Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

AI in the Sciences and Engineering **Spring Semester 2024**

Student: Carla Judith López Zurita

Project 2

Due date: Monday 20 May 2024, 23:59

The main objective of the project is to apply the concepts learned in class related to Neural Operators. The first task consists in making future (or out of samples) predictions of the fluid and solid temperature of a thermal storage system. The second task is to model the water flow on the sphere using a Spherical Fourier Neural Operator (SFNO) and compare the results with the ones obtained by a baseline neural operator. The project also includes an optional task to test the universal approximation property of the Convolutional Neural Operator (CNO). My attempt at solving each of the mentioned tasks will be described in detail below.

1. Time Series Forecasting with Neural Operators

The first task is based on the mathematical model of a thermal storage system described by the following PDEs:

$$\epsilon\rho_f C_f \frac{\partial T_f}{\partial t} + \epsilon\rho_f C_f u_f(t)\frac{\partial T_f}{\partial x} = \lambda_f \frac{\partial^2 T_f}{\partial x^2} - h_v(T_f - T_s) \quad \text{for} \quad x \in [0, L], \quad t \in [0, T], \tag{1}$$

$$(1-\epsilon)\rho_s C_s \frac{\partial T_s}{\partial t} = \lambda_s \frac{\partial^2 T_s}{\partial x^2} + h_v(T_f - T_s) \quad \text{for} \quad x \in [0, L], \quad t \in [0, T], \tag{2}$$

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 system is also defined with appropriate boundary conditions defined in the project description. The goal is to predict the fluid and solid temperature at future time steps using a Neural Operator. One of the main challenges of this task was to arrange the data in the correct format to feed it to the network, namely the data had to be arranged in a series of windows. I used a window generator to create the windows with a length of 34 for both inputs and outputs, which was chosem because it is the size of the final prediction we have to make. The outputs were taken from the provided measurements, but corresponding to the next 34 time steps. Fig. 1 shows an example of a window with prediction.

The data was normalized using the MinMaxScaler from the sklearn library. The data was split into training and validation sets with a ratio of 80% and 20% respectively. I chose to use a Fourier Neural Operator (FNO) to solve this task. I used a single network to predict both the fluid (T_f) and solid temperature (T_s), so that it had three channels as input (T_f, T_s and time) and two channels as output (T_f and T_s). The network consisted of 3 layers with 3 Fourier features each. It was trained using the Adam optimizer with an initial learning rate of 10^{-3} and a batch size of 10. The learning rate was adjusted using `StepLR` with step size of 50 and gamma 0.5. The loss function used was the L2 norm. The network was trained for 1,000 epochs. The loss function of the model during training is shown in Fig. 2. Traing takes around two minutes on a single core, in a M1 MacBook Pro.

The model was evaluated on the test set and the predictions were compared with the ground truth. Fig. 3 shows the predictions of the model taken from various windows in the test set. The

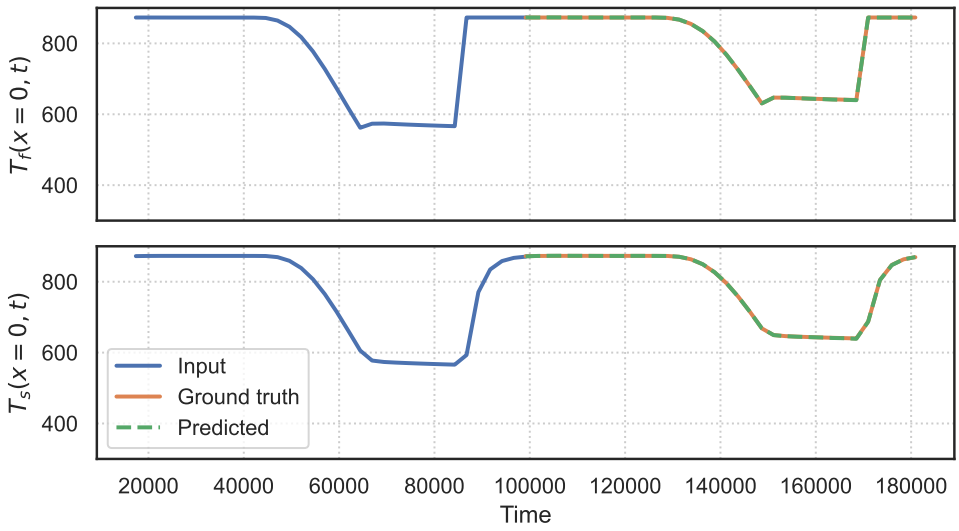

Figure 1: Example of a window with prediction.

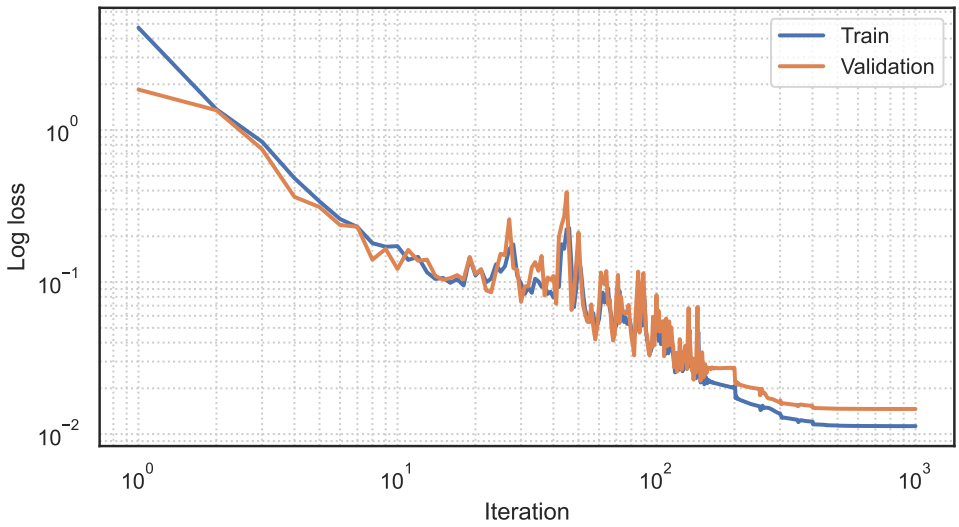

Figure 2: Loss function of the FNO model during training.

dashed lines represent the predicted values, while the solid lines depict the ground truth. The model seems to be able to capture the general trend of the data, and predict very close values.

Finally, we show the future predictions of the fluid and solid temperature in Fig. 4.

2. Modeling Water Flow on the Sphere with FNO

The next task consists in modeling the water flow on the sphere using Neural Operators. The idea of the task is to use two different architectures to predict the velocity field of the water on the sphere for various resolutions while training on a lower resolution. In this manner, we can see how the model generalizes to out of sample data. We will implement an Spherical Fourier Neural Operator (SFNO) and compare the results with those of a baseline neural operator, in this case, a Fourier Neural Operator (FNO). The main difference between the two models is that the SFNO uses spherical harmonics to represent the data, while the FNO uses Fourier features. This was implemented by building a custom layer that computes the spherical harmonics using function from Bonev et al's library [1]. The architectures were based on the paper and public code repository

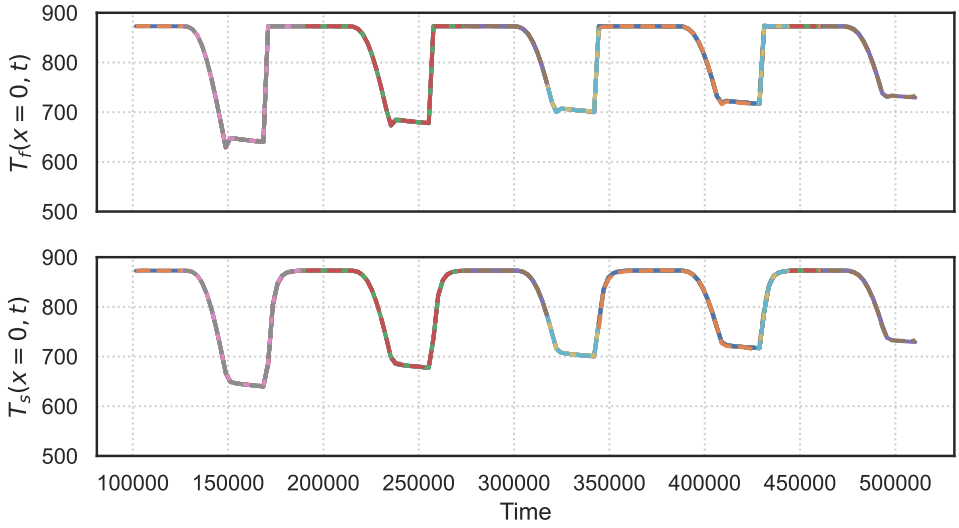

Figure 3: Predictions of the FNO model taken from various windows in the test set. Dashed lines represent the predicted values, solid lines depict ground truth.

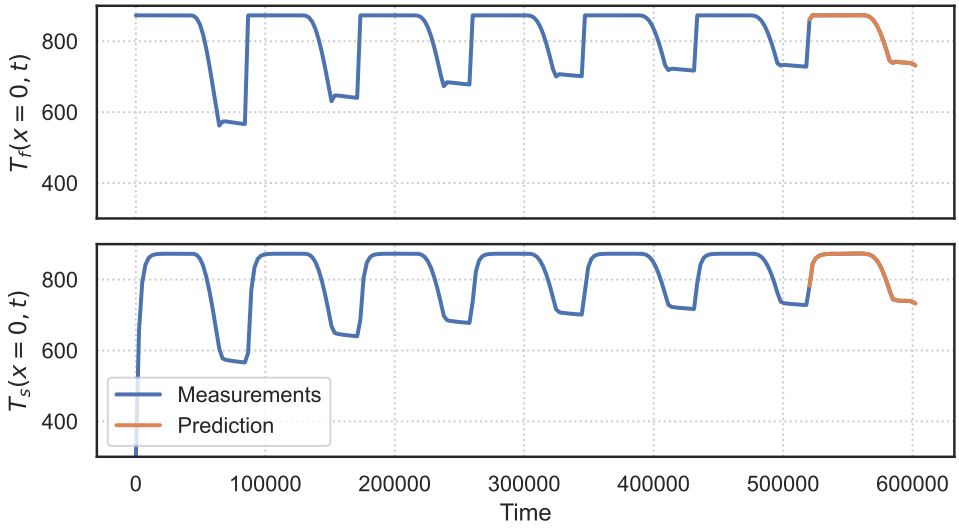

Figure 4: Future predictions of the fluid and solid temperature.

by Zongyi Li et al [3].

The data was retrieved from the Kossaifi et al's library [2]; it contains the velocity field of water on the sphere, as well as the x and y coordinates of the sphere. The data was split using 200 samples for training with (32, 64) resolution and 50 samples for validation for (32, 64) and (64, 128) resolutions. It was normalized using the StandardScaler from the sklearn library.

The FNO model was trained using Adam optimizer with an initial learning rate of $8e^{-4}$ and a batch size of 4 for training and 10 for testing. The learning rate was adjusted using `CosineAnnealingLR` with a $T_{\max}$ of 30. The loss function used was the L2 norm. The network was trained for 20 epochs, which took around 20 minutes on a single core, in a M1 MacBook Pro. The loss function of the model using FNO during training is shown in Fig. 5. The SFNO model was trained using the same parameters as the FNO model, and took appropriately the same time to train. The loss function of the model using SFNO during training is shown in Fig. 6. We can see that both models converge in a similar manner, but the SFNO model seems to reach a lower loss value in the training set, as shown in Table 1. Nevertheless, the FNO model seems to generalize better to the validation set.

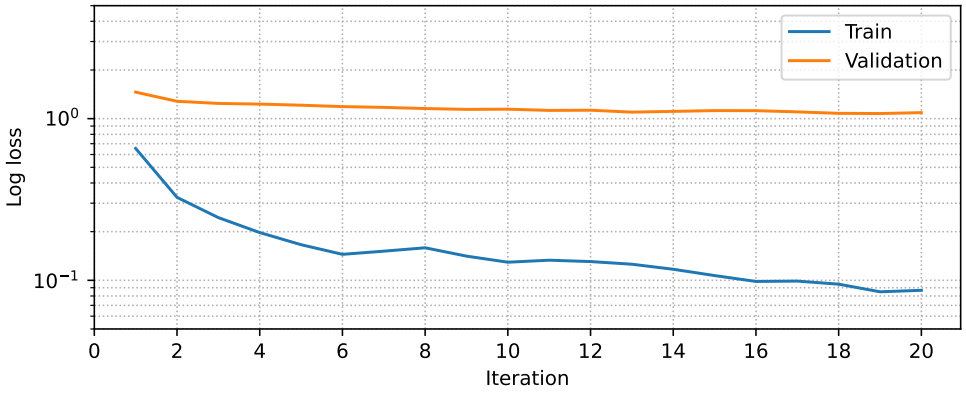

Figure 5: Loss function of the FNO model during training.

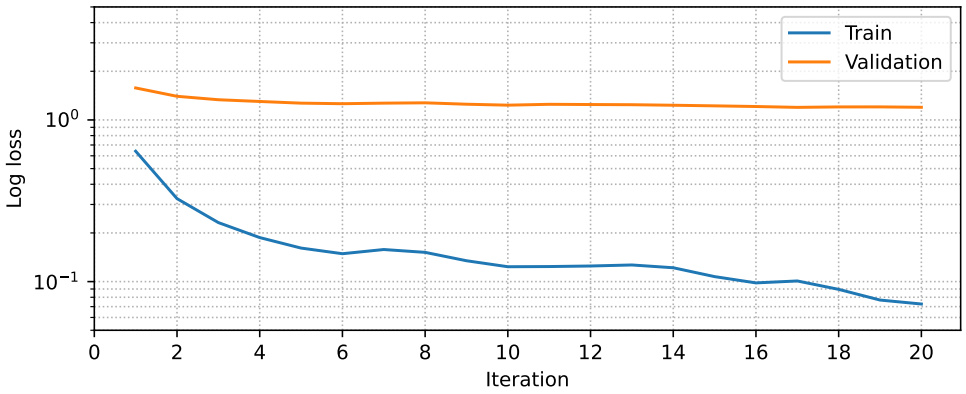

Figure 6: Loss function of the SFNO model during training.

Model	**Train Norm**	**Test Norm**
FNO	0.086722	1.089633
SFNO	0.0726737	1.197494

Table 1: Value of the relative L2 norm for the FNO and SFNO model after 20 epochs.

It is important to mention that the loss values are directly comparable between the two models, since the data and training was handled in the same way. It is also reasonable to see why the values of the loss function between training and validation sets are different, as the validation set at higher resolution is unseen data for the model when training. This is bound to give a higher loss value. Training for more epochs did not seem to improve the results of both models, even as the loss function of the training set kept decreasing, the loss function of the validation set started increasing, which is a sign of overfitting.

Finally, we show an example of the velocity field of water on the sphere using the FNO model in Fig. 7 and the SFNO model in Fig. 8. Qualitatively, both models seem to capture the general trend of the data correctly, specially in the resolution they were trained on. Both models seem to create some aritifacts in the velocity field, and also seem to retain some more details compared to the ground truth. It is hard to say which model is better without further analysis. More examples of the velocity field of water on the sphere using the FNO model and the SFNO model can be found in the `results` folder attached to this document.

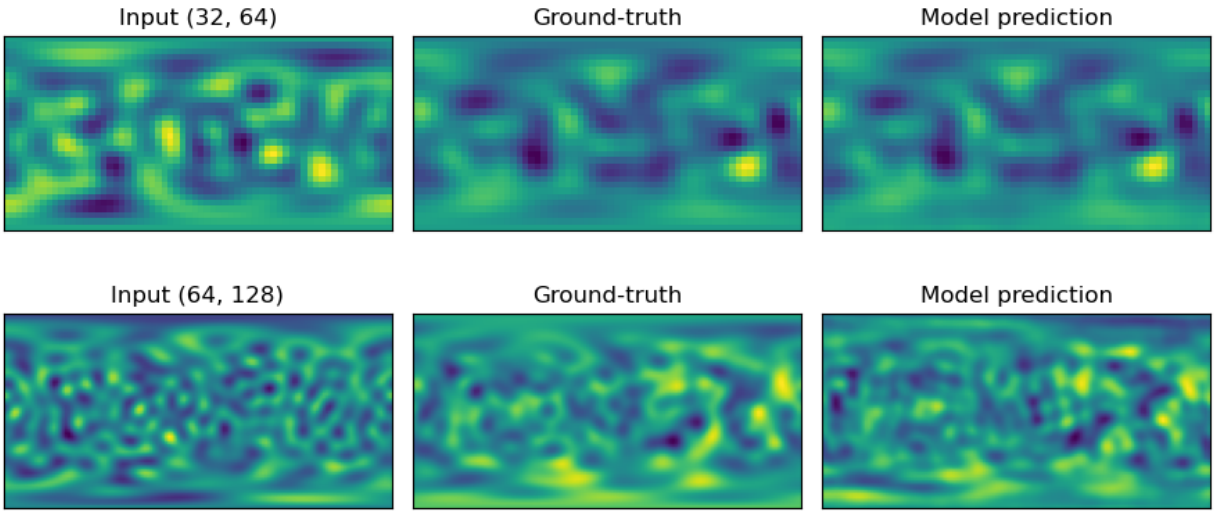

Figure 7: Example of the velocity field of water on the sphere using a FNO.

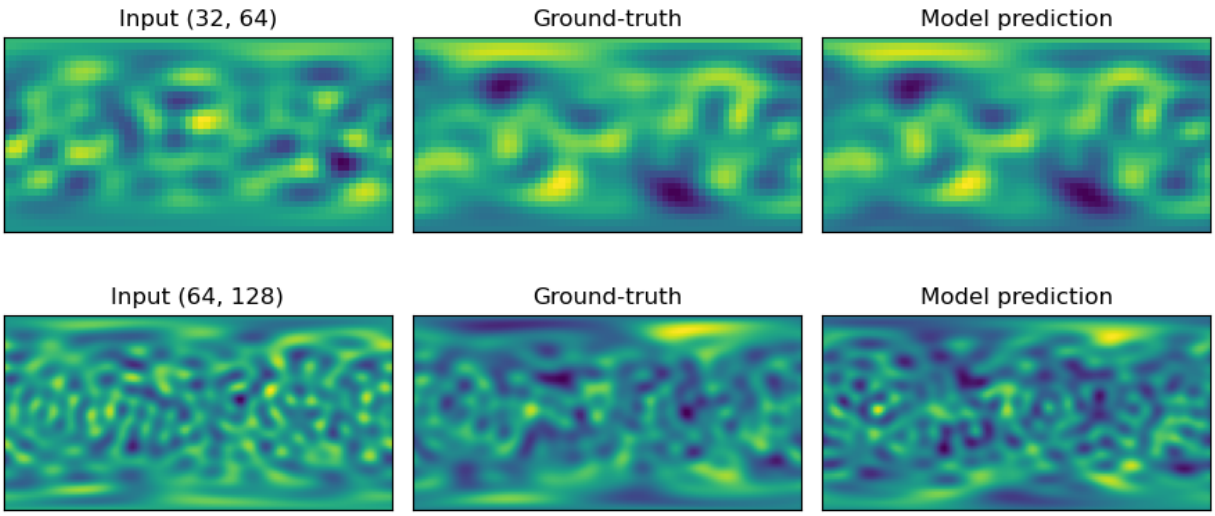

Figure 8: Example of the velocity field of water on the sphere using a SFNO.

3. Universal Approximation for CNO

Consider the following abstract PDE in the 2d torus $D = T^2$, $\mathcal{L}(u) = 0$, $\mathcal{B}(u) = 0$, 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 $\mathcal{G}^\dagger : X^* \in H^r(D) \to H^r(D) : a \to u$, with u being the solution of the PDE. $H^r(D)$ is Sobolev space of order r.

The CNO $\mathcal{G}$ is a compositional mapping between function spaces. It is defined as $\mathcal{G} = \mathcal{N} \circ \mathcal{G}^\dagger$, where $\mathcal{N}$ is a neural network. It can have many layers and different activation functions.

The universality theorem for CNO states that for any $\epsilon > 0$ and any sufficiently regular operator $\mathcal{G}^\dagger$, there exists a CNO G such that for every $a \in X^*$ with $|a|H^r(D) \leq B$ it holds,

$$\|\mathcal{G}(a) - \mathcal{G}^\dagger(a)\|_{H^r(D)} \leq \epsilon. \tag{3}$$

In practice, this means that the CNO can approximate the solution operator $\mathcal{G}^\dagger$ with an arbitrary precision.

[4] Apart from the condition of sufficiently regular solution operators $\mathcal{G}^\dagger$, we can impose some additional regularization constraints on $\mathcal{G}^\dagger$, so that the universality theorem holds. For example, we can require that the solution operator $\mathcal{G}^\dagger$ is Lipschitz continuous with respect to the L^2 norm. This is a common regularization constraint in practice.

The following sketch aims to illustrate the universality theorem for CNO.

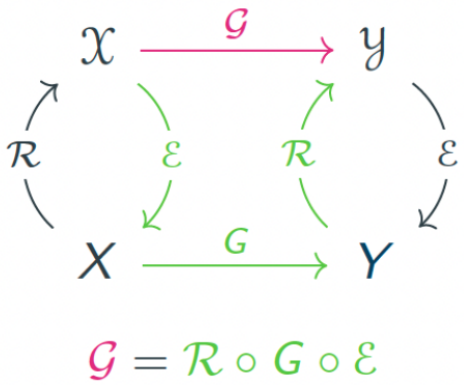

Figure 9: Sketch illustrating the universality theorem for CNO.

References

[1] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere, 2023.

[2] Jean Kossaifi, Nikola Kovachki, Zongyi Li, and Anima Anandkumar. Training a sfno on the spherical shallow water equations. `https://neuraloperator.github.io/neuraloperator/dev/auto_examples/plot_SFNO_swe.html`, 2024. Accessed: 2024-05-20.

[3] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.

[4] Bogdan Raonić, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes, 2023.