AISE assignment 2

Lorenzo Tarricone

May 2024

1 Time series forecasting with Neural Operator

For this section, I've predicted to predict the out-of-sample temperatures with FNO. I decided to train two different FNOs for the two different temperatures

The general idea behind my approach was one of dividing the training set into chunks of equal size and letting the network learn how to predict a chunk starting from the previous one. This was done in two different ways/approaches, according to two different intuitions (see below).

To speed up the training of the model the data were first scaled into the interval $[0, 1]$ and then scaled back to the original scale to generate the final predictions

1.1 Approach 1: Non-overlapping window

The intuition behind this first approach was to notice that the size of the test set is almost a multiple of the one of the training set, for a ratio of train:test = 6:1.

Having the ordered chunks $\{1, 2, 3, 4, 5, 6\}$ in the training set at each epoch we predict 2 with 1, 3 with 2, ... , 6 with 5. In order to generate predictions one would then just generate the next chunk from 6.

And produced a final (on the training set) average MSE of $1.322 \cdot 10^{-5}$ and relative L2 of 0.451% for T_f and a final (on the training set) average MSE of $2.404 \cdot 10^{-5}$ and relative L2 of 0.538% for T_s 1

Plotting the results on the test set (after rescaling to the original dimension) yields qualitatively good results 2

1.2 Approach 2: Overlapping windows

The rationale behind this approach was the one of trying to create some constant overlap between windows to mimic a sliding kernel of a CNN. This would allow to "fit more frames" because for example with an overlap of 50% you would be now able to have double the chunks with respect to the previous setting

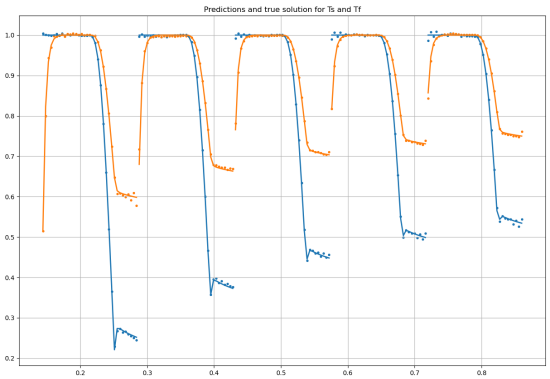


Figure 1: Ground truth value for T_f and T_s (solid line) and predicted values at the end of the training (dots). [The values plotted are normalized]

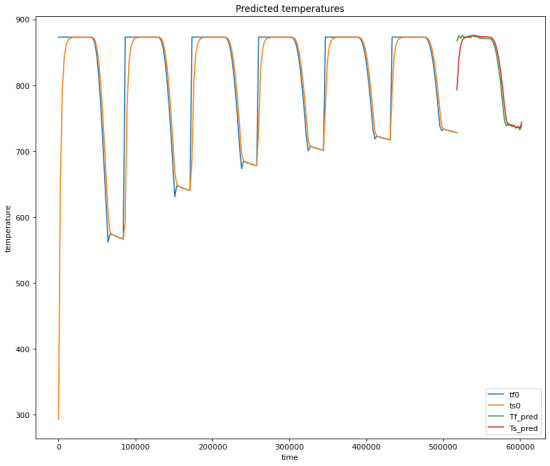


Figure 2: Ground truth value in the training data for T_f and T_s (blue and orange) and predicted values on the test set (green and red). [The values plotted are not normalized]

And produced a final (on the training set) average MSE of $5.627 \cdot 10^{-5}$ and relative L2 of 0.886% for T_f and a final (on the training set) average MSE of $1.000 \cdot 10^{-4}$ and relative L2 of 1.127% for T_s 1

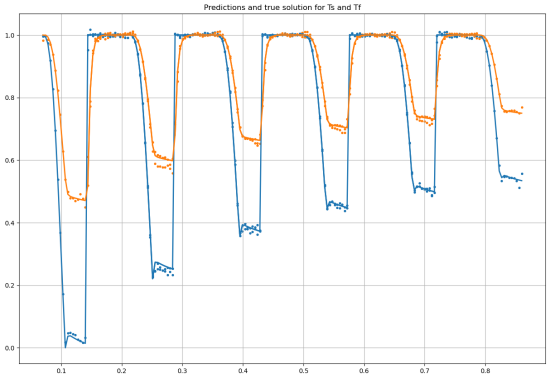


Figure 3: Ground truth value for T_f and T_s (solid line) and predicted values at the end of the training (dots). [The values plotted are normalized]

Plotting the results on the test set (after rescaling to the original dimension) yields qualitatively worse results 4, as we can see that the predicted values have the correct shape but tend to be "flipped upside down". Maybe the reason for this is the fact that the overlap reduces the information that needs to be learned from one chunk to the following one and this penalizes the training. It might also be that a better set of hyperparameters would lead to correct results.

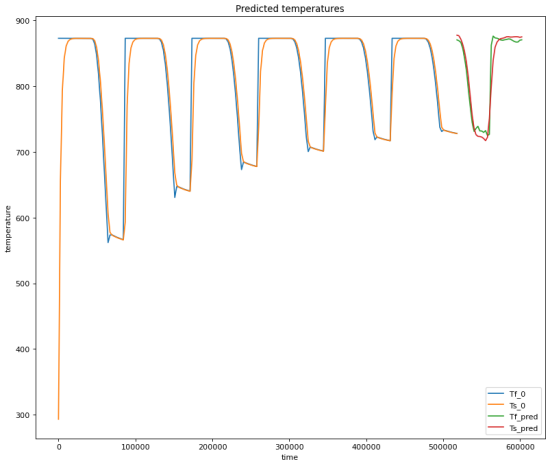


Figure 4: Ground truth value in the training data for T_f and T_s (blue and orange) and predicted values on the test set (green and red). [The values plotted are not normalized]

For the reasons mentioned just above I have decided to use the first approach to produce the prediction on the given test set

2 Modeling Water Flow on the Sphere with FNO

For this second task, I adapted the code for the 2dFNO provided in class.

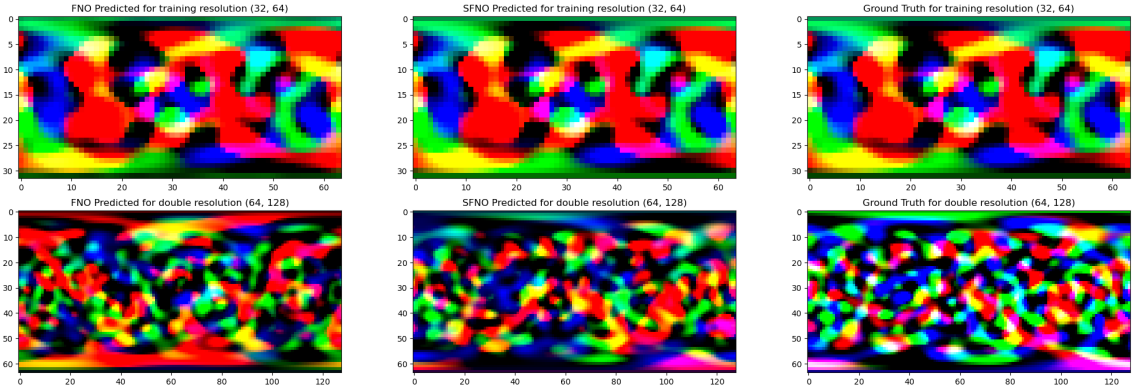
As suggested, I used the function `load_spherical_swe` to create the dataloader for the training and the two dataloaders for the test. Every time the dataloaders for the test were created one was generated with the same resolution as the training loader and the other with double resolution.

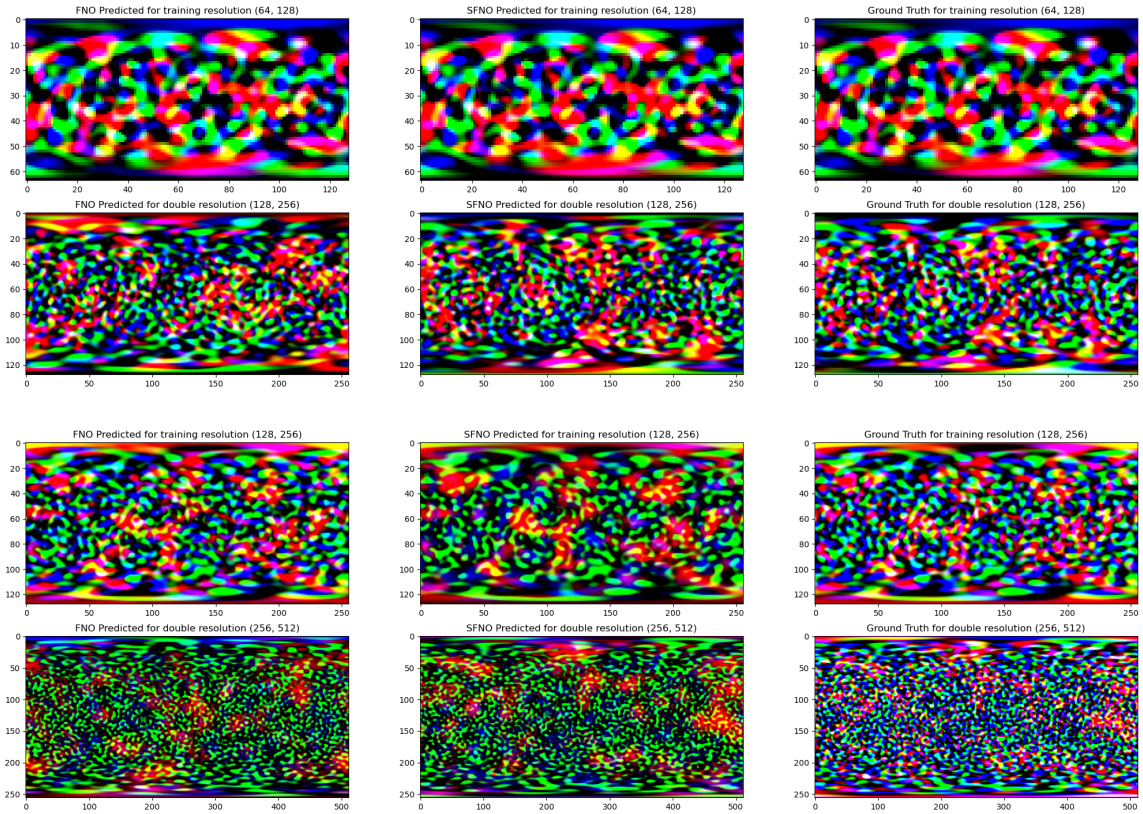
2.1 Architecture and preprocessing of the data

The class `SFNO2d` just changes (with respect to 2dFNO) just some permutation of the data during the forward pass (to coincide with the format of the previous implementation) and the use of `RealSHT` and `InverseRealSHT` instead of `torch.fft` and `torch.ifft`.
Moreover, the data were preprocessed to add the spatial information (as two channels) to the already present three channels.
As a baseline, I used the standard 2dFNO provided in the tutorial adapted to handle the format of the dataloaders.

2.2 Comparing results

I trained both networks at three different resolutions. In each case I have trained the model on a given resolution and then produced predictions for both the same resolution that was used for the training and double that resolution. All the plots are displayed below. In general, I have noticed that while the results on the training resolution are comparable between FNO and SFNO, SFNO generalizes better on the larger size, especially around the "poles" (most upper part and most lower part of the images)





3 Universal approximation for CNO

3.1 Definition of CNO

In class, we defined CNO as a Representation equivalent Neural Operator (ReNO) with a specific architecture (that in a way mimics a U-Net) that implements a compositional mapping between functional spaces. We can say that the CNO $\mathcal{G}$ is a map between band-limited functions (i.e. $\mathcal{G} : \mathcal{B}_w \to \mathcal{B}_w$ and $\mathcal{B}_w(D) := \{f \in L^2(D) : supp\hat{f} \subseteq [-w, w]^2\}$) As suggested in the hint CNO is defined as a compositional mapping between functions spaces.

$$\mathcal{G} : u \to P(u) = v_0 \to v_1 \to v_2 \to \cdots v_L \to Q(v_L) = \overline{u}$$

Where for each step in the middle (i.e. excluding the initial lifting and final projection) we have

$$v_{l+1} = \mathcal{P}_l \circ \Sigma_l \circ \mathcal{K}_l(v_l) \quad 1 \leq l \leq L-1$$

$\mathcal{P}_l$ can be an upsampling or a downsampling operator (depending on where we are in the U-Net architecture). upsampling boils down to simply enlarging the

considered grid and downsampling requires a convolution with the interpolation sink filter.

Σ_l is the activation operator that itself has inside: an upsampling, an application of the activation point-wise and a downsampling. We have seen in class that the upsampling procedure is important at this step to accommodate newly created higher modes.

$\mathcal{K}_l$ is the convolution operator that for a single channel $\mathcal{K}_w f(x) = (\mathcal{K}_w \star f)(x) = \int_D \mathcal{K}_w(x-y)f(y)dy = \sum_{i,j=1}^{k} k_{ij} f(x - z_{ij}) \quad \forall x \in D$. Where k is the kernel size and i, j are grid points

The lifting operator is used to take the original channels into a new latent space more suitable for learning (usually with many more channels) and the projection operator brings back the original number of channels. The complete architecture that shows the multi-scale processing is shown below

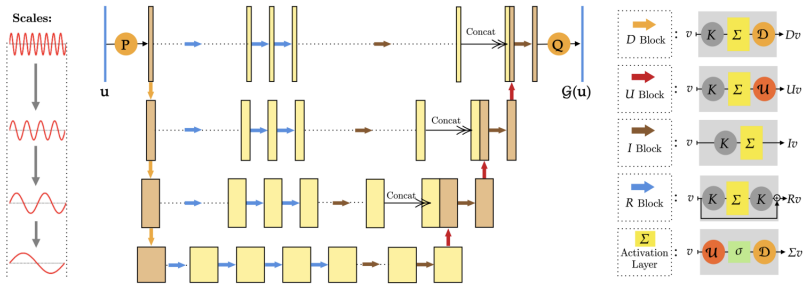


Figure 5: CNO architecture

3.2 Universality theorem

The universality theorem in practice means that the CNO $\mathcal{G}$ applied on band-limited functions can approximate with arbitrary precision any sufficiently regular operator $\mathcal{G}^\dagger$ and that works $\forall a \in \chi^*$ with $||a||_{H^r(D)} \leq B$, meaning that we are not learning a representation of the operator, but the operator itself.

3.3 Regularity of $\mathcal{G}^\dagger$

We would like the operator $\mathcal{G}^\dagger$ to be continuous and to respect the following modulus of continuity:

$$||\mathcal{G}^\dagger(a) - \mathcal{G}^\dagger(a')||_{L^p(\mathbb{T}^2)} \leq \omega\left(||a - a'||_{H^\sigma(\mathbb{T}^2)}\right)$$

for some $p \in \{2, \infty\}$ and $0 \leq \sigma \leq r-1$, and where $\omega : [0,\infty) \to [0,\infty)$ is a monotonously increasing function with $lim_{y \to 0}\omega(y) = 0$

3.4 Proof

The proof can be constructed in two steps:

3.4.1 Construction of $\mathcal{G}$ that approxiamtes $\mathcal{G}^\dagger$

We want to construct a continuous $\mathcal{G} : H^r(\mathbb{T}^2) \to C(\mathbb{T}^2)$. We start by setting

$$G : (a(x^M), e^{M/2}(x^M)) \to \mathcal{G}^\dagger(P_M(a))(x_{0,0})$$

Where P_M is a trigonometric polynomial interpolation operator, x^M the sampling points, $e^{M/2}$ the Fourier basis and $x_{0,0}$ the approximation point.
We know by the Nyquist–Shannon sampling theorem and the Whittaker–Shannon interpolation formula that there is a bijection between the discrete values $(a(x^M), e^{M/2}(x^M))$ and $P_M a$ and $e^{M/2}$ so G will be well definded. We can choose a shalow NN Ψ to approxiamte G by the universal approximation theorem and we can extend it to the whole $\mathbb{T}^2$ by using $e^{M/2}$ (call this extension Ψ^*. In the end our final construct will be

$$\mathcal{G}(a)(z) = (P_N \circ \Psi^*)(P_M a)(z)$$

3.4.2 This construction corresponds to a CNO

Because of the bijection identified above, we just need to find a CNO with an input such that the continuous representation will correspond to $\mathcal{G}(a)$. It can be shown that we can construct $\Psi = \pi \circ \Phi$ (where π is a projection onto the fist coordinate and Φ is a shallow NN) and a CNN $\hat{\Phi}$ so that $\Psi^*(x^N) = \hat{\Phi}(P_M a(x^N), e^{M/2}(x^N))$ and therefore

$$\mathcal{G}(a)(x^N) = (P_N \circ \Psi^*)(P_M a)(x^N) = h_N \star \hat{\Phi}(P_M a(x^N), e^{M/2}(x^N))$$