Comparison of LSTM, GRU and Reservoir

Tsuyoshi Yoneda
Hitotsubashi University

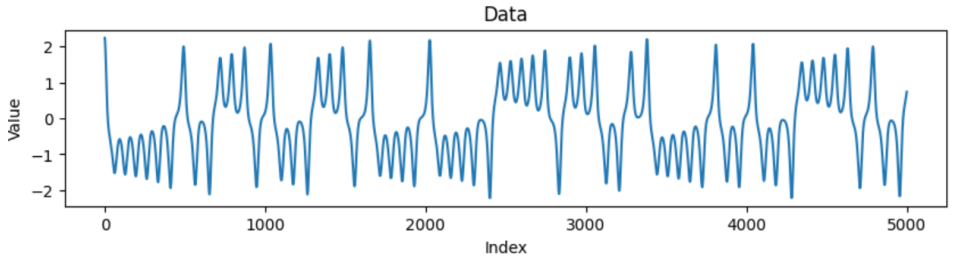
The foundation of modern AI: Backpropagation
$\Rightarrow$ Vanishing gradient problem

- Hochreiter-Schmidhuber (1997)
  Solved the vanishing gradient problem in RNNs by introducing LSTM
- Mikolov et al. (2010–2013), Sutskever-Vinyals-Le (2014)
  LSTM began to be applied to natural language generation
- However, since 2017, Transformers (Vaswani et al., 2017) have become dominant, and LSTMs are no longer at the forefront

Personal perspectives and questions

- My research topic: Forecasting of stock price data and weather data
- Small-scale datasets (less than 10,000 samples)
- For small datasets, is it really necessary to pursue the "context-level dependencies" that LSTMs and Transformers are good at?

Experimental setup for model comparison (1D Lorenz data)



Parameters fixed for fair accuracy comparison:

- Number of trainable parameters: 20,000
  (nodes: Reservoir ~ 150, LSTM ~ 70, GRU ~ 80)
- Training data size: 5,000
- Prediction horizon: 100 steps ahead
- L^2 regularization parameter: 0.0001

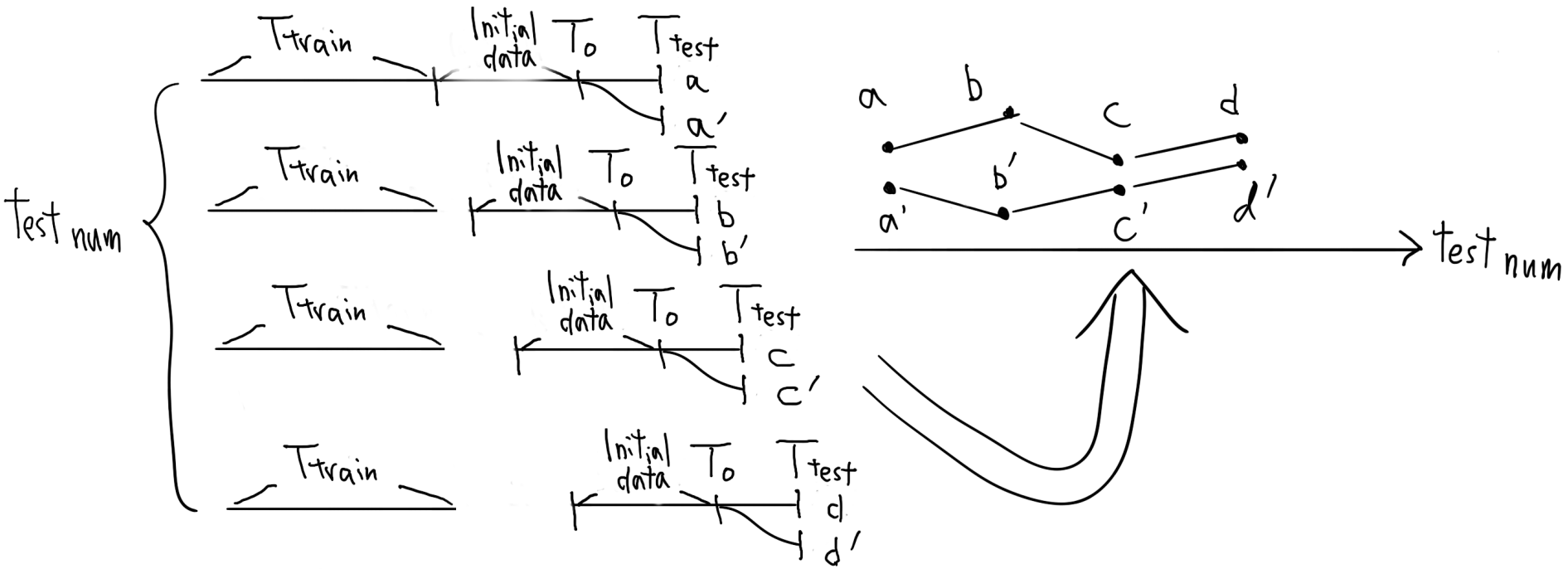
A low-spec machine on Google Colab (free version) is sufficient!

Learning models for accuracy comparison

- Standard LSTM implemented with TensorFlow (Adam optimizer for gradient descent)
- LSTM with a novel gradient descent method without backpropagation
- GRU with a novel gradient descent method without backpropagation

Algorithm for model performance comparison (ensuring robustness)

- Perform predictions 3000 times (test num), each shifted by one step, and compute the MAE (mean absolute error).
- Repeat the same process until the average MAE stabilizes.



The key point of my scheme

Delay Coordinates (see for e.g. Nakai-Saiki '21)

$d_t = (data(t), data(t-\tau), \cdots, data(t-(M-1)\tau))$

We train by using this d_t.

TensorFlow-LSTM: $\tau = 1$, M is chosen prior to training

Proposed Gradient Descent: τ and M are also learned during training

Sigmoid function h, $\odot$: element-wise product

LSTM

$$
\begin{aligned}
\text{Input gate} \quad & i_t = h(W_{datain} d_t + W_{gatein} x_{t-1} + bias), \\
\text{Forget gate} \quad & f_t = h(W_{dataforget} d_t + W_{forget} x_{t-1} + bias), \\
\text{Output gate} \quad & o_t = h(W_{dataout} d_t + W_{gateout} x_{t-1} + bias), \\
\text{Memory cell} \quad & c_t = i_t \odot \tanh(W_{datacell} d_t + W_{cell} c_{t-1} + bias) + f_t \odot c_{t-1}, \\
\text{Output} \quad & x_t = o_t \odot \tanh(c_t), \quad \widehat{d}_{t+1} = W_{out} x_t
\end{aligned}
$$

Grad descent without Backpropagation (Bayes Opt)

Random matrices

$W_{datain}, W_{dataforget}, W_{dataout}, W_{datacell}, W_{gatein}, W_{forget}, W_{gateout}, W_{cell}$

By ridge regression (convexity), we first determine the following W_{out}:

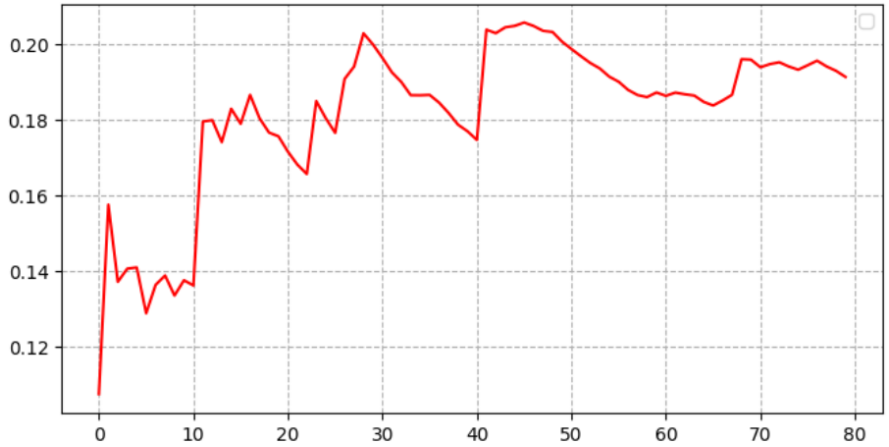
$$d_{t+1} \simeq W_{out} x_t \qquad \text{(the idea of the conventional Reservoir)}.$$

By exploiting convexity (matrix computation), W_{out} is obtained. Note that we are not directly measuring the difference between d_{t+1} and the estimate $\widehat{d}_{t+1}$ so far. In the hyper-parameter space, we apply Bayesian Optimization (Optuna) to minimize the following objective function (RMSE):

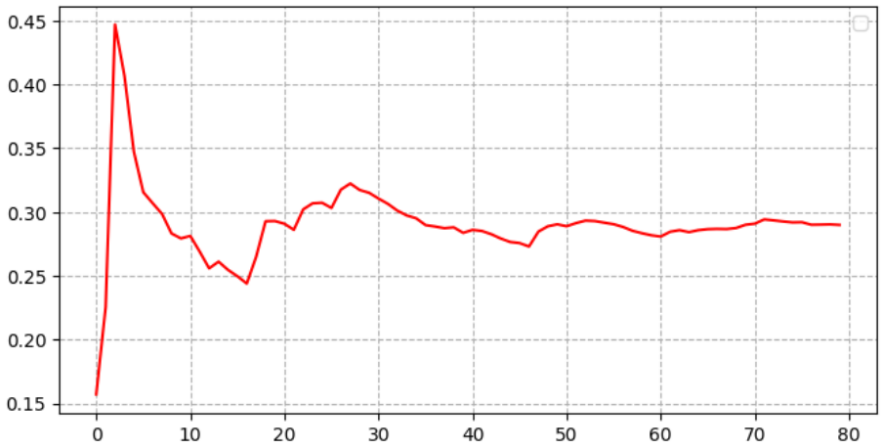
$$\left(\frac{1}{T} \sum_{t=0}^{T-1} |d_{t+1} - W_{out} x_t|^2 \right)^{1/2}.$$

Note: It is necessary to implement coding such that the random matrices vary continuously with continuous changes in the seed value!

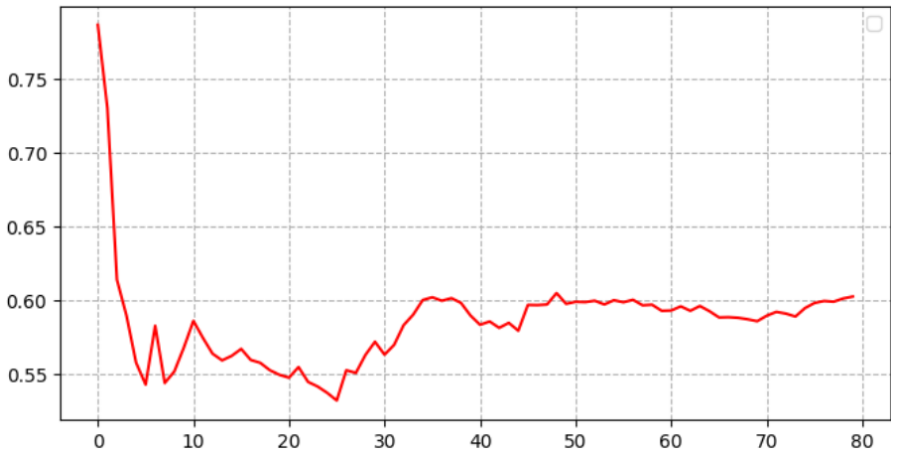
Results (GRU-Bayes, LSTM-Bayes, LSTM-TensorFlow)



GRU-Bayesian Optimization: MAE 0.19, 1h 30min



LSTM-Bayesian Opt: MAE 0.29,
1h 44min



LSTM-TensorFlow: MAE 0.60, 2h 59min

Additional Insights

Data Characteristics

ADF test, p-value: 0.00: The Lorenz data is statistically quite stationary
$\Rightarrow$ Even lighter NN architectures may yield good prediction results.
$\Rightarrow$ propose Fast-lightweight LSTM

Considering an Advanced Version of Bayesian Optimization

$\Rightarrow$ Construct a double-loop training scheme by securing test data from the training dataset for model selection.
$\Rightarrow$ Since double-loop training is computationally heavy, reduce the dataset size to 1500+150 (training + model selection).

Additional models for performance comparison:

- Fast-lightweight LSTM
- Reservoir computing (online) with double-loop training via Bayesian Optimization

Fast-lightweight LSTM (only cell layer) · probabilistic ESP

$$\begin{aligned}
\text{input gate:} \quad & i_t = tanh(W_{datain} d_t) \\
\text{forget gate:} \quad & f_t = h(W_{dataforget} d_t + bias) \\
\text{memory cell:} \quad & c_t = i_t + f_t \odot c_{t-1} \\
\text{Ridge regression:} \quad & d_{t+1} \simeq W_{out} c_t
\end{aligned}$$

$$\text{induction arg:} \quad c_t = \sum_{s=0}^{t} \left(i_s \prod_{r=s+1}^{t} (\odot f_r) \right), \quad \prod_{r=t+1}^{t} f_r := 1.$$

Now assume that the following probability:

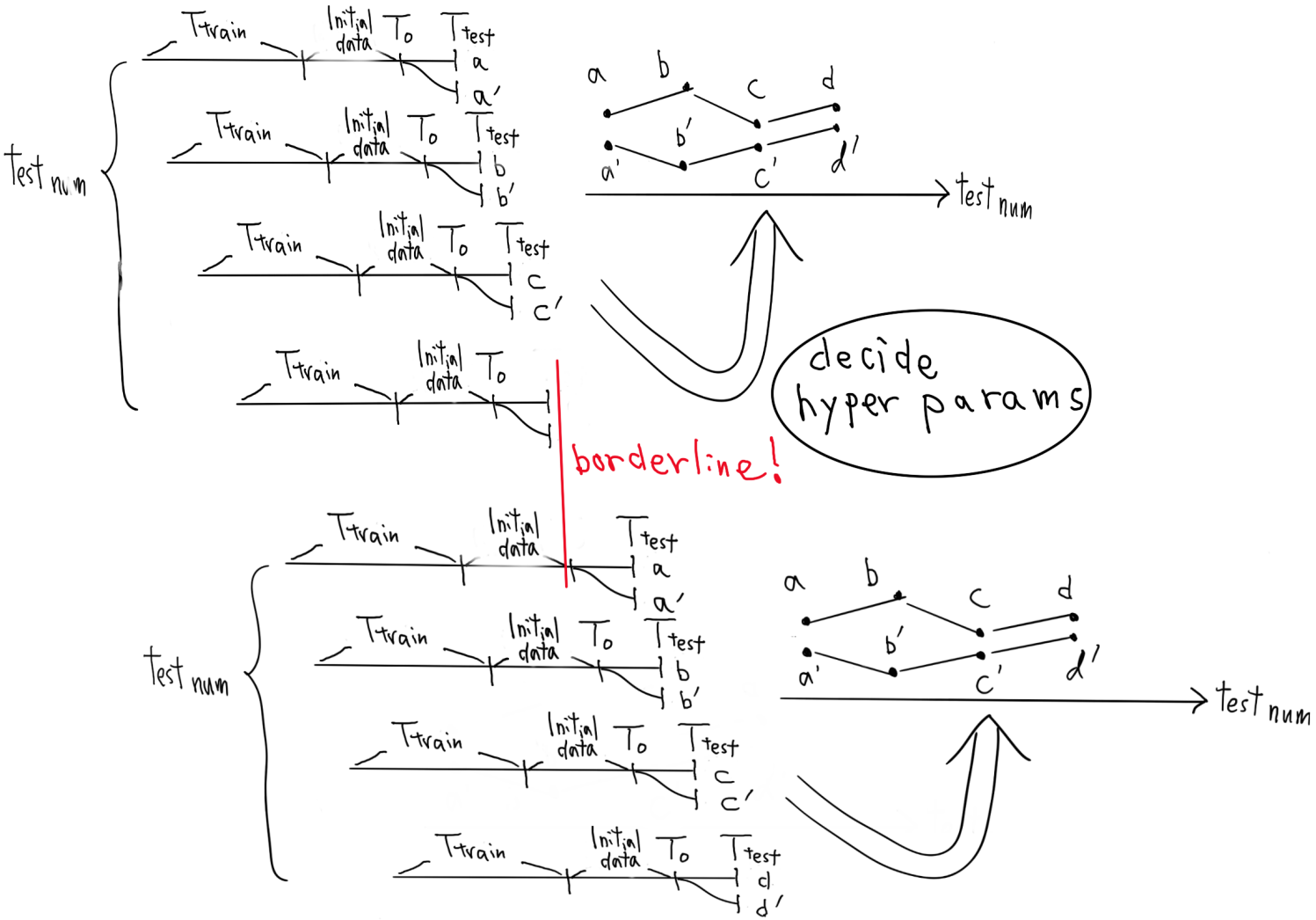
$$\gamma := P(f_r \cdot e_j \neq 0) \quad (0 < \gamma < 1)$$

is independent of j and r, where e_j is the j'th unit vector. Also assume f_r and $f_{r'}$ $(r \neq r')$ are probabilistically independent. Then we have

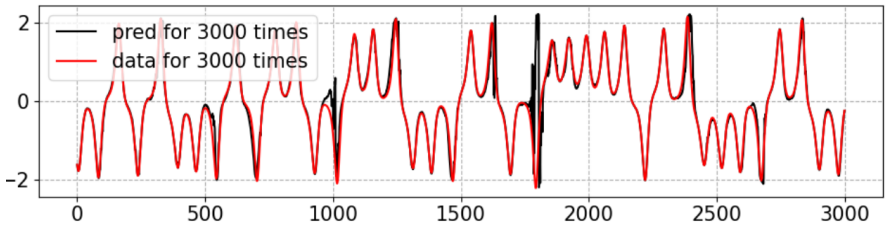
$$P\left(\prod_{r=s+1}^{t} f_r \cdot e_j \neq 0 \right) = \prod_{r=s+1}^{t} P(f_r \cdot e_j \neq 0) = \gamma^{t-s} \to 0 \quad (s \ll t \to \infty).$$

Double-Loop Learning (Online)

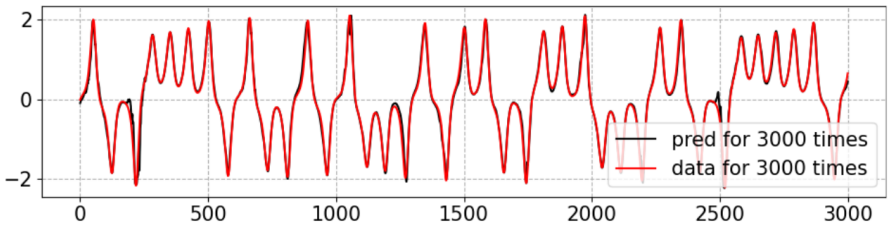
The first loop: Ridge regression, The second loop: Bayesian optimization using model selection data.



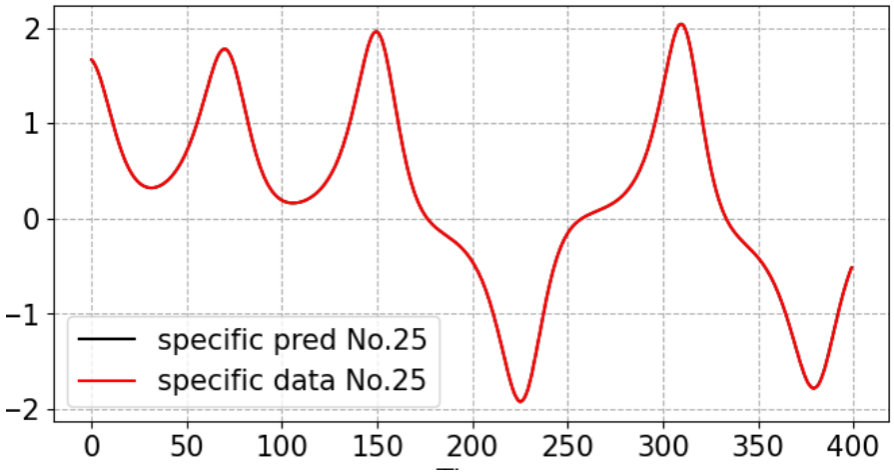
Both the fast-lightweight LSTM and the double-loop online Reservoir computing achieved good results.



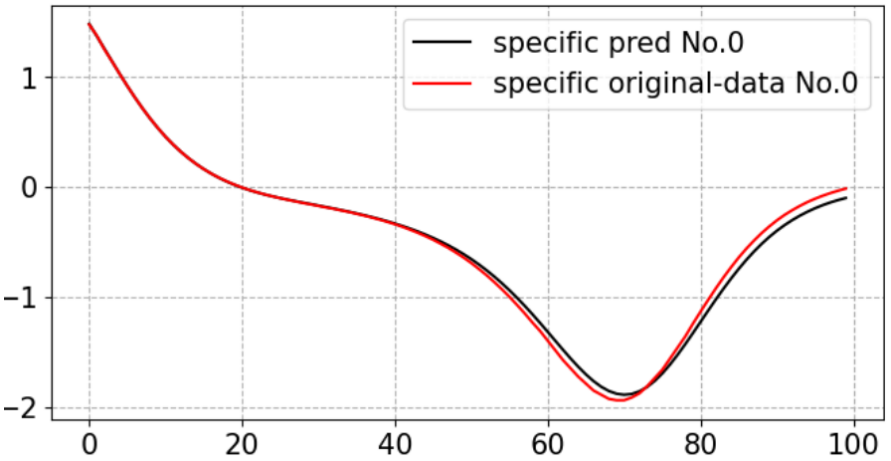
FastLightLSTM: MAE 0.11, prediction 400 steps ahead



Reservoir: MAE 0.067, 1500 training samples

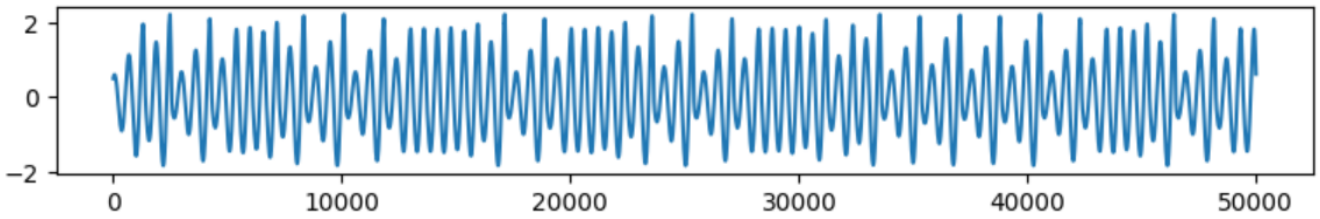


FastLightLSTM: MAE 0.11, prediction 400 steps ahead



Reservoir: MAE 0.067, 1500 training samples

Replication Test: Rössler Equation



- ADF test, p-value: 0.00 $\Rightarrow$ The Rössler data is statistically highly stationary
- Number of trainable parameters: 20,000
  (Reservoir ∼150 nodes, LSTM ∼70, GRU ∼80)
- Training data size: 5,000 (Reservoir uses 1,500+150)
- Prediction: 100 steps ahead (FastLightLSTM: 400 steps ahead)
- L^2 regularization parameter: 0.0001

Learning Results (MAE)

- FastLightweightLSTM: 0.005
  (prediction 400 steps ahead)
- DoubleRoopRes: 0.030
  (1,500 training samples)
- Bayes-LSTM: 0.055
- Bayes-GRU: 0.061
- TensorFlow-LSTM: 0.52

Thank you!

Conclusions:

- For time series prediction with around 5,000 data points, designing an architecture based on the data characteristics achieves better learning performance than using standard libraries directly.
- Even a simple architecture, such as the fast-lightweight LSTM, can achieve excellent learning results.
- The reservoir with double-loop online learning can achieve strong performance even with only 1,500 + 150 data points (see also Nakai-Saiki 2024).