HyperTime: Implicit Neural Representations for Time Series –Supplementary Material –

Anonymous Author(s)
Affiliation
Address
email

1 Datasets

We use univariate and multivariate time series datasets from the UCR archive [2]. We selected four univariate and four multivariate datasets with different characteristics, which are summarized in Table 1. Additionally, in Section 4.3, we use two datasets that were used in Fourier Flows [1], Google stocks data and UCI Energy data, which we downloaded from the project's Github repository.

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
DuckDuckGeese	50	270	1345	
MotorImagery	278	3000	64	
PhonemeSpectra	3315	217	11	

2 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

Submitted to NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research. Do not distribute.

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 Reconstruction

Figure 1 shows the comparison of the losses for the univariate datasets encoded with implicit networks with different activation functions ans Figure 2 shows the reconstruction of random samples of each univariate dataset using an INR with Sine activation (SIREN).

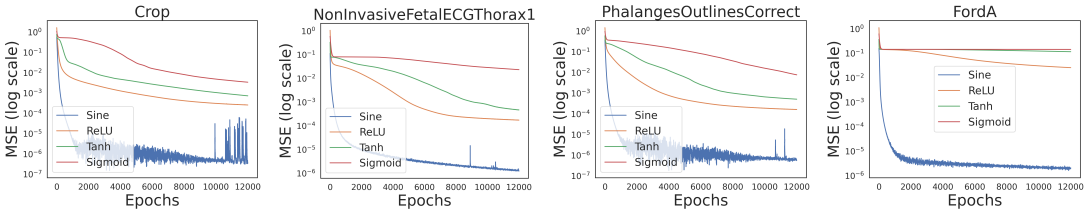

Figure 1: Comparison of implicit networks using different activation functions for univariate datasets.

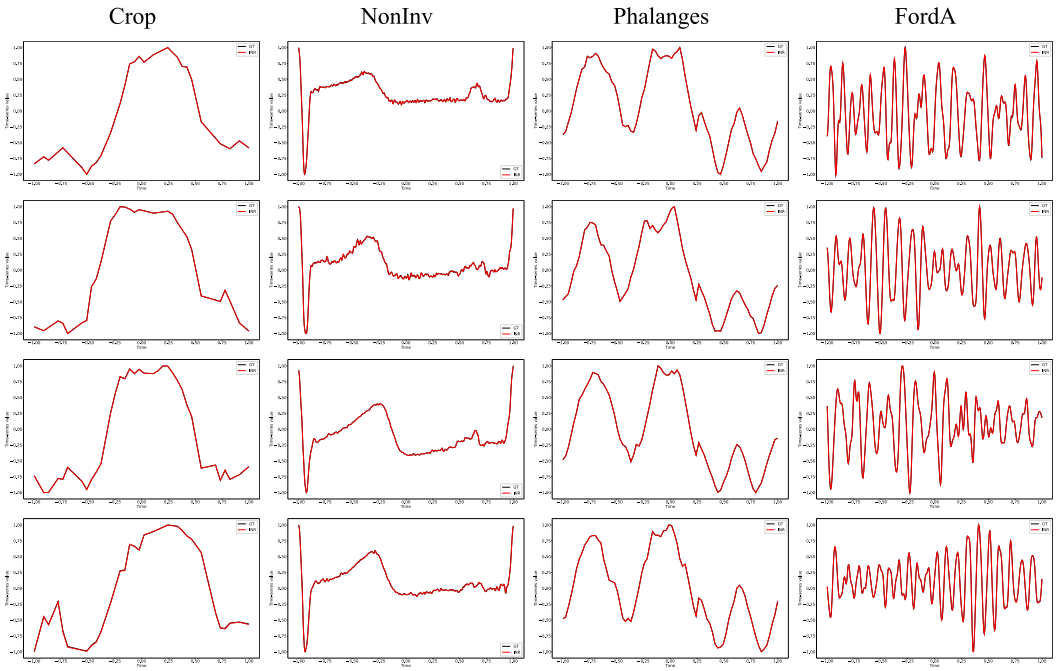

Figure 2: SIREN reconstructions of random samples from each univariate dataset.

Additionally, Figure 3 shows the comparison of implicit networks using different activation functions for the multivariate datasets (in log scale), and Figure 4 shows the reconstruction of random samples of the multivariate dataset using SIREN. We do not show DuckDuckGeese because it contains over a 1000 features.

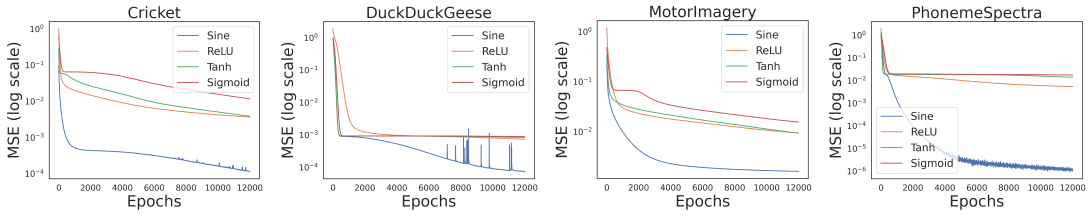

Figure 3: Comparison of implicit networks using different activation functions for multivariate datasets.

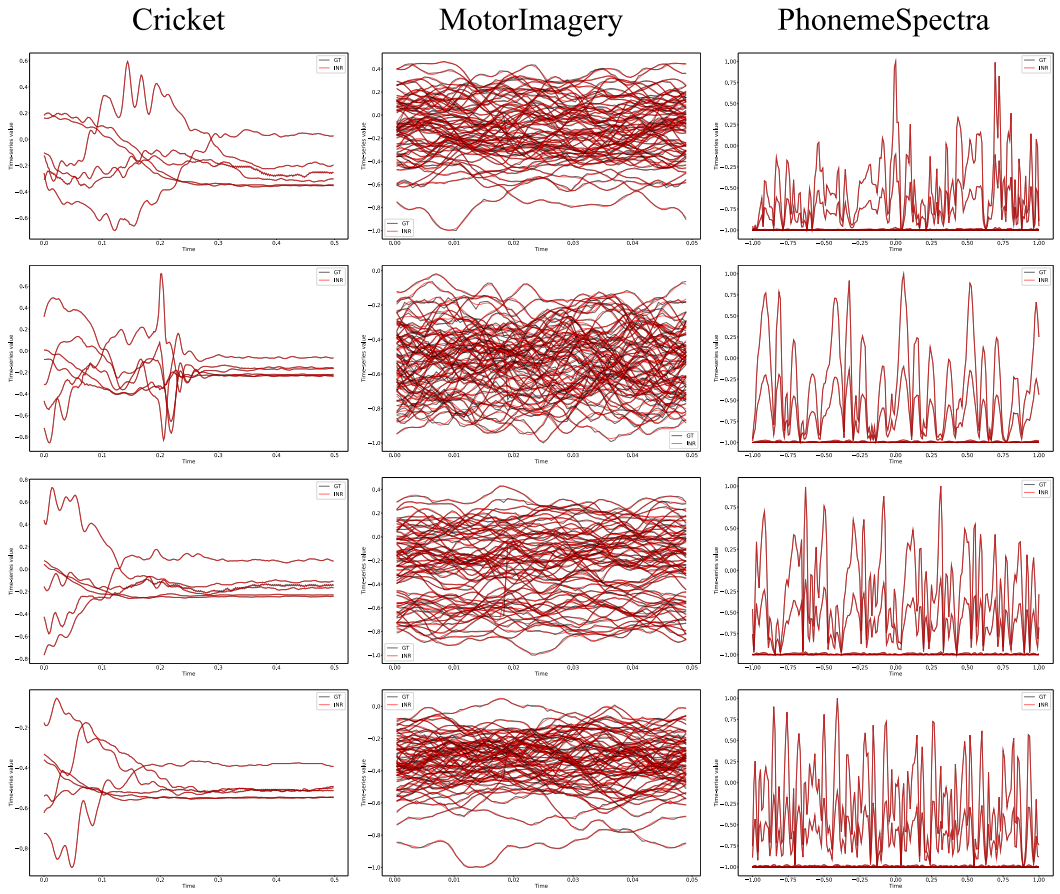

Figure 4: INR reconstructions of random samples from three multivariate datasets.

3.1 Implementation & Reproducibility Details

We use a two-layer MLP architecture with 60 neurons for all experiments. For the univariate datasets we sample 300 time series from the available training samples or the maximum number of time series if the number is below 300. We train for 12000 epochs using Adam optimizer with a learning rate of 1e-4. With this configuration, it takes approximately 180 minutes to train 300 time series on a *g4dn.2xlarge* AWS instance with a NVIDIA T4 GPU.

3.2 Code

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

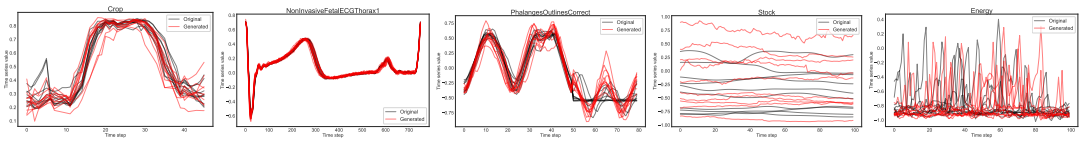

Figure 5: Random samples of generated time series using HyperTime, compared to random samples of original time series.

4 Time Series Generation

Figure 5 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.

4.1 Baselines

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

- Fourier Flows [1]: `https://github.com/ahmedmalaa/Fourier-flows`
- TimeGAN [3]: `https://github.com/jsyoon0823/TimeGAN`

4.2 Implementation & Reproducibility Details

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 $40 \times 128 \times 7500$. The output of the hypernetwork is a one-dimensional vector that contains the network weights of a SIREN with input dimension 1 and 3 hidden layers of 60 neurons.

For the Phalanges and NonInv datasets we train for 300 epochs, with Adam optimizer and a learning rate of $5e-5$. The training takes approximately 210 and 110 seconds for each dataset, respectively. For Crop, Stock and Energy, we train for 1000 epochs, and the training times are approximately 6 min for the Crop and Stock dataset and 30 min for the Energy dataset. All experiments were run on a *g4dn.2xlarge* AWS instance with a NVIDIA T4 GPU.

References

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

[2] 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.

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