HYPERTIME: IMPLICIT NEURAL REPRESENTATION FOR TIME SERIES –SUPPLEMENTARY MATERIAL –

Anonymous authors
Paper under double-blind review

1 ADDITIONAL RELATED WORK

Implicit Neural Representations INRs (or coordinate-based neural networks) have recently gained popularity in computer vision applications. The usual implementation of INRs consists of a fully-connected neural network (MLP) that maps coordinates (*e.g.* xyz-coordinates) to the corresponding values of the data, essentially encoding their functional relationship in the network. One of the main advantages of this approach for data representation, is that the information is encoded in a continuous/grid-free representation, that provides a built-in non-linear interpolation of the data. This avoids the usual artifacts that arise from discretization, and has been shown to combine flexible and accurate data representation with high memory efficiency (Sitzmann et al., 2020b; Tancik et al., 2020). Whilst INRs have been shown to work on data from diverse sources, such as video, images and audio (Sitzmann et al., 2020b; Chen et al., 2021; Rott Shaham et al., 2021), their recent popularity has been motivated by multiple applications in the representation of 3D scene data, such as 3D geometry (Park et al., 2019; Mescheder et al., 2019; Sitzmann et al., 2020a; 2019) and object appearance (Mildenhall et al., 2020; Sztrajman et al., 2021). In early architectures, INRs showed a lack of accuracy in the encoding of high-frequency details of signals. Mildenhall et al. (2020) proposed positional encodings to address this issue, and Tancik et al. (2020) further explored them, showing that by using Fourier-based features in the input layer, the network is able to learn the full spectrum of frequencies from data. Concurrently, Sitzmann et al. (2020b) tackled the encoding of high-frequency data by proposing the use of sinusoidal activation functions (SIREN: Sinusoidal Representation Networks), and Benbarka et al. (2022) showed the equivalence between Fourier features and single-layer SIRENs. Our INR architecture for time series data (Section 3) is based on the SIREN architecture by Sitzmann *et al*. In Section 4 we compare the performance of different activation layers, in terms of reconstruction accuracy and training convergence speed, for both univariate and multivariate time series.

Hypernetworks A hypernetwork is a neural network architecture designed to predict the weight values of a secondary neural network, denominated a HypoNetwork (Sitzmann et al., 2020a). The concept of hypernetwork was formalized by Ha et al. (2017), drawing inspiration from methods in evolutionary computing (Stanley et al., 2009). Moreover, while convolutional encoders have been likened to the function of the human visual system (Skorokhodov et al., 2021), the analogy cannot be extended to convolutional decoders, and some researchers have argued that hypernetworks much more closely match the behavior of the prefrontal cortex (Russin et al., 2020). Hypernetworks have been praised for their expressivity, compression due to weight sharing, and for their fast inference times(Skorokhodov et al., 2021). They have been leveraged for multiple applications, including few-shot learning (Rusu et al., 2019; Zhao et al., 2020), continual learning (von Oswald et al., 2020) and architecture search (Zhang et al., 2019; Brock et al., 2018). Moreover, in the last two years some works have started to leverage hypernetworks for the training of INRs, enabling the learning of latent encodings of data, while also maintaining the flexible and accurate reconstruction of signals provided by INRs. This approach has been implemented with different hypernetwork architectures, to learn priors over image data (Sitzmann et al., 2020b; Skorokhodov et al., 2021), 3D scene geometry (Littwin & Wolf, 2019; Sitzmann et al., 2019; 2020a) and material appearance (Sztrajman et al., 2021). Tancik et al. (2021) leverage hypernetworks to speed-up the training of INRs by providing learned initializations of the network weights. Sitzmann et al. (2020b) combine a set encoder with a hypernetwork decoder to learn a prior over INRs representing image data, and apply it for image in-painting. Our hypernetwork architecture from Section 3 is similar to Sitzmann *et al*.'s,

however we learn a prior over the space of time series and leverage it for new data synthesis through interpolation of the learned embeddings. Furthermore, our architecture implements a Fourier-based loss, which we show to be crucial for the accurate reconstruction of time series datasets (Section 4).

Interpretable Time Series Seasonal-trend decomposition techniques are standard tools in time series analysis used to decompose a time series into trend, seasonal, and remainder components. The trend component encapsulates the slow time-varying behavior of the time series, while seasonal components capture recurring (i.e., periodic) fluctuations in the data. These techniques enable an intuitive and interpretable analysis of time series data which play an important role in a variety of downstream tasks, including forecasting and anomaly detection. The classic approach for performing the decomposition is the widely used STL algorithm (Cleveland et al., 1990). To account for outliers and distributional shifts, a robust version of the algorithm, called Robust STL, has also been proposed (Wen et al., 2019). Additional challenges in seasonal-trend decomposition involve dealing with complex time series data that exhibit multiple seasonal components, to which techniques such as multiple STL (MSTL) have been proposed (Bandara et al., 2022). The ability to break time series into interpretable components has been a topic of recent interest in the context of anomaly detection, forecasting, and generation. Relevant to this work is the recently proposed N-BEATS architecture (Oreshkin et al., 2020), a deep learning-based univariate time series forecasting solution that provides time series interpretability capabilities without considerable loss in predictive performance. The N-BEATS architecture explicitly encodes seasonal-trend decomposition into the network by defining two blocks: a trend block which uses a small ordered polynomial to capture slow varying behaviors, and a seasonality block which uses a Fourier series to capture cyclical patterns. Little work, however, has been done in the design of *generation* schemes that allow for decomposition of time series data into interpretable components. While TimeVAE (Desai et al., 2021) proposes a VAE architecture where the decoder has trend and seasonality blocks to allow for interpretable generation, no results highlighting the advantage of this capability were demonstrated.

2 DATASETS

We use publicly available univariate and multivariate time series datasets from the UCR archive (Bagnall et al., 2017). We selected four univariate datasets andv two datasets that were used in Fourier Flows (Alaa et al., 2021), Google stocks data and UCI Energy data, which we downloaded from the project's Github repository. Additionally we use three multivariate datasets with different characteristics, which are summarized in Table 1.

Table 1: Main characteristics of the datasets used.

Dataset	Number of Samples	Length of Time series	No of Features	Source
Crop	7200	46	1	URL
NonInvasiveFetalECGThorax1	1800	750	1	
PhalangesOutlinesCorrect	1800	80	1	
FordA	3601	500	1	
Stock	3585	100	1	URL
Energy	19635	100	1	
Cricket	108	1197	6	URL
MotorImagery	278	3000	64	
PhonemeSpectra	3315	217	11	

3 FOURIER-BASED LOSS

As part of the training of our HyperTime architecture, we propose a Fourier spectrum reconstruction loss. For a discrete-time signal $\mathbf{f} = \{f_0 = f(0), f_1 = f(1), \ldots, f_N = f(N)\}$, the N-point discrete Fourier transform (DFT) is utilized to obtain the corresponding frequency domain representation of $\mathbf{f}$ through the following operation:

$$F_k = [\mathcal{F}_T\{\mathbf{f}\}]_k = \sum_{n=0}^{N-1} f_n e^{-2\pi j\left(\frac{kn}{N}\right)}, \quad 0 \leq k \leq N-1,$$

where $j = \sqrt{-1}$ corresponds to the imaginary unit of a complex number. The coefficient $F_k \in \mathbb{C}$ quantifies the strength in representation of the kth frequency component of the signal. The DFT has a time complexity of $\mathcal{O}(N^2)$. In practice, an algorithm called the fast Fourier transform (FFT) is used to compute the DFT due to its lower time complexity (i.e., $\mathcal{O}(N \log N)$). Using the FFT to obtain the frequency domain representations of two discrete-time signals $\mathbf{f}$ and $\hat{\mathbf{f}}$, we introduce a Fourier-based reconstruction loss as follows:

$$\mathcal{L}_{\text{FFT}} = \frac{1}{N} \sum_{k=0}^{N-1} \|F_k - \hat{F}_k\|.$$

Here, we utilized the PyTorch implementation of the FFT to obtain the DFT for each signal. It is important to note that the DFT is only well-defined for regularly sampled signals. In the case of this work, the discrete-time signal $\mathbf{f}$ is obtained by deterministically sampling the function $f(t)$ via a discretized grid of time steps $t \in \{0, 1, \ldots, N\}$.

3.1 Analysis of Fourier-based loss

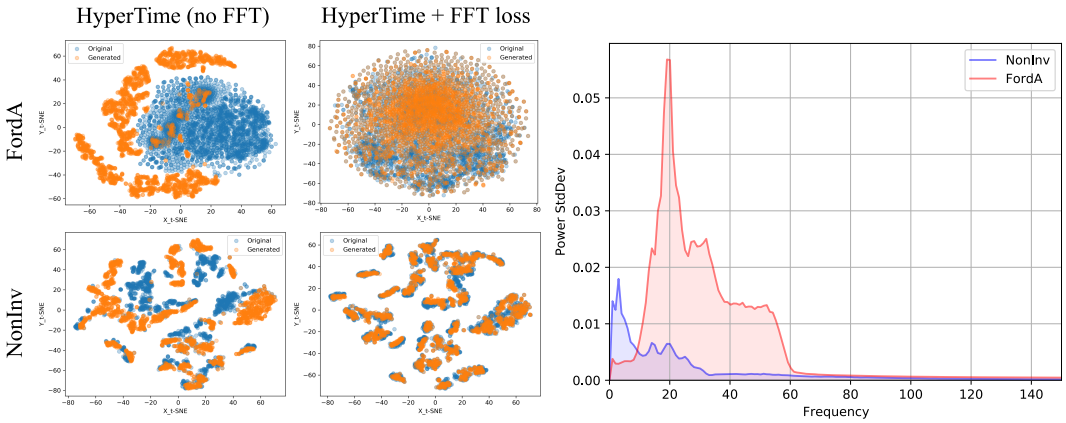


Figure 1: *Left:* t-SNE visualization of ground truth and generated data on two univariate datasets (NonInv and FordA), using HyperNet with and without the Fourier-based loss $\mathcal{L}_{\text{FFT}}$ (Eq. 3). *Right:* Standard deviation of the power spectra for the time series of the same two datasets. FordA shows a considerably larger number of variations in the distributions of the power spectra, which explains the difficulty of HyperTime to learn patterns from the data.

Here, we analyze in more detail the importance of the Fourier-based loss $\mathcal{L}_{\text{FFT}}$ from equation 3 on the training of HyperTime. In Figure 1-left we display t-SNE visualizations of time series synthesized by HyperTime with and without the use of the FFT loss during training, for two datasets (NonInv and FordA). In order to visualize the datasets better, we use 3000 samples for generated and original data, instead of the 1000 samples that we used in Figure 5. In both cases, the addition of the $\mathcal{L}_{\text{FFT}}$ loss results in an improved matching between ground truth and generated data. However, in the case of FordA, the addition of this loss becomes crucial to guide the learning process. This is also reflected in the numerical evaluations from Table 2, which shows steep improvements in performance for the FordA dataset.

A likely explanation for the difficulty of the network to learn meaningful patterns from the data of this dataset is provided by the right plot in Figure 1. Here we show the standard deviation of the power spectrum for both datasets, as a function of the frequency. The difference in the distributions indicates that FordA is composed of spectra that present larger variability, while NonInv's spectra are considerably more clustered. Further research on the characteristics of the datasets that benefit the more from the $\mathcal{L}_{\text{FFT}}$ loss should be further investigated, especially focusing on non-stationary time series.

4 RECONSTRUCTION

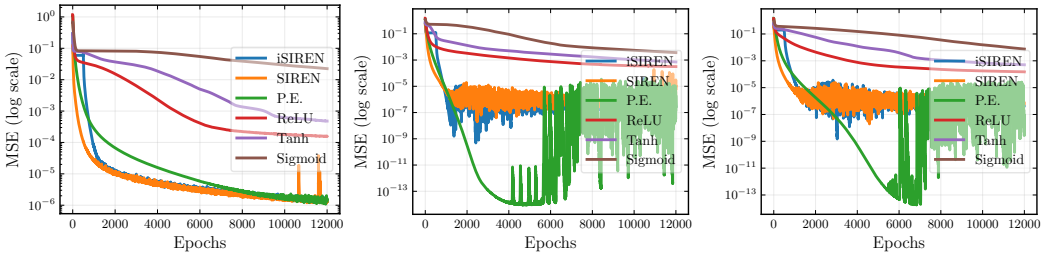
4.1 UNIVARIATE

For each dataset, we sample 100 time series, train the INR and average the metrics. Figure 2 shows the comparison of the losses running over many epochs. We can see that iSIREN and SIREN converge quickly, so for all subsequent experiments we used 1500 epochs for training. In the case of iSIREN, we first train the trend block for 100 epochs and then train both blocks for 1400 epochs. Figure 3 shows the reconstruction error of the power spectral density for this configuration. We show an extended version of reconstruction results in Table 2, with the comparison of reconstruction of iSIREN with other INRs across all datasets, with standard deviations in parenthesis.

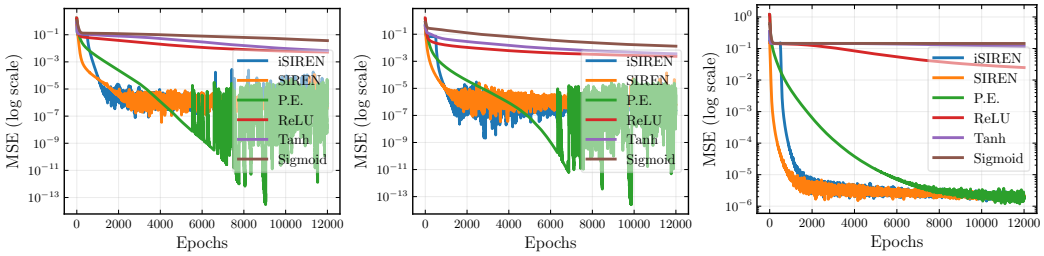
Figures 4 to 9 show additional results accross all datasets of the output of iSIREN and its trend and seasonality blocks.

Table 2: Comparison using MSE on time space and MAE in frequency space (FFT) of implicit networks using different activation functions and of iSIREN on univariate and multivariate datasets (standard deviations are shown in parenthesis).

Dataset	iSIREN (Ours)		SIREN		P.E.		ReLU		Tanh	
	FFT	MSE	FFT	MSE	FFT	MSE	FFT	MSE	FFT	MSE
Univariate										
Crop	1.4e-3	5.6e-6	1.4e-3	1.6e-6	**6.8e-4**	**7.3e-7**	5.4e-1	2.1e-2	8.6e-1	6.0e-2
	(6.7e-3)	(3.4e-5)	(3.7e-3)	(8.1e-6)	(3.0e-3)	(4.6e-6)	(3.2e-1)	(2.1e-2)	(4.4e-1)	(5.4e-2)
Energy	**4.1e-3**	**5.3e-6**	1.8e-2	1.2e-5	1.3e-1	7.7e-4	1.5e+0	4.9e-2	1.9e+0	8.3e-2
	(9.0e-3)	(1.9e-5)	(1.7e-2)	(1.7e-5)	(1.2e-1)	(1.2e-3)	(3.8e-1)	(2.2e-2)	(4.6e-1)	(3.7e-2)
FordA	**1.7e-2**	**4.9e-6**	1.9e-2	6.2e-6	3.1e-1	2.1e-3	2.5e+0	1.3e-1	2.8e+0	1.4e-1
	(9.5e-3)	(1.1e-5)	(1.1e-2)	(1.2e-5)	(2.5e-1)	(4.5e-3)	(5.2e-1)	(3.1e-2)	(5.3e-1)	(3.4e-2)
NonInv	**3.6e-2**	**1.2e-5**	4.0e-2	1.3e-5	1.1e-1	1.3e-4	1.0e+0	2.2e-2	1.3e+0	4.6e-2
	(8.7e-3)	(8.0e-6)	(1.1e-2)	(7.1e-6)	(3.4e-2)	(9.9e-5)	(1.8e-1)	(7.7e-3)	(1.6e-1)	(1.4e-2)
Phalanges	**1.4e-3**	2.1e-6	3.8e-3	**1.8e-6**	7.6e-3	1.2e-5	2.4e-1	3.8e-3	7.5e-1	8.4e-2
	(4.0e-3)	(1.4e-5)	(3.9e-3)	(1.0e-5)	(1.3e-2)	(5.2e-5)	(1.5e-1)	(5.5e-3)	(1.8e-1)	(3.9e-2)
Stock	**2.5e-3**	5.1e-6	4.4e-3	**1.4e-6**	4.3e-2	1.2e-4	6.2e-1	1.2e-2	8.9e-1	3.8e-2
	(9.1e-3)	(2.6e-5)	(4.0e-3)	(3.0e-6)	(3.9e-2)	(2.1e-4)	(2.5e-1)	(7.1e-3)	(3.2e-1)	(2.2e-2)
Multivariate										
Cricket	**3.9e-1**	**4.1e-4**	4.5e-1	4.2e-4	1.7e+0	3.7e-3	3.5e+0	1.7e-2	3.9e+0	3.1e-2
	(2.5e-1)	(5.9e-4)	(2.4e-1)	(5.9e-4)	(7.4e-1)	(2.2e-3)	(1.2e+0)	(8.7e-3)	(1.2e+0)	(1.1e-2)
MotorImagery	**5.1e+0**	**2.1e-3**	7.2e+0	6.2e-3	1.1e+1	2.4e-2	1.0e+1	2.6e-2	1.1e+1	3.0e-2
	(8.5e-1)	(7.7e-4)	(9.6e-1)	(1.8e-3)	(1.3e+0)	(5.3e-3)	(1.6e+0)	(6.3e-3)	(1.8e+0)	(6.9e-3)
PhonemeSpectra	**2.9e-2**	**2.1e-6**	4.2e-1	2.7e-4	1.8e+0	5.9e-3	3.0e+0	1.5e-2	3.4e+0	2.0e-2
	(1.5e-2)	(3.2e-6)	(1.9e-1)	(2.3e-4)	(8.0e-1)	(3.6e-3)	(1.0e+0)	(5.5e-3)	(9.9e-1)	(6.8e-3)

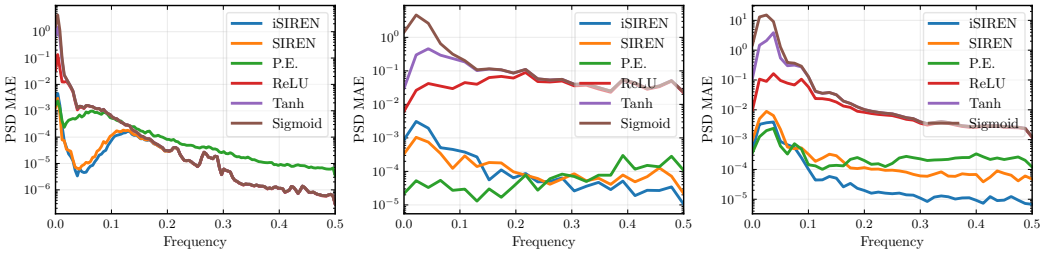


(a) NonInv (b) Crop (c) Phalanges

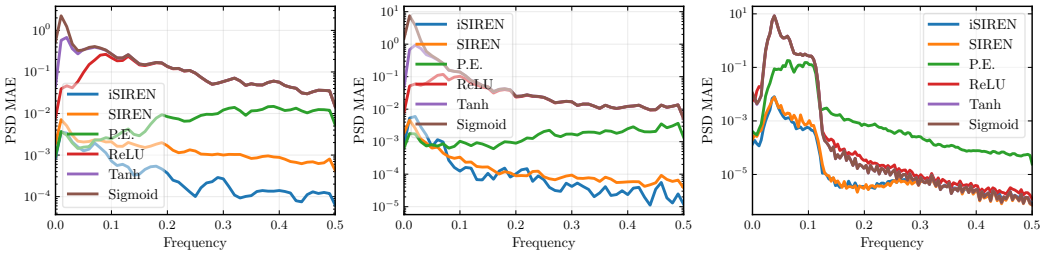


(d) Energy (e) Stock (f) FordA

Figure 2: Comparison of reconstruction of iSIREN with other INR encodings fitting multiple time series.

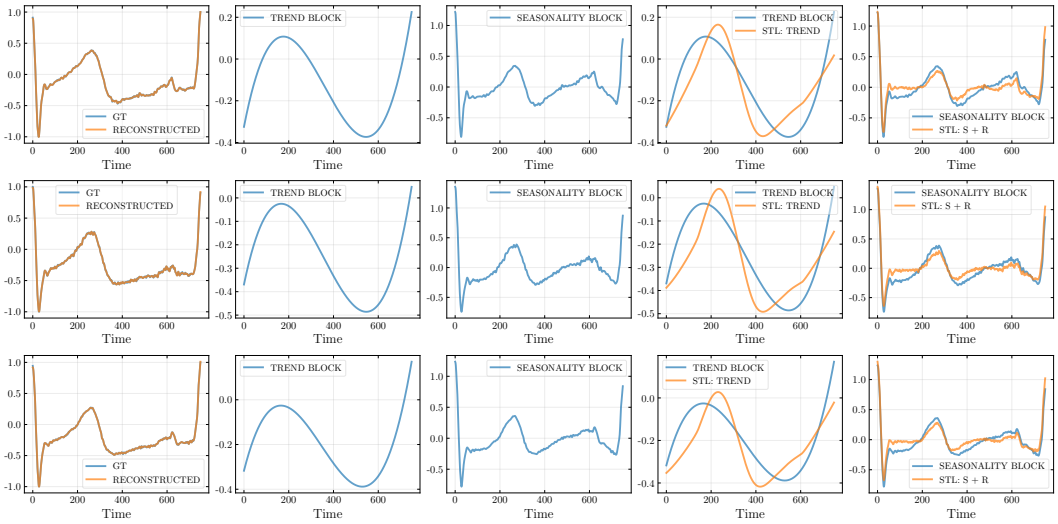


(a) NonInv (b) Crop (c) Phalanges



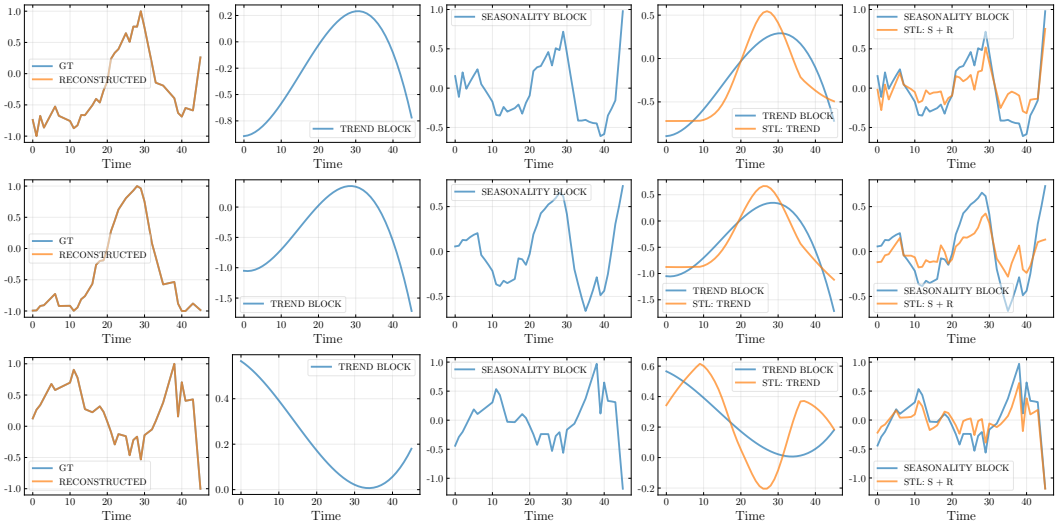
(d) Energy (e) Stock (f) FordA

Figure 3: Comparison of frequency reconstruction of iSIREN with other INR encodings fitting multiple time series evaluating MAE of power spectral density.



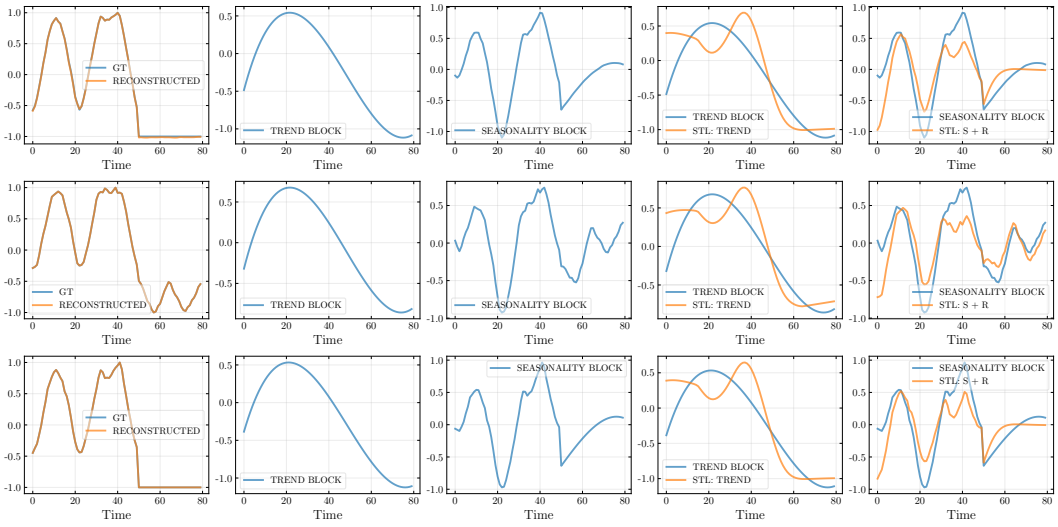
(a) Reconstruction (b) Trend (c) Seasonality (d) STL: trend (e) STL: seas

Figure 4: Additional outputs of iSIREN on the `NonInv` dataset. Each row is a randomly selected sample. Column (a) shows the actual value (GT) and the iSIREN reconstruction. Columns (b) and (c) show the output of the trend and seasonality blocks, respectively; the reconstruction is their summation. Columns (d) and (e) show the comparison of the interpretable blocks with classic STL decomposition.



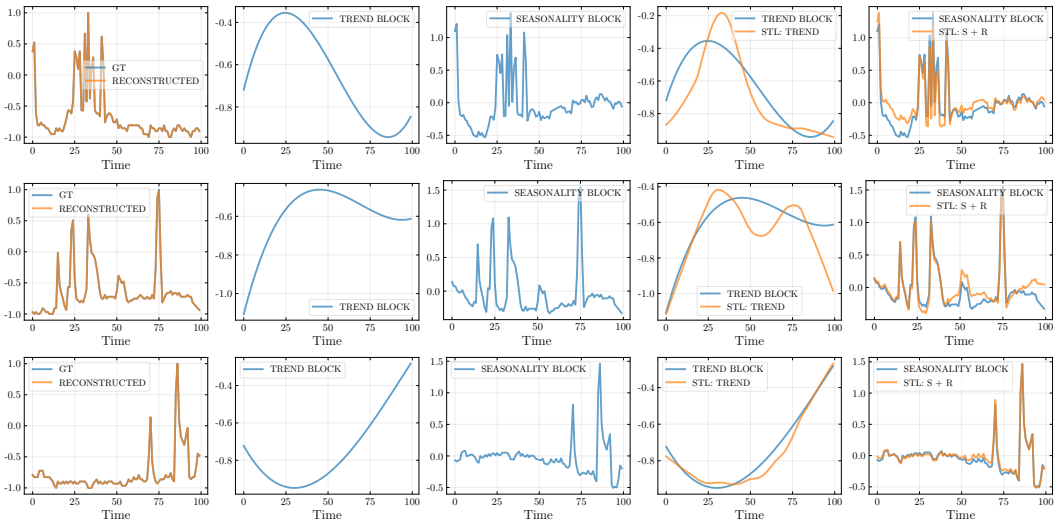
(a) Reconstruction (b) Trend (c) Seasonality (d) STL: trend (e) STL: seas

Figure 5: Additional outputs of iSIREN on the `Crop` dataset. Each row is a randomly selected sample. Column (a) shows the actual value (GT) and the iSIREN reconstruction. Columns (b) and (c) show the output of the trend and seasonality blocks, respectively; the reconstruction is their summation. Columns (d) and (e) show the comparison of the interpretable blocks with classic STL decomposition.



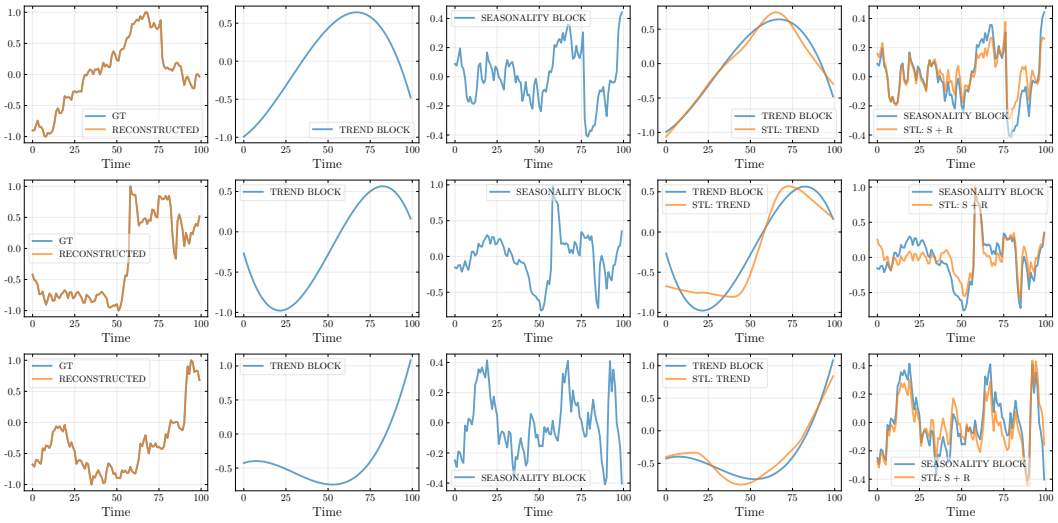
(a) Reconstruction (b) Trend (c) Seasonality (d) STL: trend (e) STL: seas

Figure 6: Additional outputs of iSIREN on the `Phalanges` dataset. Each row is a randomly selected sample. Column (a) shows the actual value (GT) and the iSIREN reconstruction. Columns (b) and (c) show the output of the trend and seasonality blocks, respectively; the reconstruction is their summation. Columns (d) and (e) show the comparison of the interpretable blocks with classic STL decomposition.



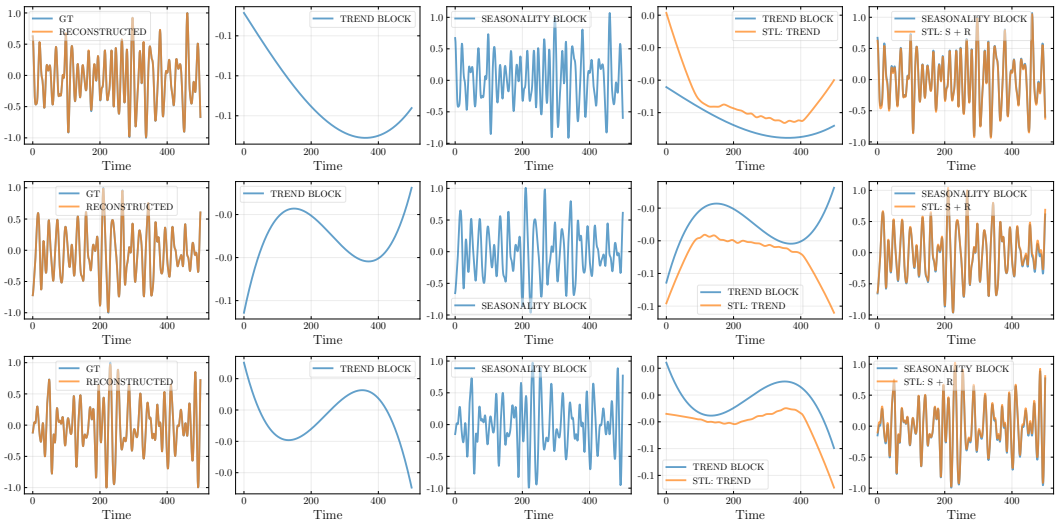
(a) Reconstruction (b) Trend (c) Seasonality (d) STL: trend (e) STL: seas

Figure 7: Additional outputs of iSIREN on the `Energy` dataset. Each row is a randomly selected sample. Column (a) shows the actual value (GT) and the iSIREN reconstruction. Columns (b) and (c) show the output of the trend and seasonality blocks, respectively; the reconstruction is their summation. Columns (d) and (e) show the comparison of the interpretable blocks with classic STL decomposition.



(a) Reconstruction (b) Trend (c) Seasonality (d) STL: trend (e) STL: seas

Figure 8: Additional outputs of iSIREN on the `Stock` dataset. Each row is a randomly selected sample. Column (a) shows the actual value (GT) and the iSIREN reconstruction. Columns (b) and (c) show the output of the trend and seasonality blocks, respectively; the reconstruction is their summation. Columns (d) and (e) show the comparison of the interpretable blocks with classic STL decomposition.



(a) Reconstruction (b) Trend (c) Seasonality (d) STL: trend (e) STL: seas

Figure 9: Additional outputs of iSIREN on the `FordA` dataset. Each row is a randomly selected sample. Column (a) shows the actual value (GT) and the iSIREN reconstruction. Columns (b) and (c) show the output of the trend and seasonality blocks, respectively; the reconstruction is their summation. Columns (d) and (e) show the comparison of the interpretable blocks with classic STL decomposition.

4.2 Multivariate

Additionally, Figure 10 shows the reconstruction of random samples of the multivariate dataset using iSIREN.

4.3 Implementation & Reproducibility Details

Architecture We use a three-layer MLP architecture with 60 neurons for all experiments.

Hyperparameters We train for 12000 epochs using Adam optimizer with a learning rate of 1e-4 to generate the plots in Fig. 2 and Fig. 3. The results on Table 2 are generated training for 1500 epochs.

Runtime It takes approximately 50 minutes to train 100 INRs for 12000 iterations.

Hardware The experiments are run using a *g4dn.2xlarge* AWS instance with a NVIDIA T4 GPU.

4.4 Code

The code can be provided upon request and will be placed on a public repository after the revision period.

5 Time Series Generation

Table 3 shows an extended version of generation results, with standard deviations in parenthesis and including training and inference times (in seconds).

Figure 11 shows generated time series using HyperTime and compares it with random samples of original time series for different datasets. We can see that the generated time series show a strong resemblance with the original time series and is in line with the t-SNE visualization on Figure 5 of the paper.

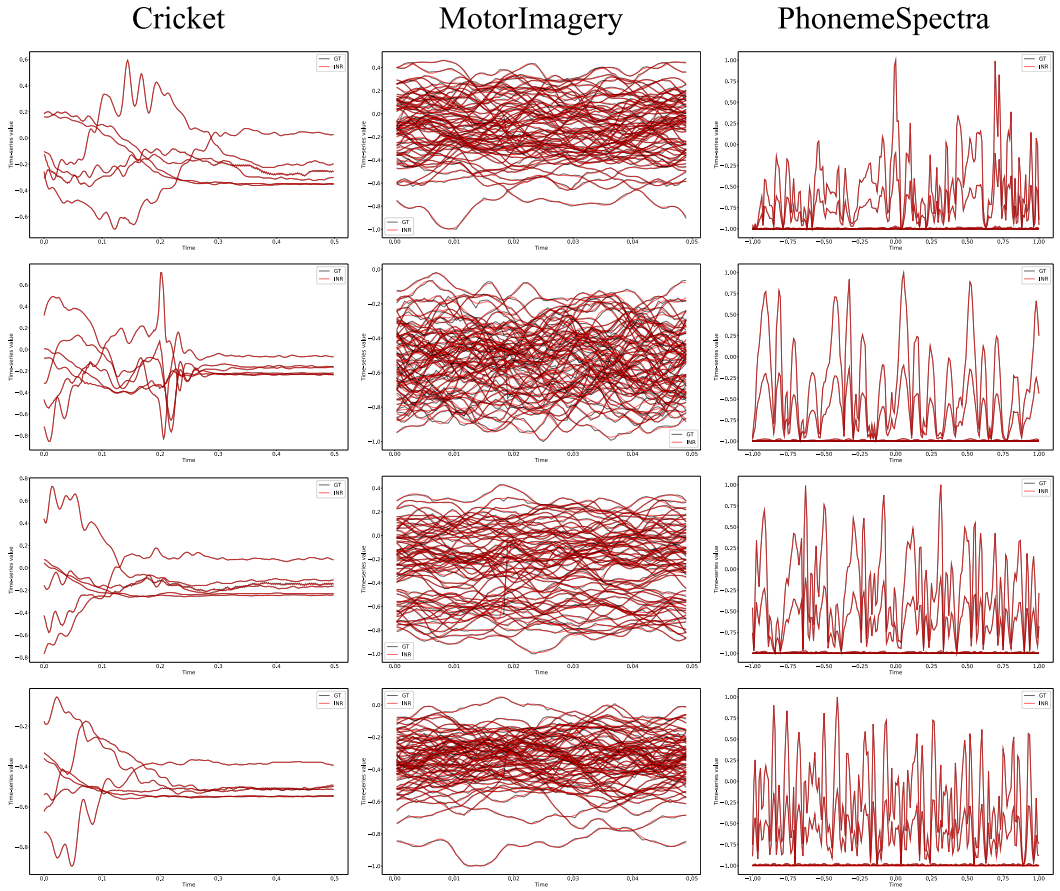


Figure 10: INR reconstructions of random samples from three multivariate datasets.

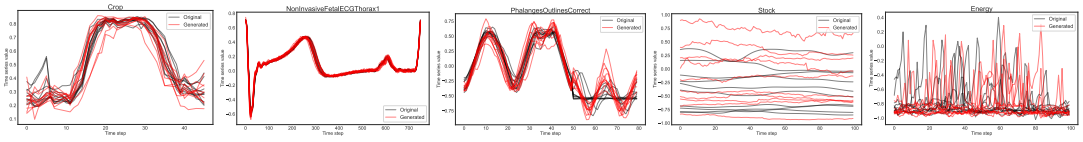


Figure 11: Random samples of generated time series using HyperTime, compared to random samples of original time series.

Additionally, figures 12 to 17 show the linear interpolation in seasonality, leaving the trend fixed and the interpolation in the trend space, leaving the seasonality fixed. The plots correspond to several datasets, and we can see that in all cases the transition of either block is smooth.

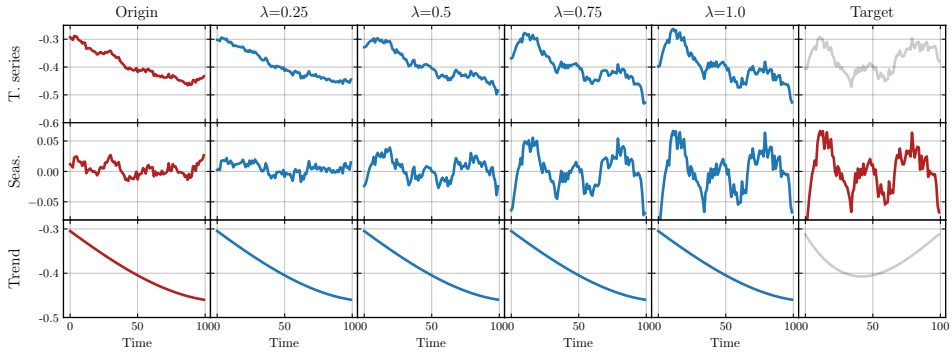
5.1 BASELINES

We use the following methods with publicly available code as benchmark for our method:

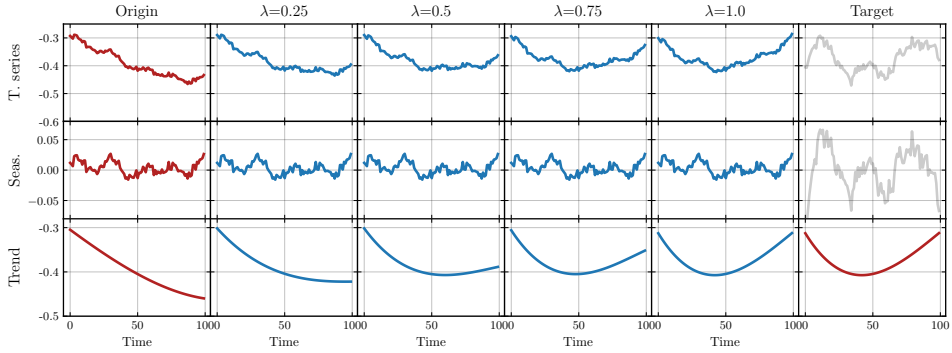
- Fourier Flows Alaa et al. (2021): `https://github.com/ahmedmalaa/Fourier-flows`
- RealNVP Alaa et al. (2021): `https://github.com/ahmedmalaa/Fourier-flows`
- TimeGAN (Yoon et al., 2019): `https://github.com/jsyoon0823/TimeGAN`

Table 3: Performance scores for data generation using baselines (TimeGAN, Fourier Flows, RealNVP) and multiple hypernet models: **HT (no FFT):** SIREN hyponetwork, trained without spectral loss. **HT (w/FFT):** SIREN hyponetwork. **HT (iSIREN):** iSIREN hyponetwork. **iHT:** interpretable HT.

	Crop	**NonInv**	**Phalan.**	**Energy**	**Stock**	**FordA**
RealNVP						
MAE	0.170	0.038	0.073	0.036	0.019	0.115
	(1.5e-2)	(3.6e-4)	(1.1e-4)	(7.2e-4)	(2.4e-3)	(4.9e-4)
F1 Score	0.981	0.986	0.976	0.964	0.977	0.999
	(7.3e-4)	(3.0e-4)	(1.8e-3)	(2.7e-3)	(3.4e-3)	(8.0e-6)
Time (training)	51.9	100.3	19.6	242.6	46.3	73.1
Time (generation)	0.03	0.20	0.06	0.07	0.06	0.09
TimeGAN						
MAE	0.048	–	0.108	0.056	0.173	–
	(1.7e-2)		(9.6e-3)	(1.2e-1)	(1.7e-1)	
F1 Score	0.831	–	0.960	0.479	0.938	–
	(2.4e-2)		(4.3e-2)	(2.4e-1)	(1.1e-2)	
Time (training)	1799	–	3177	3942	3872	–
Time (generation)	0.6	–	0.6	0.7	0.7	–
Fourier Flows						
MAE	0.040	0.018	0.056	**0.030**	**0.010**	0.024
	(4.5e-3)	(3.3e-3)	(9.1e-4)	(2.9e-5)	(1.5e-3)	(1.7e-3)
F1 Score	0.991	0.990	0.992	0.936	0.990	0.998
	(2.7e-3)	(3.9e-5)	(3.1e-3)	(1.8e-3)	(6.6e-4)	(1.5e-5)
Time (training)	864.1	543.7	75.2	1652.1	172.6	607.2
Time (generation)	0.07	1.46	0.08	0.09	0.09	1.2
HyperTime						
HT (no FFT)						
MAE	0.040	0.005	**0.023**	0.058	0.012	0.17
	(1.4e-4)	(1.5e-4)	(8.4e-4)	(1.6e-3)	(8.1e-4)	(4.7e-5)
F1 Score	**0.999**	0.996	0.996	**0.998**	**0.995**	0.084
	(1.4e-4)	(5.9e-4)	(7.1e-4)	(8.0e-5)	(9.0e-4)	(2.7e-3)
Time (training)	211.5	145.8	64.1	466.1	92.5	210.0
Time (generation)	0.5	0.4	0.3	2.1	0.7	0.5
HT (w/ FFT)						
MAE	0.040	0.005	**0.023**	0.057	0.013	**0.007**
	(1.8e-4)	(5.4e-5)	(8.2e-4)	(1.1e-3)	(1.1e-3)	(4.9e-5)
F1 Score	**0.999**	**0.997**	**0.999**	0.997	0.994	**0.998**
	(2.5e-4)	(5.9e-4)	(2.7e-4)	(9.3e-5)	(3.8e-4)	(9.2e-5)
Time (training)	218.1	147.1	64.4	477.0	95.4	213.4
Time (generation)	0.5	0.4	0.3	2.2	0.7	0.5
HT (iSiren)						
MAE	**0.039**	**0.004**	0.024	0.057	0.013	0.008
	(4.0e-4)	(5.1e-4)	(6.3e-4)	(3.6e-3)	(7.8e-4)	(5.7e-4)
F1 Score	**0.999**	**0.997**	**0.999**	0.997	**0.995**	0.997
	(9.6e-5)	(1.0e-4)	(1.7e-4)	(6.6e-5)	(4.5e-4)	(1.9e-4)
Time (training)	217.8	147.8	74.7	477.9	102.7	207.1
Time (generation)	0.5	0.4	0.3	2.1	0.7	0.5
iHT						
MAE	**0.039**	**0.004**	0.024	0.056	0.011	0.009
	(5.9e-4)	(1.5e-4)	(9.2e-4)	(6.4e-3)	(4.1e-4)	(1.8e-4)
F1 Score	**0.999**	**0.997**	0.997	0.997	**0.995**	0.996
	(1.9e-4)	(2.8e-4)	(2.4e-4)	(1.3e-4)	(5.8e-4)	(3.6e-4)
Time (training)	208.5	146.3	71.4	443.5	98.2	205.3
Time (generation)	0.5	0.4	0.3	2.1	0.7	0.5



(a) Interpolation in seasonality, leaving trend fixed.



(b) Interpolation in trend, leaving seasonality fixed.

Figure 12: Linear interpolation in seasonality (a) and trend (b) between two time series from the `Stock` dataset, leaving trend and seasonality fixed, respectively. In red are the original time series (first column), and target seasonality/trend (last column).

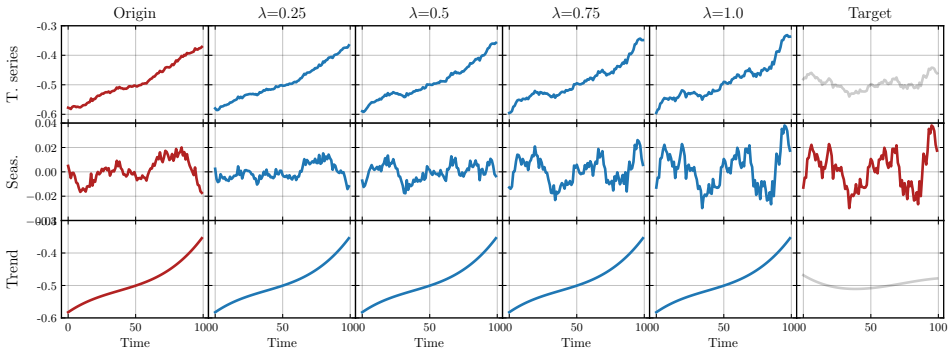
5.2 Implementation & Reproducibility Details

Architecture HyperTime is composed of a set encoder corresponding to a SIREN with input dimension 2, two hidden layers of 128 neurons and an output layer (embedding) of 40 neurons. The decoder (hypernetwork) is an MLP with ReLU activations with dimensions $40 \times 128 \times n_W$, where n_W corresponds to the number of weights of the hypo network given that the output of the hypernetwork is a one-dimensional vector that contains the hypo network weights. The hypo networks are MLPs, following the description of Section NUMBER.

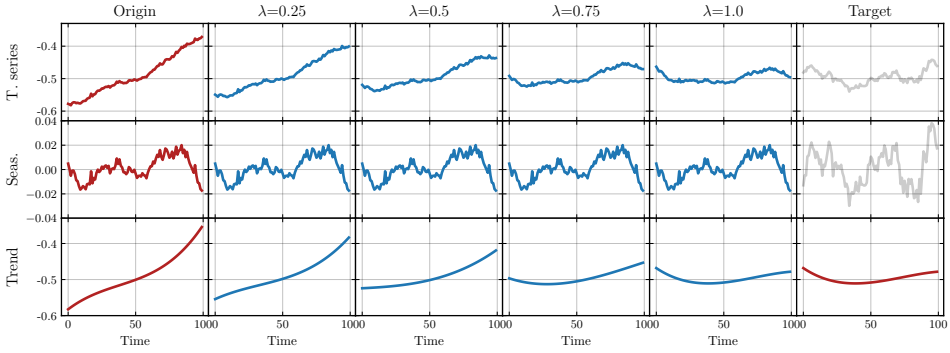
Hyperparameters We train all datasets for 300 epochs, with Adam optimizer and a learning rate of $5e-5$. We use loss parameters $\lambda_1 = 1.0 \times 10^{-3}$, $\lambda_2 = 1.0$ and $\lambda_3 = 1.0 \times 10^{-2}$. For HyperTime using iSIREN for the HypoNet and for iHyperTime, the training is performed in two stages, in order to improve stability. We first train 100 epochs the trend hypernetwork and trend block, using as loss only the reconstruction between ground truth and the output of the trend block, and then we train for 200 more epochs, using both the trend and seasonality blocks.

Runtime It takes between 1 and 4 minutes to train the datasets. More details are in table 3.

Hardware The experiments are run using a *g4dn.2xlarge* AWS instance with a NVIDIA T4 GPU.

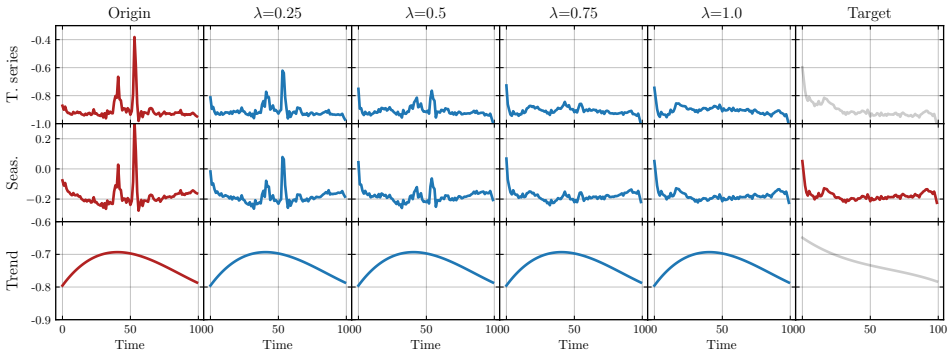


(a) Interpolation in seasonality, leaving trend fixed.

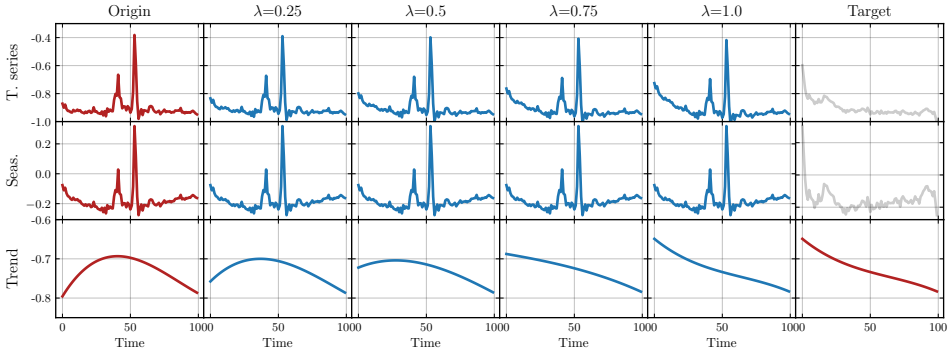


(b) Interpolation in trend, leaving seasonality fixed.

Figure 13: Linear interpolation in seasonality (a) and trend (b) between two time series from the `Stock` dataset, leaving trend and seasonality fixed, respectively. In red are the original time series (first column), and target seasonality/trend (last column).

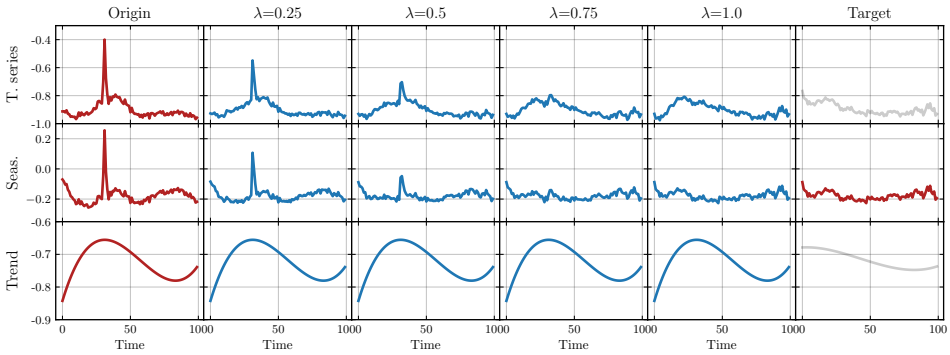


(a) Interpolation in seasonality, leaving trend fixed.

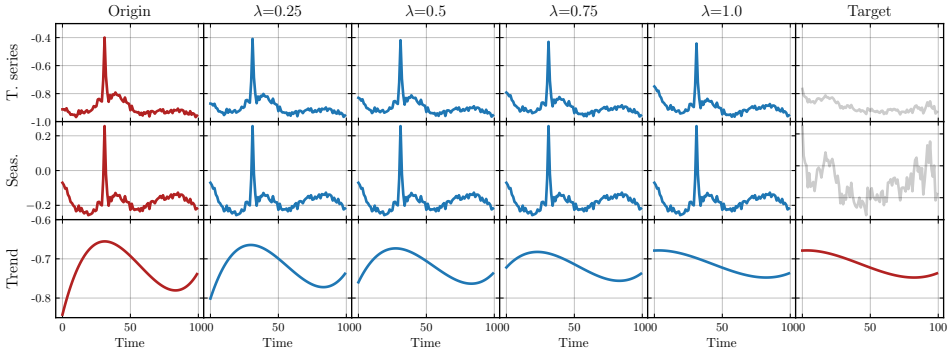


(b) Interpolation in trend, leaving seasonality fixed.

Figure 14: Linear interpolation in seasonality (a) and trend (b) between two time series from the `Energy` dataset, leaving trend and seasonality fixed, respectively. In red are the original time series (first column), and target seasonality/trend (last column).

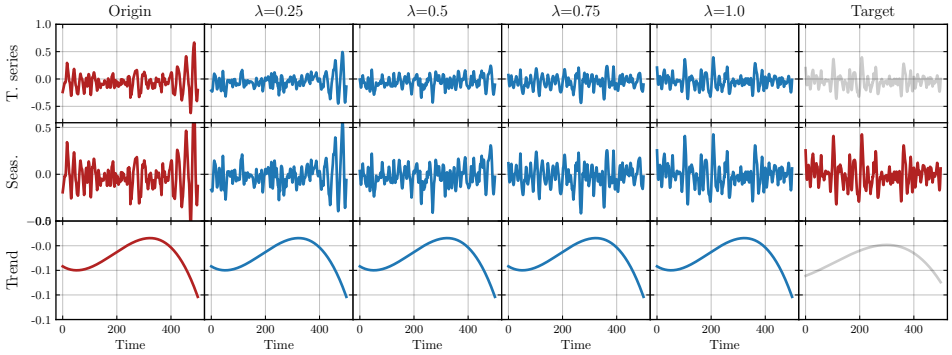


(a) Interpolation in seasonality, leaving trend fixed.

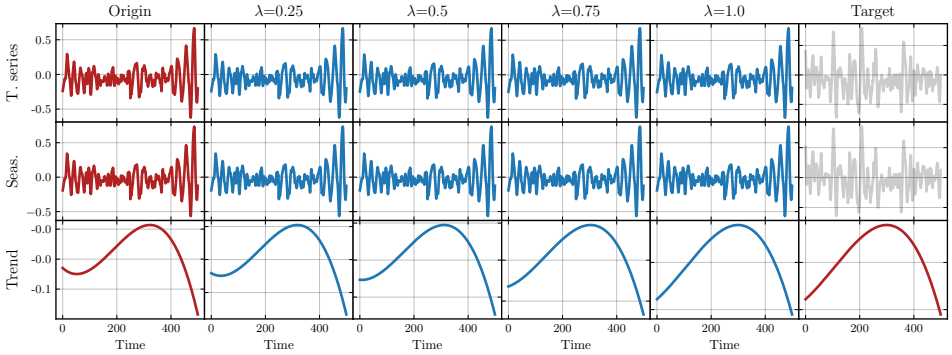


(b) Interpolation in trend, leaving seasonality fixed.

Figure 15: Linear interpolation in seasonality (a) and trend (b) between two time series from the `Energy` dataset, leaving trend and seasonality fixed, respectively. In red are the original time series (first column), and target seasonality/trend (last column).

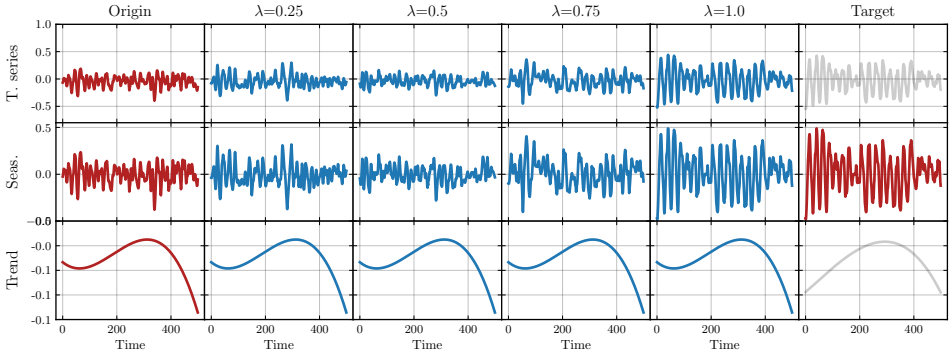


(a) Interpolation in seasonality, leaving trend fixed.

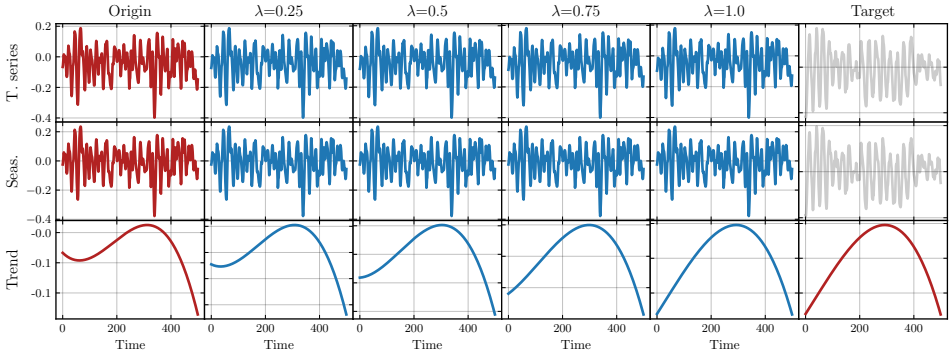


(b) Interpolation in trend, leaving seasonality fixed.

Figure 16: Linear interpolation in seasonality (a) and trend (b) between two time series from the `FordA` dataset, leaving trend and seasonality fixed, respectively. In red are the original time series (first column), and target seasonality/trend (last column).



(a) Interpolation in seasonality, leaving trend fixed.



(b) Interpolation in trend, leaving seasonality fixed.

Figure 17: Linear interpolation in seasonality (a) and trend (b) between two time series from the `FordA` dataset, leaving trend and seasonality fixed, respectively. In red are the original time series (first column), and target seasonality/trend (last column).

REFERENCES

Ahmed Alaa, Alex James Chan, and Mihaela van der Schaar. Generative time-series modeling with fourier flows. In *International Conference on Learning Representations*, 2021.

A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31:606–660, 2017.

Kasun Bandara, Rob J. Hyndman, and C. Bergmeir. Mstl: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. *International Journal of Operational Research*, 2022.

Nuri Benbarka, Timon Höfer, Hamd ul Moqeet Riaz, and Andreas Zell. Seeing implicit neural representations as fourier series. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2283–2292, 2022.

Andrew Brock, Theo Lim, J.M. Ritchie, and Nick Weston. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rydeCEhs-.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8628–8638, 2021.

Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition procedure based on loess (with discussion). *Journal of Official Statistics*, 6: 3–73, 1990.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1824–1833, 10 2019. doi: 10.1109/ICCV.2019.00191.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Tamar Rott Shaham, Michael Gharbi, Richard Zhang, Eli Shechtman, and Tomer Michaeli. Spatially-adaptive pixelwise networks for fast image translation. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

Jacob Russin Russin, Randall O'Reilly, and Yoshua Bengio Bengio. Deep learning needs a prefrontal cortex. In *Bridging AI and Cognitive Science ICLR 2020 Workshop*, 2020.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJgklhAcK7.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. In *Proc. NeurIPS*, 2020a.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020b.

Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10753–10764, June 2021.

Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life*, 15(2):185–212, 04 2009. ISSN 1064-5462. doi: 10.1162/artl.2009.15.2.15202. URL https://doi.org/10.1162/artl.2009.15.2.15202.

Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. Neural brdf representation and importance sampling. *Computer Graphics Forum*, 40(6):332–346, 2021. doi: https://doi.org/10.1111/cgf.14335. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14335.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021.

Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgwNerKvB.

Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. Robuststl: A robust seasonal-trend decomposition algorithm for long time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5409–5416, Jul. 2019.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *NeurIPS*, 2019.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rkgW0oA9FX.

Dominic Zhao, Seijin Kobayashi, João Sacramento, and Johannes von Oswald. Meta-learning via hypernetworks. In *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*. NeurIPS, 2020. doi: 10.3929/ethz-b-000465883.