

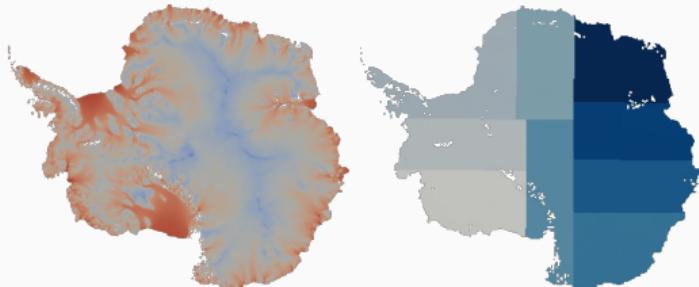
Domain decomposition for neural networks

Alexander Heinlein¹

NHR Conference 2024, Darmstadt, Germany, September 9-12, 2024

¹Delft University of Technology

Domain Decomposition Methods



Images based on Heinlein, Perego, Rajamanickam (2022)

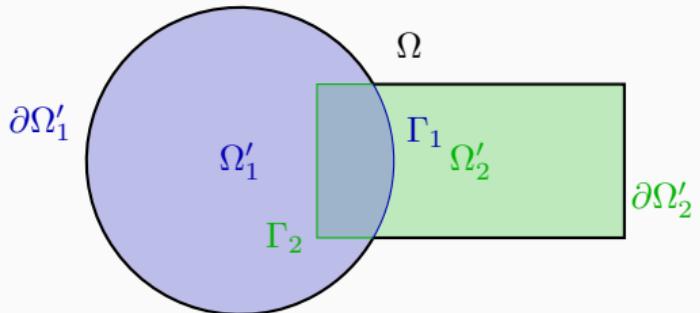
Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

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

Idea

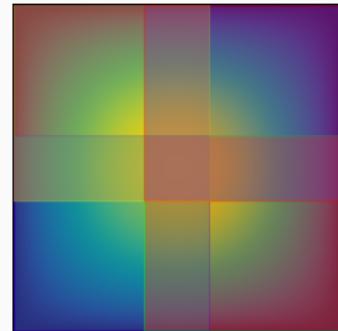
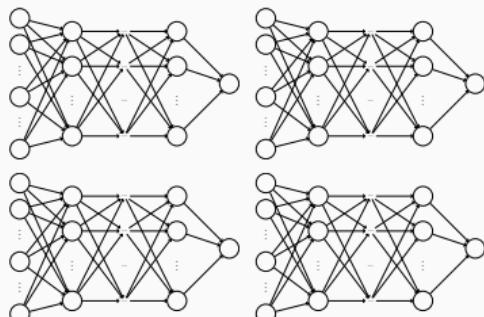
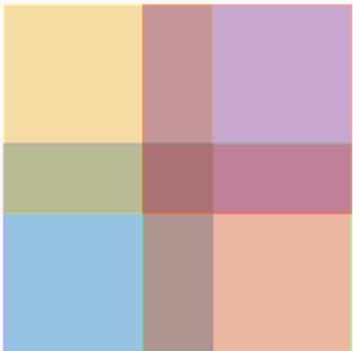
Decomposing a large **global problem** into smaller **local problems**:

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



Domain Decomposition for Neural Networks

I)



II)

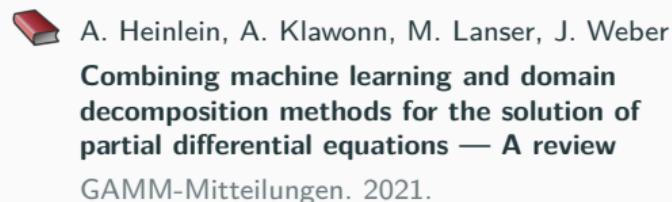


Domain Decomposition Methods and Machine Learning – Literature

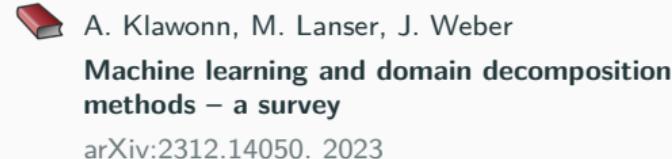
A non-exhaustive literature overview:

- 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 (2024); Verburg, Heinlein, Cyr (subm. 2024)

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



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



Outline

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

Based on joint work with

Damien Beecroft

(University of Washington)

Victorita Dolean

(Eindhoven University of Technology)

Amanda Howard, Bruno Jacob, and Panos Stinis

(Pacific Northwest National Laboratory)

Ben Moseley and Siddhartha Mishra

(ETH Zürich)

Sarah H. Murphy

(University of North Carolina)

2 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

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

by solving an **optimization problem**

$$\min_{\theta} \sum_{x_i} G(x_i, \Psi_t(x_i, \theta), \nabla\Psi_t(x_i, \theta), \nabla^2\Psi_t(x_i, \theta))^2$$

where $\Psi_t(x, \theta)$ is a **trial function**, 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**:

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

- 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

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

by solving an **optimization problem**

$$\min_{\theta} \sum_{x_i} G(x_i, \Psi_t(x_i, \theta), \nabla\Psi_t(x_i, \theta), \nabla^2\Psi_t(x_i, \theta))^2$$

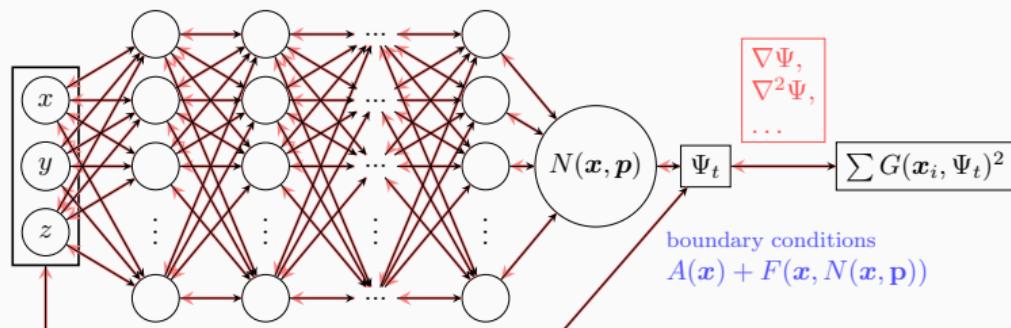
where $\Psi_t(x, \theta)$ is a **trial function**, x_i sampling points inside the domain Ω and θ are adjustable parameters.

Construction of the trial functions

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

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

- 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



Lagaris et. al's Method – Motivation

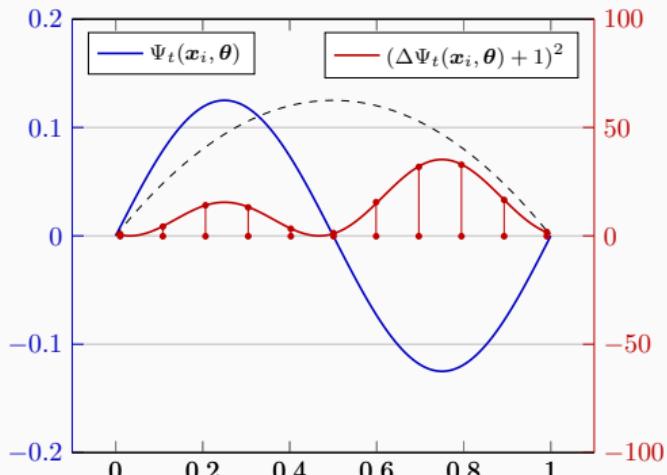
Solve the **boundary value problem**

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

$$\Psi_t(0, \theta) = \Psi_t(1, \theta) = 0,$$

via a **collocation approach**:

$$\min_{\theta} \sum_{x_i} (1 - \Delta \Psi_t(x_i, \theta))^2$$

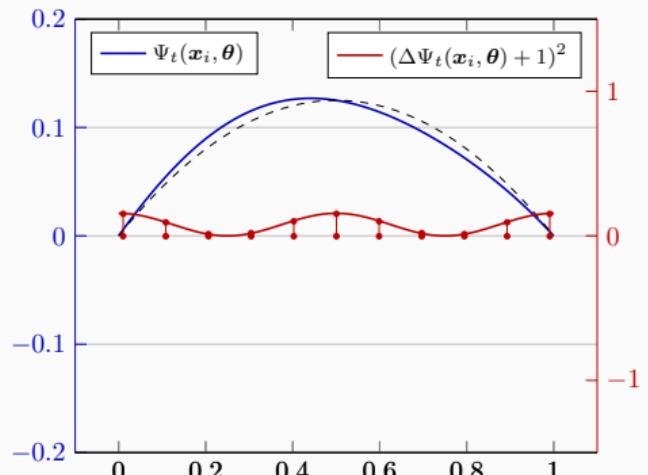


$$(\Delta \Psi_t(x_i, \theta) + 1)^2 >> 0$$

Boundary conditions

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

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



$$(\Delta \Psi_t(x_i, \theta) + 1)^2 \approx 0$$

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

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

PINNs use a **hybrid loss function**:

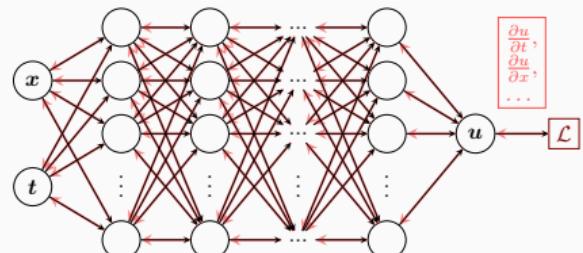
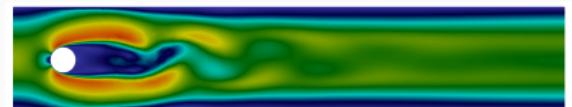
$$\mathcal{L}(\theta) = \omega_{\text{data}} \mathcal{L}_{\text{data}}(\theta) + \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}}(\theta),$$

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

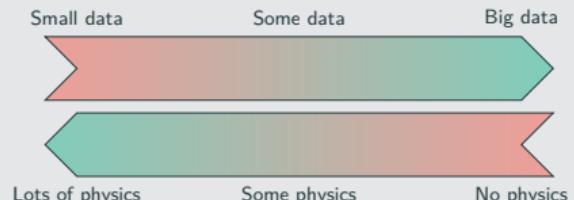
$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} (u(\hat{x}_i, \theta) - u_i)^2,$$

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} (n[u](x_i, \theta) - f(x_i))^2.$$

See also [Dissanayake and Phan-Thien \(1994\)](#); [Lagaris et al. \(1998\)](#).



Hybrid loss



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

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

Theoretical Result for PINNs

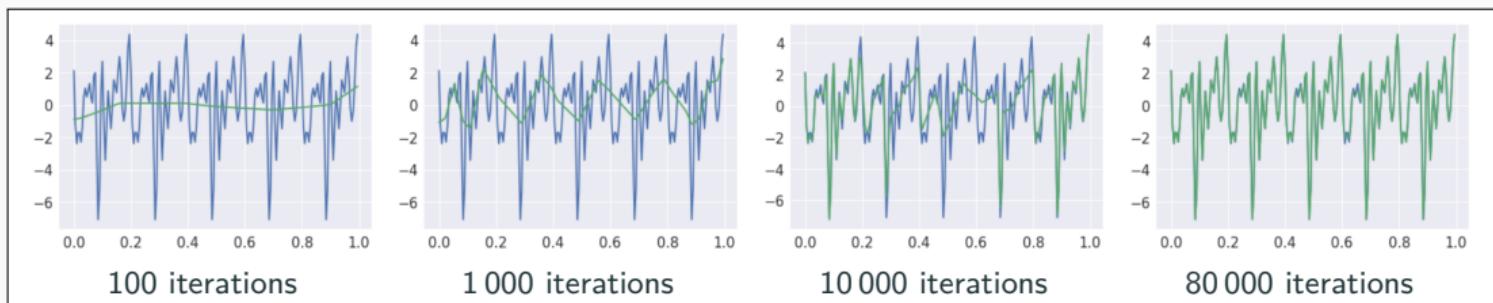
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

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

- $\mathcal{E}_G = \mathcal{E}_G(\mathbf{X}, \theta) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **general. error** (V Sobolev space, \mathbf{X} training data set)
- $\mathcal{E}_{\mathcal{T}}$ **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, quadrature, and neural network**

Rule of thumb: “As long as the PINN is **trained well**, it also **generalizes well**”



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

Motivation – Some Observations on the Performance of PINNs

Solve

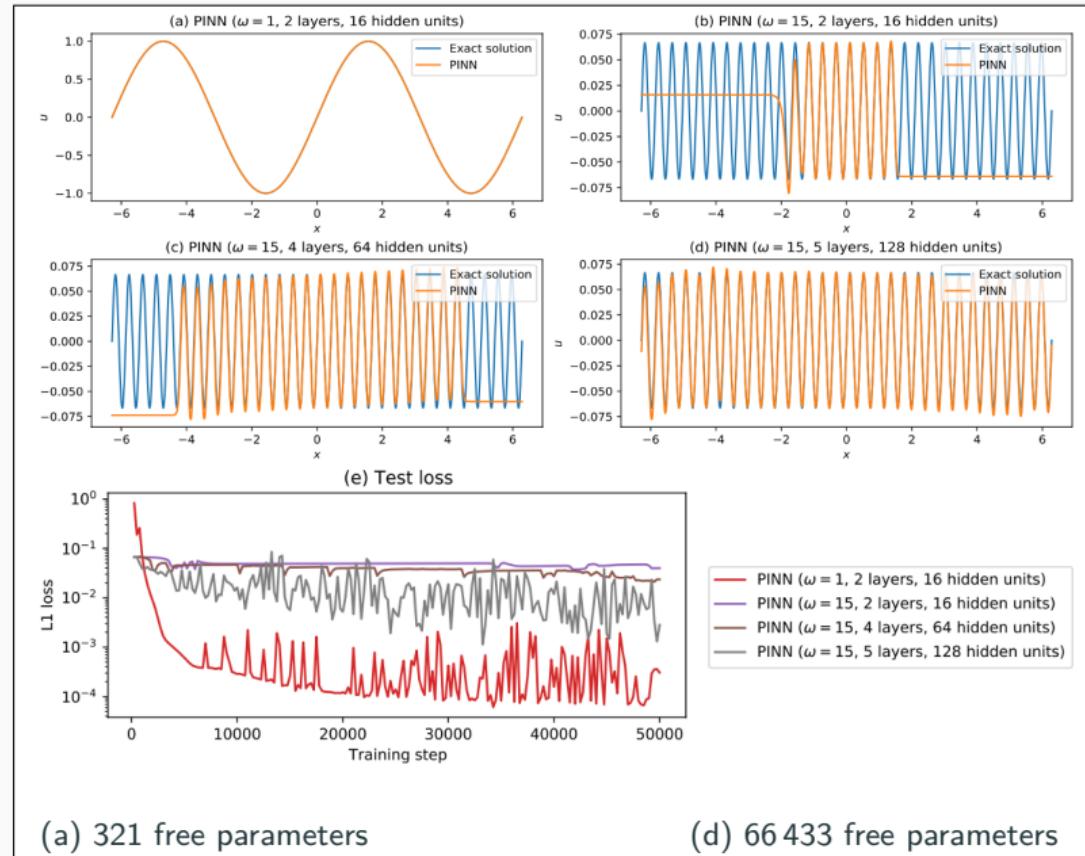
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

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)



Motivation – Some Observations on the Performance of PINNs

Solve

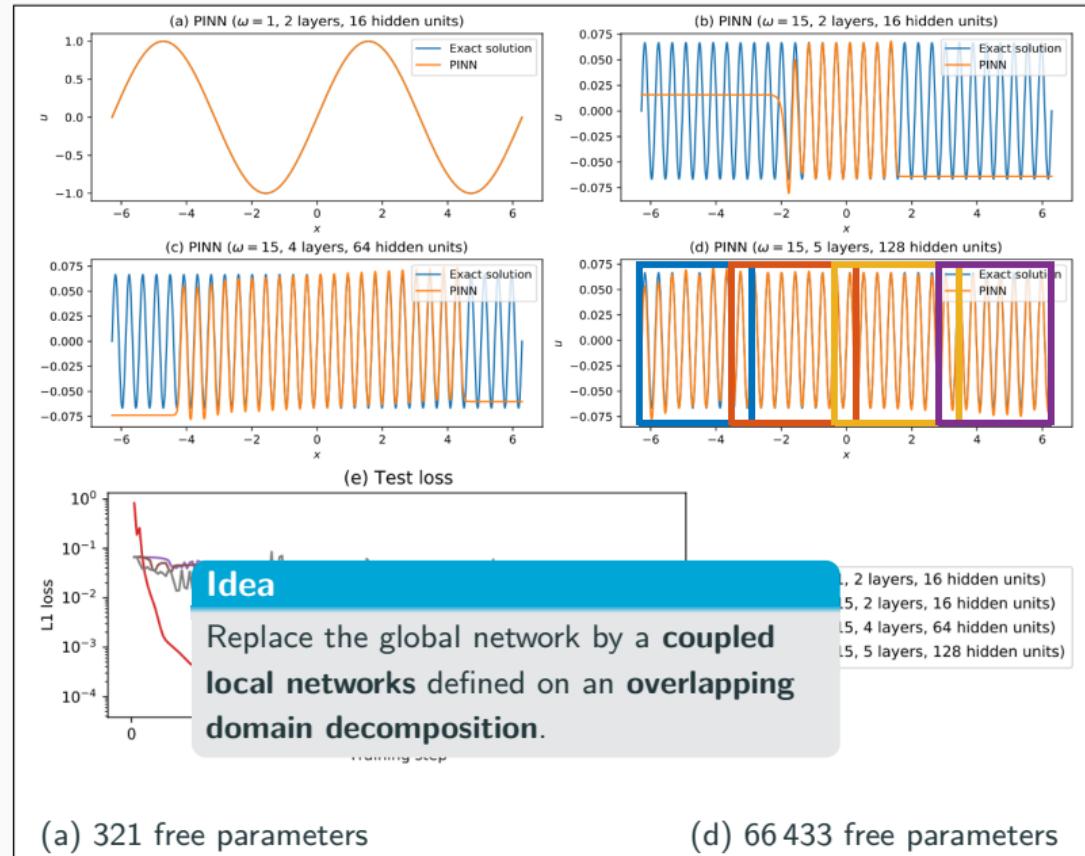
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

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)



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**

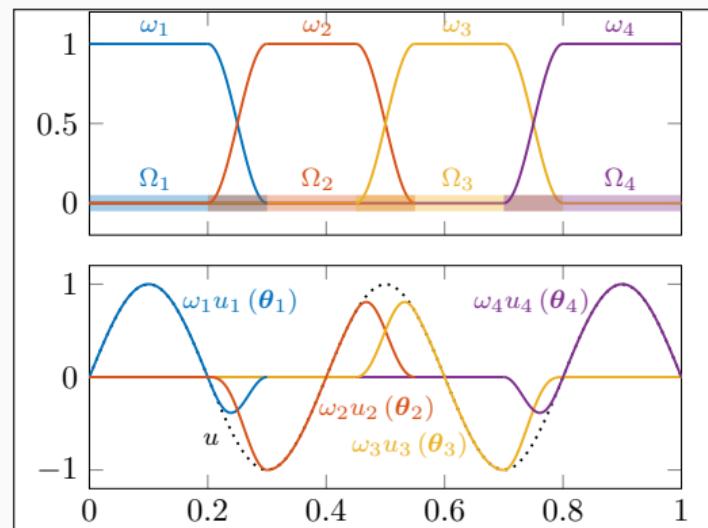
$$u(\theta_1, \dots, \theta_J) = \mathcal{C} \sum_{j=1}^J \omega_j u_j(\theta_j)$$

and the **loss function**

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

Here:

- **Overlapping DD:** $\Omega = \bigcup_{j=1}^J \Omega_j$
- **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

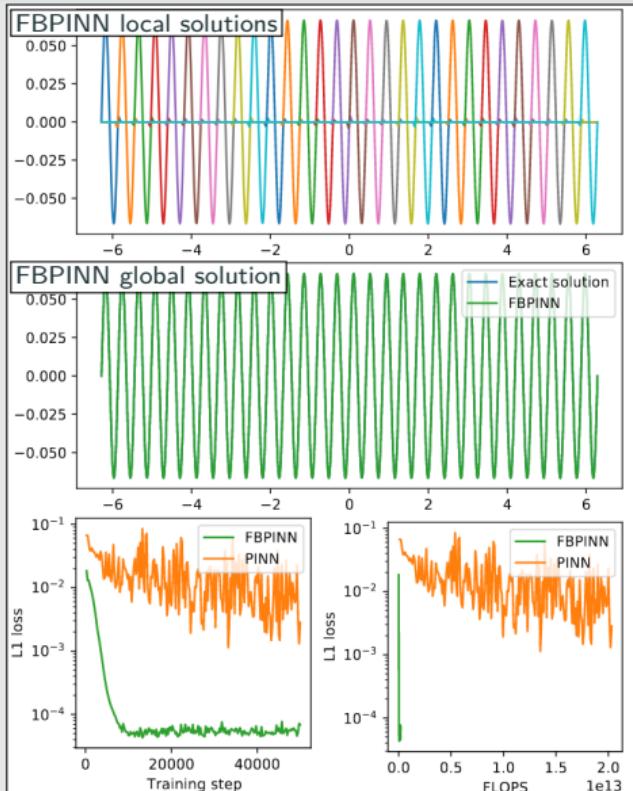
Loss function

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

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

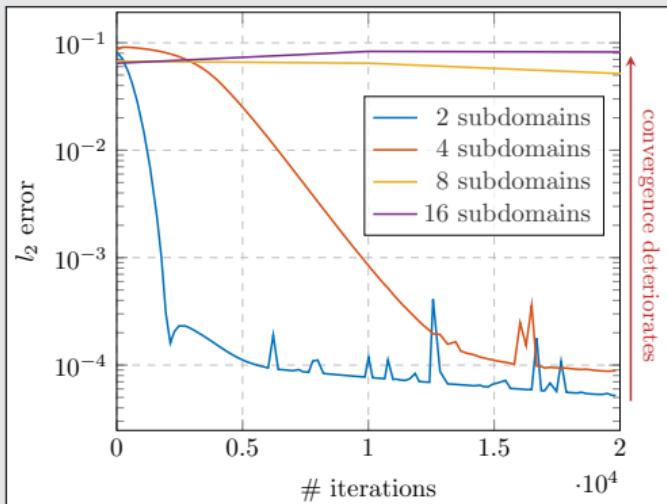
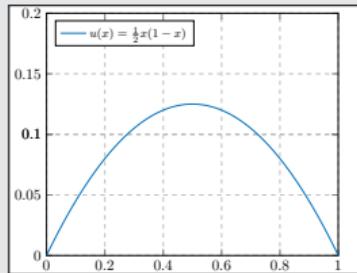
Consider the simple boundary value problem

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

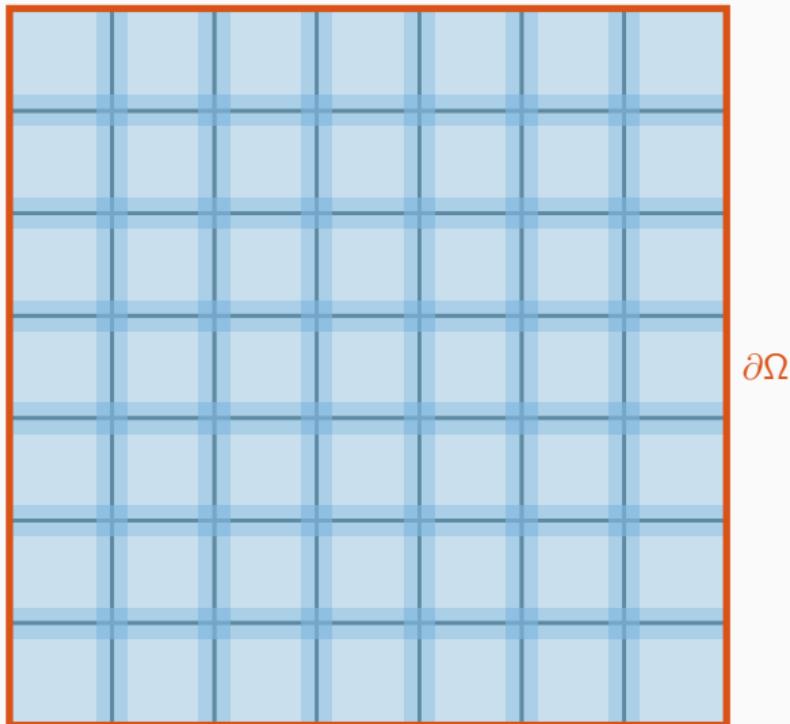
$$u(0) = u(1) = 0,$$

which has the solution

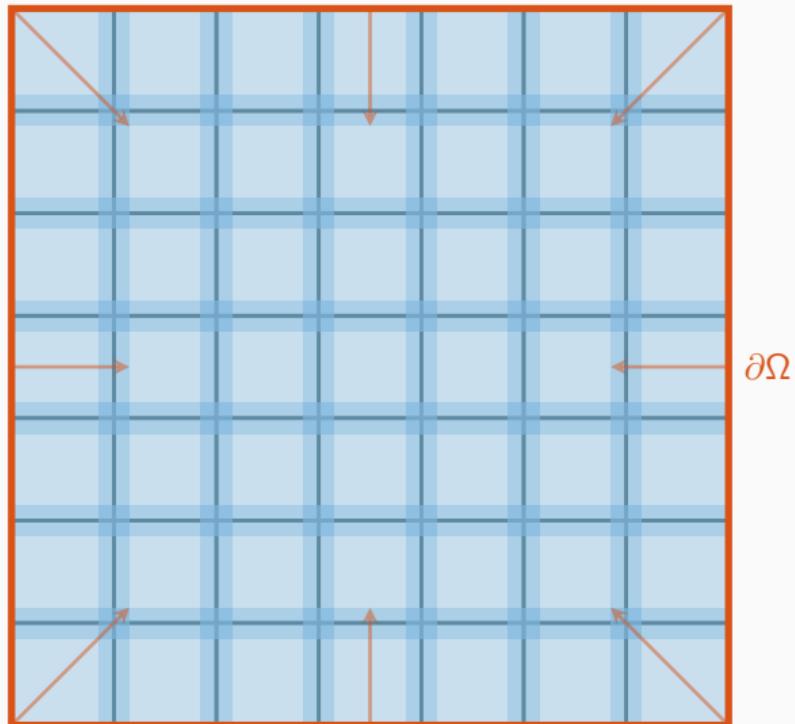
$$u(x) = \frac{1}{2}x(1-x).$$



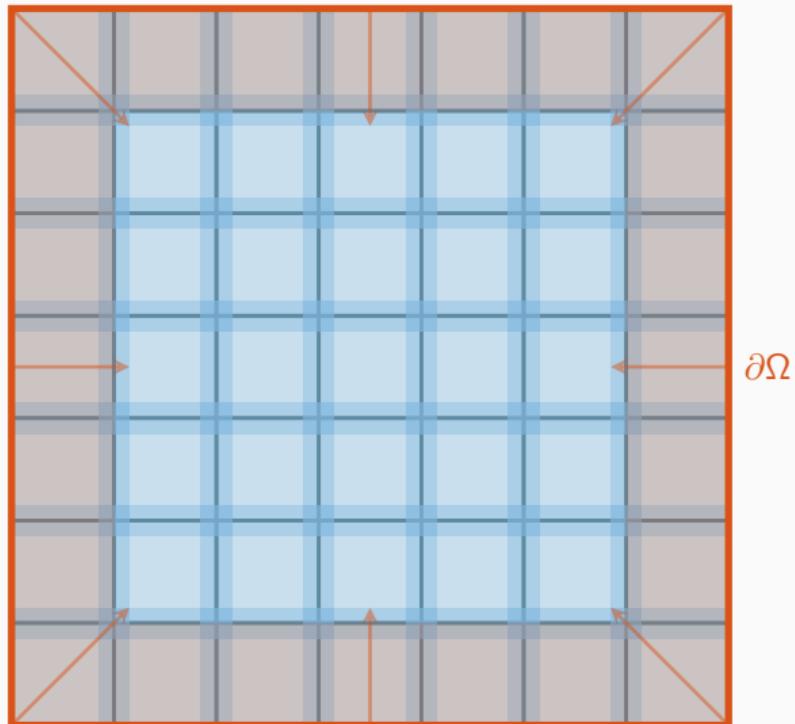
Transport of Information – One-Level Overlapping Schwarz Methods



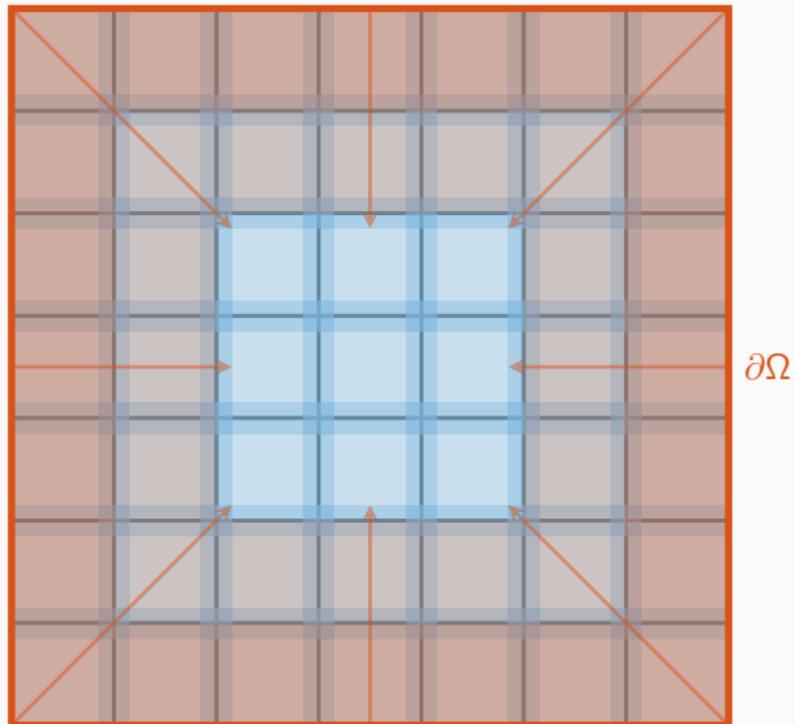
Transport of Information – One-Level Overlapping Schwarz Methods



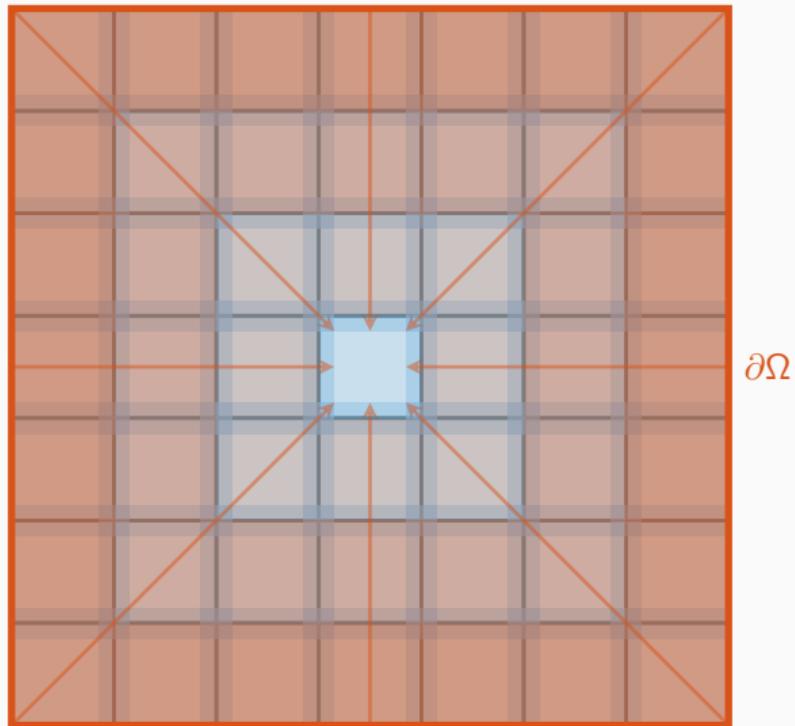
Transport of Information – One-Level Overlapping Schwarz Methods



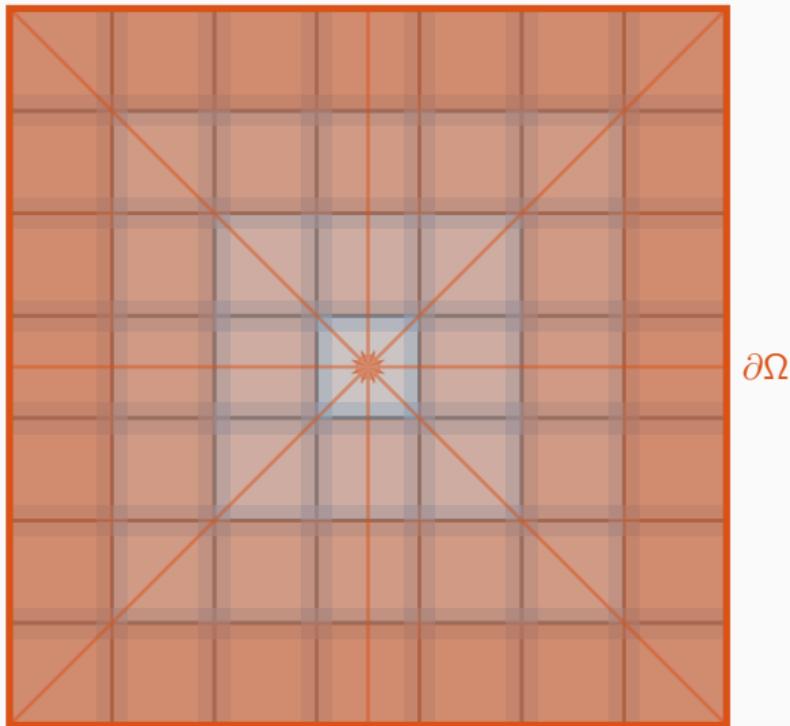
Transport of Information – One-Level Overlapping Schwarz Methods



Transport of Information – One-Level Overlapping Schwarz Methods



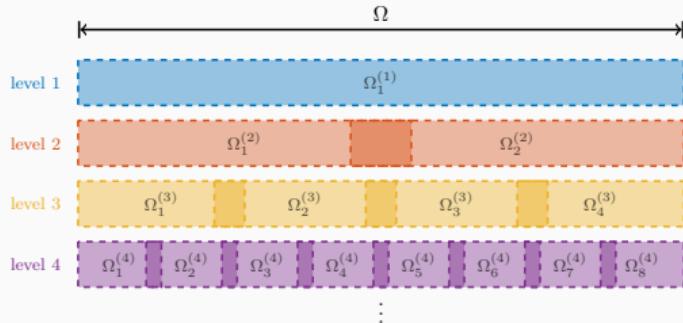
Transport of Information – One-Level Overlapping Schwarz Methods



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

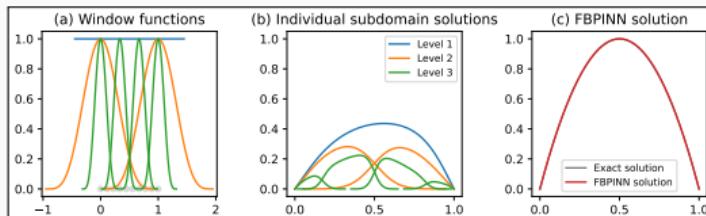
Multi-Level FBPINN Algorithm

Extension of FBPINNs to L levels; Cf. [Dolean, Heinlein, Mishra, Moseley \(accepted 2024 / arXiv:2306.05486\)](#).



L -level network architecture

$$u(\theta_1^{(1)}, \dots, \theta_{J(L)}^{(L)}) = \mathcal{C} \left(\sum_{l=1}^L \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}(\theta_j^{(l)}) \right)$$



Multi-Frequency Problem

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

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

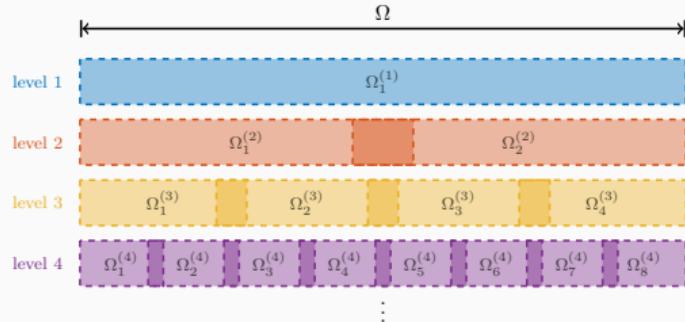
with $\omega_i = 2^i$.

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



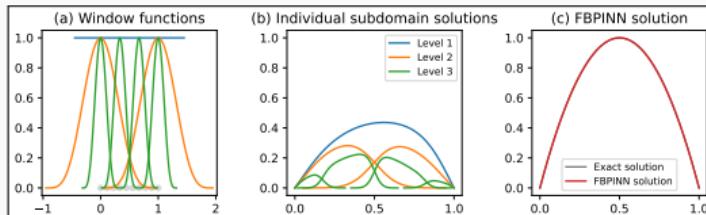
Multi-Level FBPINN Algorithm

Extension of FBPINNs to L levels; Cf. [Dolean, Heinlein, Mishra, Moseley \(accepted 2024 / arXiv:2306.05486\)](#).



L -level network architecture

$$u(\theta_1^{(1)}, \dots, \theta_{J(L)}^{(L)}) = \mathcal{C} \left(\sum_{l=1}^L \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}(\theta_j^{(l)}) \right)$$



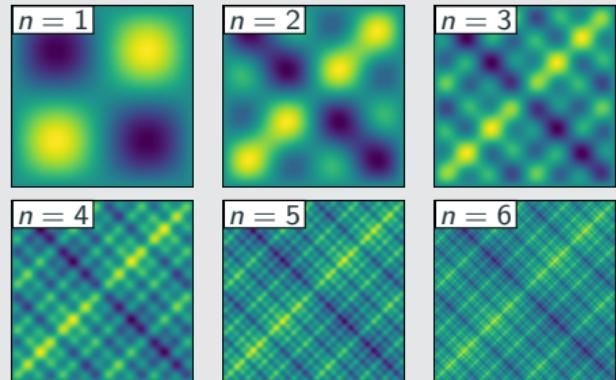
Multi-Frequency Problem

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

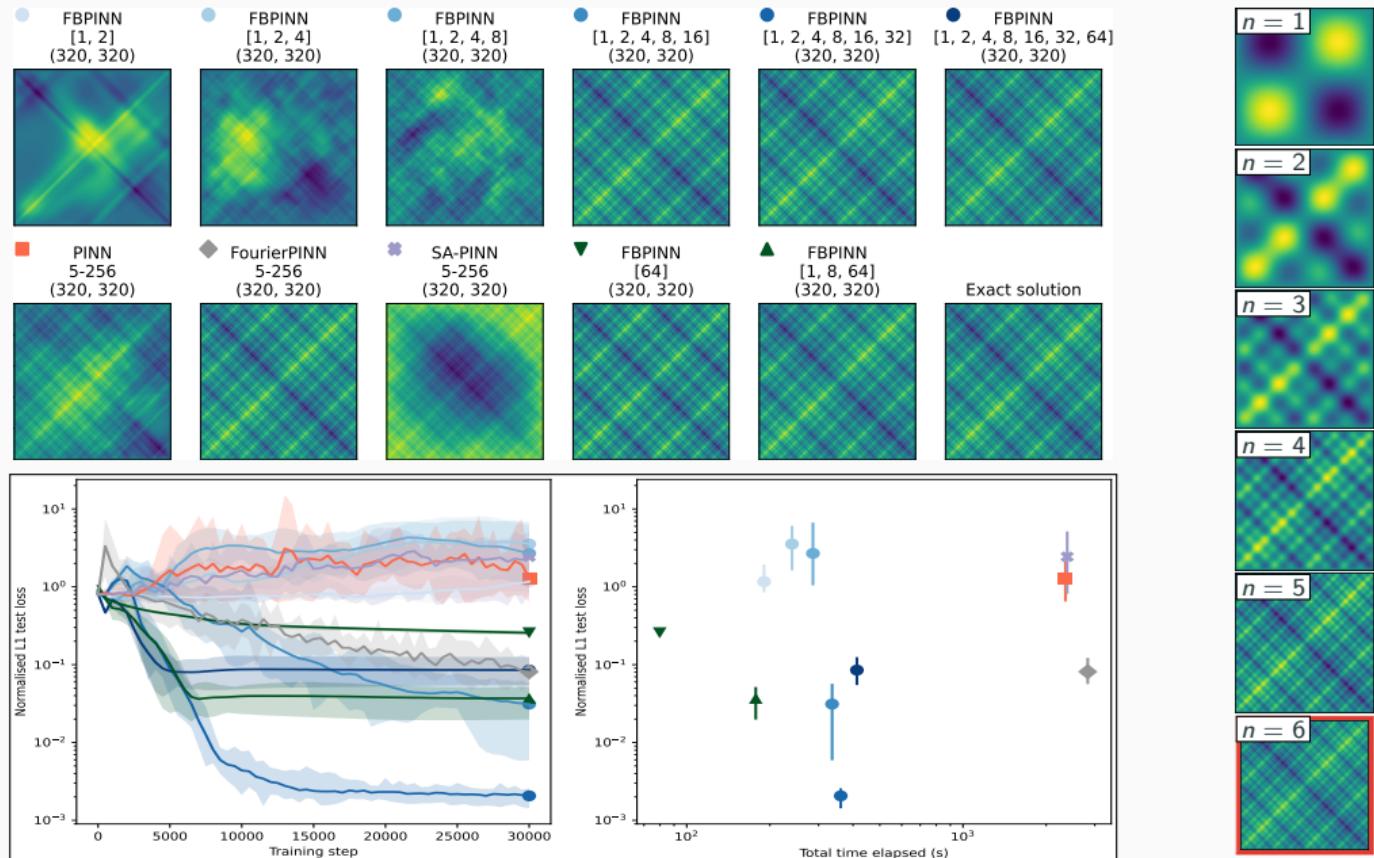
$$-\Delta u = 2 \sum_{i=1}^n (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) \quad \text{in } \Omega,$$
$$u = 0 \quad \text{on } \partial\Omega,$$

with $\omega_i = 2^i$.

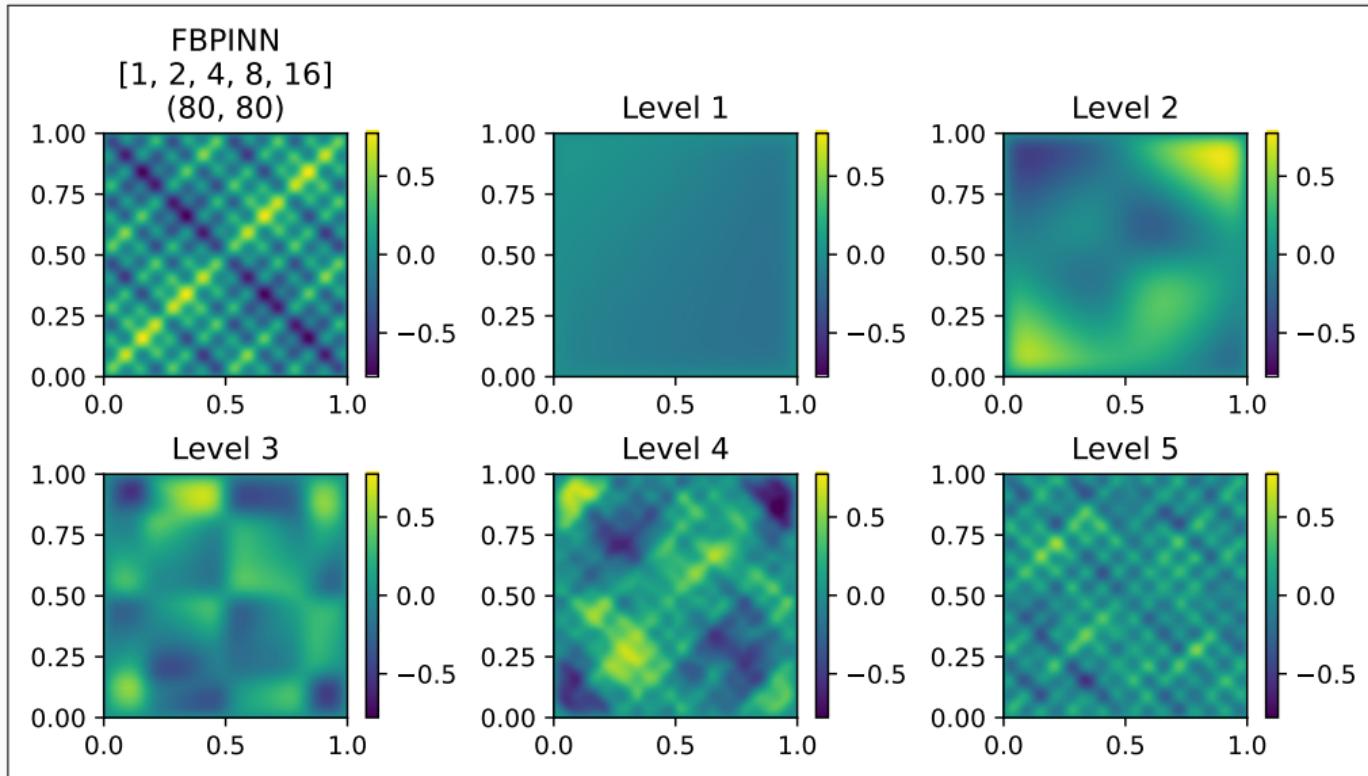
For increasing values of n , we obtain the analytical solutions:



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

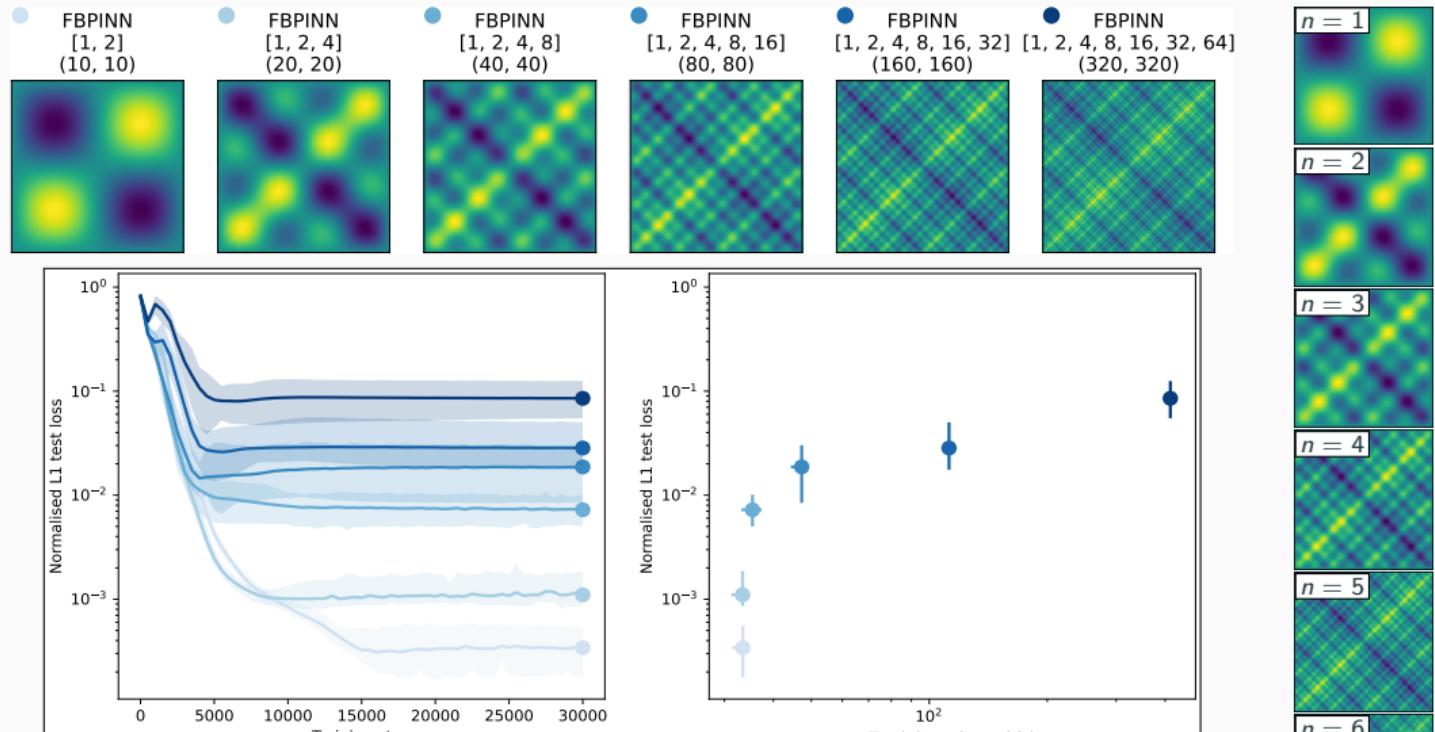


Multi-Frequency Problem – What the FBPINN Learns



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

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

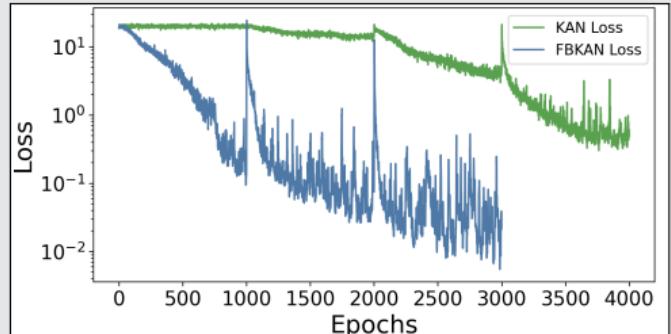
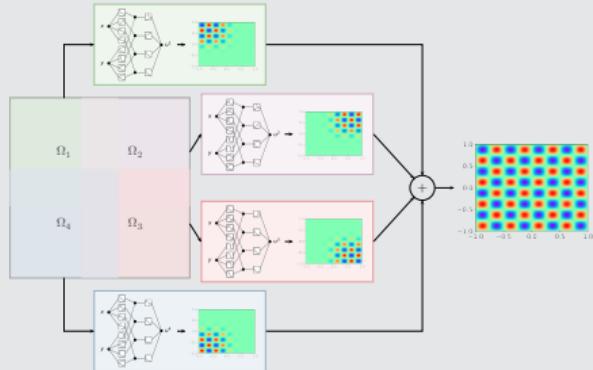


- Ongoing: analysis and improvement of the convergence

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

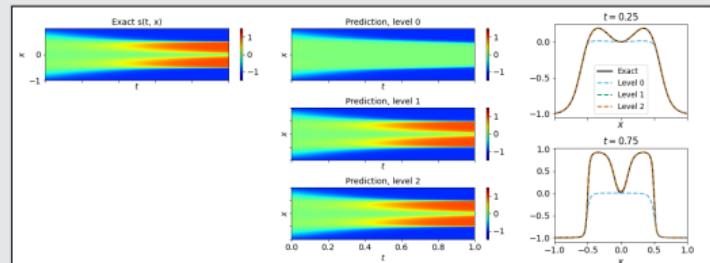
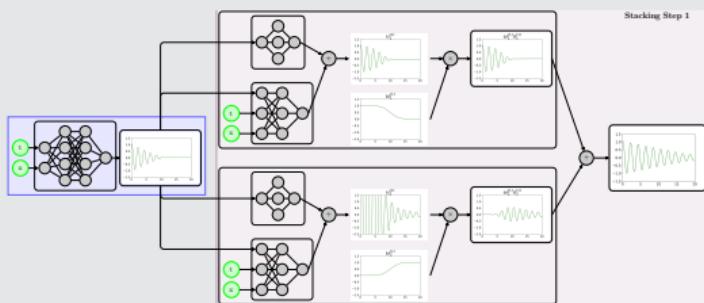
Extension to Kolmogorov–Arnold Networks (KANs)

Extension to Kolmogorov–Arnold Networks (KANs)



Cf. Howard, Jacob, Murphy, Heinlein, Stinis (arXiv 2024).

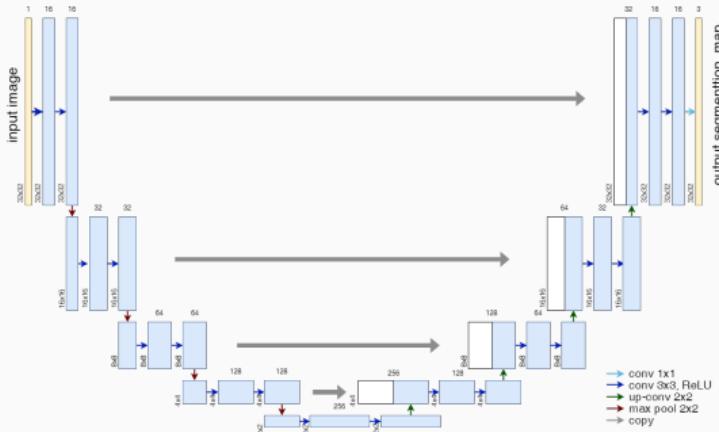
Stacking Multifidelity FBPINNs



Cf. Heinlein, Howard, Beecroft, and Stinis (acc. 2024)

Domain Decomposition for Convolutional Neural Networks

Memory Requirements for CNN Training

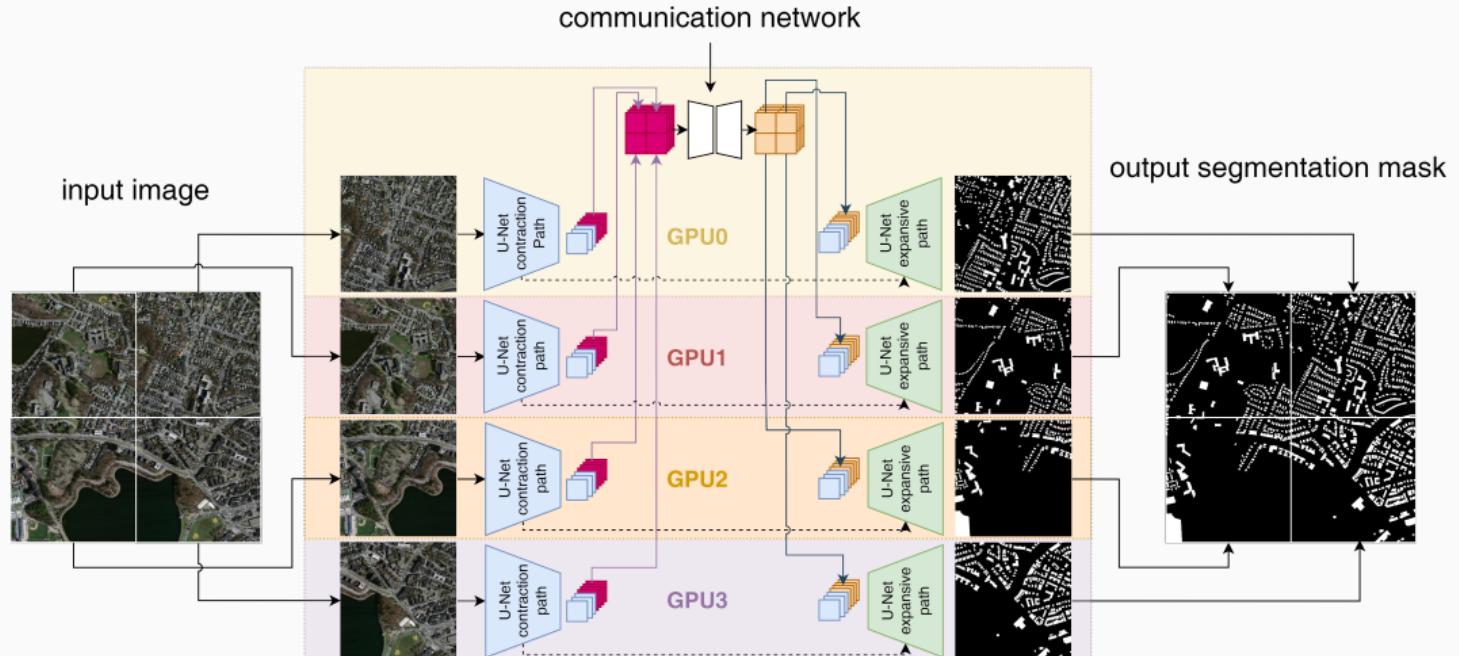


- As an example for a **convolutional neural network (CNN)**, we employ the **U-Net architecture** introduced in **Ronneberger, Fischer, and Brox (2015)**.
- The U-Net yields **state-of-the-art accuracy in semantic image segmentation** and other **image-to-image tasks**.

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

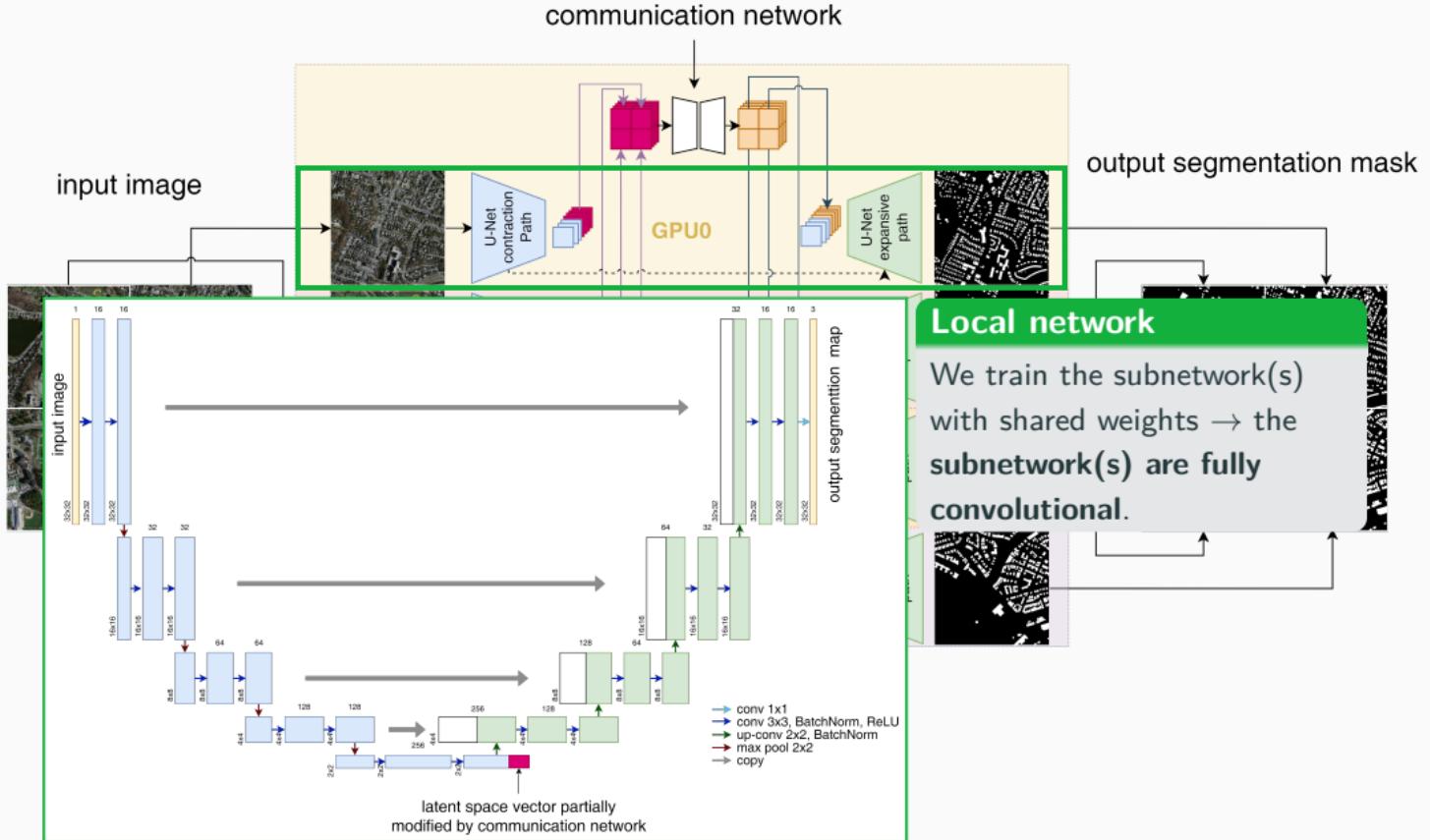
name	size	# channels		mem. feature maps		mem. weights	
		input	output	# of values	MB	# of values	MB
input block	1 024	3	64	268 M	1 024.0	38 848	0.148
encoder block 1	512	64	128	167 M	704.0	221 696	0.846
encoder block 2	256	128	256	84 M	352.0	885 760	3.379
encoder block 3	128	256	512	42 M	176.0	3 540 992	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

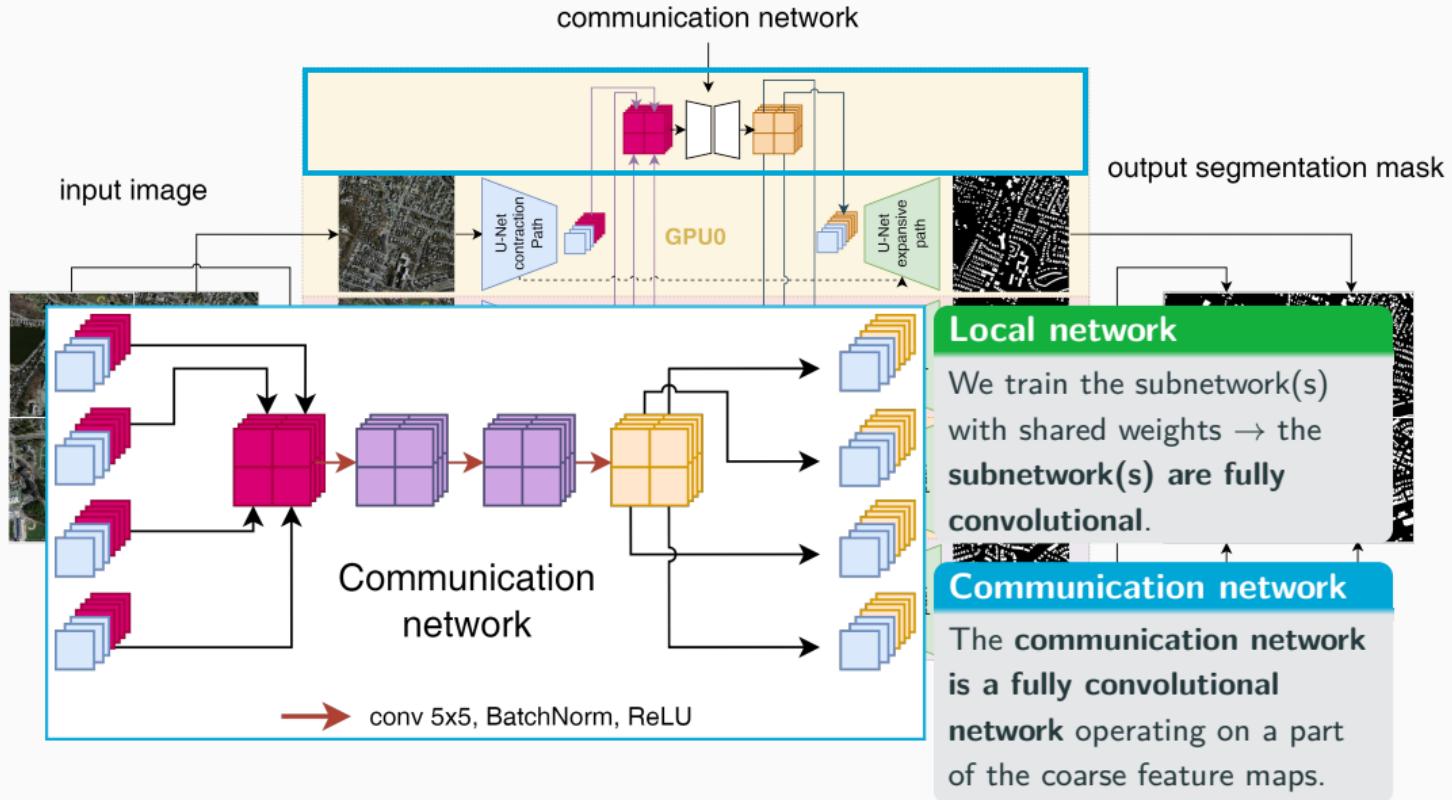


Cf. Verburg, Heinlein, Cyr (subm. 2024).

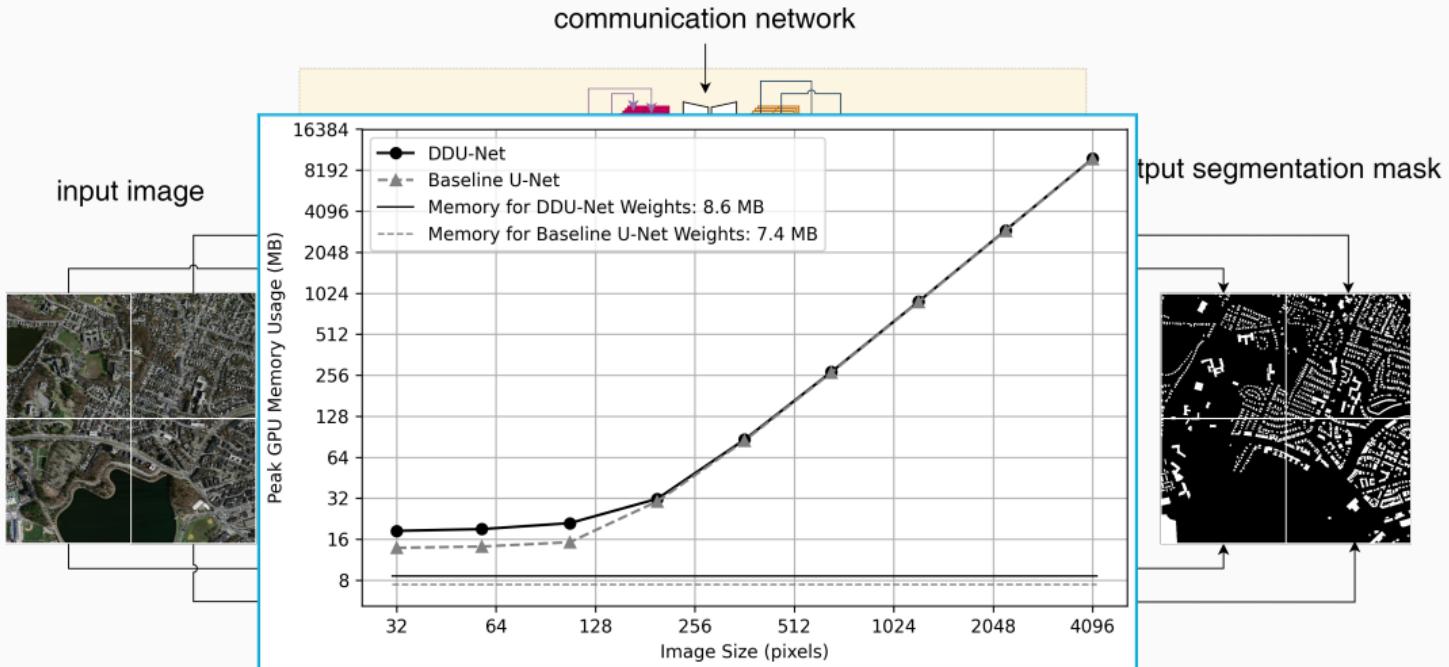
Decomposing the U-Net



Decomposing the U-Net



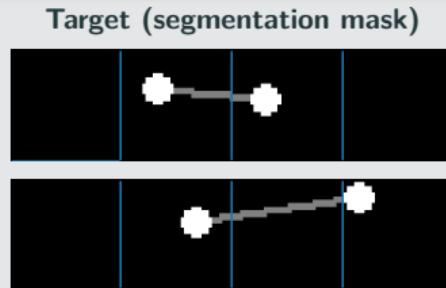
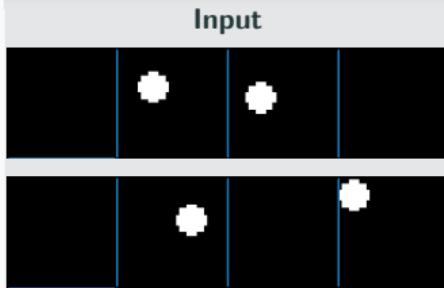
Decomposing the U-Net



- 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

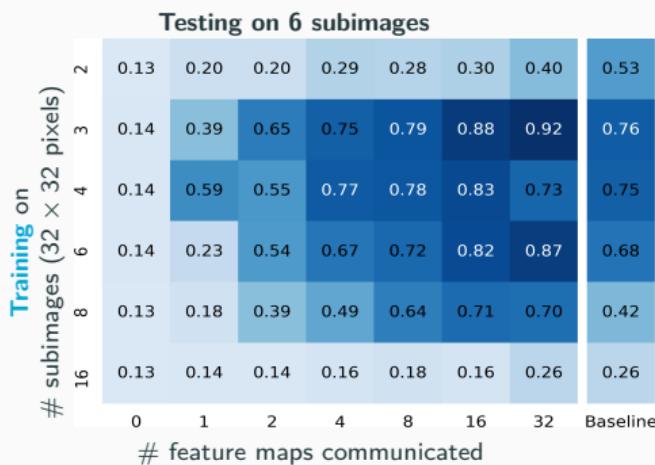
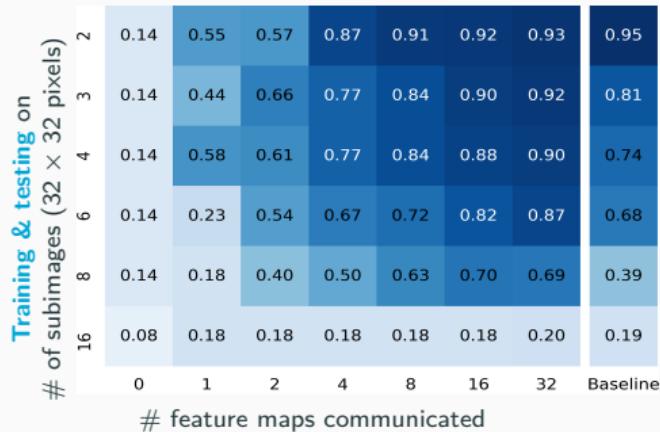
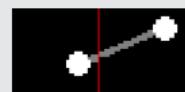
True mask



Pred. (no comm.)

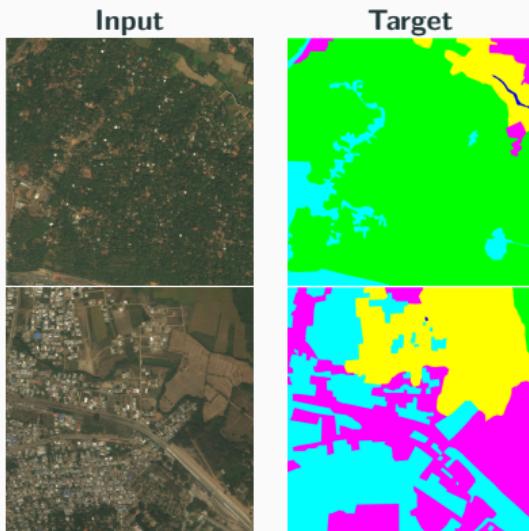


Pred. (comm.)



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

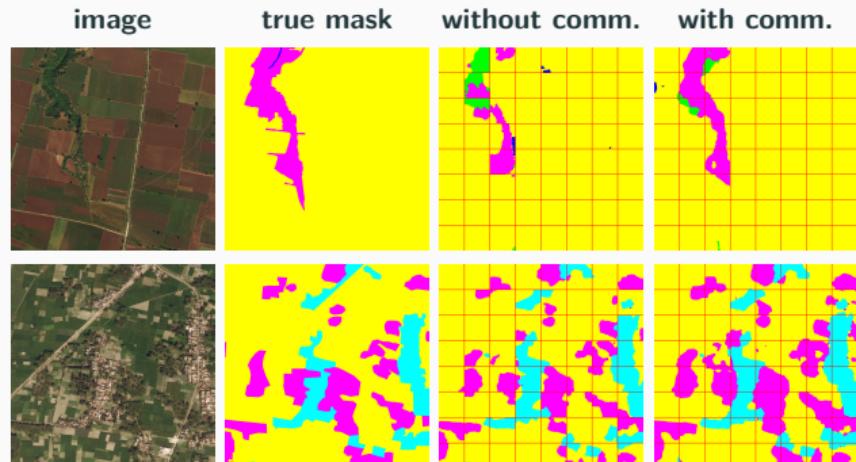
class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %



Avoiding overfitting

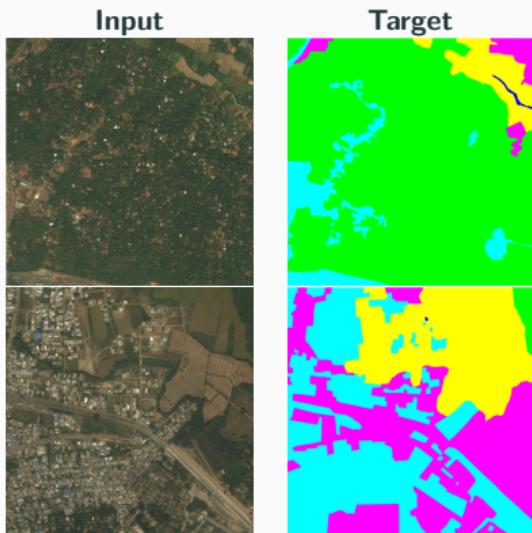
The data set includes **only 803 images**. To **avoid overfitting**, we

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



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

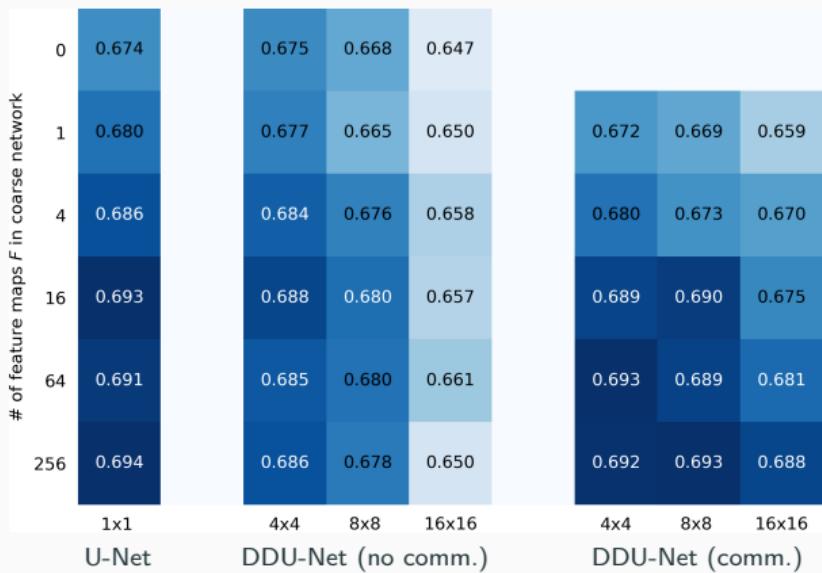
class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %



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**.
- Also applies to **KANs** and can be combined with **multifidelity stacking approaches** 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!