Project 2: Neural Operators in practice

ETH zürich

AI in the Sciences and Engineering
Due date: May 20th, 2024

Training and testing data can be found on the Moodle page: `https://moodle-app2.let.ethz.ch/course/view.php?id=22389`

IMPORTANT INFORMATION

To get ECTS credits for the course you need to submit and obtain a passing grade on **each** of the **three** projects, which are to be completed **individually**. Submitting these projects is the only form of assessment for the course - there are no group projects this year.

This document describes the **second** project. The due date for this project is **20 May 2024**.

The remaining project is scheduled to be released on 19 on the 24th of May (but these dates may change slightly depending on lecture progress). This project will also have its due date four weeks after its release.

This project consists of three tasks on different topics. For each task, you will be asked to train a learning model and provide predictions on a testing set. **The final submission consists of the predictions files, the code and a report of maximum 2000 characters per task** where you should succinctly describe the procedure followed in each task. The submissions have to be collected in a zip folder named as *yourfirstname_yoursecondname_yourleginumber.zip*.

Bonus system

- Each project will have its own **bonus, optional** question – you can get full marks on the project (6/6) without touching it.
- Each such question will be scored from 0 to 100 – due to this question being open-ended, we will decide on what a 'reasonable attempt' is based on the submissions of **all** students.
- Getting at least a 'reasonable attempt' on **all** bonus questions across **all** projects will earn you a final grade bonus of 0.5. Note the final grade is still capped at 6.

Task 1

The main objective of the Task 1 is to apply **neural operators** the a task related to the preliminary design of a *thermal energy storage*.

The device is used in solar power plants to store thermal energy during the *charging phase* and release it for production of electricity during the *discharging phase*. The thermal energy is stored due to the interaction of a fluid and a solid phase. During the charging state the fluid is injected at high temperature from one end of the storage and heats the solid up. In contrast, during the discharging phase the reverse process occurs: cold fluid flows from the opposite end and absorbs heat from the solid. Between charging and discharging *idle phases* take place, where no fluid enters the thermal storage.

Therefore, at any instant of time the thermal storage can be in one of the following states:

1. Charging;
2. Idle between charging and discharging;
3. Discharging;
4. Idle between discharging and charging;

Together the four states establish a *cycle* and the same process is repeated for several cycles until the thermal storage reaches a periodic or stationary regime.

Mathematical Model

The thermal storage is modeled by a cylinder with length L and diameter D and it is assumed that temperature variation occurs only along the axis of the cylinder (see Figure 1 for a schematic representation of the thermal storage).

The temperature evolution of the solid and fluid phases, T_s and T_f, is described by a system of two linear *reaction-convection-diffusion* equations:

$$\begin{aligned} \varepsilon\rho_f C_f \frac{\partial T_f}{\partial t} + \varepsilon\rho_f C_f u_f(t)\frac{\partial T_f}{\partial x} &= \lambda_f \frac{\partial^2 T_f}{\partial x^2} - h_v(T_f - T_s) \quad x \in [0, L],\ t \in [0, T], \\ (1-\varepsilon)\rho_s C_s \frac{\partial T_s}{\partial t} &= \lambda_s \frac{\partial^2 T_s}{\partial x^2} + h_v(T_f - T_s) \quad x \in [0, L],\ t \in [0, T], \end{aligned} \tag{1}$$

with ρ being the density of the phases, C the specific heat, λ the diffusivity, ε the solid porosity, u_f the fluid velocity entering the thermal storage and h_v the heat exchange coefficient between solid and fluid.

The fluid velocity is assumed to be uniform along the cylinder and varying only in time: $u_f = u$ during charging, $u = 0$ during idle and $u_f = -u$ during discharging, with u being a positive constant.

The system of equations has to be augmented with suitable initial and boundary conditions:

$$T_f(x, t = 0) = T_s(x, t = 0) = T_0, \ x \in [0, L] \tag{2}$$

$$\frac{\partial T_s(x,t)}{\partial x}\Big|_{x=0} = \frac{\partial T_s(x,t)}{\partial x}\Big|_{x=L} = 0, \ t \in [0, T] \tag{3}$$

The boundary conditions for the fluid instead will be different according to the current state of the thermal storage:

- **Charging State:**

$$T_f(0, t) = T_{hot}, \qquad \frac{\partial T_f(x,t)}{\partial x}\Big|_{x=L} = 0, \qquad t \in [0, T] \tag{4}$$

- **Discharging State:**

$$\frac{\partial T_f(x,t)}{\partial x}\Big|_{x=0} = 0, \qquad T_f(L, t) = T_{cold}, \qquad t \in [0, T] \tag{5}$$

- **Idle Phase:**

$$\frac{\partial T_f(x,t)}{\partial x}\Big|_{x=0} = 0, \qquad \frac{\partial T_f(x,t)}{\partial x}\Big|_{x=L} = 0, \qquad t \in [0, T] \tag{6}$$

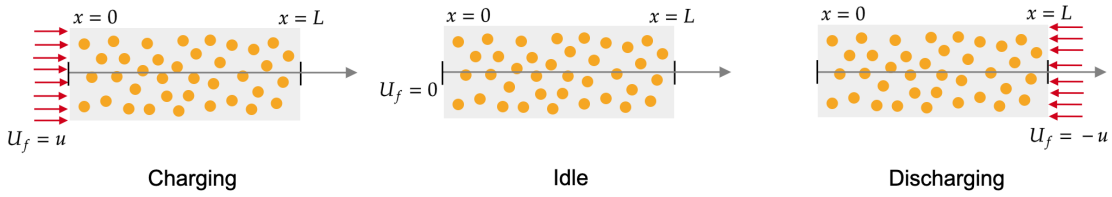

Figure 1: Schematic representation of the thermal storage

Task 1: Time Series Forecasting with Neural Operators

Let us assume that **noiseless** measurements of the fluid and solid temperature $T^0_{f,i}, T^0_{s,i}$, $i = 0, ..., 210$, are taken at the top end of the storage $x = 0$ during the entire process.

In this task we are interested in **future** (or **out of samples**) predictions of the fluid and solid temperature. In other words, the training data consists of time measurements t_i registered in a frame $[0, T]$, $T = 520000s$ and we would like to forecast the fluid and solid temperature in the frame $[T, T_{end}]$, $T_{end} = 602168s$.

Train a learning model **based on Neural Operators** with the data given in **Task3/TrainingData.txt** and make predictions on the test set **Task3/TestingData.txt**. The first column of the training data contains the time frames t_i, $i = 0, ..., 210$, the second and the third column the fluid and solid temperatures $T^0_{f,i}$, $T^0_{s,i}$, $i = 0, ..., 210$, respectively. See Figure 2 for a schematic representation of the training data.

Afterwards, save your predictions in the file *yourfirstname_yoursecondname_yourleginumber/Task1.txt*. **The format of the file has to be the same as the file Task1/SubExample.txt.**

Hint: observe that $t_{i+1} - t_i = \Delta t = const$, for all, $i = 0, ..., 209$.

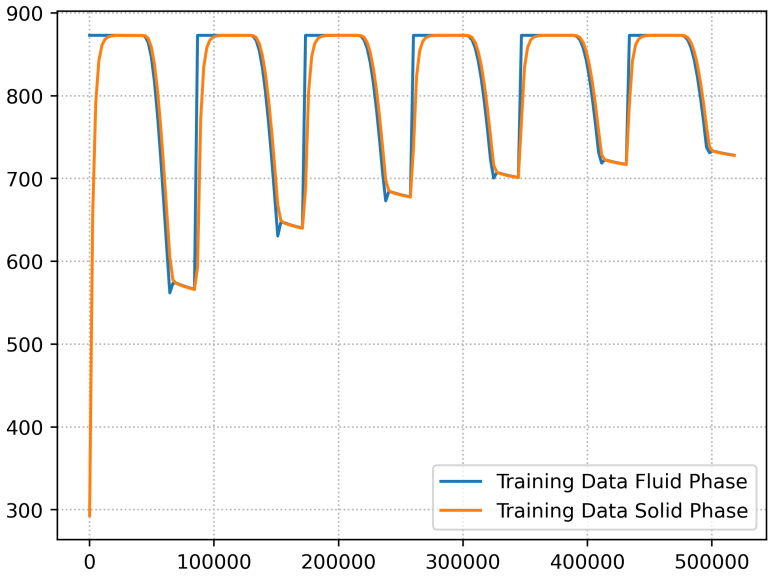

Figure 2: Task 1 training data

Task 2: Modeling Water Flow on the Sphere with FNO

The *Spherical Fourier Neural Operator* (SFNO) presents a promising approach for solving geophysical fluid dynamics problems, particularly the Shallow Water Equations (SWE), which are fundamental for modeling atmospheric and oceanic dynamics. SFNO leverages the spherical geometry of Earth's surface, allowing for more accurate and efficient predictions compared to traditional methods.

Specifications

1. **SFNO Implementation**: You will develop an implementation of SFNO tailored specifically for solving the Shallow Water Equations.
2. **Dataset Creation and Loading**: To facilitate experimentation, you can use pre-existing code for creating and loading the dataset. The `load_spherical_swe` function from `https://neuraloperator.github.io/neuraloperator/dev/auto_examples/plot_SFNO_swe.html` can be used for this purpose.
3. **Restricted Package Usage**: While students can utilize the dataset utilities, they are prohibited from using the SFNO implementation available in the provided package.
4. **Spherical Convolutional Layer**: SFNO requires a specialized convolutional layer that operates on the spherical harmonics decomposition of the data. You will implement this layer using the `RealSHT` and `InverseRealSHT` modules from `https://github.com/NVIDIA/torch-harmonics`.
5. **Resolution Experimentation**: Experimentation will involve training SFNO on datasets at different resolutions, starting with $(32, 64)$. The results obtained will be compared against those of a baseline neural operator (e.g. FNO) trained on the same dataset but without considering the spherical geometry.
6. **Baseline Implementation**: For the baseline neural operator, you are encouraged to explore existing code available online. However, you must perform the training yourselves using the provided dataset.

Deliverables

1. **SFNO Implementation**: A fully functional implementation of SFNO tailored for solving the Shallow Water Equations.
2. **Experimental Results**: Detailed experimentation results comparing the performance of SFNO with the baseline neural operator at different resolutions.
3. **Documentation**: Latex typeset document, detailing the implementation process, experimental setup, and results analysis – 3 page max.

Resources

- https://neuraloperator.github.io/neuraloperator/dev/auto_examples/plot_SFNO_swe.html
- https://github.com/NVIDIA/torch-harmonics
- https://developer.nvidia.com/blog/modeling-earths-atmosphere-with-spherical-fourier-neural-ope

Task 3 - Universal Approximation for CNO (Optional):

Attention: We do not expect fully-fleshed out maths here, even though that's welcome if you feel like you can do it – we're only looking for you to be as quantitative as possible in your line of thought.

Attention: You can write up to 2 **pages** for this task.

Problem Setup:

Consider the following abstract PDE in the 2d torus $D = \mathbb{T}^2$,

$$\mathcal{L}(u) = 0, \quad \mathcal{B}(u) = 0, \tag{7}$$

with $\mathcal{L}$ being a differential operator and $\mathcal{B}$ a boundary operator. We assume that the differential operator $\mathcal{L}$ only depends on the coordinate x through a *coefficient* function $a \in H^r(D)$. The corresponding *solution* operator is denoted by $\mathcal{G}^\dagger : \mathcal{X}^* \subset H^r(D) \to H^r(D) : a \mapsto u$, with u being the solution of the PDE (7). Note that $H^r(D)$ is *Sobolev space* of order r.

The goal of this task is to outline the proof of **universality theorem** for CNO. The theorem states that for any $\epsilon > 0$ and any **sufficiently regular** operator $\mathcal{G}^\dagger$, as defined above, there exists a CNO $\mathcal{G}$ such that for every $a \in \mathcal{X}^*$ with $\|a\|_{H^r(D)} \leq B$ it holds,

$$\|\mathcal{G}^\dagger(a) - \mathcal{G}(a)\|_{L^p(D)} < \epsilon. \tag{8}$$

Questions

- Explain briefly how the CNO $\mathcal{G}$ is defined. (Hint: CNO is a compositional mapping between function spaces)
- Explain briefly what the universality theorem means in practice.
- Note that we stated that universality theorem holds for sufficiently regular solution operators $\mathcal{G}^\dagger$. What additional regularization constraints would you impose on $\mathcal{G}^\dagger$, so that the universality theorem holds?
- Make a rough sketch of the universality theorem proof. Please explain the steps. The sketch does not need to be rigorous.

Resources

- `https://arxiv.org/pdf/2302.01178.pdf`
- `https://arxiv.org/abs/2107.07562`
- `https://arxiv.org/abs/2108.08481`