Pablo Lahmann - 19-947-035
AI in the Sciences and Engineering
May 20, 2024

Project 2: Neural Operators in Practice

DUE: May 20th, 8pm.

Question 1. Time Series Forecasting with Neural Operators

For task 1 I decided to base my code on tutorial 5. I used the implementation of the classes SpectralConv1d and FNO1d. To put the training data into a format fit for training, I use the data points to aggregate them into sliding windows of length 34. Additionally, it turned out helpful, to normalize all temperature and time values into $[0, 1]$. Also, we do not perform a train-test split, since we use the predictions on the provided training data for validation of our models.
For training I decided to use one neural network for T_f and T_s each. We later optimize these models independently from each other. In experimentation, it turned out that optimizing them together gives no difference in the results.
I achieved the best final results for a learning rate of $r = 0.005$ on 50 epochs, and with a padding of 7. Increasing the amount of epochs would lead to over-fitting, and give worse results in the prediction, while lowering the amount of epochs would give much worse results as well.
In the current implementation, the results are good. I tried many different combinations of hyper parameters for improving the performance, but did not manage to get better predictions.

Question 2. Modeling Water Flow on the Sphere with FNO

Implementation:
My implementation of SFNO is based on the implementation of FNO2d from tutorial 5. Though, it was necessary to make some adjustments. Conceptually, in SFNO we don't extract fourier features inside our specialized convolutional layer. Instead, we work with a shperical harmonics decomposition of our data. This could essentially be achieved by using the provided functions RealSHT and InverseRealSHT. These essentially replace the fourier transformation. Another important part of implementation was to extract the data with the provided function `load_spherical_swe`, as well as using the training function from the same library. This required some debugging, since the trainer would pass data in a different format to the model, than what was implemented before. Lastly training the model on several different hyperparameters, I found the current implementation to work best.
Implementing the baseline FNO algorithm was similar, since it required much of the same debugging as for the implementation for SFNO. With the implemented hyperparameters, the model gives sensible predictions.

Experimental Setup:
I trained the SFNO and FNO each on training data at resolutions (32, 64), (64, 128), (128,256) and (256,512). Training at any higher resolutions would probably take several hours of training, and was not feasible to do. Each model was then tested on data at resolutions (32, 64), (64, 128), (128,256), (256,512), (512, 1024). For each model I provide a saved figure plot of the performance on all training resolutions. Further, I added a plot which summarizes the MSE of every model on each training resolution. Lastly, my submission includes the code for implementation of the SFNO architecture.

Analysis:

SFNO: We can clearly see that SFNO consistently performs better at test resolutions on which is was trained. Thus it is possible that SFNO generally performs better on testing resolutions, which it was trained on. But note that in operator learning this is not optimal. This indicates that SFNO is overfitting on the training resolution. Generally, we can observe that the performance generally worsens as we deviate from the training resolution. This indicates that SFNO might generally have difficulty with generalizing to resolutions, which it has not seen before.

FNO: Like SFNO, also FNO perform best at the resolution that it was trained on, and tends to perform worse as we deviate in testing from the training resolution. But we notice that FNO shows more significant drops in performance than SFNO, when tested on resolutions different to the training resolution. This lets me believe that FNO might be more sensitive to changes in resolution.

Comparison FNO and SFNO: Generally SFNO has lower MSE values compared to FNO, when trained and tested on the same resolution. In total, this indicates that SFNO might be generally be a more effective model in this specific setting.
Another interesting observation is that both SFNO and FNO show improved generalization, when trained on higher resolutions. Though SFNO does this more effectively. This indicates that for both models it is of advantage to train on high resolutions. But I have noticed that training on higher resolutions takes significantly longer.

Conclusion:
Both the SFNO and FNO models tend to perform best on the resolution they were trained on, with performance decreasing as the testing resolution deviates from the training resolution. SFNO shows more consistent and generally lower MSE values compared to FNO across different training and testing resolutions. This indicates that SFNO performs better and is more stable. Training on higher resolutions improves noth models' ability to generalize to other resolutions, suggesting the importance of high-resolution training data for better model performance. Though, naturally this comes at a price of high computation cost. My observations suggest that while both models benefit from training on higher resolutions, SFNO might be a more reliable choice for tasks which should generalize across different resolutions.

Question 3. Universal Approximation for CNO

How is the CNO $\mathcal{G}$ defined?

$\mathcal{G}$ is an operator on the space of bandlimited functions. Specifically it is given by a lifting operator, then repeatedly applying the composition of an up-/downsampling, convolution, and activation operator, and at the very end applying a projection operator. Mathematically formulated, it has the following structure

$$\mathcal{G} : u \mapsto P(u) = v_0 \mapsto v_1 \mapsto \dots v_L \mapsto Q(v_L) = \bar{u}, \tag{1}$$

where

$$v_{l+1} = \mathcal{P}_l \circ \Sigma_l \circ \mathcal{K}_l(v_l), \quad 1 \leq \ell \leq L-1. \tag{2}$$

For u being the the input bandlimited function, Q the operator projecting onto the output space, P the lifting operator projecting onto a higher dimensional latent space, $\mathcal{P}_l$ the up-/downsampling operator, Σ_l the activation operator, and $\mathcal{K}_l$ the convolution operator.

What does the universality theorem mean in practice?

The universality theorem proves that CNO's can, in practice, arbitrarily well approximate the true underlying solution to a differential equation, given that the PDE is sufficiently regular. Thus it is a potentially very powerful model. Although note, that this doesn't mean that is easy to find a good CNO for a given PDE.

Proposals for regularization constraints for $\mathcal{G}^\dagger$, such that the universality theorem holds.

Since $\mathcal{G}$ maps onto bandlimited functions, it would be helpful for $\mathcal{G}^\dagger$ to be sufficiently smooth/nice, to be approximated by $\mathcal{G}$ (which is essentially a composition of "nice" functions). This is achieved by assuming there exists a continuity modulus for the operator.

Make a rough sketch of the universality theorem proof.

Mathematically we try to prove that for any operator $\mathcal{G}^\dagger$, there exists a CNO $\mathcal{G}$, which can approximate it to arbitrary accuracy. For this we regard a generalized solution $\tilde{\mathcal{G}}$, which is characterized by sobolev functions $a, f_1, ..., f_l$. The operator of interest $\mathcal{G}^\dagger$ then maps function a to the solution given by the differential operator $\mathcal{L}$
We make the following important assumptions. Firstly, the differential operator (e.g. describing the PDE) $\mathcal{L}$ depends on $a, f_1, ..., f_l$. Secondly, the solution to this operator is given by the continuous operator $\mathcal{G}^\dagger$. Thirdly, $\mathcal{G}^\dagger$ fulfills a continuity and stability condition, which bounds the difference of solutions by the functions $f_1, ..., f_l$. Lastly, assume that the activation function of the CNO is at least r-times continuously differentiable.
The proof is split into two sections. In the first section we construct the operator $\mathcal{G}$, and in section 2 we prove that it is a CNO.
The first half, starts with using fourier features, to approximate the functions $a, f_1, ..., f_l$ through polynomials arbitrarily well. Then we define a mapping $\mathcal{G}$, which maps these fourier features, and a discrete version of a onto the solution of the PDE. From the universal approximation theorem, we additionally know that this mapping $\mathcal{G}$ can be approximated with a shallow neural network. This neural network can then be extended to the continuous domain. We do this by defining an operator Ψ^*, which has the same accuracy as the discrete version. Thus we can approximate the solution.
In the second half of the proof we prove that $\mathcal{G}$ is a CNO. For this we first show that Ψ^* can be formulated as a CNN with certain convolutional filters. Then we prove that exactly these previously described filters and discretizations can approximate $\mathcal{G}^\dagger$ on the entire domain.
Finally, we combine the results about approximation accuracy, and stability, to deduce that $\mathcal{G}$ can approximate $\mathcal{G}^\dagger$ with arbitrary accuracy. This proves the desired theorem.