ETH zürich

Time dependent Fourier Neural Operator for the Wave Equation

Benedict Armstrong
benedict.armstrong@inf.ethz.ch

January 2025

Introduction

This study focuses on implementing a Fourier Neural Operator (FNO) to solve the one-dimensional wave equation, following the approach proposed by Li et al. [[1]]. The FNO architecture is designed to learn the spatial and temporal dynamics of partial differential equations (PDEs) directly from data, enabling efficient and generalizable solutions. The task was divided into four main subtasks and an additional bonus task, with the objective of training both a fixed-time FNO and a time-dependent FNO. The trained models are available in the `models` directory of this repository.

One-to-One Training

The FNO implementation was based on the model from the tutorials and slightly modified. The model, located in `lib/fno.py`, was trained for 5000 epochs using the `Adam` optimizer and `ReduceLROnPlateau` scheduler, with the relative L_2 error[1] as the loss function. The training details are summarized in Table 4. The model achieved an average relative L_2 error of 0.028 on the test set for the mapping $u_0 \to u(t = 1.0)$. Some examples are plotted in the notebook `task2.ipynb` and in Figure 1. The training data resolution was 64.

[1] rel. L_2 error: $l_2(u, u_{\text{ref}}) = \frac{\|u - u_{\text{ref}}\|_2}{\|u_{\text{ref}}\|_2}$

Testing on Different Resolutions

To evaluate FNO's ability to generalize across resolutions the model was tested on datasets with varying resolutions. The relative L_2 errors are listed in Table 1. The model performed best at its training resolution of 64, with slight error increases at higher resolutions and a significant increase at the lowest resolution of 32. This may be due to lower resolution data failing to capture higher frequency components.

Resolution	Average relative L_2 error
32	0.1070
64	**0.0288**
96	0.0396
128	0.0479

Table 1: Predictions on test data, Training res.: 64

Visualization of the model's predictions on the test data at different resolutions can be found in the `task2.ipynb` notebook and in Figure 2.

Testing on Out-of-Distribution (OOD) Dataset

The model was also evaluated on a provided OOD dataset, yielding a relative L_2 error of 0.091, higher than the in-distribution test error, as expected since the OOD data features data with higher frequency components. Examples of the model's predictions on the OOD dataset can

be found in the `task3.ipynb` notebook and in Figure 3.

All2All Training

The final subtask was to train a time dependent FNO: *TFNO*. The key modifications included adding time-conditional batch normalization (FILM) layers and time as an additional input channel. The time is also lifted to 32 dimensions through a linear layer before being passed to the FILM layer. The implementation is spilt up into the `lib/tfno.py` and `lib/layers.py` files. The main challenge for this task was the limited amount of training data.

Data

As outlined in the task description I used All2All sampling which involves generating training samples by considering every possible ordered combination of timesteps within the dataset. For each timestep, we create pairs where the current timestep is used as the input, and all future timesteps (including the current one) are used as the target. This means that for a given timestep t, we generate pairs $(t, t + \Delta t)$ for all $t \geq 0$ (including the identity mapping where $\Delta t = 0$). Additionally I augmented the data by adding the flipped (along the x-axis) and negated version of the data, doubling the size of the dataset again. For N samples with T timesteps and a factor 2 to account for the augmentation this results in a total number of samples:

$$|S| = 2 * \left[\frac{T * (T-1)}{2} + T \right] * |N|$$

In our case with $N_{\text{train}} = 64$ and $T = 5$, this results in $|S| = 1920$ samples.

Training

The model was trained for 1500 epochs and using the same loss function as for the regular FNO. The full parameters used to train the model are listed in Table 5. The model achieved a relative L_2 error of 0.251 on the test data. The model's predictions on the test data can be found in the `task4.ipynb` notebook.

Results

To compare the model, I first ran a baseline using the FNO from the previous task, adjusted to accept time dependent data. Table 2 confirms that the TFNO model trained on the augmented data significantly outperforms the FNO model, however, the error is still much higher than the one observed in the first subtask. The limited number of trajectory samples even after augmentation is likely a reason for the higher error.

Training Type	Average rel. L_2 err.
TFNO + augmented data	**0.211**
TFNO	0.312
baseline (FNO)	0.405

Table 2: Predictions on test dataset ($t = 0 \rightarrow t = 1$)

Bonus Task

Table 3 shows, that the model performs best on predictions from $t = 0$ to $t = 1$. The relative L_2 error increases for predictions further into the future. This makes sense as for the smallest time step the model sees 4 examples for each sample in the original dataset. For the largest time step the model only sees one example. Some examples of the model's predictions on the test data for different time steps can be found in the `task4bonus.ipynb` notebook and in Figure 4 and Figure 5 in the appendix.

Dataset, Δt	Average rel. L_2 err.
Test, $\Delta t = 0.25$	0.215
Test, $\Delta t = 0.50$	0.309
Test, $\Delta t = 0.75$	0.318
Test, $\Delta t = 1.00$	**0.211**
Test, All Δt	0.263
OOD, $\Delta t = 1.00$	0.297

Table 3: Predictions on test dataset at different Δt

As seen with the FNO implementation, the model performs worse on the OOD dataset, achieving an L_2 error of 0.297. Interestingly, this model seems to generalize better to the OOD data. This might be due to the higher model complexity and the addition of dropout terms as well as the

increased number of samples through the All2All training.

Bibliography

[1] Z. Li *et al.*, "Fourier Neural Operator for Parametric Partial Differential Equations." [Online]. Available: https://arxiv.org/abs/2010.08895

Appendix

Parameter	Value
Epochs	5000
Batch size	10
Optimizer	Adam
Weight decay	1e-5
Learning rate	1e-4
Scheduler	ReduceLROnPlateau
Patience	50
Loss	Relative L_2 error
Modes	16
Width	64
Fourier Layers	2

Table 4: Parameters used to train the FNO model

Parameter	Value
Epochs	1500
Batch size	512
Optimizer	AdamW
Learning rate	1e-3
Scheduler	CosineAnnealingWarmRestarts
eta_min	1e-6
T_0	15
Loss	Relative L_2 error + $1e^{-7}$ × smoothness_loss
Modes	16
Width	64
Fourier Layers	4

Table 5: Parameters used to train the FNO model

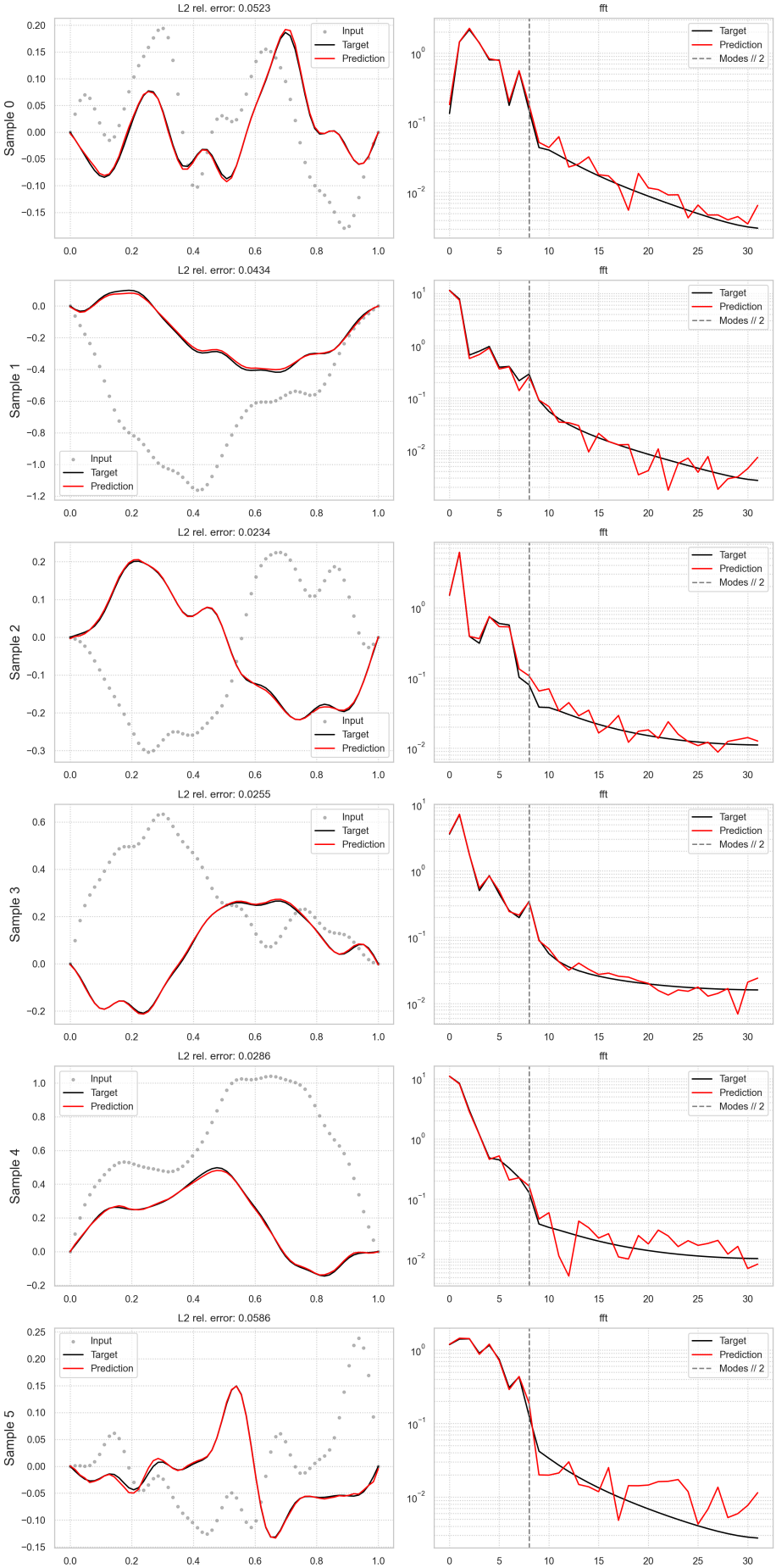

Figure 1: Predictions on test data for the FNO model

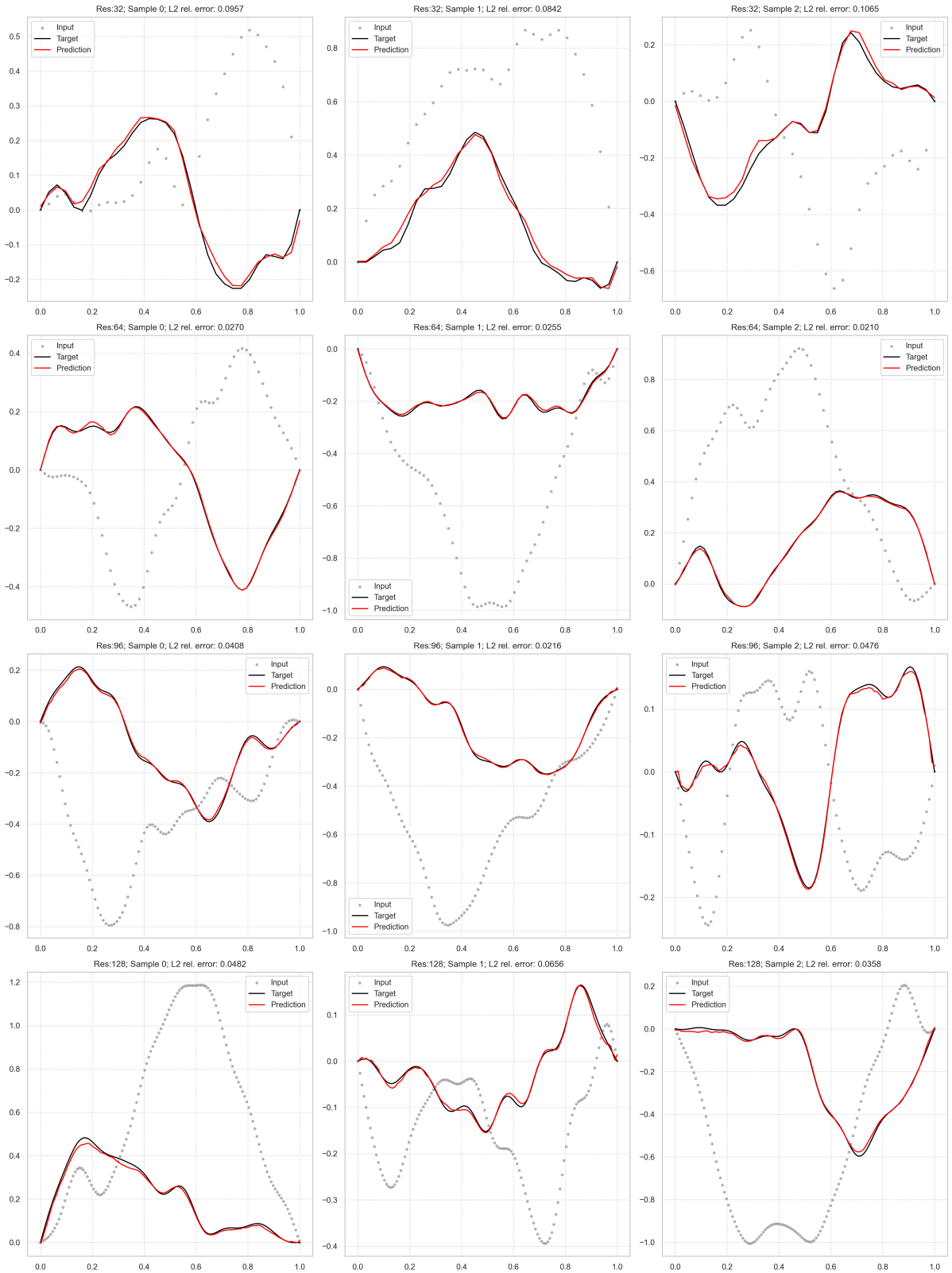

Figure 2: Predictions on test data for the FNO model at different resolutions

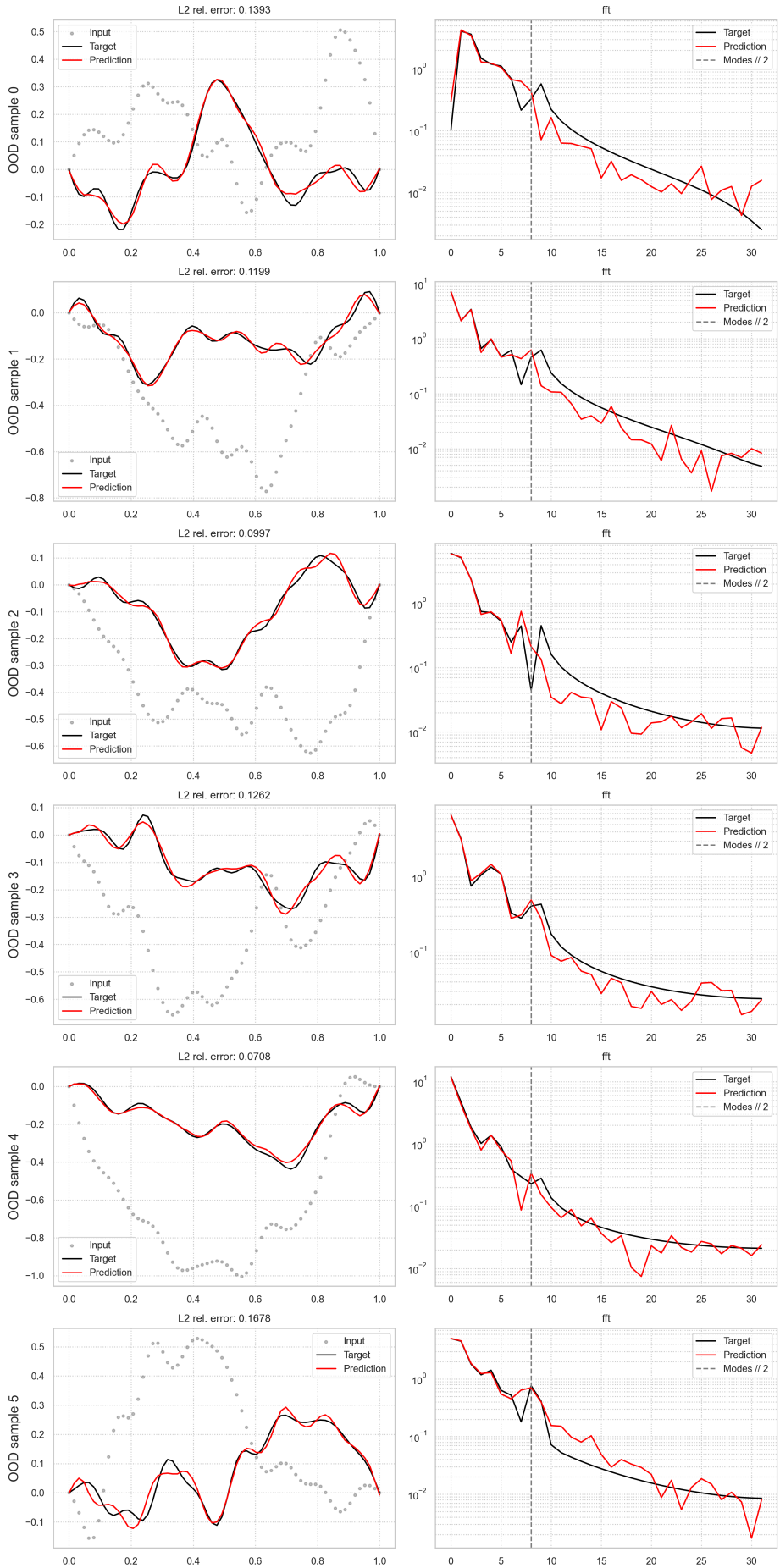

Figure 3: Predictions on OOD data for the FNO model

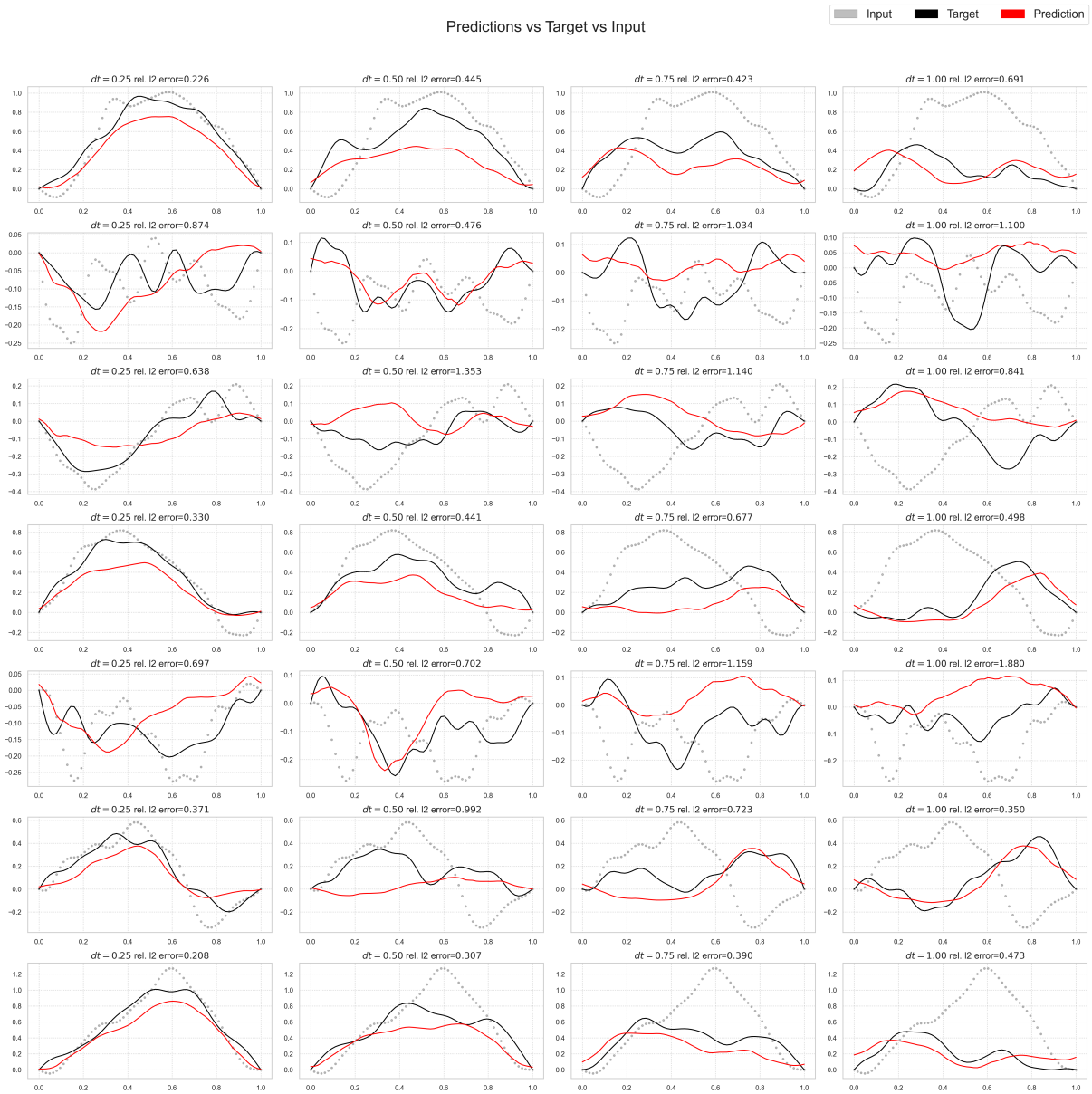

Figure 4: Predictions on test data for the TFNO model

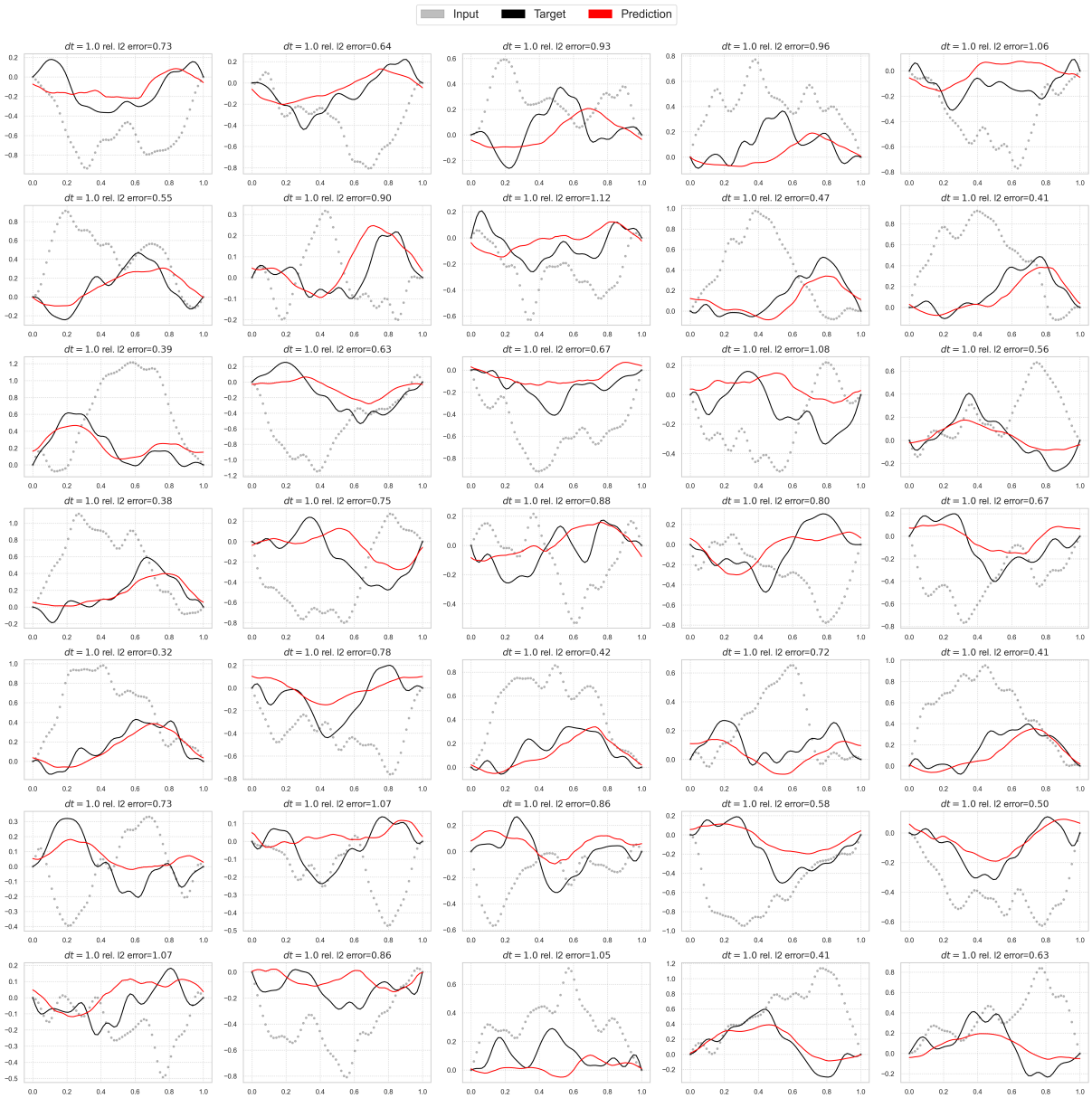

Figure 5: Predictions on OOD data for the TFNO model