Dynamical Systems Theory in Machine Learning and Data Science - Final Project

https://github.com/ZiqiuZhou/DST_final_project

Ziqiu Zhou[*]
fo249@stud.uni-heidelberg.de

Christoph Bender[†]
Christoph.Bender@stud.uni-heidelberg.de

March 8, 2022

[*]matriculation number: **3635588**. I need a grade, and I want to pick ex03, ex05, ex06, ex07, ex08, ex09 and ex10 for grading.

[†]matriculation number: **4012810**. I do not want a grade (only a pass and the 6 CP).

1 Summary LSTM [3]

The invention of LSTM was motivated by the regularization of recurrent neural networks (RNNs). In addition to inputs $\boldsymbol{i}_t \in \mathbb{R}^{d_i}$, RNNs use also loops in order to include informations from previous hidden states $\boldsymbol{h}_{t'}$(where $t' < t$) in the calculation of the current state $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$ at time t. The Elman network [1] is for example defined by [1]: However

$$\boldsymbol{h}_t = \sigma_h(\boldsymbol{W}_{hi} \cdot \boldsymbol{i}_t + \boldsymbol{W}_{hh} \cdot \boldsymbol{h}_{\boldsymbol{t-1}} + \boldsymbol{b}_h) \quad (1)$$

$$\boldsymbol{o}_t = \sigma_o(\boldsymbol{W}_{oh} \cdot \boldsymbol{h}_t + \boldsymbol{b}_o) \in \mathbb{R}^{d_o} \quad {}^{2} \quad (2)$$

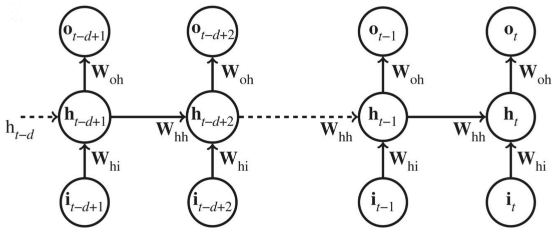


Figure 1: Schematic Sketch of a RNN [3]

RNNs struggle to recognize long-term dependencies and furthermore the gradient can vanish or explode, which also leads to problems. Because of this one uses gates. The corresponding model is called LSTM (Long Short-Term Memory). This is characterised by the following formula:

$$\begin{aligned}
\boldsymbol{g}^f &= \sigma_f(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{i}_t] + \boldsymbol{b}_f) & \boldsymbol{g}_t^i &= \sigma_i(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{i}_t] + \boldsymbol{b}_i) \\
\tilde{\boldsymbol{C}}_t &= \tanh(\boldsymbol{W}_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{i}_t] + \boldsymbol{b}_C) & \boldsymbol{C}_t &= \boldsymbol{g}_t^f \cdot \boldsymbol{C}_{t-1} + \boldsymbol{g}_t^i \cdot \tilde{\boldsymbol{C}}_t \\
\boldsymbol{g}_t^o &= \sigma_h(\boldsymbol{W}_h \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{i}_t] + \boldsymbol{b}_h) & \boldsymbol{h}_t &= \boldsymbol{g}_t^o \cdot \tanh(\boldsymbol{C}_t)
\end{aligned} \tag{3}$$

Whereby $\boldsymbol{g}_\bullet \in \mathbb{R}^{d_h \times (d_h + d_i)}$ represents the gate signal and $\boldsymbol{C}_t \in \mathbb{R}^{d_h}$ the so-called cell state, which is jointly with the hidden state referred as "LSTM states". To map the hidden state to the wanted output space one also uses an additional fully connected layer $\boldsymbol{W}_{oh}$:

$$\boldsymbol{o}_t = \boldsymbol{W}_{oh} \cdot \boldsymbol{h}_t = f^w(\boldsymbol{z}_t, \boldsymbol{h}_{t-1}, \boldsymbol{C}_{t-1}) \approx F^w(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \ldots, \boldsymbol{z}_{t-d+1}) \quad {}^{3} \tag{4}$$

The so-called LSTM cell-function f^w can be rewritten by iterative repition as F^w, where w includes all trainable parameters. In the last step one uses the assumption that d-time steps are sufficient to compute the current output and thus $\boldsymbol{h}_{t-d}, \boldsymbol{C}_{t-d}$ can be omitted. The model is data-driven, one has not to incoperate prior knowledge (like underlying equations) in the system necessarily. Goal of the LSTM is to predict the state derivative $\dot{\boldsymbol{z}}_t$ using a short time memory of the d previous states $\boldsymbol{z}_{t:t-d+1}$. Therefore, the loss $\mathcal{L}$ shall be minimized by searching for the best parameters w^*:

$$w^* = \arg\min_w \mathcal{L}(\{\boldsymbol{z}_{1:T}, \dot{\boldsymbol{z}}_{1:T}\}, w) = \arg\min_w \frac{1}{T-d+1} \sum_{t=d}^{T} ||F^w(\boldsymbol{z}_{t:t-d+1}) - \dot{\boldsymbol{z}}_t||^2 \tag{5}$$

[1] see also `https://en.wikipedia.org/wiki/Recurrent_neural_network#Elman_networks_and_Jordan_networks`

[2] where $\boldsymbol{\sigma}_\bullet$ are activation functions, $\boldsymbol{W}_{*\bullet}$ weight matrices and $\boldsymbol{b}_\bullet$ bias summands.

[3] where in the following $\boldsymbol{z}_t$ is used as the input and describes the system time series.

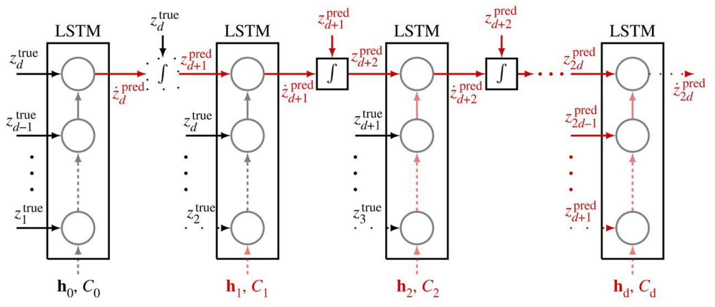


Figure 2: LSTM Model[3] Predictions are marked with red arrows, while black arrows indicate the short-term memory. The known $\boldsymbol{z}_{1:d}^{\text{true}}$ are used to predict $\dot{\boldsymbol{z}}_d^{\text{predict}}$, which can be integrated in order to get the next state $\boldsymbol{z}_{d+1}^{\text{predict}}$.

The group of Vlachas trained the LSTM by using backpropagation. For this, a mini-batch optimization with the Adam method [2] and an initial learning rate $\epsilon_{init} = 10^{-4}$) was applied. The inizialization of the weights is based on the method of Xavier. One problem the group had to face was to fine tune the dimension of the hidden stated d_h. They observed that a small d_h shortened the ability to fit to the time series well, but a big d_h increased the risk of overfitting and the computation time.

In the paper they compared the LSTM Model with the GPR (Gaussian process regression) and MSM (Mean Stochastic Model). Three applications (The Lorenz 96 sytem, the Kuromoto-Sivashinsky equation an a barotropic climate model) were considered in order to evaluate the predictive accuracy of the models. They showed that in all cases the LSTM outperformed the other two models, i.e. the LSTM model was able to catpure the local dynamics more efficient.

2 Power spectrum metric

The second task of the project was to implement the power-spectrum correlation (PSC). This features two hyperparameters: Namely

- the smoothing factor σ, which is the width of the Gaussian kernel smoothing the spectrum
- and the cutoff, which excludes all frequencies above the given frequency threshold from the calculation of the power spectrum correlation.

The code snippets of *psc.py* was added in *global_utils.py* and executed in *rnn_statefull.py*. Furthermore the plotting functions are added in the *plotting_utils.py* file. In order to ensure that the model draws a random initial condition and then generates a time series of length T. We sampled random indices in the data generation and for each prediction the model takes one of these random integers during testing. And starts to predict the time series starting at the chosen starting point of the whole test series.

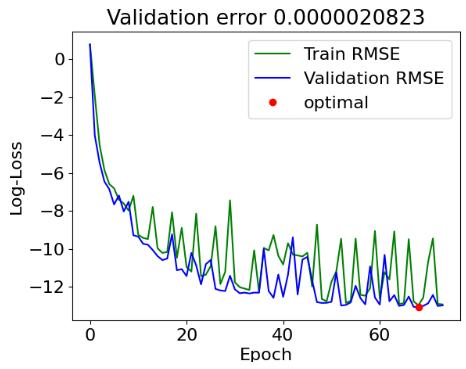
To illustrate the LSTM model and also to make sure whether the LSTM model is working, we first apply the model on a very simple sinosoidal dataset before applying it on the Lorenz datasets (see section 3). For this we generate a time series consisting of 100000 points with time step size $\Delta t = 0.5$ by computing:

$$z_t = A \cdot \sin(2\pi f t + \phi_0) + c \tag{6}$$

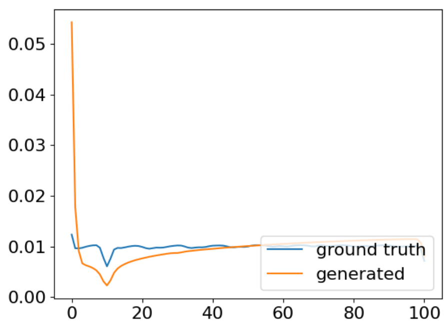
The chosen parameters are:

- amplitude $A = 2$
- frequency $f = 0.1$
- phase $\phi_0^{\text{train}} = 0$, $\phi_0^{\text{test}} = 0.5$
- offset $c = 0.5$

Using the implemented LSTM Model of Vlachas et alium [3][4] one can train the LSTM model and test the LSTM model via the command line. The root mean squared error is chosen to be the loss, which is optimized during training. The loss decreases with increasing epochs for the training data and the validation data as well. This can be seen in fig. 3a. Since validation loss is not increasing one can exclude the existence of overfitting.



(a) Training and validation loss

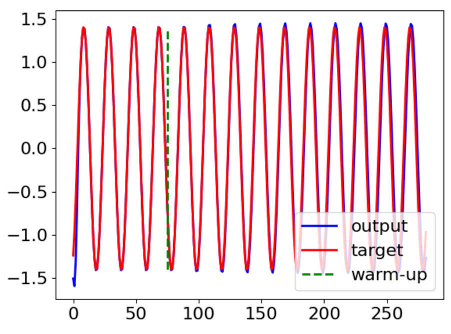


(b) Power spectrum for test predictions

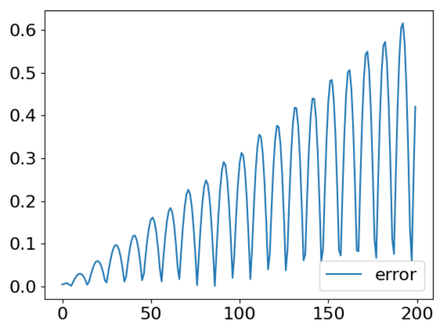
The plotted power spectrum shows a clear negative peak at 10, which corresponds to the chosen frequency of 0.1. The power spectrum of the generated predictions show the same course of the curve as the ground truth, but appearently is shifted a bit. For the power spectrum plot we used a smoothing factor $\sigma = 1$ and a frequency cutoff at 5000.

Furthermore in fig. 4 and fig. 5 one can see the predictions of the LSTM of two test series with random chosen initial conditions and the corresponding errors. One can see that the LSTM can easily extract the sinosoidal behavior of the system and is able to fit to the data very well. However, one can see that the curves are a little bit off. Probably because the regressed frequency is not optimally. This explains why the error is increasing and accumulating the more the model predicts in the future.

[4]`https://github.com/pvlachas/RNN-RC-Chaos`

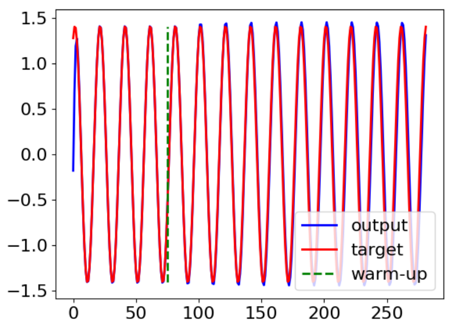


(a) Prediction

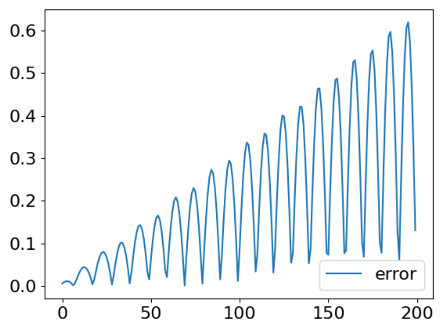


(b) Error

Figure 4: Test results for random initial condition 70417



(a) Prediction



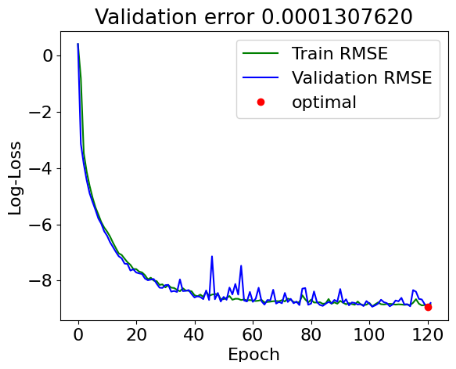
(b) Error

Figure 5: Test results for random initial condition 98804

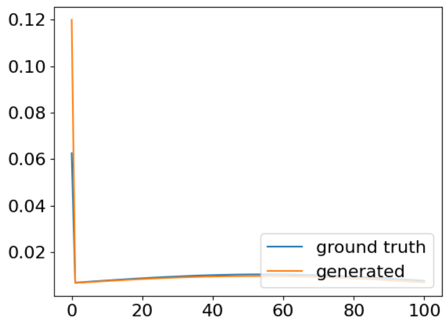
3 Applying LSTM on Lorenz-datasets

The main part of the final project is to apply the LSTM model on the Lorenz-datasets. For this the code of the LSTM is modified and also the data has to be rearranged.

3.1 Lorenz-63



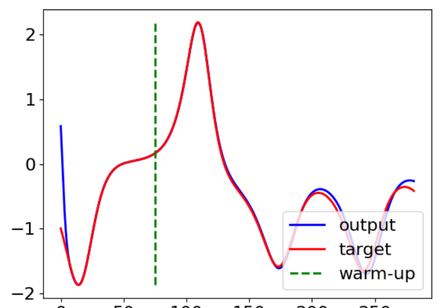
(a) Training and validation loss



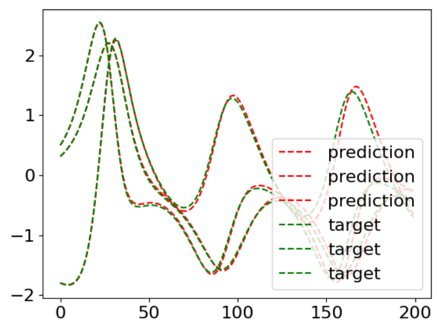
(b) Power spectrum for test predictions

Figure 6: Loss and power spectrum

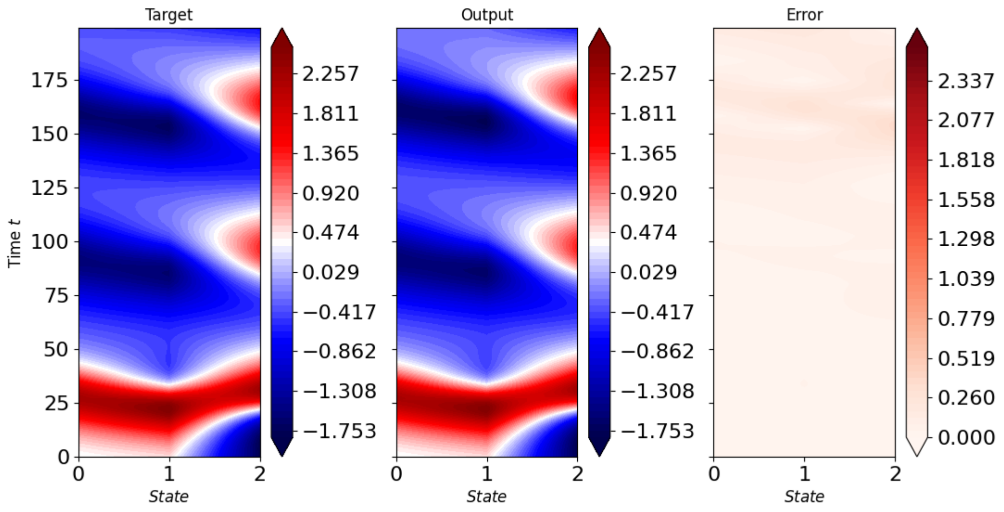
fig. 6a proves that the model learns successfully without overfitting. This can be further evidenced by the fact that the predictions of the test data looks very promising. Even after the warm up, the prediction of the model agrees very well with the ground truth. Only after about 100 predicition steps does the difference become apparent. This same behaviour is also observed with the other dimensions. the agreement of the three-dimensional prediction with the ground truth is visually illustrated by the contour plots (fig. 7 and fig. 5). For the power spectrum we use a smoothing factor $\sigma = 50$ and a cutoff-frequency $f_{\text{cut}} = 5000$. The spectra also seems to fit well, however it is hard to see the differences due to the peak at low frequencies.



(a) Prediction of first dimension

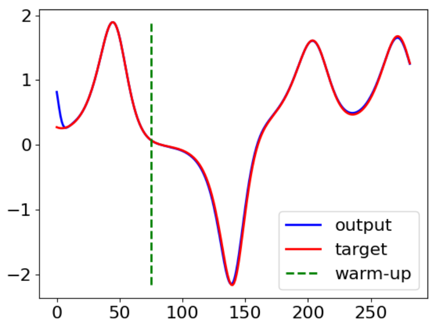


(b) Predictions of all dimensions

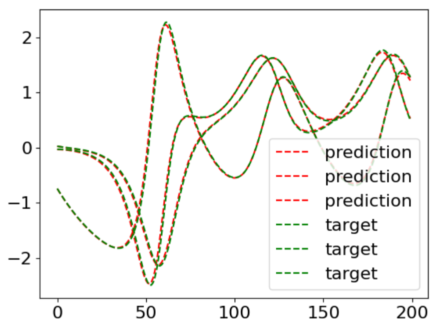


(c) Contour plot

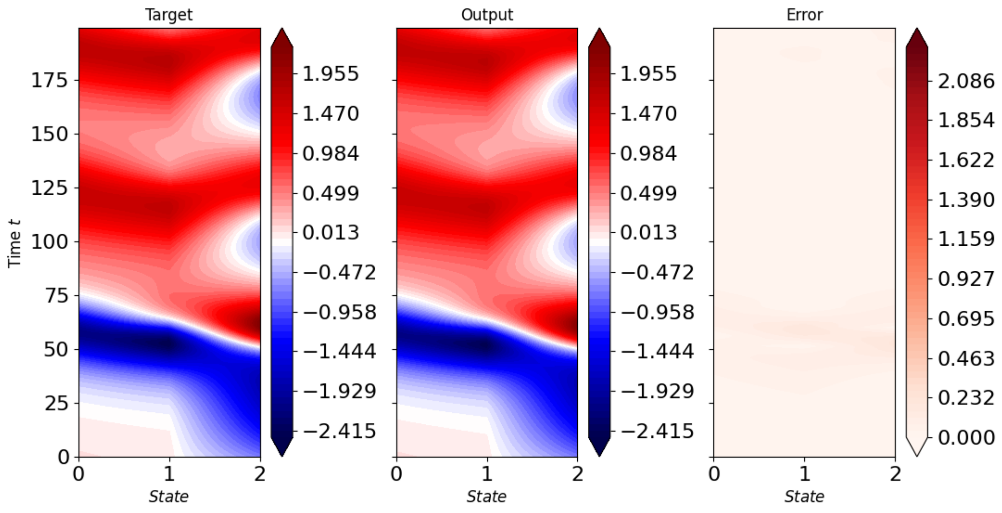
Figure 7: Test results for random initial condition 33278



(a) Prediction of first dimension



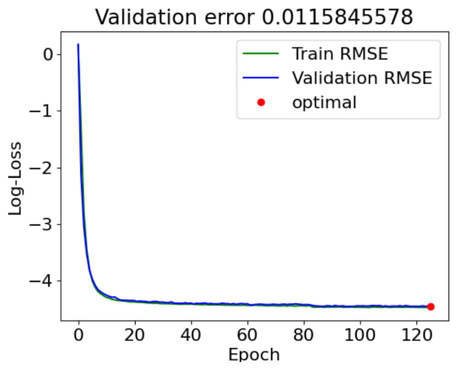
(b) Predictions of all dimensions



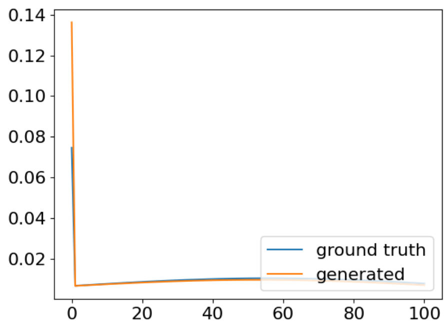
(c) Contour plot

Figure 8: Test results for random initial condition 45336

3.2 Lorenz-96



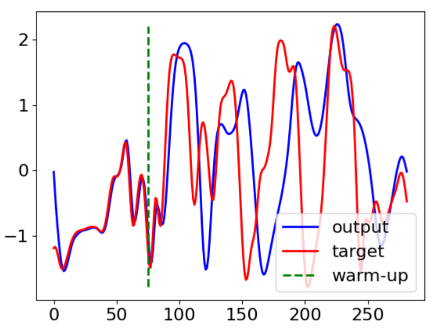
(a) Training and validation loss



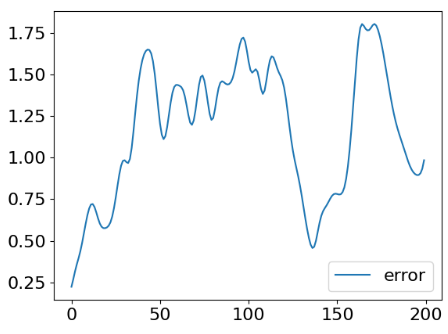
(b) Power spectrum for test predictions

Figure 9: Loss and power spectrum

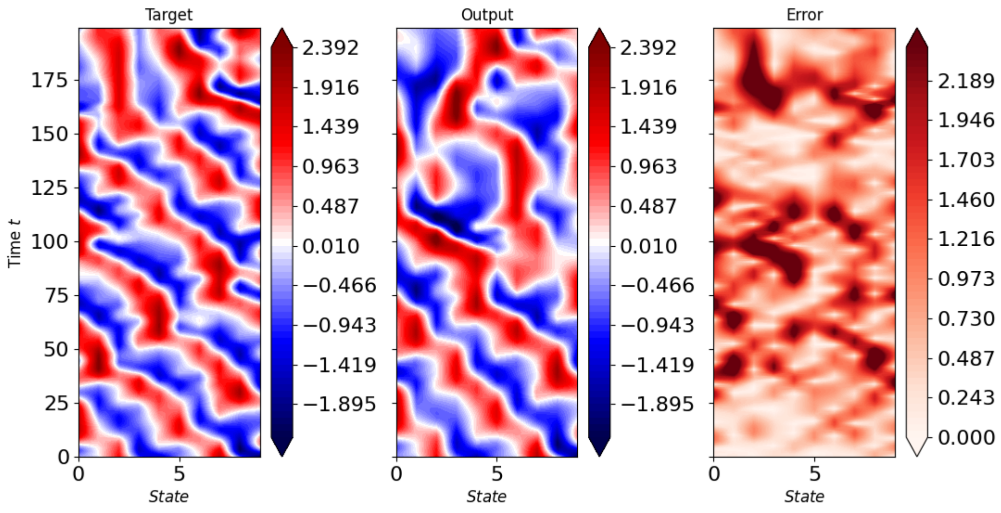
Again fig. 9a proves that the model learns and again no overfitting can be detected. However learning the Lorenz-96 proves to be significantly more difficult. This is to be expected, as this dataset is 10-dimensional, among other things. We therefore searched a lot for goot hyperparameters and also improved the learning rate scheduler, but the training always converged at a validation error of about 0.0011. The model's limited ability to predict the system is substantiated by the quality o the predictions of the test data. One can clearly see that only the first steps are correct but than prediction is off and evolves different. However, the Paper of Vlachas et alium, suggested that the LSTM should be able to fit to the Lorenz-96 system. This could be because we did not find the perfect hyperparameter. Futhermore, they used the stateless LSTM model while we used the statefull LSTM model. They also tried other models like the MSM or the combination MSM-LSTM. These models could eventually lead to better results. For the power spectrum we again use a smoothing factor $\sigma = 50$ and a cutoff-frequency $f_{\text{cut}} = 5000$. The spectra seems to fit well, but the differences are not clearly visible due to the peak at low frequencies.



(a) Prediction of first dimension

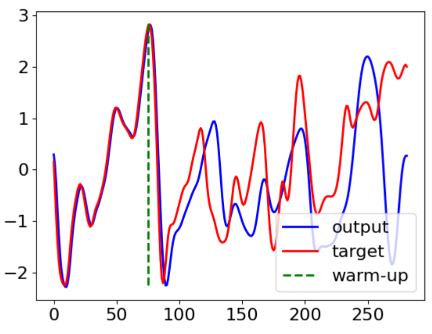


(b) Error of first dimension

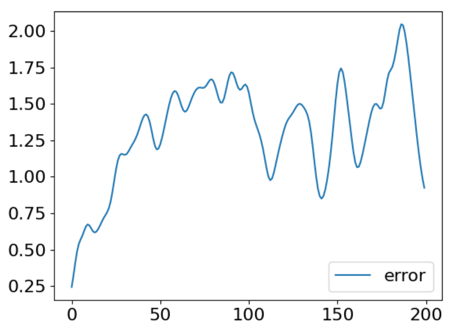


(c) Contour plot

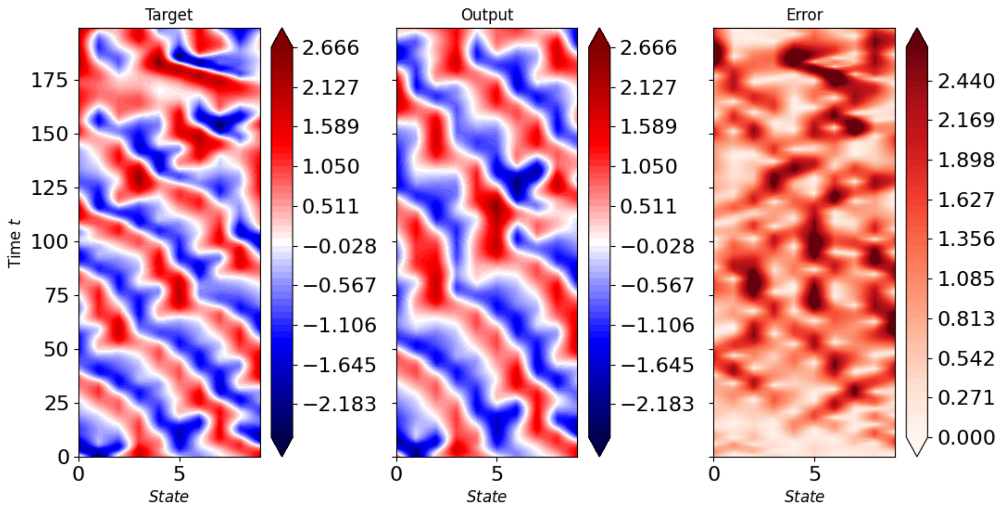
Figure 10: Test results for random initial condition 25172



(a) Prediction of first dimension



(b) Error of first dimension



(c) Contour plot

Figure 11: Test results for random initial condition 50168

References

[1] Jeffrey L. Elman. "Finding Structure in Time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. DOI: https://doi.org/10.1207/s15516709cog1402_1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1402_1. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1.

[2] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: 1412.6980 [cs.LG].

[3] Pantelis R. Vlachas et al. "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2213 (2018), p. 20170844. DOI: 10.1098/rspa.2017.0844. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2017.0844. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2017.0844.