CSE 598-002 - Homework 1 (Fall 2025)

Due: September 25, 2025 at 10:00 AM

Logistics

Submission Guidelines

You may optionally work with a partner on this assignment, though you can work alone if you want.

The due date for this assignment is **September 25, 2025 at 10:00 AM**. The submission is tracked using Gradescope. Note that any change you make to the homework already submitted on Gradescope counts as a resubmission. If you make any changes to the assignment after the due date has passed, you will be assigned a late penalty based on the number of days that have passed. For example, if you edit an assignment on September 28 and it was due on September 25, you will be assigned a 30% penalty (10% per day). This is non-negotiable.

You must submit the following files to Gradescope (note that these and other files are given to you in `https://github.com/complex-ai-lab/cse598/`):

1. The completed `hw1_pinn.ipynb` file.
2. The completed `hw1_neural_ops.ipynb` file.

Important: To avoid issues with package dependencies, you should complete this assignment in **Google Colab**. Running the notebooks in Colab ensures consistency across environments.

Lastly, if you work with a partner, you must put both people's names on the Gradescope submission.

Late Day Tokens

To accommodate for coinciding deadlines you may have from other courses, or personal unforeseen events such as sickness, each person has 5 slip day tokens. Each token provides an automatic extension of 1 calendar day—no questions asked.

1. Each late day token covers the whole homework for extension of 1 calendar day.
2. Each homework has a Gradescope submission for the late day token.
3. Late days are rounded up to the nearest integer. For example, a submission that is 4 hours late will count as one day late.
4. Extreme circumstances, like medical emergencies, etc.: In such cases, additional, no-penalty extensions will be granted. Contact the faculty instructor with some written

documentation (like a doctor's note).

5. Emails requesting exceptions for other reasons will be ignored with no reply.
6. Homework turned in after three days will not be accepted.

Grading Policy

You have one week to submit a regrade request once the grades for your homework are out. We will provide solutions, along with the exact grading rubric used.

Collaboration

Collaborating with people is encouraged, but plagiarism is strictly prohibited. As mentioned in **Submission Guidelines**, you may optionally work with a partner on the coding sections, though you can work alone if you want.

Generative AI

Learning how to use AI functions such as ChatGPT, Claude, etc, is important for all of us. Used properly, ChatGPT can enhance our work; used improperly, it can border on plagiarism. You are allowed to use ChatGPT or similar only for conceptual questions, and **it is strictly forbidden for programming**.

1 Physics-informed neural networks [50 Points]

1.1 PINNs for Burgers' equation [20 Points]

Burgers' equation is a fundamental partial differential equation (PDE) in fluid mechanics, used to model various physical phenomena, such as gas dynamics, traffic flow, and turbulence. It can be derived from the Navier–Stokes equations for the velocity field by dropping the pressure gradient term. In one space dimension the equation can be expressed as:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2},$$

where $u = u(x, t)$ is the velocity of a fluid at position x and time t and ν is a viscosity of the fluid.

In this task, you'll solve a specific instance of the Burgers' equation with the domain $x \in [-1, 1]$ and $t \in [0, 1)$, and $\nu = \frac{0.01}{\pi}$. The initial and boundary conditions are provided as:

- Initial condition: $u(x, 0) = -\sin(8\pi x)$
- Boundary conditions: $u(-1, t) = 0, u(1, t) = 0$

Your task: Solve the forward problem for $u = u(x, t)$ for the domain $x \in [-1, 1]$, $t \in [0, 1)$.

We give you a Google Colab notebook with some starter code and blanks to fill. Follow the next steps to implement a PINN that solves the PDE. See `https://github.com/AI-Complexity-Lab/cse598/tree/main/hw1`.

(a) Implement a forward pass of the PINN. (10 Points)

(b) Implement boundary condition (BC) and PDE residual losses. (5 Points)

(c) Train neural network. Aim to achieve a test error of at least 0.3. (5 Points)

1.2 PINNs for Reaction-diffusion equation [10 Points]

Next, you will solve the reaction-diffusion system given by:

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0, \quad x \in \Omega,\ t \in (0, T],$$

with initial condition u(x, 0) = h(x), where ν is the diffusion coefficient and ρ is the reaction rate. For this task, we will use $\rho = 1.75$ and $\nu = 4.25$.

Your task: Adapt the code from Section 1.1 to implement a PINN that solves the reaction-diffusion equation. Aim to achieve a test error of at least 0.2.

1.3 Advanced improvements to PINNs [20 Points]

Select one of the following methods to apply to either the Burgers' equation or the reaction-diffusion equation. Your goal is to achieve at least a 10% improvement in test error compared to your implementation in Sections 1.1 or 1.2. Partial credit will be given for implementations with lower improvement.

Option 1: Sequence-to-sequence training

Implement sequence-to-sequence training [1], discussed in class.

Option 2: Curriculum regularization

Incorporate curriculum regularization into the training process [1], discussed in class.

2 Neural operators [50 Points]

2.1 Fourier Neural Operator for Darcy flow [30 Points]

Darcy flow describes the movement of fluid through a porous medium, governed by Darcy's law, which relates the fluid velocity to the pressure gradient and the permeability of the material. This flow is common in fields like groundwater hydrology, oil recovery, and soil science. The basic form of Darcy's law is:

$$\mathbf{u} = \frac{K}{\mu} \nabla p,$$

where:

- $\mathbf{u}$ is the velocity vector describing how fast the fluid is moving through the porous medium,
- K is the permeability of the porous medium,
- μ is the dynamic velocity of the fluid, and

- $\nabla p = (\frac{\partial p}{\partial x_1}, \frac{\partial p}{\partial x_2}, ..., \frac{\partial p}{\partial x_d})$ is the pressure gradient where we are operating in d dimensions.

Here you will implement the Fourier Neural Operator (FNO) [2], a type of neural network architecture designed to learn mappings between functions, particularly in the context of solving partial differential equations (PDEs). It leverages Fourier transforms to efficiently capture complex patterns and long-range dependencies in data, making it especially useful for problems involving continuous spatial or temporal processes.

You are given the skeleton code, where you're given a number of TODO code snippets with comments directing you. See `https://github.com/AI-Complexity-Lab/cse598/tree/main/hw1`. Overall, you will:

(a) Reflection on the Darcy dataset and the meaning of the inputs and outputs (5 points).

(b) Implement the forward pass for the Fourier block class. (10 Points)

(c) Implement the constructor and forward pass for the FNO. (10 Points)

(d) Tune the parameters to make the '16_l2' metric (L2 loss in 16x16 images) on test data to be less than 0.5. (5 Points)

For training, you will simply run the code blocks in order to see how the FNO module is trained on the given small Darcy Flow dataset.

2.2 DeepONets for Darcy flow [20 Points]

In this section, you will implement the DeepONet neural operator [3], another advanced neural network model used for solving PDEs through deep learning. The trunk and branch method in DeepONet allows for the separation of input functions and locations, enabling efficient and flexible learning of nonlinear operators by independently processing the function space and input space.

You are given the skeleton code for the DeepONet class, where you're given a number of TODO code snippets with comments directing you. Overall, you will:

(a) Implement the constructor of the DeepONet class (8 Points)

(b) Implement the branch and trunk networks. (8 Points)

(c) Implement the forward pass function. (4 Points)

Following the implementation, you will simply run the code blocks in order to see how the DeepONet module is trained on the same small Darcy Flow dataset that was given for training with the FNO module.

References

[1] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

[2] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[3] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.