analysis and improvement of the convergence**

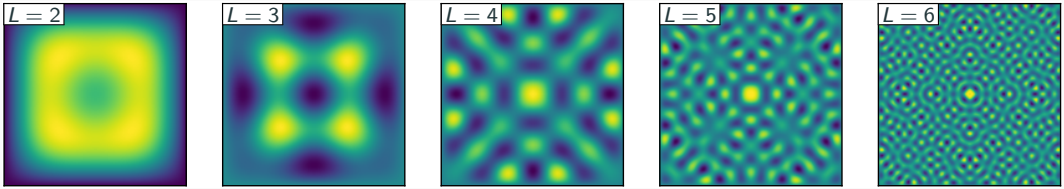
Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.

Helmholtz Problem

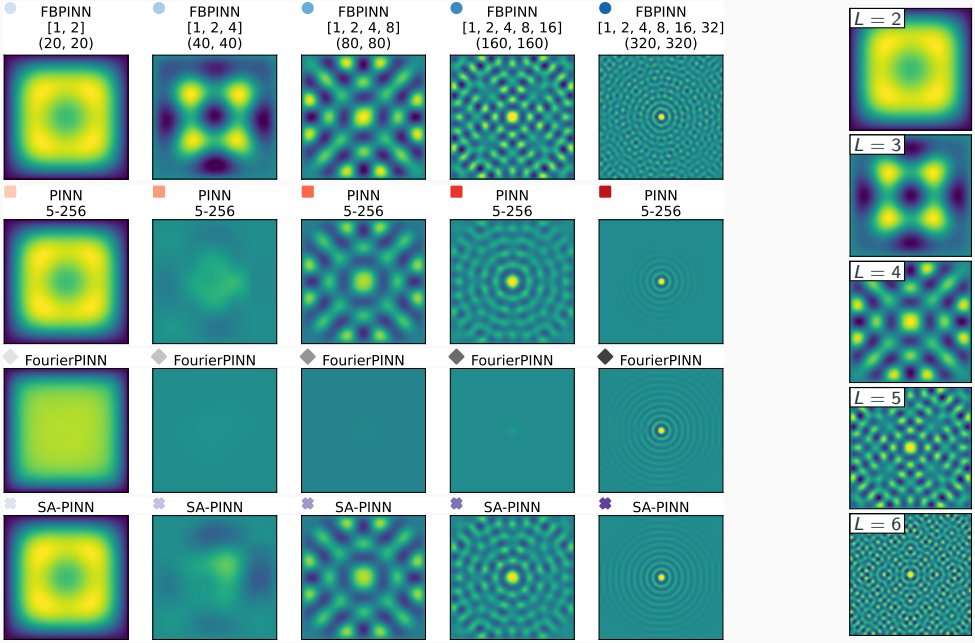
Finally, let us consider the **two-dimensional Helmholtz boundary value problem**

$$\begin{aligned} \Delta u - k^2 u &= f \quad \text{in } \Omega = [0,1]^2, \\ u &= 0 \quad \text{on } \partial\Omega, \\ f(\boldsymbol{x}) &= e^{-\frac{1}{2}(\|\boldsymbol{x}-0.5\|/\sigma)^2}. \end{aligned}$$

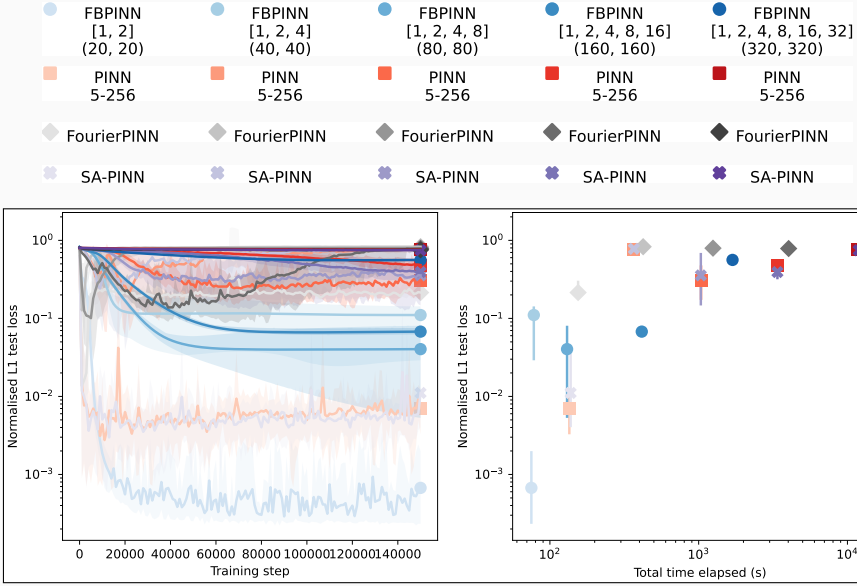
With $k = 2^L\pi/1.6$ and $\sigma = 0.8/2^L$, we obtain the **solutions**:

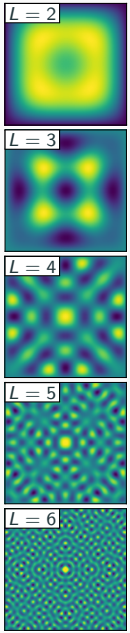


Multi-Level FBPINNs for the Helmholtz Problem – Weak Scaling



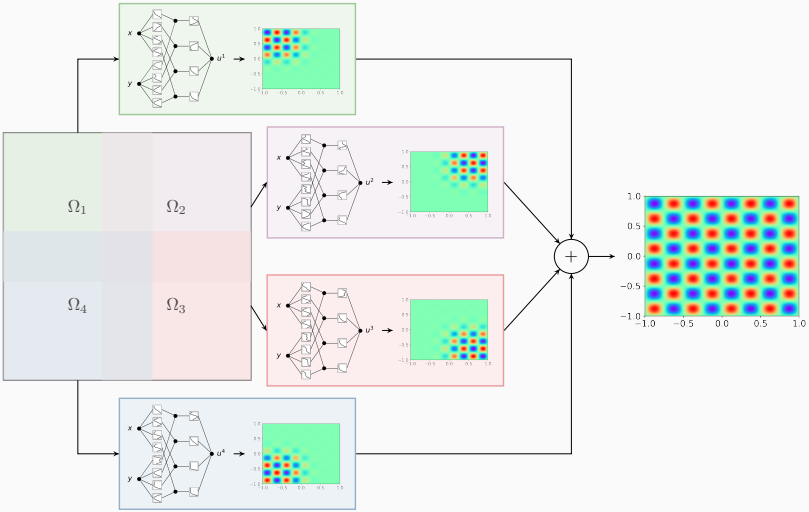
Multi-Level FBPINNs for the Helmholtz Problem – Weak Scaling





Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.

Extension to Kolmogorov–Arnold Networks



→ **Accurate results** for **noisy data** due to use of trainable **spline-based activation functions**.

Cf. **Howard, Jacob, Murphy, Heinlein, Stinis (arXiv:2406.19662)**.

Multifidelity domain decomposition-based physics-informed neural networks for time-dependent problems

PINNs for Time-Dependent Problems

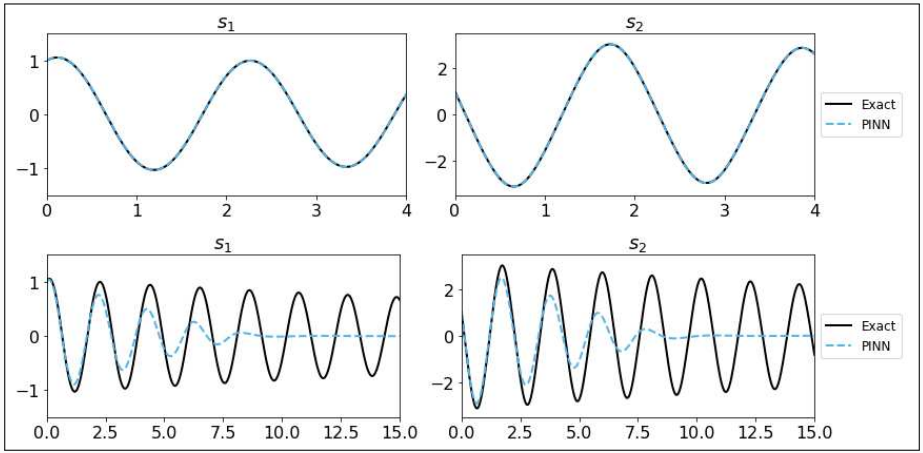
We investigate the performance of PINNs for **time-dependent problems**. Therefore, consider the simple **pedulum problem**:

$$\frac{d\vartheta_1}{dt} = \vartheta_2,$$
$$\frac{d\vartheta_2}{dt} = -\frac{b}{m}\vartheta_2 - \frac{g}{L}\sin(\vartheta_1).$$

Problem parameters

$m = L = 1$, $b = 0.05$, $g = 9.81$

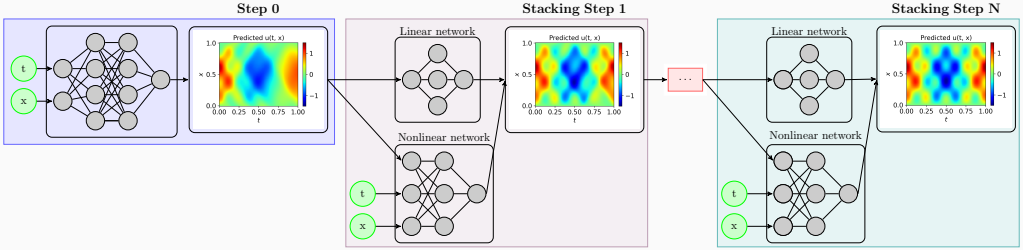
- **Top:** $T = 4$
- **Bottom:** $T = 20$



Stacking Multifidelity PINNs

In the **stacking multifidelity PINNs approach** introduced in **Howard, Murphy, Ahmed, Stinis (arXiv 2023)**, **multiple PINNs are trained in a recursive way.** In each step, a model u^{MF} is trained based on the previous model u^{SF}:

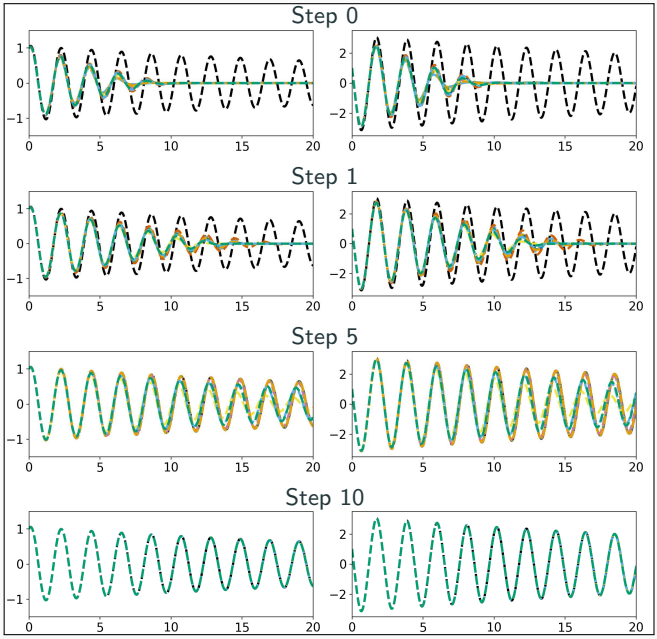
$$u^{MF}(\boldsymbol{x}, \boldsymbol{\theta}^{MF}) = (1 - |\alpha|)\, u^{MF}_{\text{linear}}(\boldsymbol{x}, \boldsymbol{\theta}^{MF}, u^{SF}) + |\alpha|\, u^{MF}_{\text{nonlinear}}(\boldsymbol{x}, \boldsymbol{\theta}^{MF}, u^{SF})$$



Related works (non-exhaustive list)

- **Cokriging & multifidelity Gaussian process regression:** E.g., **Wackernagel (1995); Perdikaris et al. (2017); Babaee et al. (2020)**
- **Multifidelity PINNs & DeepONet: Meng and Karniadakis (2020); Howard, Fu, and Stinis (arXiv 2023); Howard, Perego, Karniadakis, Stinis (2023); Murphy, Ahmed, Stinis (arXiv 2023)**
- **Galerkin, multi-level, and multi-stage neural networks: Ainsworth and Dong (2021); Ainsworth and Dong (2022); Aldirany et al. (arXiv 2023); Wang and Lai (arXiv 2023)**

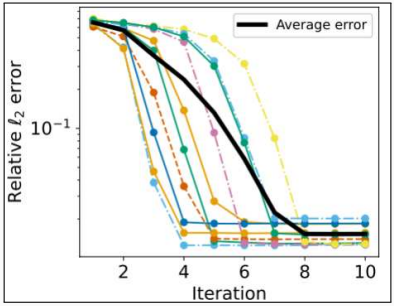
Stacking Multifidelity PINNs for the Pendulum Problem



Pendulum problem:

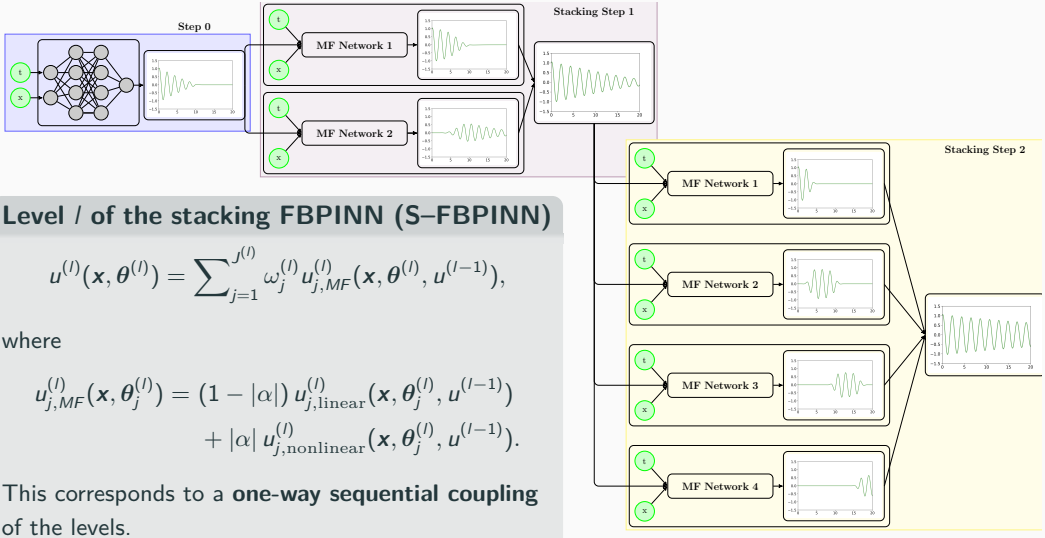
$$\frac{d\vartheta_1}{dt} = \vartheta_2,$$
$$\frac{d\vartheta_2}{dt} = -\frac{b}{m}\vartheta_2 - \frac{g}{L}\sin(\vartheta_1).$$

with $m = L = 1$, $b = 0.05$, $g = 9.81$, and $T = 20$.



Stacking Multifidelity FBPINNs

In **Heinlein, Howard, Beecroft, and Stinis (acc. 2024 / arXiv:2401.07888)**, we **combine stacking multifidelity PINNs with FBPINNs** by using an FBPINN model in each stacking step.



Level l of the stacking FBPINN (S–FBPINN)

$$u^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}^{(l)}) = \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} u_{j,MF}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}^{(l)}, u^{(l-1)}),$$

where

$$u_{j,MF}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}) = (1 - |\alpha|)\, u_{j,\text{linear}}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}, u^{(l-1)}) + |\alpha|\, u_{j,\text{nonlinear}}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}, u^{(l-1)}).$$

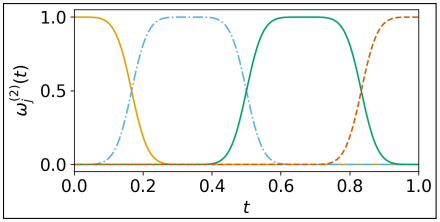
This corresponds to a **one-way sequential coupling** of the levels.

Numerical Results – Pendulum Problem

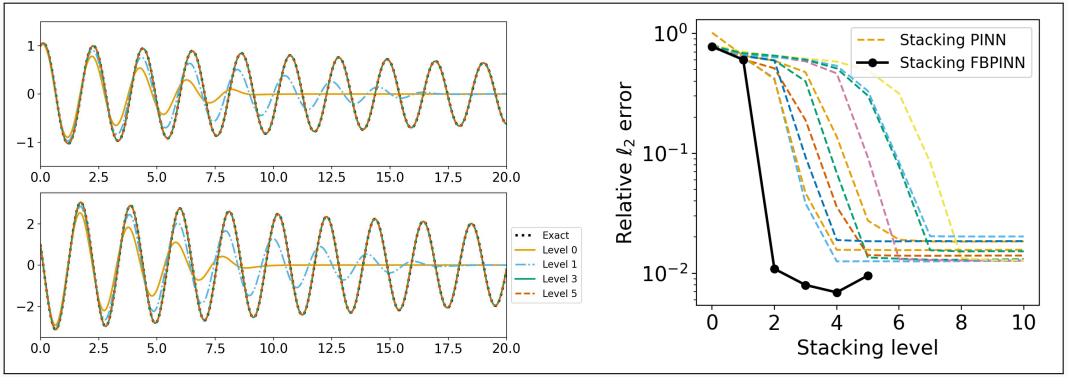
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

$$\frac{d\vartheta_1}{dt} = \vartheta_2,$$
$$\frac{d\vartheta_2}{dt} = -\frac{b}{m}\vartheta_2 - \frac{g}{L}\sin(\vartheta_1)$$

with $m = L = 1$, $b = 0.05$, $g = 9.81$, and $T = 20$.



Exemplary partition of unity in time



Numerical Results – Pendulum Problem

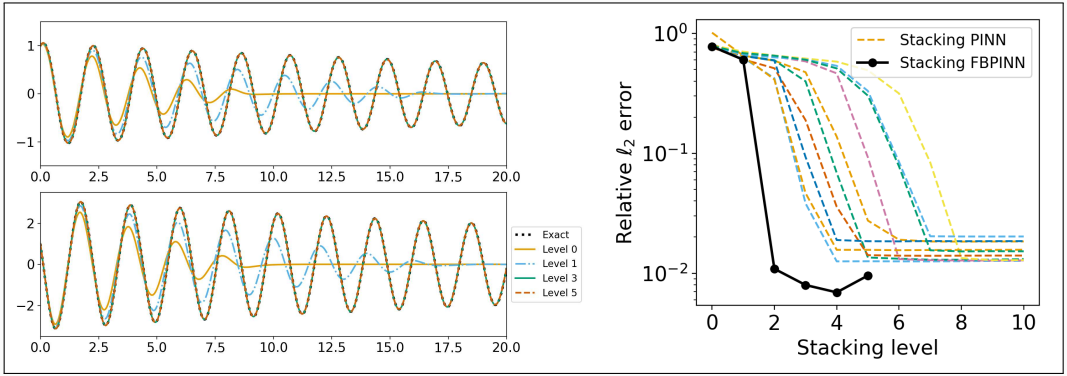
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

$$\frac{d\vartheta_1}{dt} = \vartheta_2,$$
$$\frac{d\vartheta_2}{dt} = -\frac{b}{m}\vartheta_2 - \frac{g}{L}\sin(\vartheta_1)$$

with $m = L = 1$, $b = 0.05$, $g = 9.81$, and $T = 20$.

Model details:

method	arch.	# levels	# params	error
S–PINN	5x50, 1x20	4	63 018	0.0125
S–FBPINN	3x32, 1x 4	2	34 570	0.0074



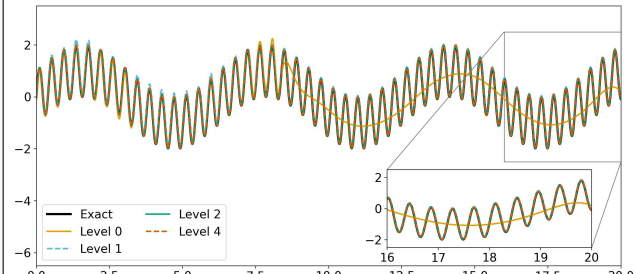
Numerical Results – Two-Frequency Problem

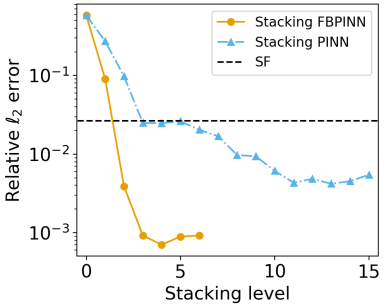
Second, we consider a **two-frequency problem**:

$$\frac{d\vartheta}{dx} = \omega_1 \cos(\omega_1 x) + \omega_2 \cos(\omega_2 x),$$
$$\vartheta(0) = 0,$$

on domain $\Omega = [0, 20]$ with $\omega_1 = 1$ and $\omega_2 = 15$.

method	arch.	# levels	# params	error
PINN	4x64	0	12 673	0.6543
PINN	5x64	0	16 833	0.0265
S–PINN	4x16, 1x5	3	4900	0.0249
S–PINN	4x16, 1x5	10	11 179	0.0061
S–FBPINN	4x16, 1x5	2	7822	0.00415
S–FBPINN	4x16, 1x5	5	59 902	0.00083



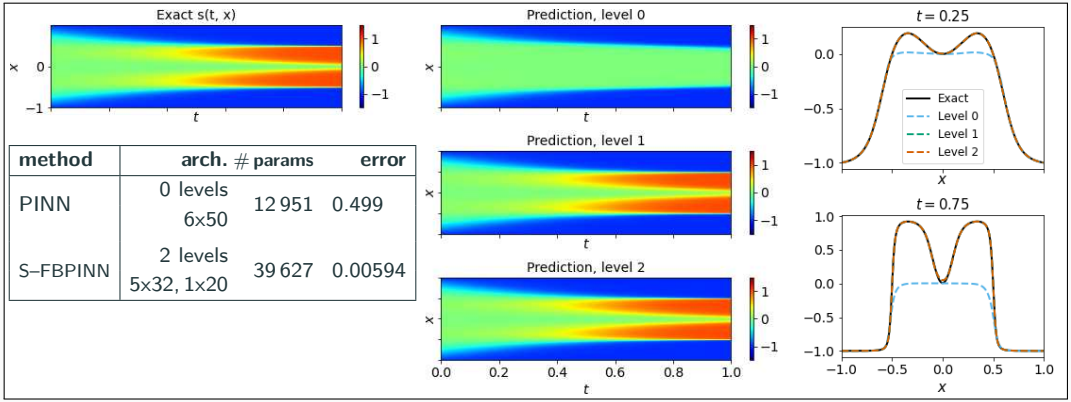


→ Due to the **multiscale structure of the problem**, the **improvements** due to the **multifidelity FBPINN approach** are **even stronger**.

Numerical Results – Allen–Cahn Equation

Finally, we consider the **Allen–Cahn equation**:

$$
\begin{aligned}
\mathfrak{s}_t - 0.0001\mathfrak{s}_{xx} + 5\mathfrak{s}^3 - 5\mathfrak{s} = 0, \qquad & t \in (0,1], x \in [-1,1], \\
\mathfrak{s}(x,0) = x^2\cos(\pi x), \qquad & x \in [-1,1], \\
\mathfrak{s}(x,t) = \mathfrak{s}(-x,t), \qquad & t \in [0,1], x = -1, x = 1, \\
\mathfrak{s}_x(x,t) = \mathfrak{s}_x(-x,t), \qquad & t \in [0,1], x = -1, x = 1.
\end{aligned}
$$

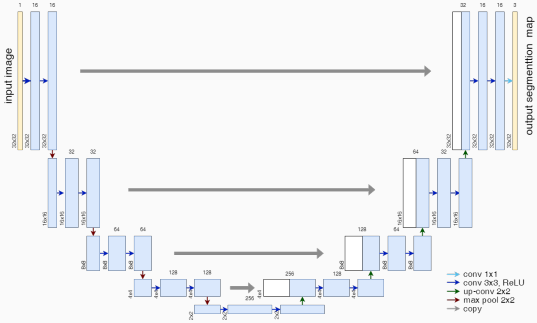


method	arch.	# params	error
PINN	0 levels 6x50	12 951	0.499
S–FBPINN	2 levels 5x32, 1x20	39 627	0.00594

PINN **gets stuck** at **fixed point of the of dynamical system**; cf. **Rohrhofer et al. (arXiv 2023)**.

Domain Decomposition for Convolutional Neural Networks

Memory Requirements for CNN Training

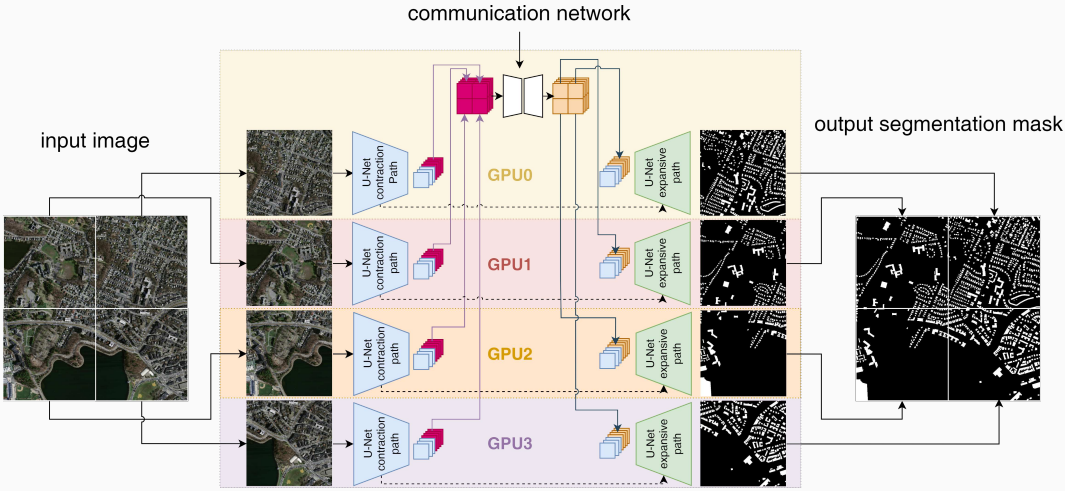


- As an example for a **convolutional neural network (CNN)**, we employ the **U-Net architecture** introduced in **Ronneberger, Fischer, and Brox (2015)**.
- The U-Net yields **state-of-the-art accuracy in semantic image segmentation** and other **image-to-image tasks**.

Below: *memory consumption for training on a single* 1024×1024 *image.*

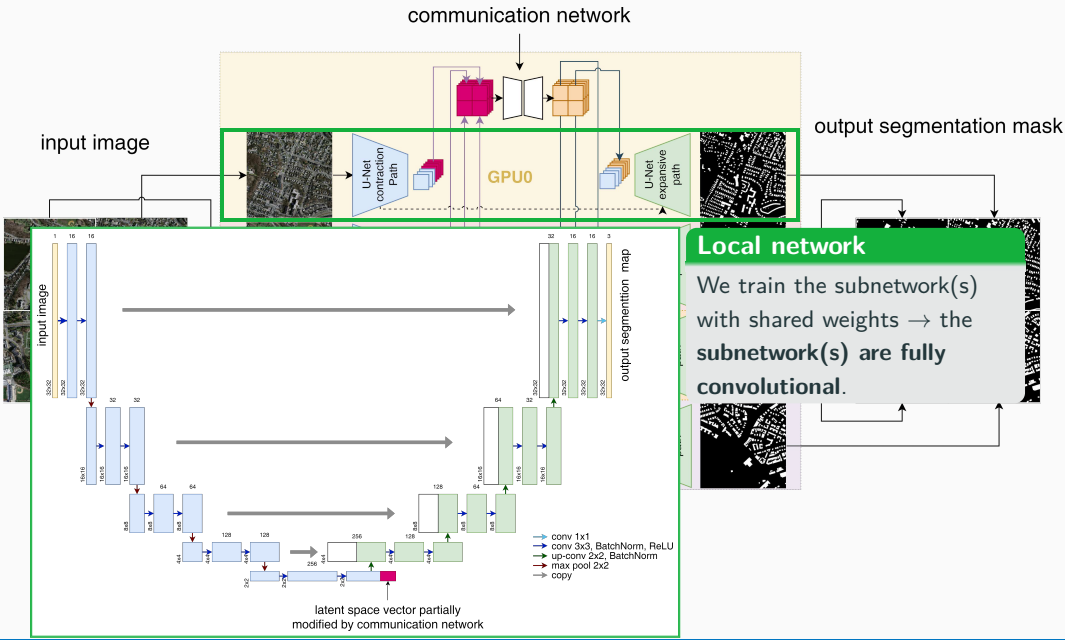
name	size	# channels		mem. feature maps		mem. weights	
		input	output	# of values	MB	# of values	MB
input block	1 024	3	64	268 M	**1 024.0**	38 848	**0.148**
encoder block 1	512	64	128	167 M	**704.0**	221 696	**0.846**
encoder block 2	256	128	256	84 M	**352.0**	885 760	**3.379**
encoder block 3	128	256	512	42 M	**176.0**	3 540 992	**13.508**
encoder block 4	64	512	1 024	21 M	**88.0**	14 159 872	**54.016**
decoder block 1	64	1,024	512	50 M	**192.0**	9 177 088	**35.008**
decoder block 2	128	512	256	101 M	**384.0**	2 294 784	**8.754**
decoder block 3	256	256	128	201 M	**768.0**	573 952	**2.189**
decoder block 4	512	128	64	402 M	**1 536.0**	143 616	**0.548**
output block	1 024	64	3	3.1 M	**12.0**	195	**0.001**

Decomposing the U-Net



Cf. **Verburg, Heinlein, Cyr (in preparation).**

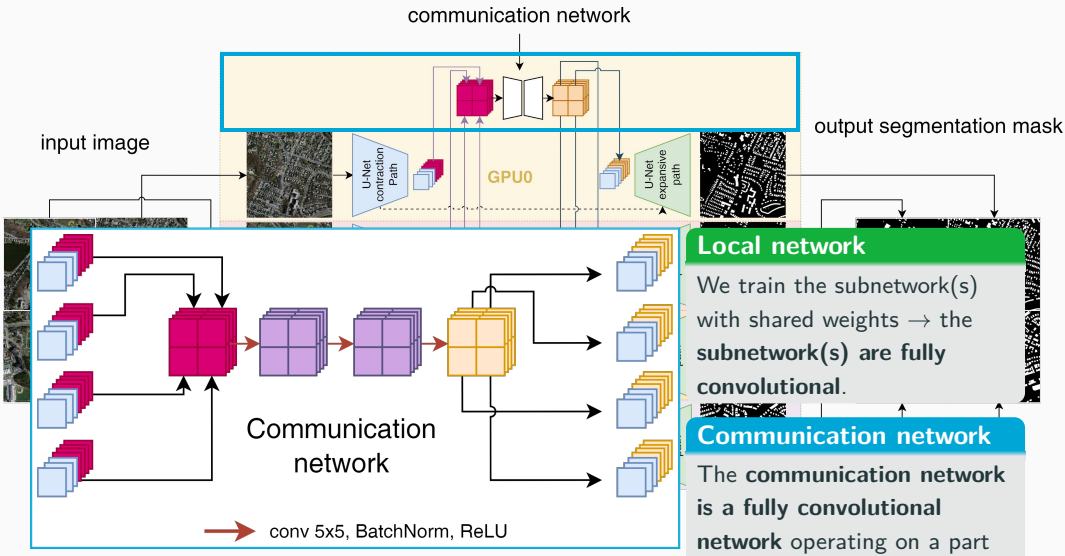
Decomposing the U-Net



Local network

We train the subnetwork(s) with shared weights → the **subnetwork(s) are fully convolutional**.

Decomposing the U-Net



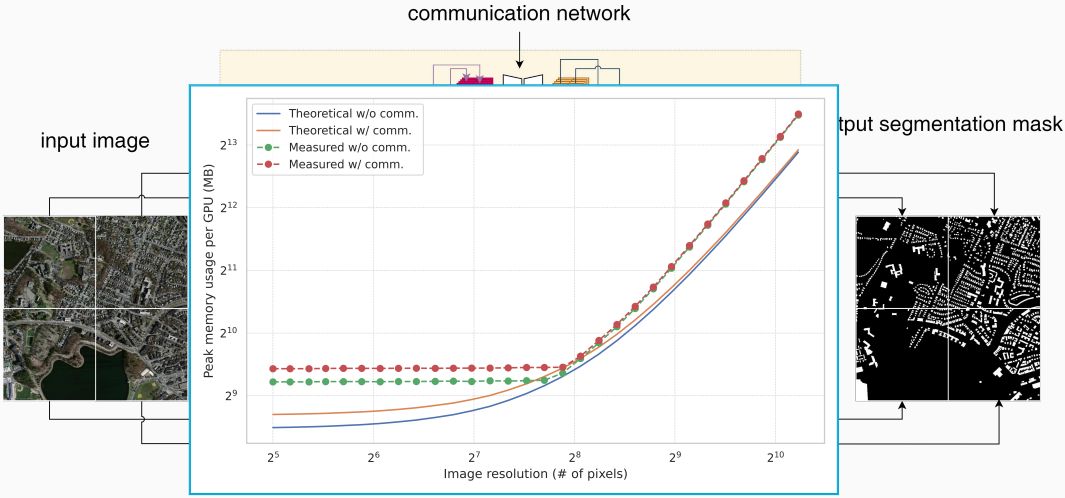
Local network

We train the subnetwork(s) with shared weights → the **subnetwork(s) are fully convolutional**.

Communication network

The **communication network is a fully convolutional network** operating on a part of the coarse feature maps.

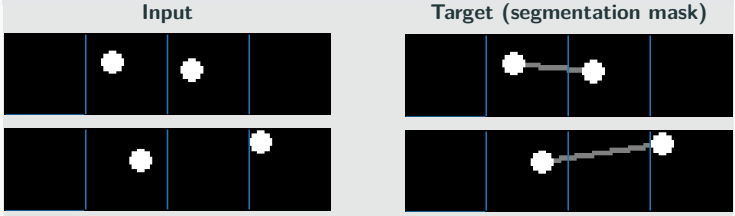
Decomposing the U-Net



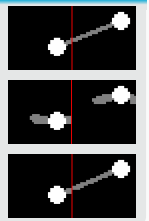
- Distribution of feature maps results in **significant reduction of memory usage on a single GPU**
- **Moderate additional memory usage** due to the **communication network**

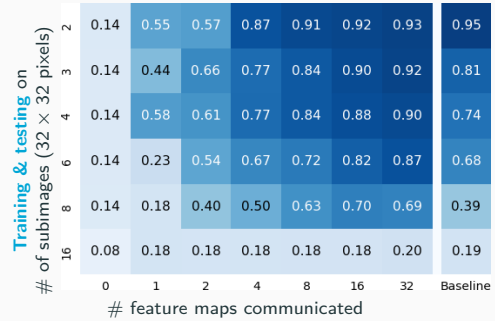
Results – Synthetic Data Set

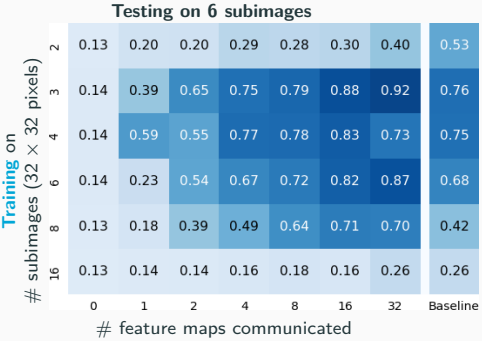
Task: Connect two dots via a line segment



Result: Communication



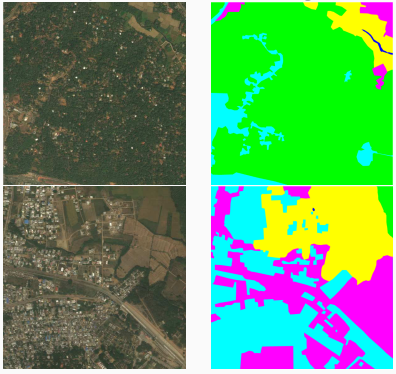




DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %

Input Target

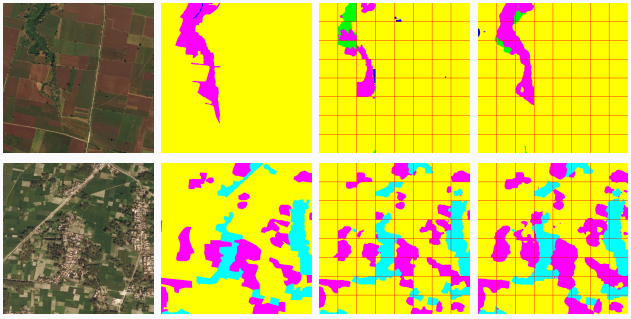


Avoiding overfitting

The data set includes **only 803 images**. To **avoid overfitting**, we

- apply **batch normalization**, use **random dropout** layers and **data augmentation**, and
- **initialize the encoder** using the **ResNet-18** (**He, Zhang, Ren, and Sun(2016)**)

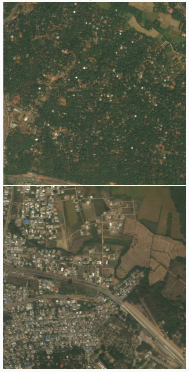
image true mask without comm. with comm.



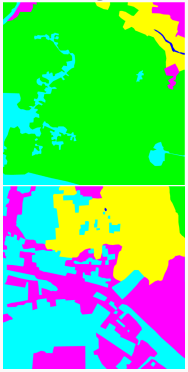
DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %

Input



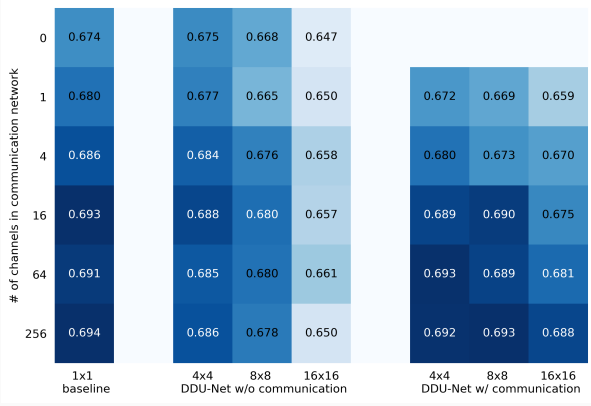
Target



Avoiding overfitting

The data set includes **only 803 images**. To **avoid overfitting**, we

- apply **batch normalization**, use **random dropout** layers and **data augmentation**, and
- **initialize the encoder** using the **ResNet-18** (**He, Zhang, Ren, and Sun(2016)**)



Multilevel FBPINNs

- **Schwarz domain decomposition architectures improve the scalability of PINNs** to **large domains** / **high frequencies**, **keeping the complexity of the local networks low**.
- As classical domain decomposition methods, **one-level FBPINNs** are **not scalable to large numbers of subdomains**; **multilevel FBPINNs enable scalability**.

Stacking Multifidelity FBPINNs

- The **combination of multifidelity stacking PINNs with FBPINNs** yields **significant improvements in the accuracy and efficiency** for **time-dependent problems**.

DDU-Net – Domain Decomposition for CNNs

- The **memory requirements for training of high-resolution images** using CNNs can be **large**, In particular, the U-Net model requires **storing intermediate feature maps**.
- Our **novel DDU-Net** approach **decouples the training on the sub-images**, allowing us to **distribute the memory load** among **multiple GPUs**. It **limits communication** to **deepest level** of the U-Net architecture using a **communication network**.

Thank you for your attention!