Exam Projects for Deep Learning for PDEs in Engineering Physics

June 30, 2025

Problem A: 1D Elastostatics

Consider a rod made of linearly elastic material subjected to some load. Static problems will be considered here, by which is meant it is not necessary to know how the load was applied, or how the material particles moved to reach the stressed state; it is necessary only that the load is applied slowly enough so that the accelerations are zero, or that it was applied sufficiently long ago that any vibrations have died away and movement has ceased.

The equations governing the static response of the rod are:

$$-\frac{d}{dx}\left(E(x)\frac{du}{dx}\right) = f, \quad x \in (0, L) \tag{1}$$

where

- $u(x)$: displacement field of the rod
- $E(x)$: Young's modulus
- $f = 9.81$: body force per unit length (e.g., gravity)
- $L = 1.$: Length of the rod

We consider the fixation of both sides of the rod, which leads to the following boundary conditions:

- $u(0) = u(1) = 0$

Task 1: Solve the displacement field $u(x)$ given the Young's modulus $E(x)$

Young's modulus $E(x)$ is given as a piecewise-constant function:

$$E(x) = \begin{cases} 5. & 0.15 < |x - 0.5| < 0.35 \\ 2. & \text{otherwise} \end{cases}$$

The goals

- Please select a suitable deep learning method for solving this problem to obtain the displacement field $u(x)$, and explain the reason for using it.
- Report your setups for the implementation, such as network structure, activation function, optimizer (with learning rate), epoch (with batch size), loss weights, and other tricks that are used for improvement.
- Compute the L^2 relative error (on testing dataset) at each training epoch and plot the `Error vs. epoch` curve (and report the final error). The L^2 relative error between the prediction u_{pred} and the truth u_{true} is defined as follows:

$$\text{error} = \sqrt{\frac{\sum_i^n |u_{\text{pred}}(x_i) - u_{\text{true}}(x_i)|^2}{\sum_i^n |u_{\text{true}}(x_i)|^2}}$$

- Plot the predicted solution (and the ground truth reference) and the pointwise absolute error using separate figures with `matplotlib`.

Dataset

The ground truth reference is given in the dataset `LinearElasticity1d.h5`:

- `x_test`: the locations where the ground truth is evaluated (Should not be used for training)
- `u_test`: the ground truth reference (Should not be used for training)
- Link to dataset: `https://www.kaggle.com/datasets/yhzang32/physicsinformedlearning4pde`

Task 2: Recover the Young's modulus $E(x)$ from the observation of displacement field $u(x)$

In this task, the Young's modulus $E(x) > 0$ of the rod is unknown. However, we observe the displacement field u_{obs} (contaminated by noise with noise level $\sim 5\%$) on a set of randomly placed sensors x_{obs} (with size $N_{\text{obs}} = 250$). Moreover, we are able to measure the Young's modulus of the rod at two boundary sides, i.e., $E(0) = E(1) = 1$.

The goals

- Please select a suitable deep learning method for solving this inverse problem to recover the Young's modulus $E(x)$, and explain the reason for using it.
- Report your setups for the implementation, such as network structure, activation function, optimizer (with learning rate), epoch (with batch size), loss weights, and other tricks that are used for improvement.
- Compute the L^2 relative error (on testing dataset) at each training epoch and plot the `Error vs. epoch` curve (and report the final error). The L^2 relative error between the prediction e_{pred} (or u_{pred}) and the truth e_{true} (or u_{true}) is defined as follows:

$$\text{error} = \sqrt{\frac{\sum_i^n |e_{\text{pred}}(x_i) - e_{\text{true}}(x_i)|^2}{\sum_i^n |e_{\text{true}}(x_i)|^2}}$$

- Plot the predicted solution (and the ground truth reference) and the pointwise absolute error using separate figures with `matplotlib`.

Dataset

The observation and the ground truth references are given in the dataset `LinearElasticity1d_inverse.h5`

- `x_obs`: the observation sensors
- `u_obs`: the observed displacement field u (contaminated by noise)

- `x_test`: the locations where the ground truth is evaluated (Should not be used for training)
- `e_test`: the ground truth reference for Young's modulus (Should not be used for training)
- `u_test`: the ground truth reference for displacement field (Should not be used for training)
- Link to dataset: `https://www.kaggle.com/datasets/yhzang32/physicsinformedlearning4pde`

Problem B: Effective Thermal Conductivity of 2-Phase Materials

Predicting the effective thermal conductivity k_{eff} of a two-phase composite material is a classical and practical problem in materials science. The goal is to estimate the bulk conductivity of a heterogeneous material made of two constituents with thermal conductivities k_1 and k_2. One efficient way to compute the effective thermal conductivity of given 2-phase materials is through numerical homogenization.

In numerical homogenization, we need to solve the steady-state heat equation on a representative volume element (RVE):

$$-\nabla \cdot (k(x,y)\nabla T) = 0, \quad (x,y) \in \Omega = [0,1]^2 \tag{2}$$

with boundary conditions:

- $T(x=0,y)=0,\ T(x=1,y)=1,$
- $\frac{\partial T}{\partial \vec{n}}(x,y=0) = \frac{\partial T}{\partial \vec{n}}(x,y=1) = 0,$

where $T(x,y)$ indicates the temperature and $k(x,y)$ indicates thermal conductivity, which is defined piecewise:

$$k(x,y) = \begin{cases} k_1 = 2., & (x,y) \in \Omega_1 \\ k_2 = 10., & (x,y) \in \Omega_2 \end{cases}$$

where Ω_1, Ω_2 denote the first phase region and the second phase region, respectively.

Task: Predicting the Temperature Field $T(x,y)$ Given 2-Phase Material $k(x,y)$

In this task, we have collected many materials $k(x,y)$ which are sampled from a distribution $\mathcal{A}$. We have also computed the corresponding temperature field $T(x,y)$ through solving the PDE problem (2) with high-precision FEM method. Now, we hope to make a fast prediction of the temperature field $T(x,y)$ (therefore, a fast prediction of the effective conductivity) of a group of materials that are sampled from the same distribution.

The Goals

- Please select a suitable deep learning method for solving this task, and explain the reason for using it.
- Report your setups for the implementation, such as network structure, activation function, optimizer (with learning rate), epoch (with batch size), loss weights, and other tricks that are used for improvement.
- Compute the L^2 relative error (on testing dataset) at each training epoch and plot the `Error vs. epoch` curve (and report the final error). The L^2 relative error between

the prediction $\{T_{\text{pred}}^{(j)}\}_j^N$ and the truth $\{T_{\text{true}}^{(j)}\}_j^N$ is defined as follows:

$$\text{error} = \frac{1}{N}\sum_j \sqrt{\frac{\sum_i \left|T_{\text{pred}}^{(j)}(x_i) - T_{\text{true}}^{(j)}(x_i)\right|^2}{\sum_i \left|T_{\text{true}}^{(j)}(x_i)\right|^2}}$$

where j indicates the j-th instance.

- Plot the material thermal conductivity, the corresponding predicted solution (and the ground truth reference), and the pointwise absolute error for the first instance in the testing Dataset using separate figures with `matplotlib`.

Dataset

The observation and the ground truth references are given in the dataset `EffectiveConductivity.h5`:

- `k_train`: size(1000, 36, 36), the collected materials (a 36×36 matrix represents each material)
- `T_train`: size(1000, 36, 36), the collected temperature field (computed on a 36×36 regular mesh)
- `k_test`: the materials that we are going to predict their corresponding temperature field (Should not be used for training)
- `T_test`: the ground truth reference of the temperature field (Should not be used for training)
- `X`: size(36, 36), the x-coordinate of locations where the temperature field computed
- `Y`: size(36, 36), the y-coordinate of locations where the temperature field computed
- Link to dataset: `https://www.kaggle.com/datasets/yhzang32/dno4pdes`

Problem C: Prediction of Traffic Flow Based on Burgers' Equation Model

Burgers' equation is a mathematical model used in various fields, including traffic flow, to represent the behavior of nonlinear systems with both convective and diffusive effects. In traffic flow, it captures the interactions between vehicles, such as the tendency for traffic to slow down when density increases (convection) and the tendency for traffic to spread out due to individual driver behavior (diffusion).

The Burgers' Equation for modeling traffic flow is given as follows:

$$u_t + uu_x = \nu u_{xx}, \quad x \in (-1, 1),\ t \in (0, 1] \tag{4}$$

where:

- $u(x, t)$: car velocity (m/s),
- ν: a diffusion coefficient (reflecting how drivers respond to nearby traffic; higher ν = more cautious drivers),
- x: position along the road,
- t: time.

We set $\nu = 0.1$ and consider the Dirichlet boundary condition, i.e.:

$$u(x = -1, t) = u(x = 1, t) = 0, \quad t \in (0, 1]$$

With this PDE model, once the initial velocity field $u(x, t = 0) = a(x)$ is given, we can predict the velocity field $u(x, t)$ at any time $t > 0$ by solving Burgers' equation.

Task: Predicting the Velocity Field $u(x, t)$ Given the Initial Field $u(x, t = 0) = a(x)$

In this task, we have collected many initial fields $u(x, t = 0) = a(x)$ which are sampled from a distribution $\mathcal{A}$. We have also computed the corresponding velocity field $u(x, t)$ by solving Burgers' equation with a high-precision FDM method. Now, we hope to make a fast prediction of the velocity field $u(x, t)$ once a new initial condition $a(x)$ is given from the same distribution.

The Goals

- Please select a suitable deep learning method for solving this task, and explain the reason for using it.
- Report your setups for the implementation, such as network structure, activation function, optimizer (with learning rate), epoch (with batch size), loss weights, and other tricks that are used for improvement.

- Compute the L^2 relative error (on testing dataset) at each training epoch and plot the `Error vs. epoch` curve (and report the final error). The L^2 relative error between the prediction $\{u^{(j)}_{\text{pred}}\}^N_j$ and the truth $\{u^{(j)}_{\text{true}}\}^N_j$ is defined as follows:

$$\text{error} = \frac{1}{N}\sum_j \sqrt{\frac{\sum_i |u^{(j)}_{\text{pred}}(x_i, t_i) - u^{(j)}_{\text{true}}(x_i, t_i)|^2}{\sum_i |u^{(j)}_{\text{true}}(x_i, t_i)|^2}}$$

  where j indicates the j-th instance.

- Plot the initial field, the corresponding predicted solution (and the ground truth reference), and the pointwise absolute error for the first instance in the testing Dataset using separate figures with `matplotlib`.

Dataset

The observation and the ground truth references are given in the dataset `TrafficFlow.h5`:

- `a_train`: size(1000, 128), the collected $N = 1000$ initial fields (on 128 sensors)
- `u_train`: size(1000, 100, 128), the collected $N = 1000$ velocity fields (on 100×128 temporal-spatial regular meshgrids)
- `a_test`: size(200, 128), the initial fields that we are going to predict their corresponding velocity field (Should not be used for training)
- `u_test`: size(200, 100, 128), the ground truth reference of the velocity field (Should not be used for training)
- `x_mesh`: size(128, 1), the spatial-coordinate of locations where the velocity field is computed
- `t_mesh`: size(100, 1), the temporal-coordinate of locations where the velocity field is computed
- Link to dataset: `https://www.kaggle.com/datasets/yhzang32/dno4pdes`

About the Project Report

You are asked to report your experiments. You are given a downloadable LaTeX template[1], which you can edit using **TUMShareLatex**. The report should contain the following sections:

1. **Title page** including name and student ID number
2. **Method** (max. three pages)
   - Establishing consistent symbols
   - Mathematical formulation of both network structures
   - Introduce the PDE and the loss
   - Theoretical advantages and disadvantages of both structures
   - Establish metrics for comparisons in the results part
   - Cite the relevant works
3. **Results**
   - Problem setup, used optimization methods (and parameters), and used hardware
   - Exact architecture of used neural networks (layers, neurons, activation functions, etc.)
   - Presentation of the results of the different numerical experiments
   - Interpretation of the results
4. **Conclusion**
5. **References**: list of all sources cited
6. **Appendices** (if needed): contains lengthy materials, procedures, tables, or figures

Important Notes

- You must provide a **link to the original code** you wrote for solving the PDE problems. This can be hosted on platforms such as GitHub, Google Colab, or Kaggle Notebooks. If possible, please include training logs or experiment outputs to support reproducibility.
- You are **not allowed to use pre-built implementations of deep learning methods** (e.g., PINNs, FNOs, etc.) from open-source packages or libraries. All key components must be implemented by you unless otherwise permitted in class.
- The use of **ChatGPT, Copilot, or any other AI coding assistants is strictly prohibited**. All code and analysis must reflect your own independent work.

[1]Please do not edit the original template.

About the Presentation

You are required to give a **10–15 minute** presentation summarizing your work on solving PDE problems using deep learning methods covered in this course. Your presentation should effectively communicate:

- Your approach to each problem,
- The deep learning methods used,
- Strategies for improving performance,
- Key results and conclusions.

This presentation is your chance to clearly show what you did, how you reasoned, and what you learned.

Presentation Requirements

- Maximum time: 15 minutes
- Use slides (PDF, PowerPoint, or Google Slides)
- Keep slides clear and uncluttered
- Include figures, equations, and tables where relevant
- Practice to ensure you stay within the time limit

Example Structure

1. Problem Setup and Method Selection (1–2 slides)

- Clearly define the PDE problem and the method you think is suitable for solving it (and why)

2. Model Architecture & Training Setup (1–2 slides)

- Describe your network design (input, output, hidden layers, activation functions)
- The loss definition you use
- List key training parameters: optimizer, learning rate, batch size, loss components
- Mention any tricks or design choices that helped

3. **Results & Evaluation (1–3 slides)**
   - Show visual results: plots, contours, loss curves, residuals
   - Include error metrics (e.g., L^2 error) or comparisons to analytical solutions if available
   - Discuss how well the model worked
4. **Challenges & Improvements (1 slide)**
   - What difficulties did you face?
   - What strategies helped improve results (adaptive sampling, normalization, reweighting loss terms, etc.)?
   - What didn't work and why?
5. **Conclusion (1 slide)**
   - Summarize your key findings
   - Which method(s) worked best for which types of PDEs?
   - Any takeaways or lessons learned?