Solving Nonlinear PDE with Physics-Informed Neural Network: 1D Burgers' Equation

Presenter: Huy Tong (VNUHCM-US)

Supervisors:

Dr. Hsien Shang

Dr. Somdeb Bandopadhyay

(ASIAA)

NCTS-TCA Summer Student Program

National Tsing Hua University, 30th August 2024

Outline

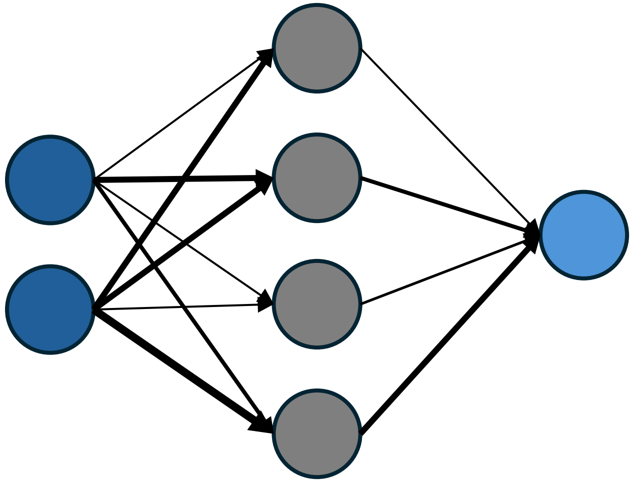
- **Motivation and objective**
- **Problem statement**
- **Method**
  - ➢Overview on physics-informed neural network
  - ➢DeepXDE: overview on usage and features
- **Result**
  - ➢Fitting result, loss during training, comparison across different models (feed-forward, ResNet)
- **Discussion and Summary**

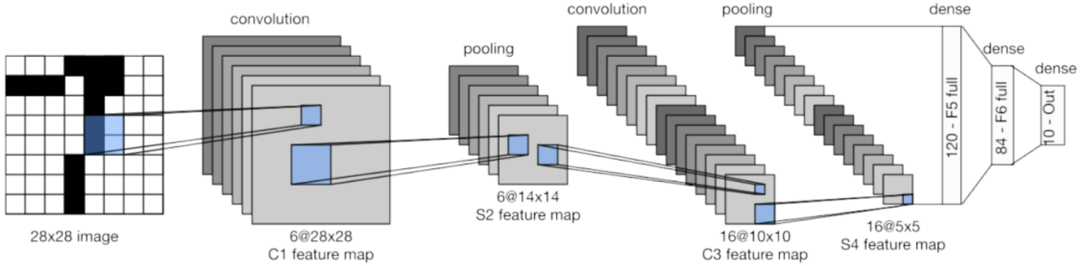
Motivation and objective

- **Motivation:**
  - ➢Physics-informed neural network (PINN) is a deep learning framework for solving PDE-related problems
  - ➢Using machine learning, we can make use of prior knowledge in our computations
- **Objective:**
  - ➢Establish an overview on PINN and usage of DeepXDE
  - ➢Evaluate the performance of PINN with different architectures

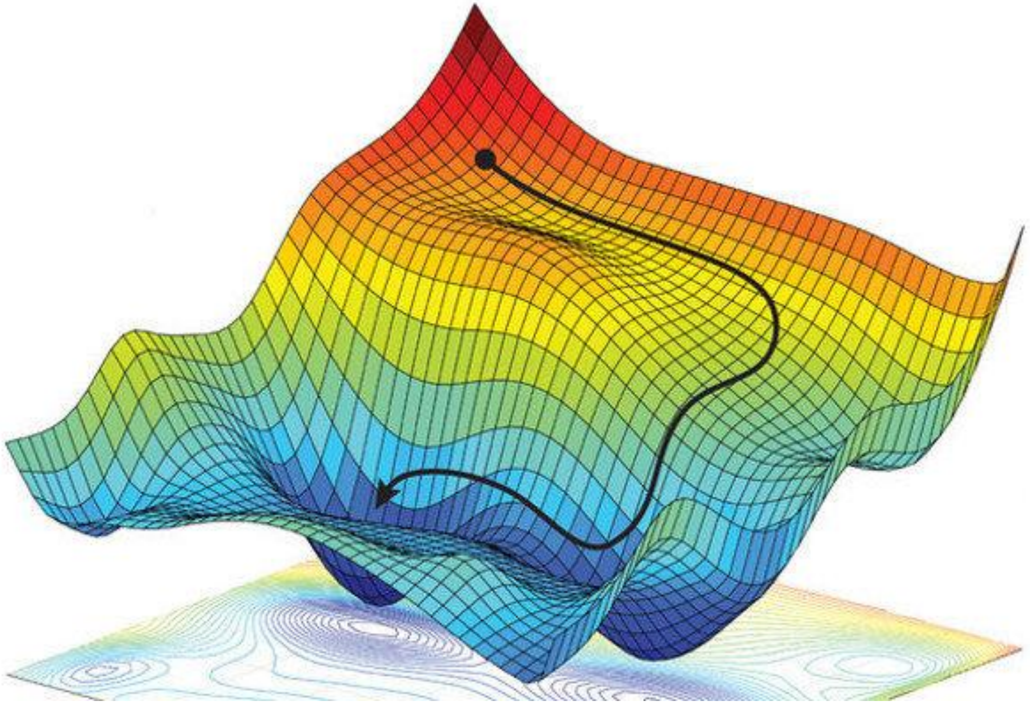
Terminologies

Architecture: structure/design of a NN





Loss function: error of the model



Training: finding the optimal model by minimizing loss

Problem statement

Partial Differential Equation

$$\frac{du}{dt} + u\frac{du}{dx} = v\frac{d^2u}{dx^2},$$

Computational Domain

$$x \in [0,1], t \in [0,1]$$

$$u(0,t) = u(1,t) = 0, \qquad u(x,0) = func(x,0)$$

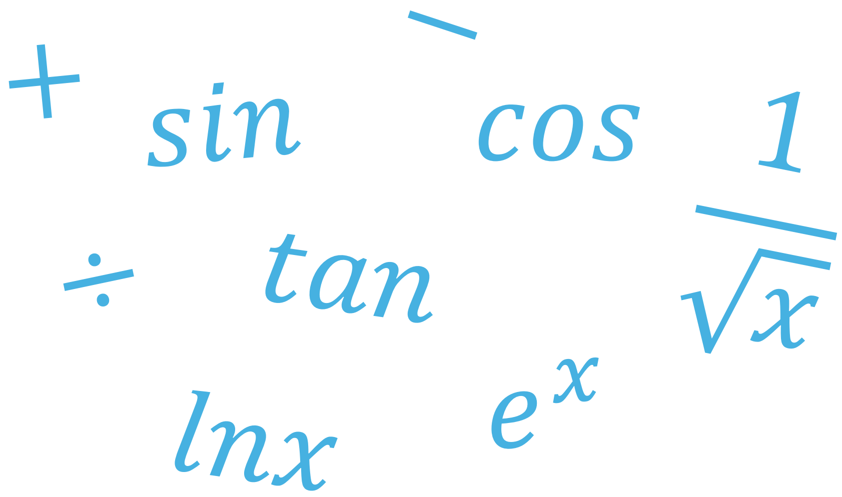
Boundary and Initial Condition

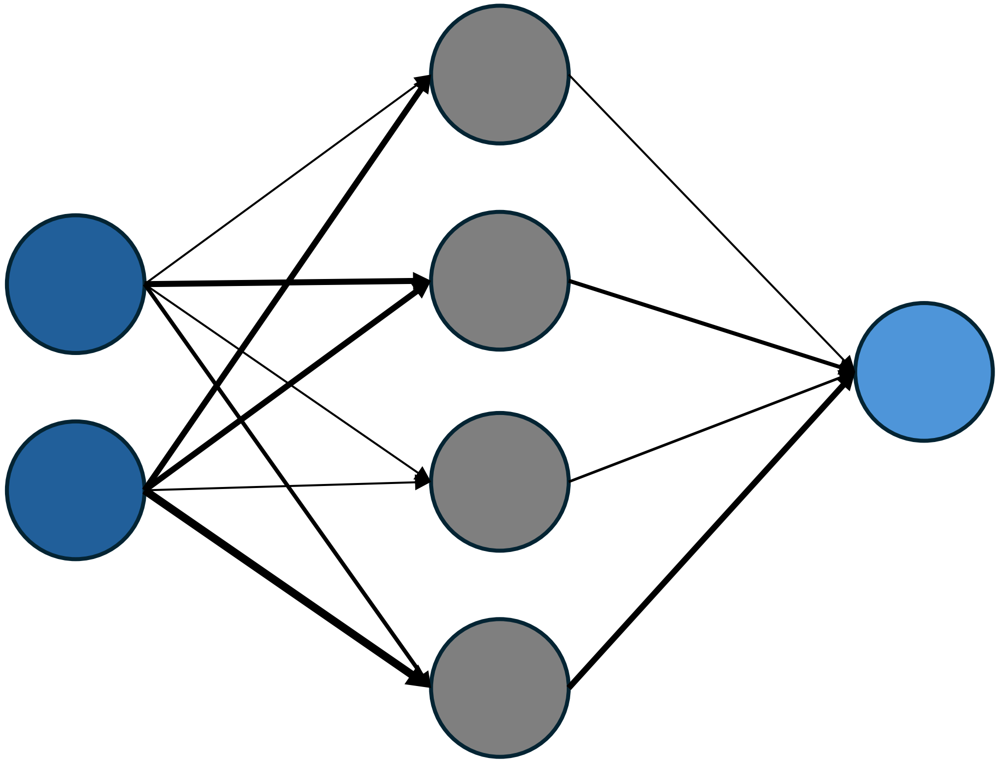
Goal: find function $u(x,t)$ that satisfy all constraints

Physics-informed neural network

$$\begin{matrix} x \\ t \end{matrix} \longrightarrow u(x,t) \longrightarrow y$$

Function $u(x,t)$ can be...





Physics-informed neural network

- **For regular neural network…**
  - ➢Need to train with data
  - ➢Optimizing using only data-model error might not be enough

- **For PINN…**
  - ➢Incorporate physical information (PDE, conditions) into the loss function

Hence the "PHYSICS-INFORMED"!

PINN: Loss function

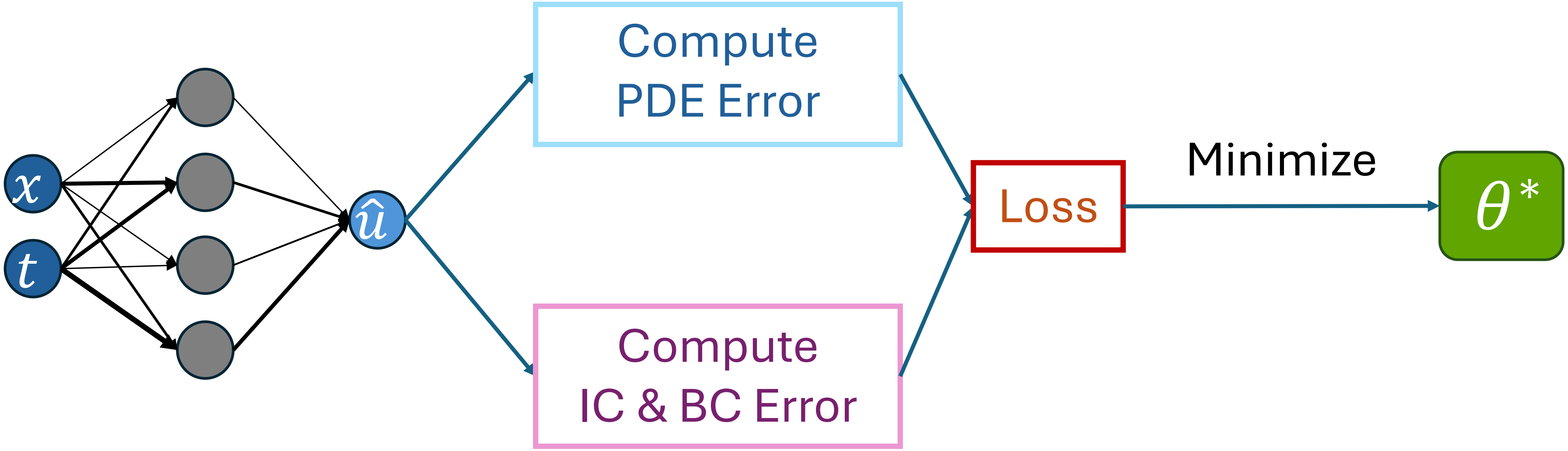
Weights

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{T}) = \boxed{w_f}\,\mathcal{L}_f(\boldsymbol{\theta};\mathcal{T}_f) + \boxed{w_b}\,\mathcal{L}_b(\boldsymbol{\theta};\mathcal{T}_b)$$

PDE Error

IC/BC Error

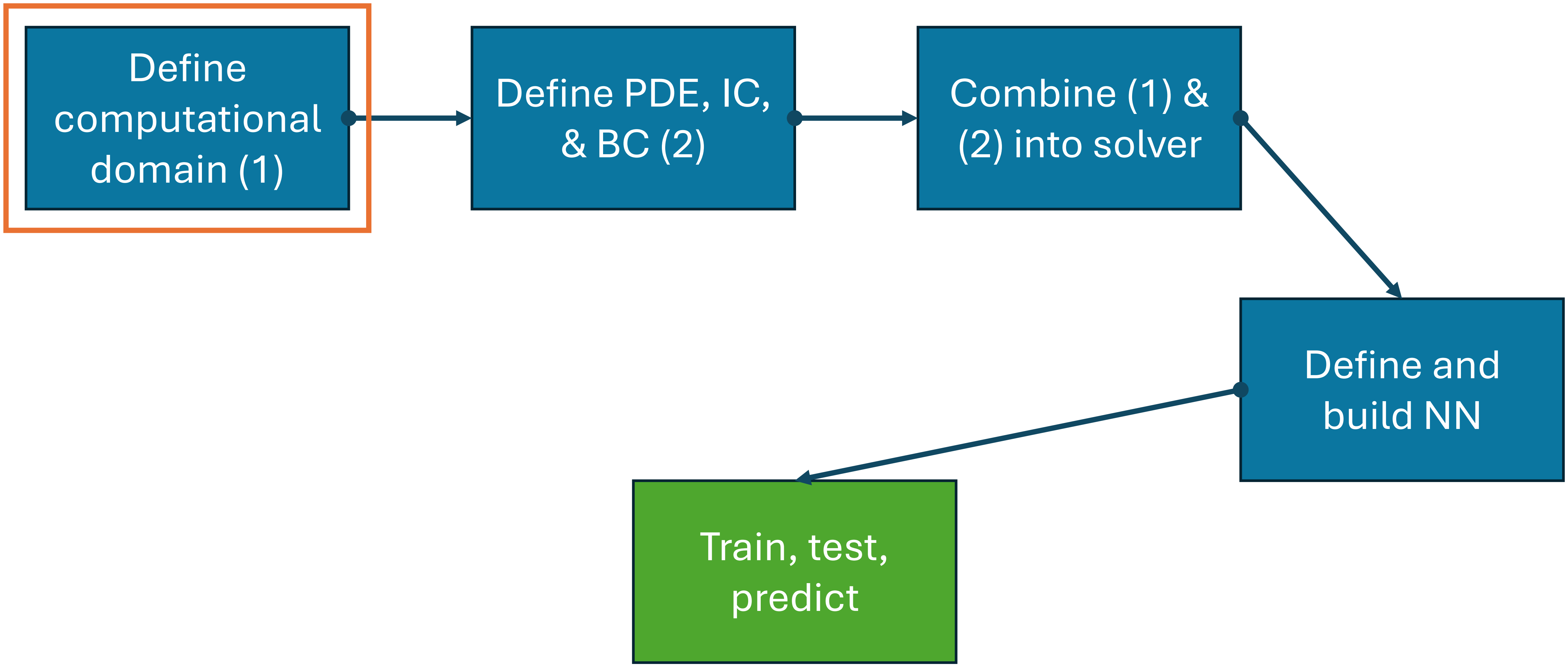
PINN: Training scheme



DeepXDE: An overview

- Developed by Lu Lu et al. 2021
- Implemented several architectures of NN: feed-forward, ResNet, DeepONet, multifidelity neural network
- Supported backends: TensorFlow, PyTorch, JAX, PaddlePaddle
- Documentation website: https://deepxde.readthedocs.io/en/latest/index.html

DeepXDE: Usage



DeepXDE: Customization

Procedure 3.2 Customization of the new geometry module **`MyGeometry`**. The class methods should only be implemented as needed.

Custom Domain

```python
class MyGeometry(Geometry):
    def inside(self, x):
        """Check if x is inside th
    def on_boundary(self, x):
        """Check if x is on the geometry boundary."""
    def boundary_normal(self, x):
        """Compute the unit normal at x for Neumann or Robin boundary conditions."""
    def periodic_point(self, x, component):
        """Compute the periodic image of x for periodic boundary condition."""
    def uniform_points(self, n, boundary=True):
        """Compute the equispaced point locations in the geometry."""
    def random_points(self, n, random="pseudo"):
        """Compute the random point locations in the geometry."""
    def uniform_boundary_points(self, n):
        """Compute the equispaced point locations on the boundary."""
    def random_boundary_points(self, n, random="pseudo"):
        """Compute the random point locations on the boundary."""
```

Procedure 3.3 Customization of the neural network **`MyNet`**.

Custom NN

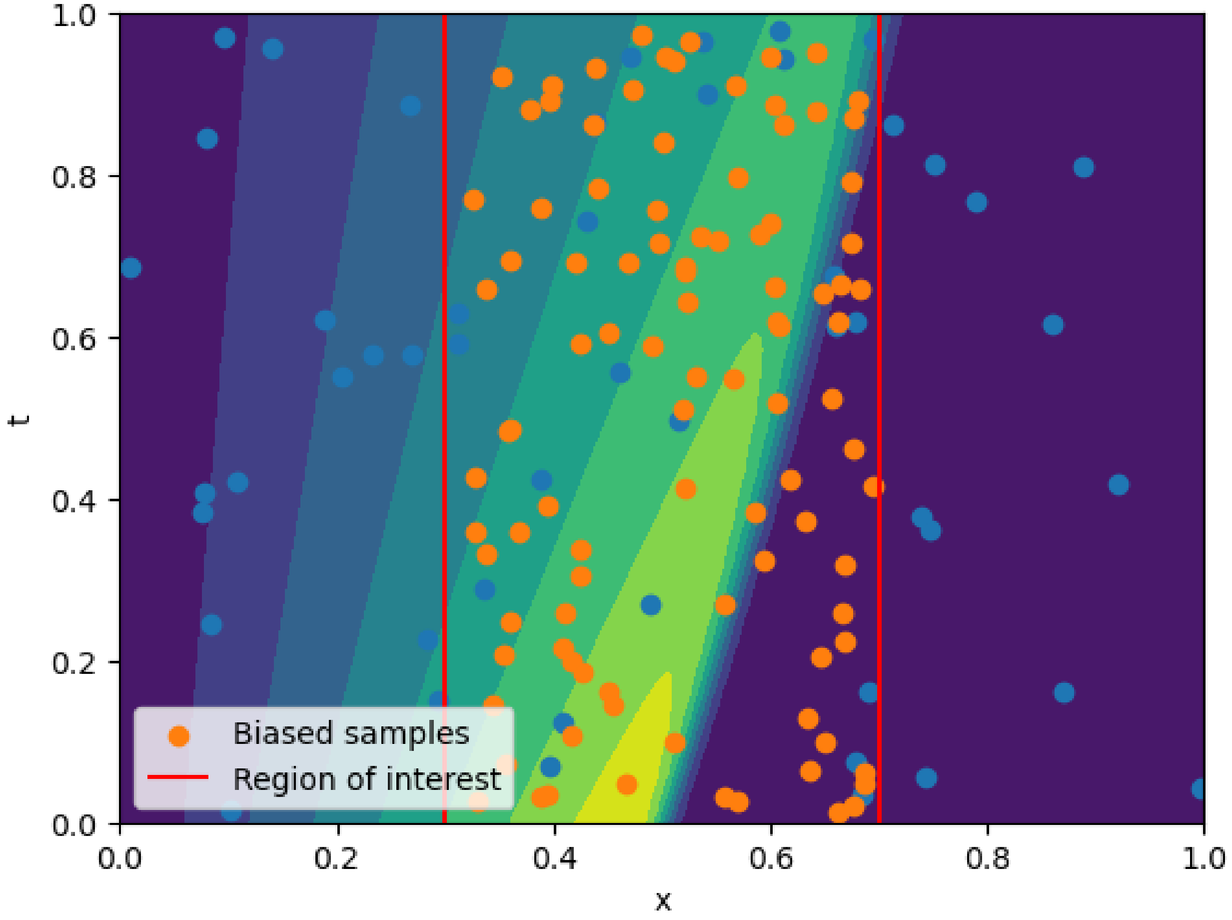
```python
class MyNet(Map):
    @property
    def inputs(self):
        """Return the net inputs."""
    @property
    def outputs(self):
        """Return the net outputs."""
    @property
    def targets(self):
        """Return the targets of the net outputs."""
    def build(self):
        """Construct the network."""
```

Procedure 3.4 Customization of the callback **`MyCallback`**. Here, we only show how to add functions to be called at the beginning/end of every epoch. Similarly, we can call functions at the other training stages, such as at the beginning of training.

Custom Callback

```python
class MyCallback(Callback):
    def on_epoch_begin(self):
        """Called at the beginning of every epoch.
    def on_epoch_end(self):
        """Called at the end of every epoch."""
```

Training data overview



Exact solution:

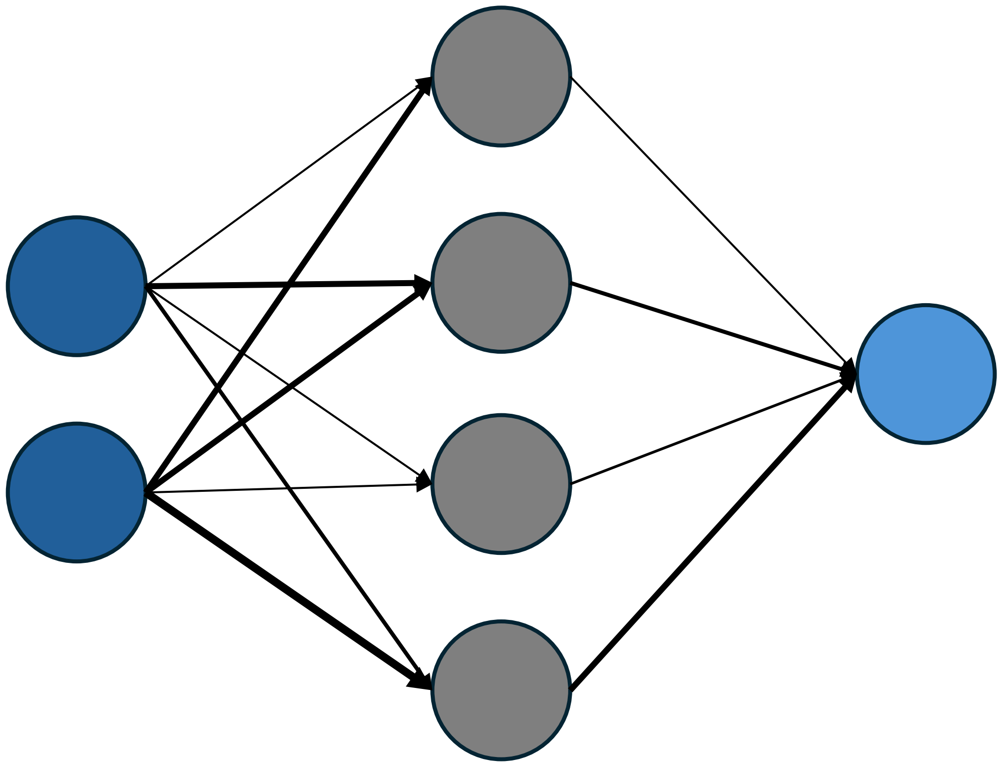
$$\frac{x}{t+1} \div \left(1 + \sqrt{\frac{t+1}{t_0}} \times exp\left(\frac{R_{num} \times x^2}{4t+4}\right)\right)$$

With: $t_0 = \ exp\left(\frac{R_{num}}{8}\right)$

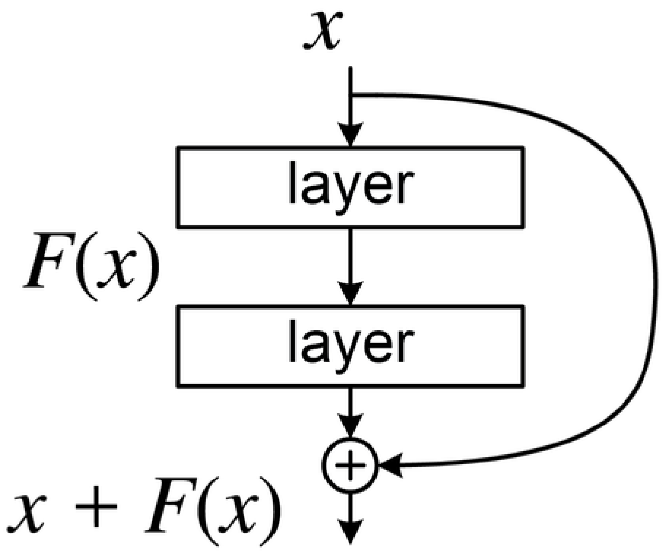
- **Biased sampling:** generate more sample points at region of interest
- Useful for helping the model learn interesting properties in the RoI

Architectures used

Fully-connected NN

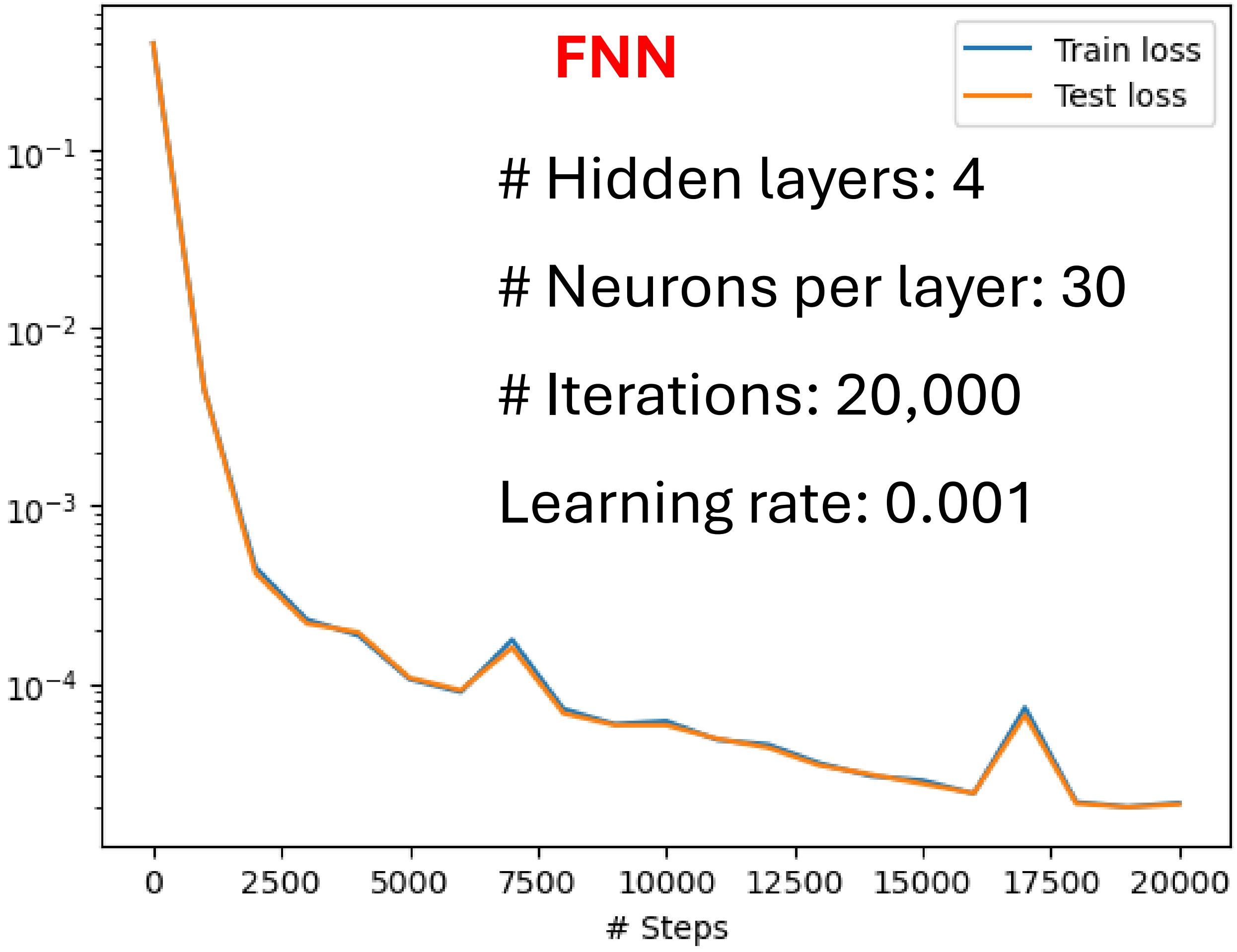


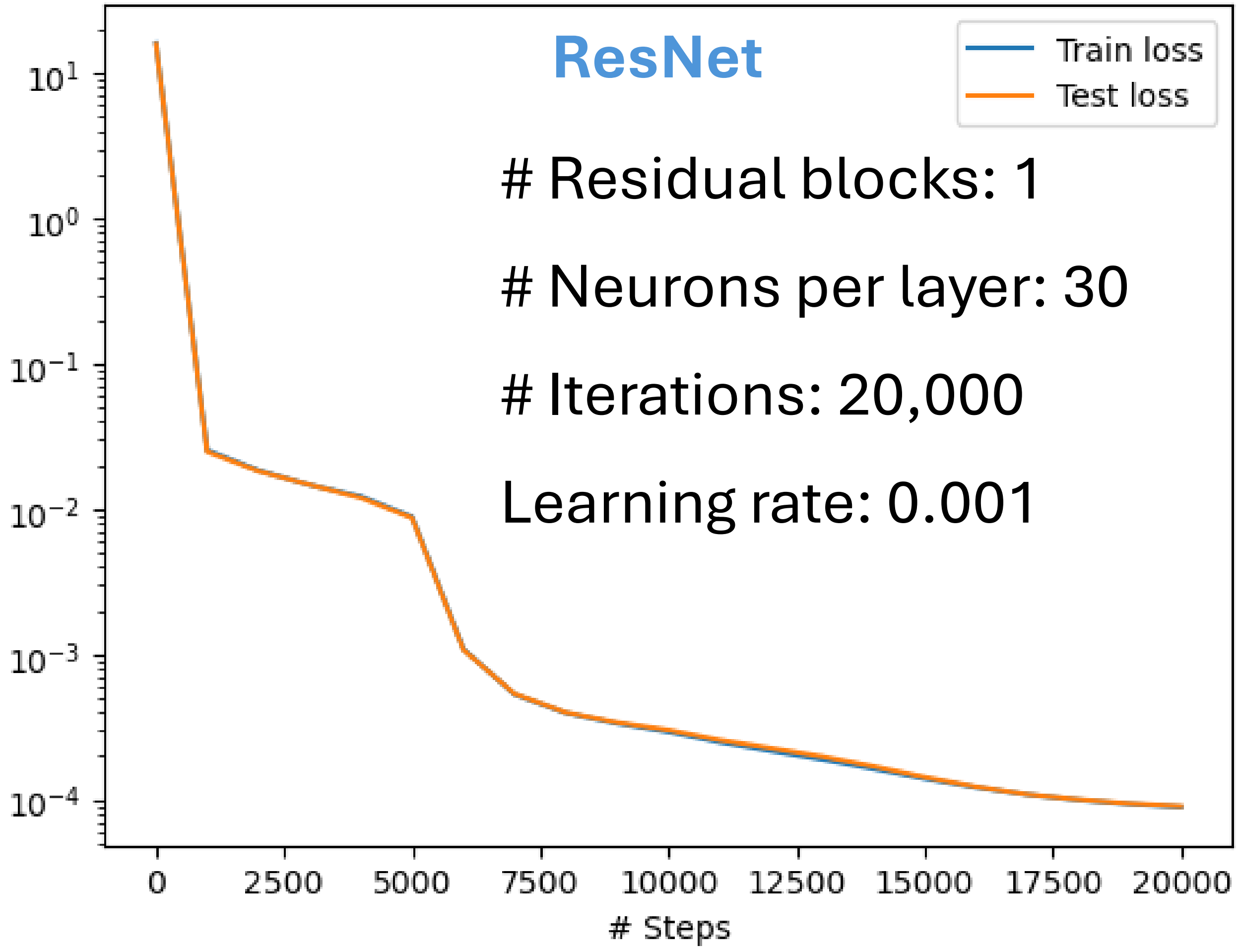
Residual NN



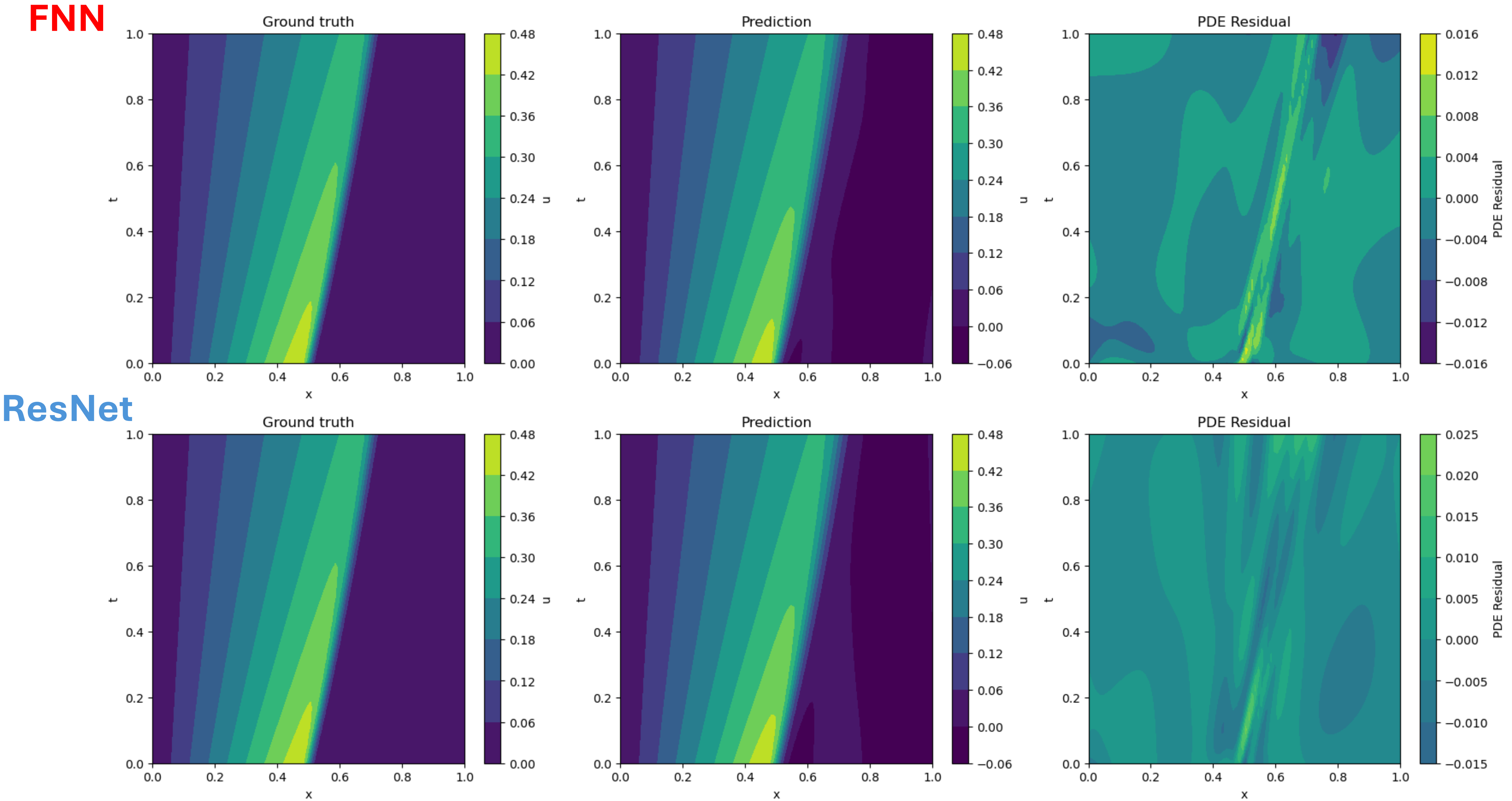
Aim to tackle the problem of vanishing gradient in deep networks

Result: Train loss

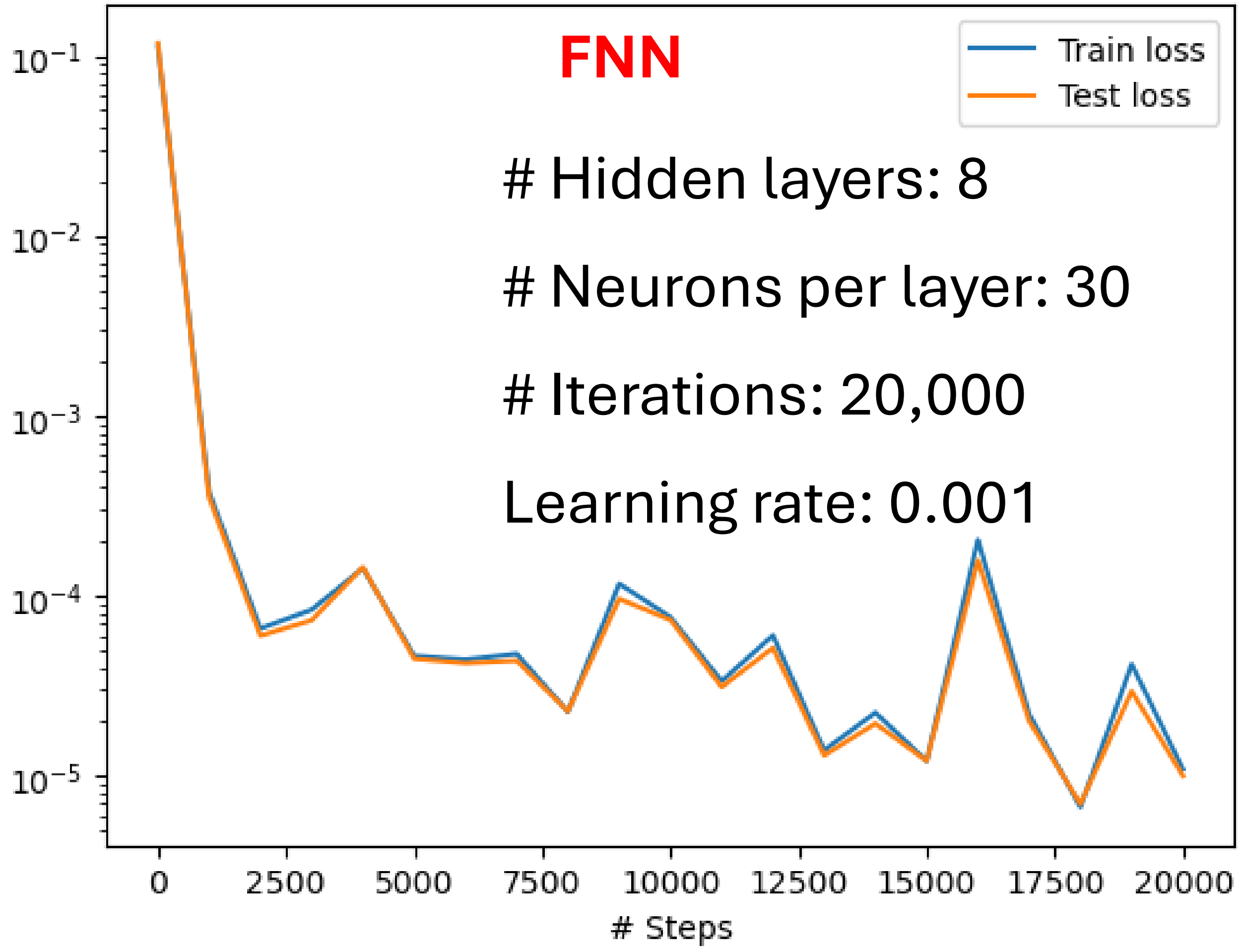


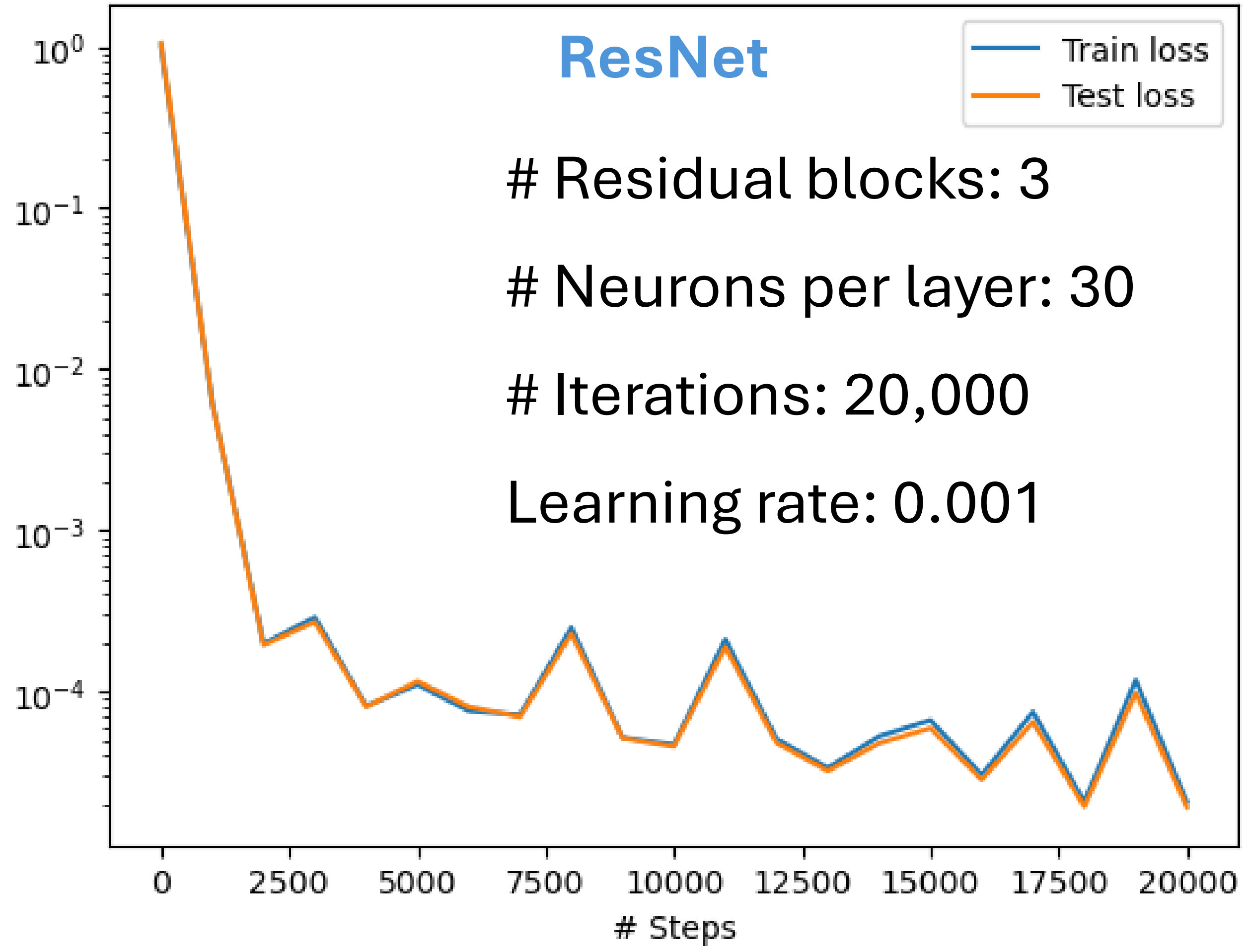


Result: Visualization

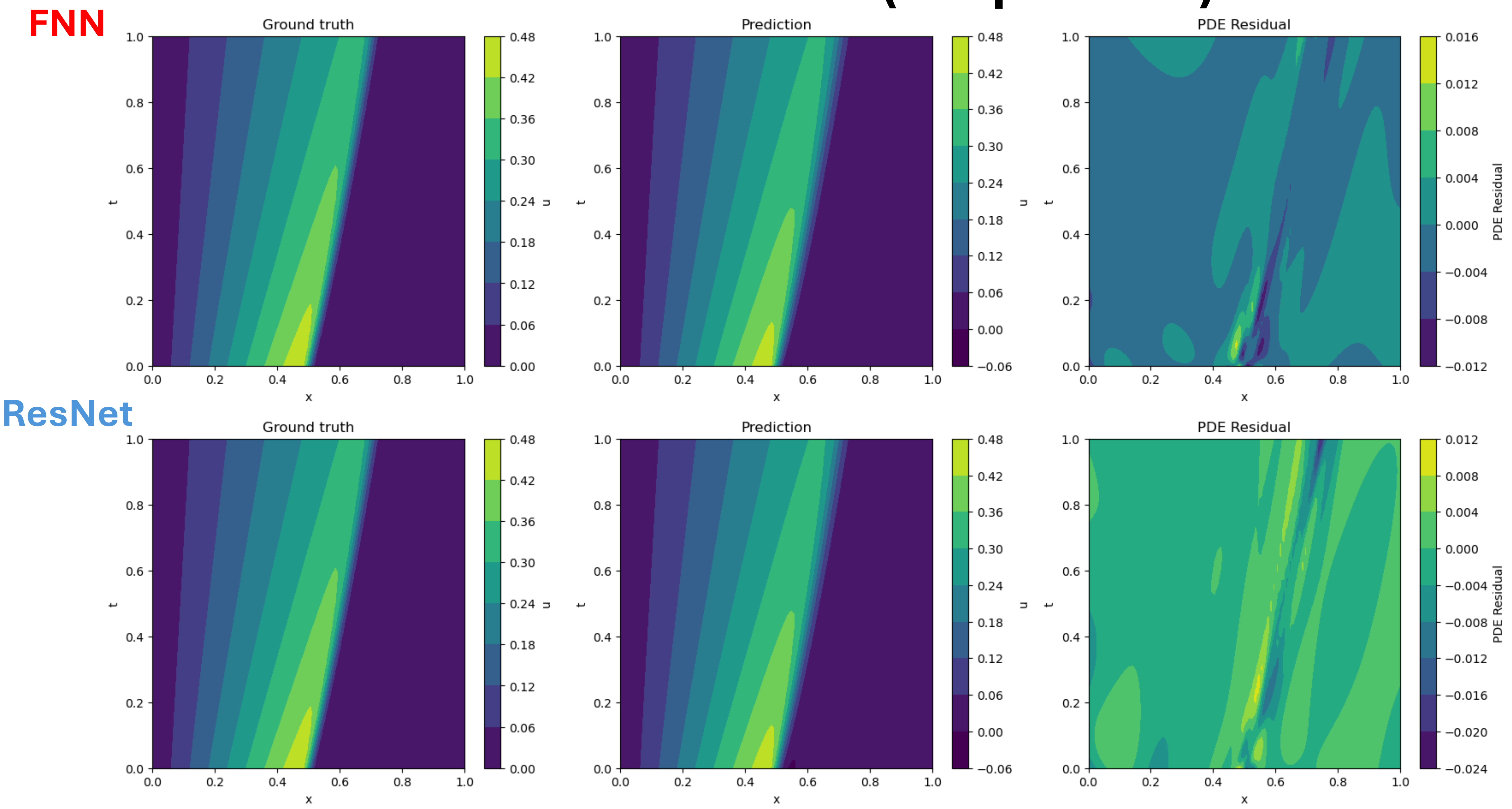


Result: Train loss (deeper nets)





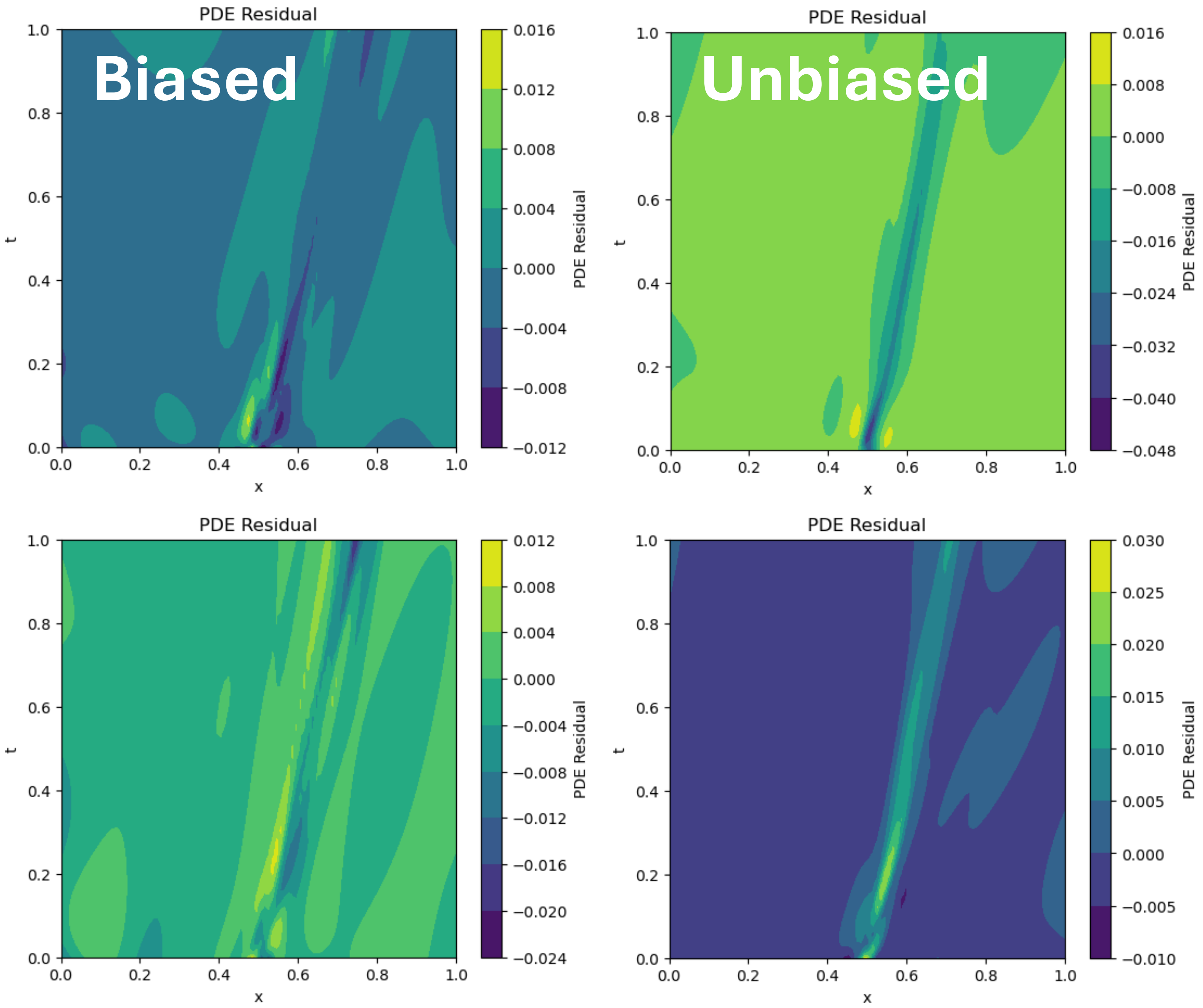
Result: Visualization (deeper nets)



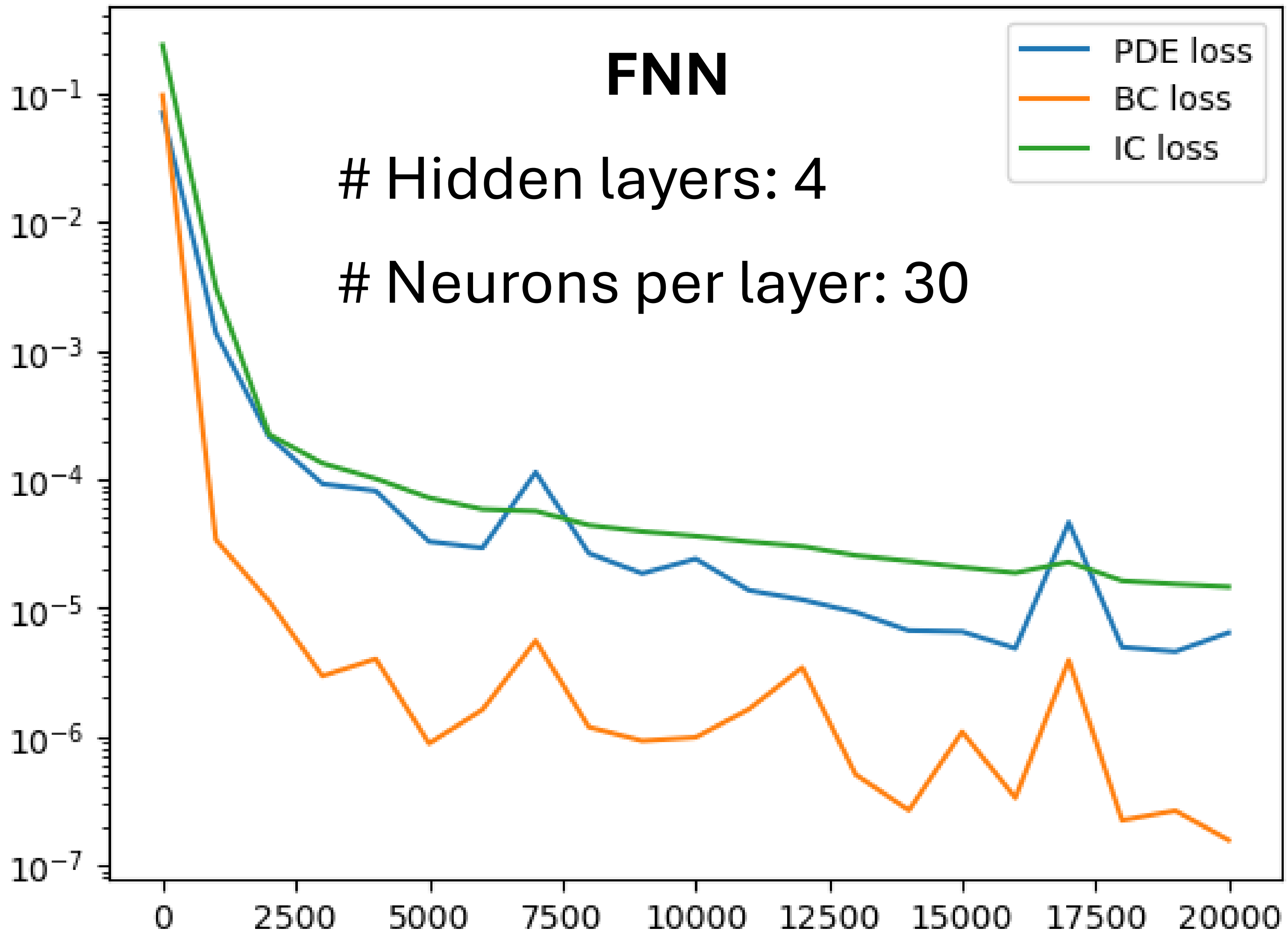
Biased vs unbiased sampling

FNN

ResNet



Discussion on loss



Which component of the loss function is the most important?

Which component should we prioritize?

Optimal combination of loss weights?

Discussion

- With PINN's loss function design, the model can also learn from a small dataset
- The PINN framework gives us the freedom to experiment with different architecture

- **Outlook:**
    - ➢Experiment with different architectures
    - ➢Try out combinations of loss weights
    - ➢Use different optimizing algorithms

Summary

- Neural network can be used in place of a solution to a PDE
- PINN aims to incorporate physical information into the training process
- There are still rooms for experimentation (architectures, loss function components, optimizing strategies, ...)

The outlook of PINN is as exciting as machine learning!

Thank you for your attention!