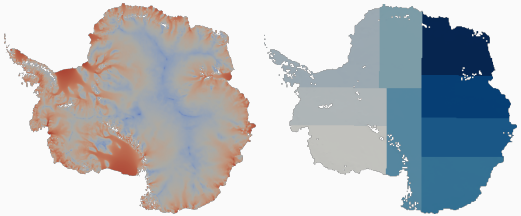
# Domain decomposition for neural networks

Alexander Heinlein[1]

IGHASC: Indo-German Workshop on Hardware-aware Scientific Computing, Heidelberg, Germany, October 28-30, 2024

[1]Delft University of Technology

# Domain Decomposition Methods



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

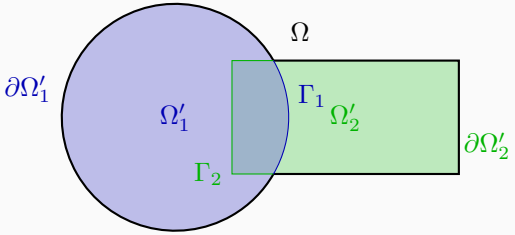
**Idea**

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

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

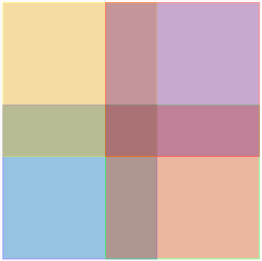
**Historical remarks:** The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

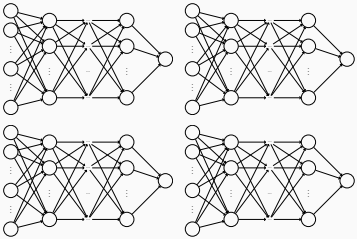
- Schwarz used the algorithm to establish the **existence of harmonic functions** with prescribed boundary values on **regions with non-smooth boundaries**.

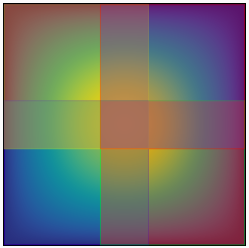


# Domain Decomposition for Neural Networks

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); Dolean, Heinlein, Mercier, Gratton (subm. 2024 / arXiv:2408.12198); 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:

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** | (Eindhoven University of Technology) |
   | **Ben Moseley** and **Siddhartha Mishra** | (ETH Zürich) |

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

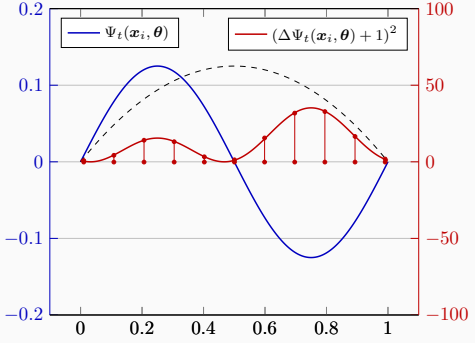
# Physics-Informed Neural Networks (PINNs) – Idea

In **Lagaris et al. (1998)**, the authors solve the **boundary value problem**

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

via a **collocation approach**:

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



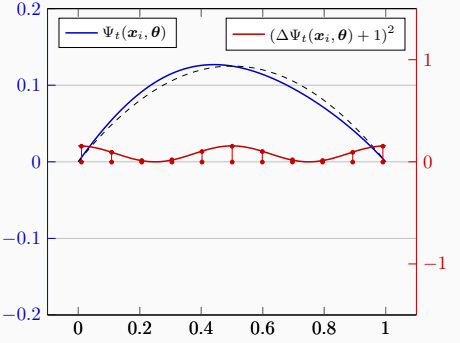

$(\Delta\Psi_t(\boldsymbol{x}_i,\boldsymbol{\theta}) + 1)^2 >> 0$

**Boundary conditions ...**

... can be **enforced explicitly** via the ansatz:

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

- $A$ **satisfies the boundary conditions**
- $F$ **does not contribute to the boundary conditions**




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

# 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 $\omega_{\text{data}}$ and $\omega_{\text{PDE}}$ are **weights** and

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

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

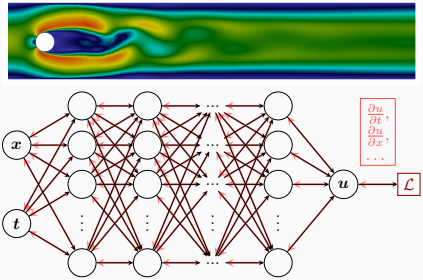
See also **Dissanayake and Phan-Thien (1994); Lagaris et al. (1998)**.

## Advantages

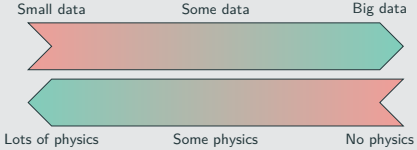
- "Meshfree"
- Small data
- Generalization properties
- High-dimensional problems
- Inverse and parameterized problems

## Drawbacks

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




## Hybrid loss




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

# Theoretical Result for PINNs

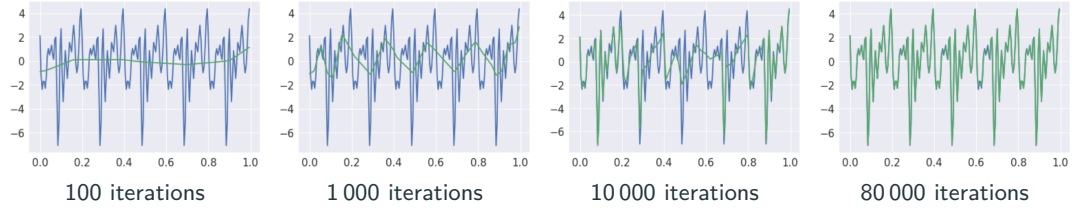
**Estimate of the generalization error (Mishra and Molinaro (2022))**

The generalization error (or total error) satisfies

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

- $\mathcal{E}_G = \mathcal{E}_G(\boldsymbol{X}, \theta) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **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 $\alpha$ **convergence rate of the quadrature**
- $C_{\text{PDE}}$ and $C_{\text{quad}}$ **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

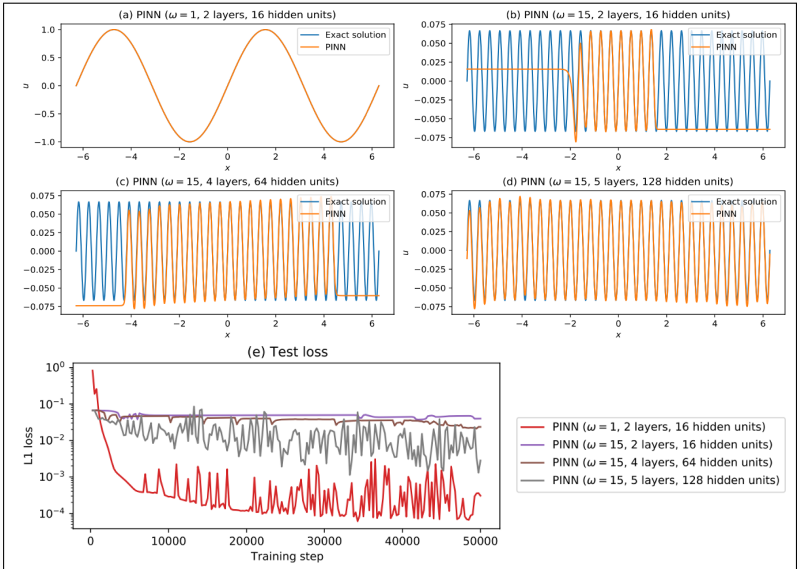
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

for different values of $\omega$ 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

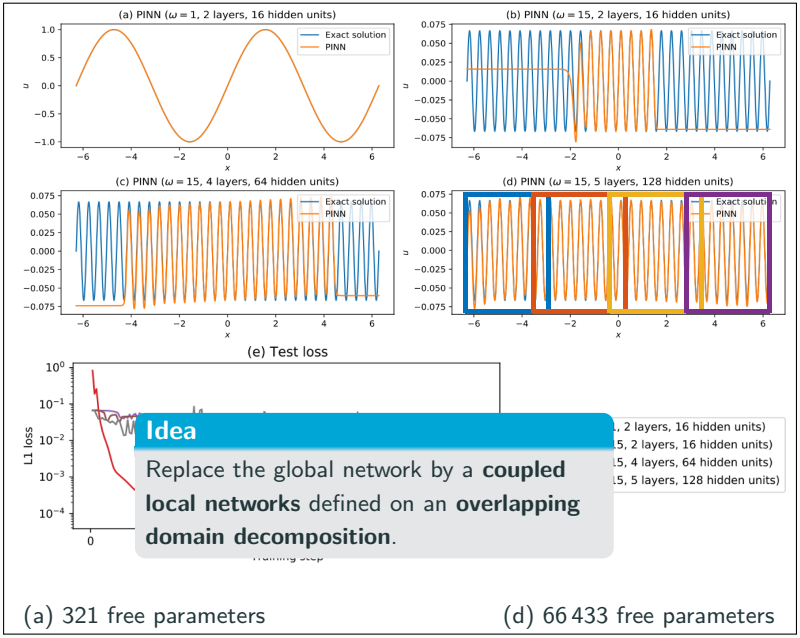
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

for different values of $\omega$ 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**

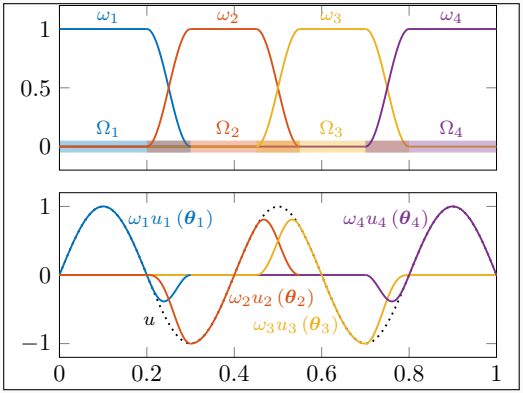
$$u(\theta_1, \dots, \theta_J) = \mathcal{C} \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

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




**Hard enf. of boundary conditions**

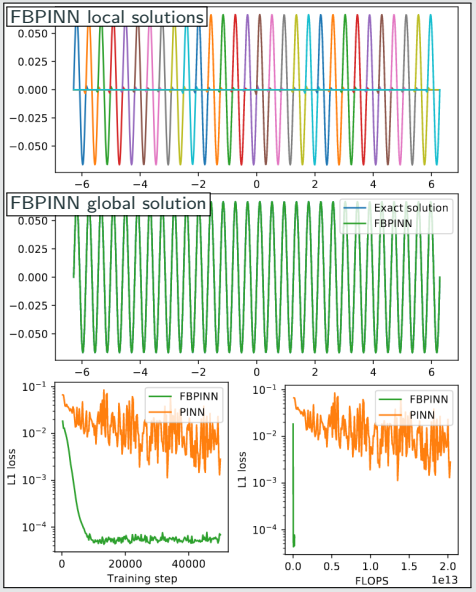
Loss function

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

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

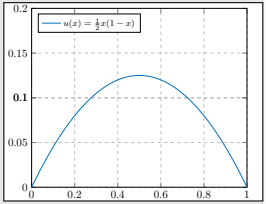
Consider the **simple boundary value problem**

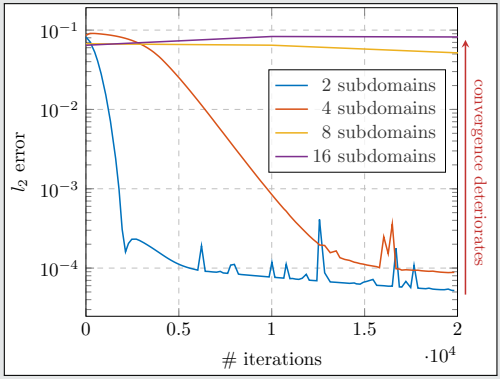
$$-u'' = 1 \text{ in } [0,1],$$

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

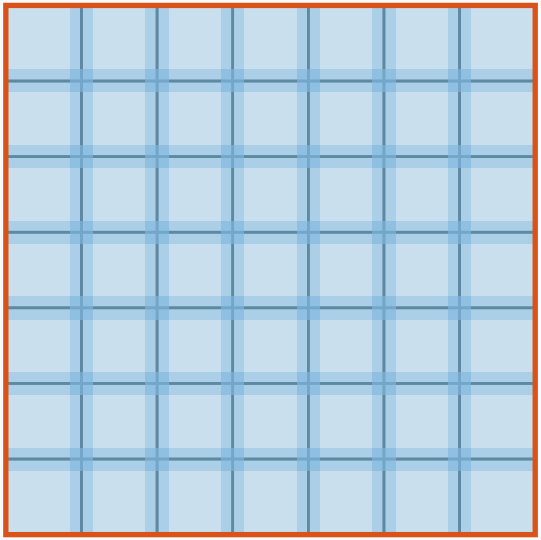
which has the **solution**

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

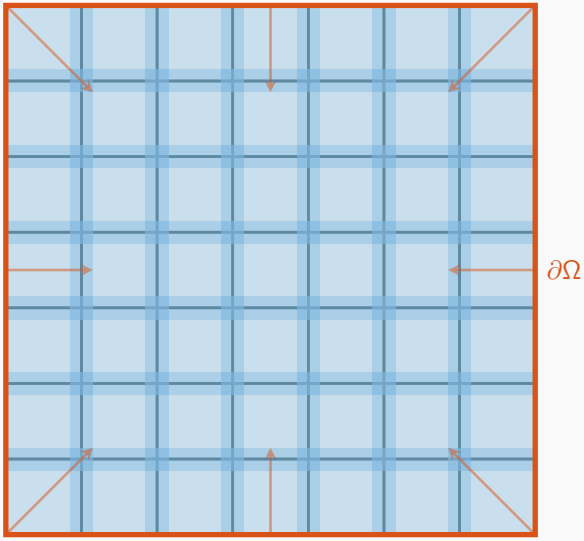


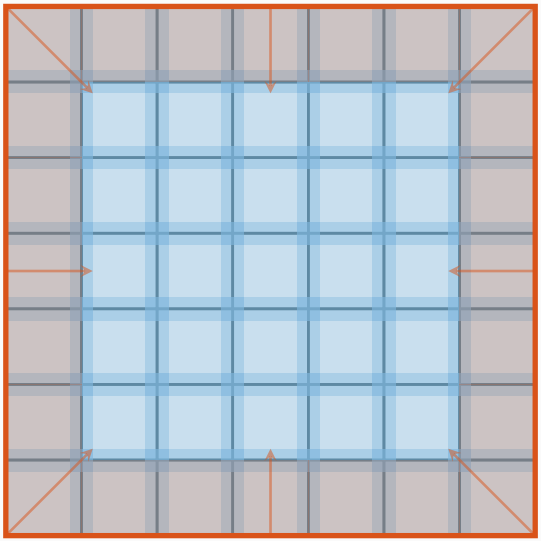
# Transport of Information – One-Level Overlapping Schwarz Methods



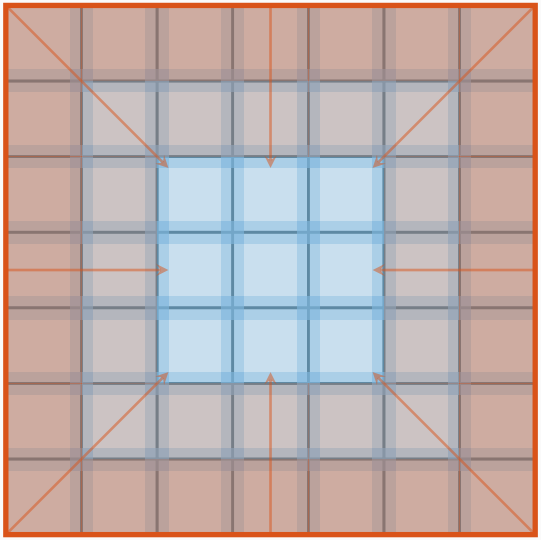
# Transport of Information – One-Level Overlapping Schwarz Methods



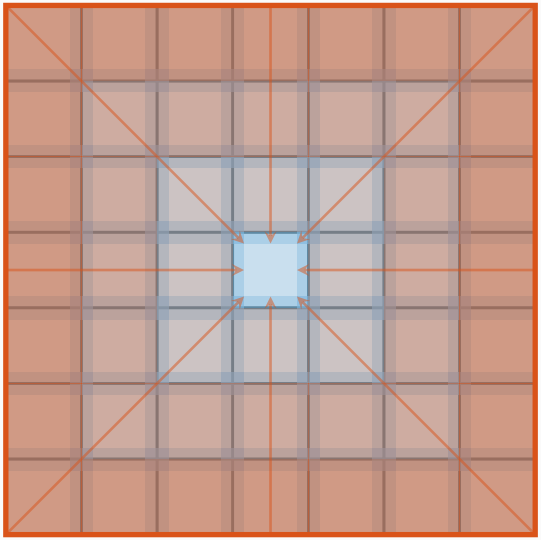
# Transport of Information – One-Level Overlapping Schwarz Methods



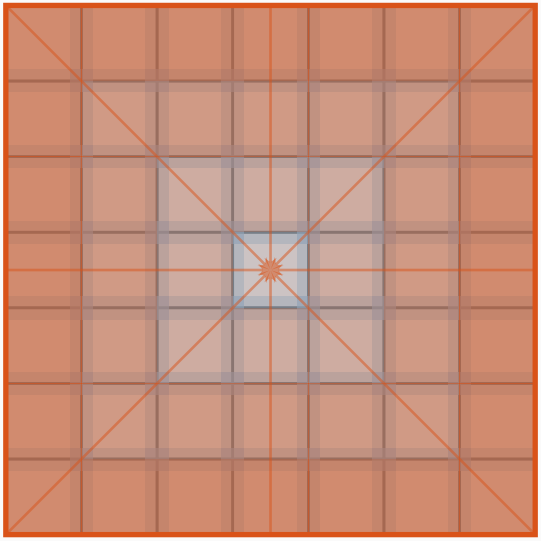
# Transport of Information – One-Level Overlapping Schwarz Methods



# Transport of Information – One-Level Overlapping Schwarz Methods



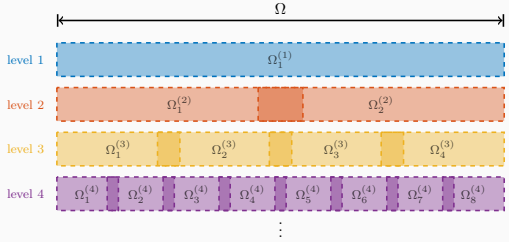
# Transport of Information – One-Level Overlapping Schwarz Methods




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

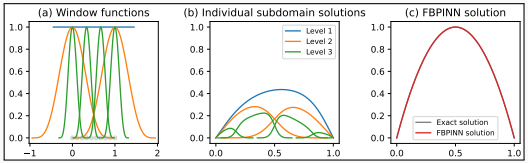
# Multi-Level FBPINN Algorithm

Extension of FBPINNs to $L$ levels; Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.




## $L$-level network architecture

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




## Multi-Frequency Problem

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

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

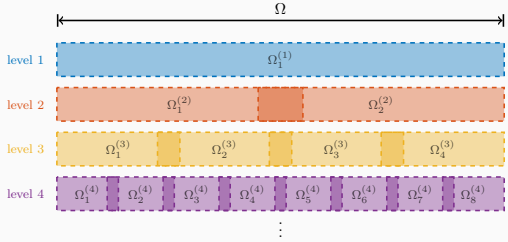
$$u = 0 \quad \text{on } \partial\Omega,$$

with $\omega_i = 2^i$.

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

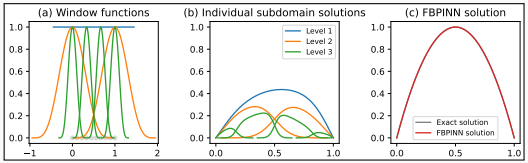
# Multi-Level FBPINN Algorithm

Extension of FBPINNs to $L$ levels; Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.




## $L$-level network architecture

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




## Multi-Frequency Problem

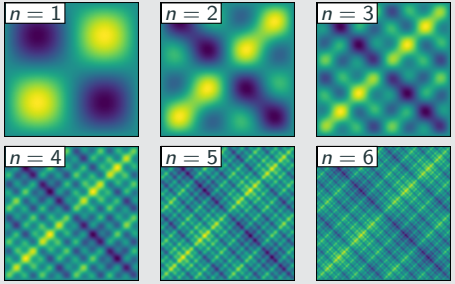
Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

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

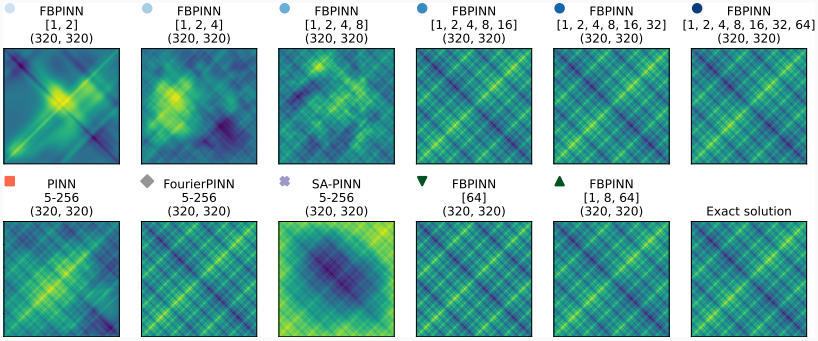
$$u = 0 \quad \text{on } \partial\Omega,$$

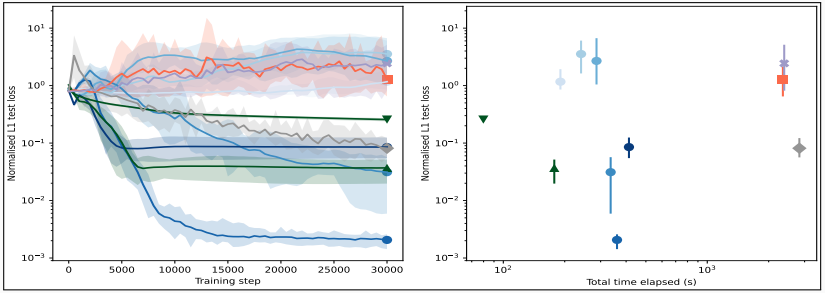
with $\omega_i = 2^i$.

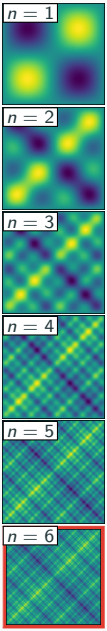
For increasing values of $n$, we obtain the **analytical solutions**:



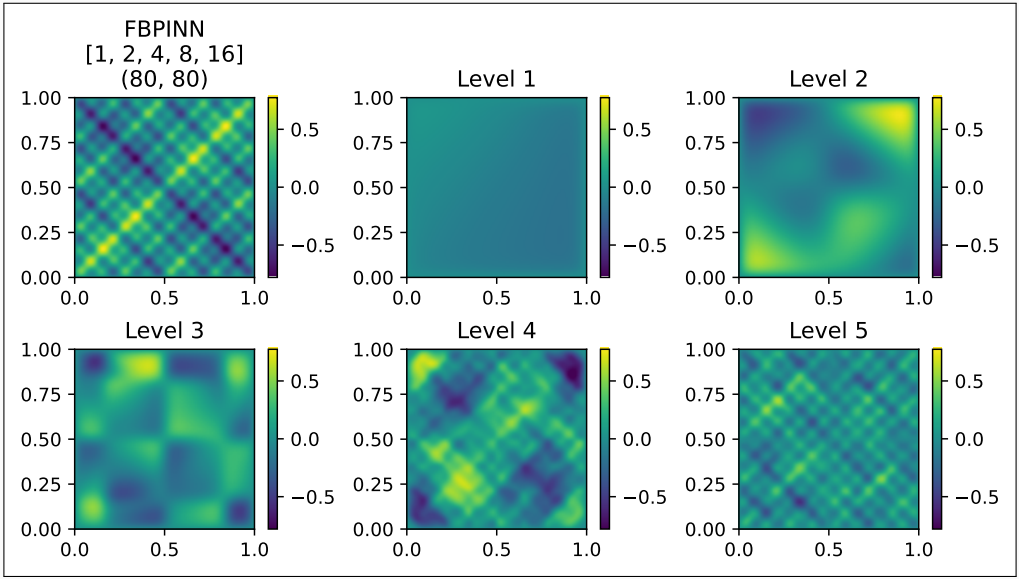
# Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling



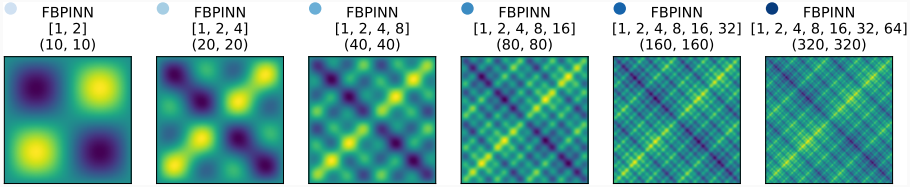



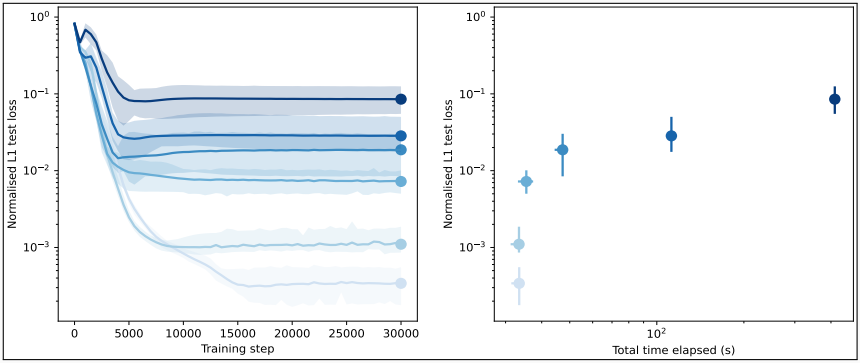
# Multi-Frequency Problem – What the FBPINN Learns

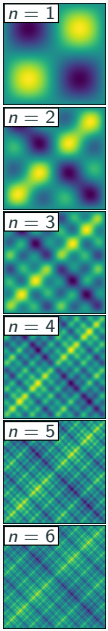



Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.

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



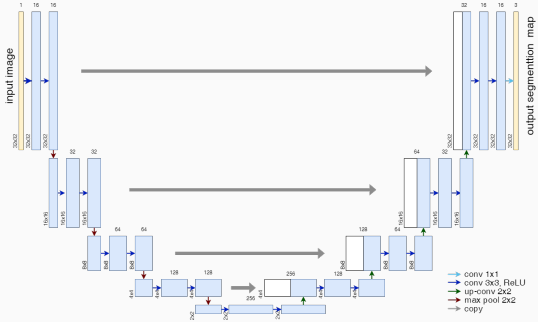




- Ongoing: **analysis and improvement of the convergence**

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

# Domain Decomposition for Convolutional Neural Networks

# Memory Requirements for CNN Training

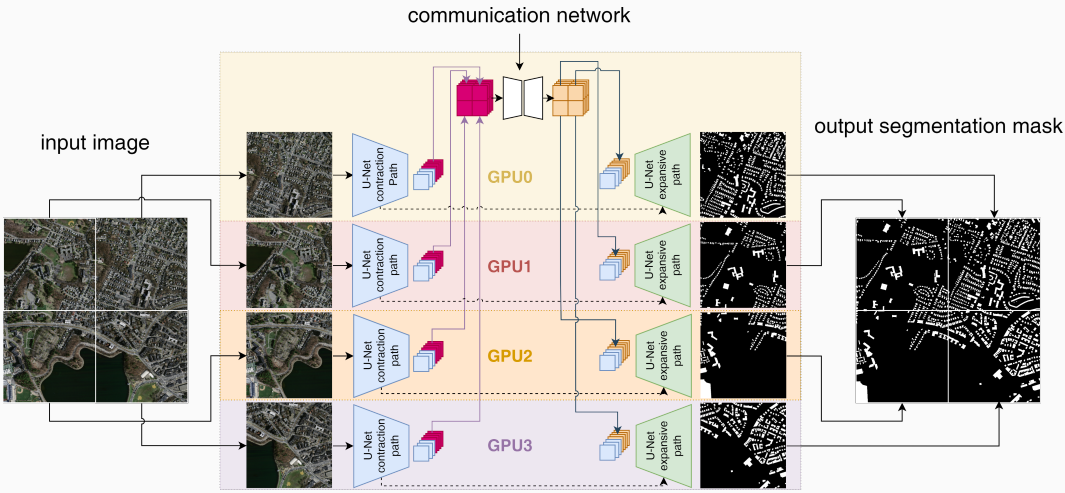



- As an example for a **convolutional neural network (CNN)**, we employ the **U-Net architecture** introduced in **Ronneberger, Fischer, and Brox (2015)**.
- The U-Net yields **state-of-the-art accuracy in semantic image segmentation** and other **image-to-image tasks**.

***Below:*** *memory consumption for training on a single* $1024 \times 1024$ *image.*

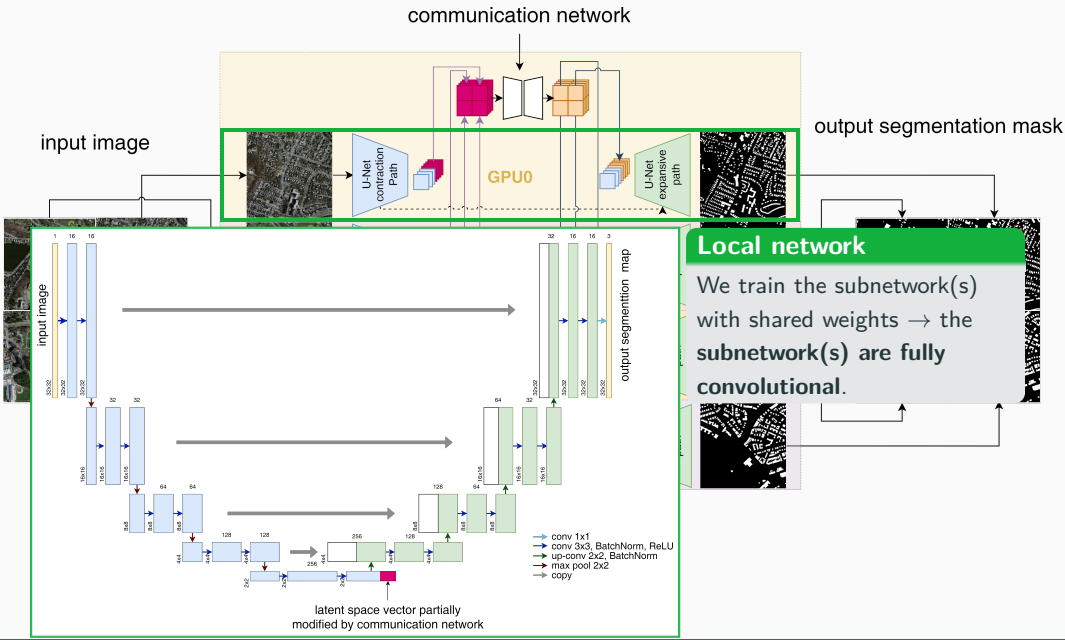
| name | size | # channels | | mem. feature maps | | mem. weights | |
|---|---|---|---|---|---|---|---|
| | | input | output | # of values | MB | # of values | MB |
| input block | 1 024 | 3 | 64 | 268 M | **1 024.0** | 38 848 | **0.148** |
| encoder block 1 | 512 | 64 | 128 | 167 M | **704.0** | 221 696 | **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).**

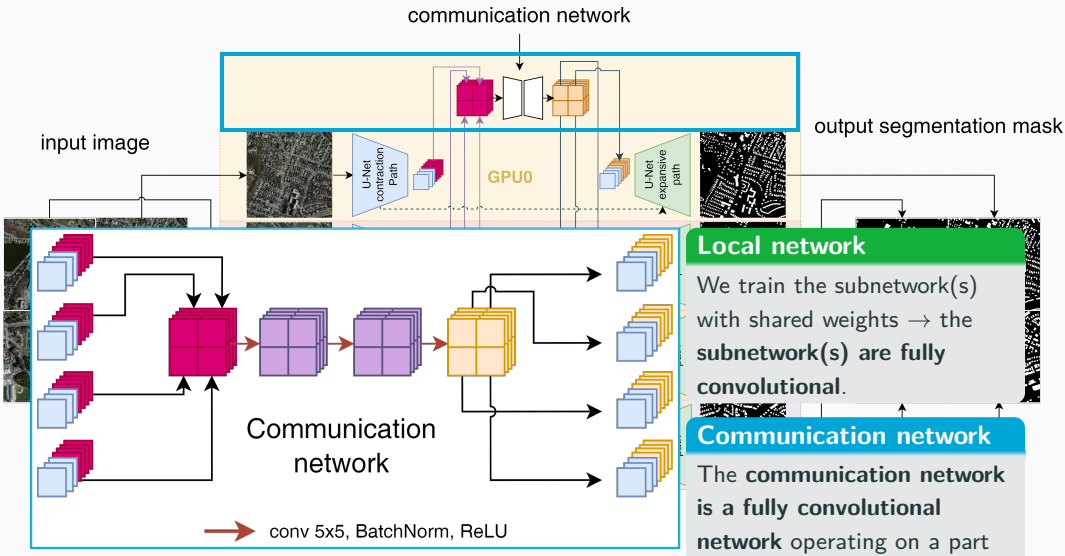
# Decomposing the U-Net




**Local network**

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

# Decomposing the U-Net



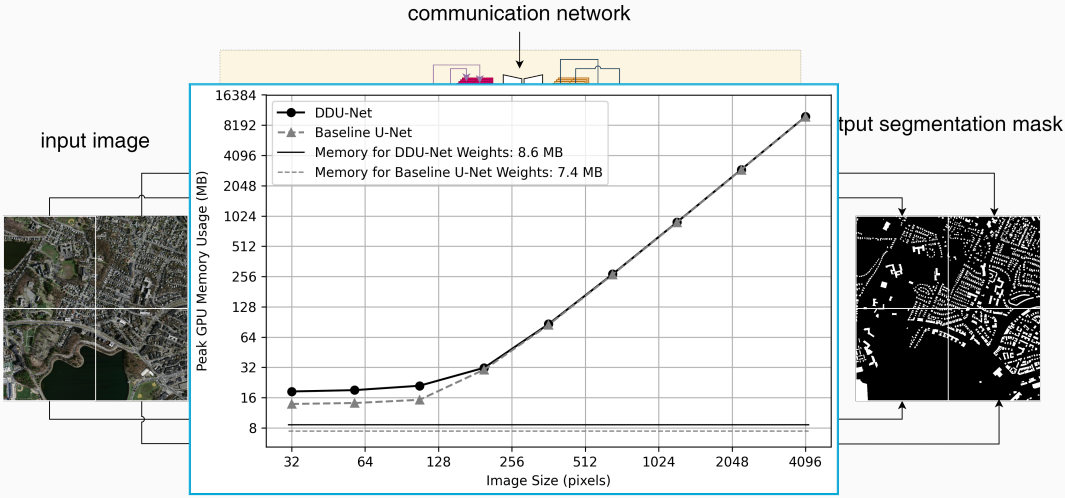

**Local network**

We train the subnetwork(s) with shared weights $\rightarrow$ 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.

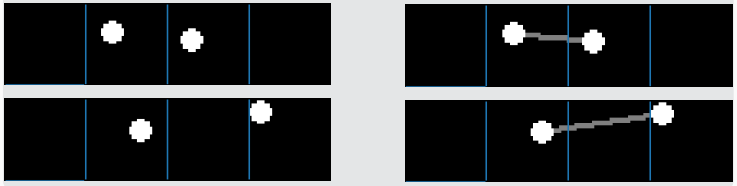
# Decomposing the U-Net



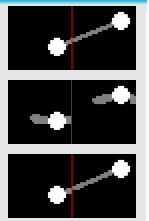

- Distribution of feature maps results in **significant reduction of memory usage on a single GPU**
- **Moderate additional memory usage** due to the **communication network**

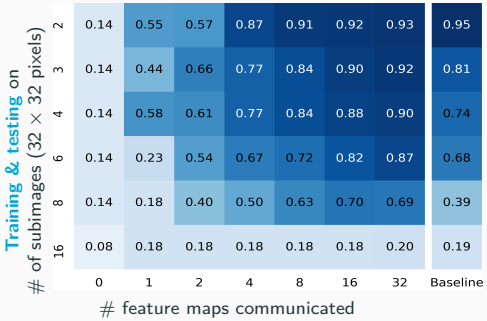
# Results – Synthetic Data Set

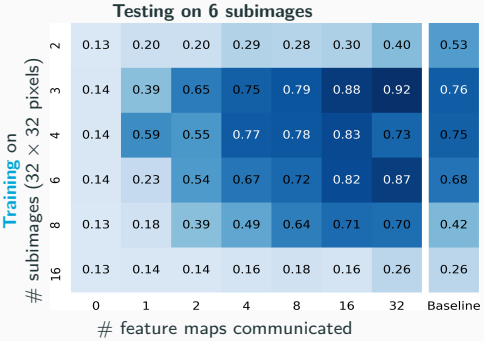
## Task: Connect two dots via a line segment




## Result: Communication



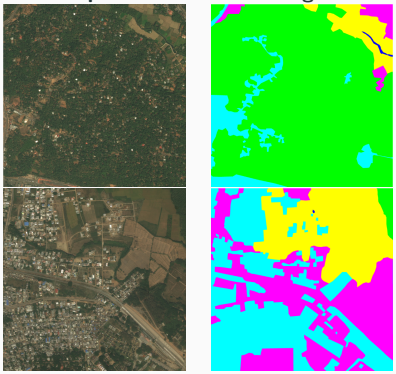



# DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

| 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

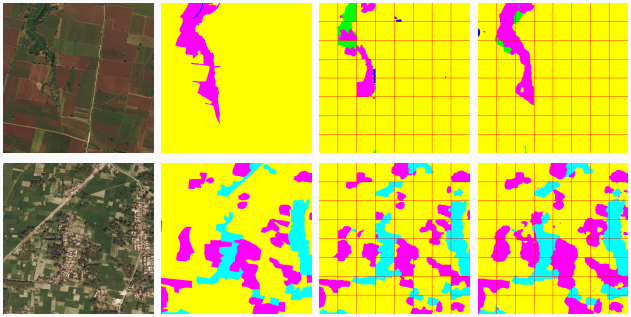


## Avoiding overfitting

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)**)

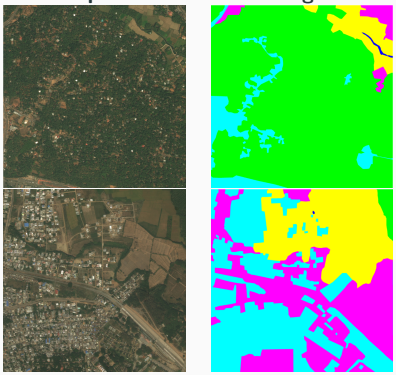
image true mask without comm. with comm.



# 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 % |

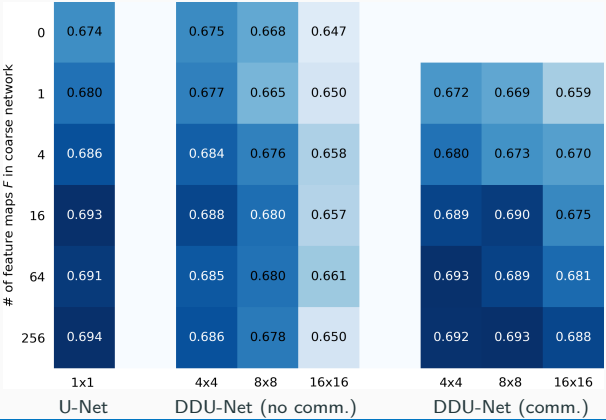
Input Target



## 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)**)



## FBPINNs – Domain Decomposition for Physics-Informed Neural Networks

- **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**.

## 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!