# Machine Learning in Climate Science Research Project
# How much Data do S2S-Neural-Networks need?
# An ENSO Showcase

Felix Bötte

October 3, 2023

## Abstract

The importance of ENSO in global climate science is significant, yet its irregularity poses a prediction challenge, deeply impacting climate research and risk management. This academic report works towards a comprehensive exploration of ENSO forecasting by integrating Sequence-to-Sequence Prediction Neural Networks and addressing data sparsity concerns through synthetic data generation by integrating the Linear Inverse Model. These methodologies aspire to advance the field of ENSO prediction and contribute to a deeper understanding of this critical climatic phenomenon. The findings from this study reveal that the LIM effectively generates synthetic data mirroring the original data's temporal dynamics. Expanding the training data for neural network models by a factor of 3 using synthetic data results in performance enhancement. Further increase shows no significant benefit. Additionally, the complexity and type of recurrent model for prediction do not exhibit significant improvements over a vanilla LSTM. Nonetheless, selecting the optimal prediction horizon during training significantly impacts performance, with larger horizons generally yielding favorable outcomes for the models.

## 1 Introduction

The El Niño-Southern Oscillation (ENSO) phenomenon is a primary driver of climate variability worldwide and holds great significance in the realm of climate science. Accurate forecasts would enable anticipating and managing climate-related risks, ranging from extreme weather events to agricultural productivity. However, the inherent challenge in ENSO prediction lies in the irregularity of the event which occurs only every 2-7 years with an average of around 3 years [1]. The sparsity of available data results in a significant challenge when trying to study and forecast its pattern. The first data available dates back to around 1950 until today, effectively providing only a very limited number of ENSO events. Historically, the Linear Inverse Model (LIM) proposed by Penland et. al. [2] has served as a benchmark for ENSO prediction. It assumes a linear relationship between the observed data and the underlying parameters. It is based on covariance statistics that estimates the best-fit linear Markov process to the data, utilizing stochastic differential equations. The dynamical assumption of stable linearity is implicit in the method. Nowadays, machine learning approaches in climate and weather prediction are becoming more popular due to the ever increasing amount of data available data that can be utilized. Nevertheless, the efficacy of such models is compromised when confronted with limited and sparse data. Consequently, there is a need to explore innovative approaches for improving ENSO forecasting capabilities.
The following project addresses the problem of data sparsity by generating synthetic data through integration of LIM, aiming to simulate realistic ENSO dynamics. This approach enables us to vary the

number of available data points for training neural networks while keeping the inherently stochastic ENSO dynamics. The primary type of machine learning algorithm employed are Sequence-to-Sequence neural networks with the aim of utilizing the hidden states of the model in order to capture the underlying temporal dynamics of a sequence of data points. Here we use the principal components of sea surface height and temperature of the pre-industrial control runs of the Community Earth System Model 2.

# 2 Methods

## 2.1 Data

The following study is based on data from the Community Earth System Model 2 (CESM2) which refers to a specific type of general circulation model (GCM) model, designed to represent the pre-industrial control state of the Earth‘s climate system [3]. In piControl simulations, the external forcings, such as greenhouse gas concentrations and aerosols, are held constant at pre-industrial levels. This provides a baseline for understanding how the Earth's climate system would behave in the absence of human-induced changes.
In this section, I outline the data preprocessing steps undertaken for the analysis of the CESM2 PiControl run data, with a specific focus on the natural variability of sea surface temperature (SST) and sea surface height (SSH). The data preprocessing pipeline includes principal component analysis (PCA), generation of empirical orthogonal functions (EOFs) plots and time series for validating correctness, data normalization, and the selection of the region of interest for the study of the El Niño-Southern Oscillation (ENSO) phenomenon. Additionally, the resulting data is split into a train-/validation-/ and test-set for conducting the following experiments. All results are performed on the test-set.

### 2.1.1 Selection of ENSO Region

To focus on the ENSO phenomenon, we sliced the data to the region spanning 130°E to 70°W and 30°S to 30°N. This region encompasses the central and eastern tropical Pacific, where ENSO events are most prominent. Additionally a land-sea mask was applied to filter out any part of the respective region that represents mainland as illustrated in Figure 1.

### 2.1.2 Principal Component Analysis (PCA)

PCA was applied to both SST and SSH datasets separately to reduce their dimensionality and capture the dominant modes of variability. The PCA process involves calculating the covariance matrix and then performing eigendecomposition to obtain the PCs and their corresponding eigenvalues [4]. The formula for PCA can be expressed as:

$$\mathbf{X} = \mathbf{USV}^T$$

Where: $\mathbf{X}$ represents the original data matrix. $\mathbf{U}$ contains the eigenvectors (principal components). $\mathbf{S}$ is a diagonal matrix with the square root of the eigenvalues. $\mathbf{V}^T$ represents the transpose of the loading matrix. For SST, we retained the top 20 PCs, while for SSH, we retained the top 10 PCs. These numbers were determined based on the cumulative explained variance to capture a significant portion of the dataset's variability. Afterwards, the PCs are concatenated, resulting in a 30-dimensional dataset.
After selecting the PCs, we generated Empirical Orthogonal Functions (EOFs) plots and time series to visually inspect the spatial patterns and temporal behavior of the leading modes of variability in SST and SSH data [5].

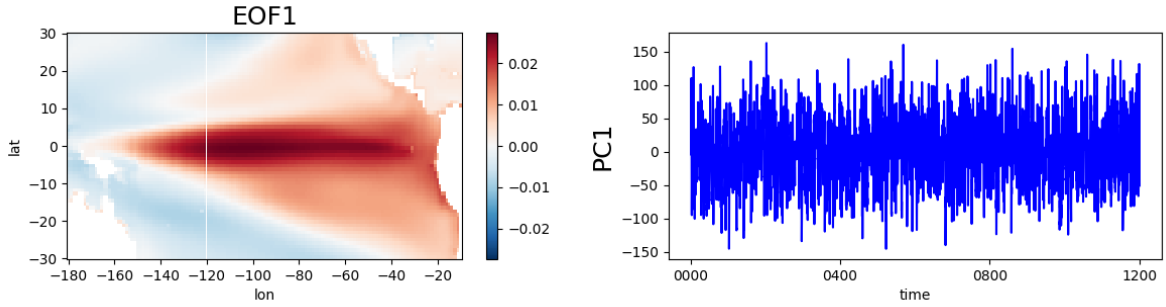



Figure 1: EOF of first principal component with stationary timeseries of the corresponding PC.

### 2.1.3 Normalization

Both SST and SSH datasets were normalized separately for each dimension to have zero mean and unit variance. The so called z-normalization formula is given by:

$$\hat{x} = \frac{x - \mu}{\sigma}$$

Where: $\hat{\mathbf{x}}$ represents the normalized data. $\mathbf{x}$ is the original data. $\mu$ is the mean of the data. $\sigma$ is the standard deviation of the data.

## 2.2 Linear Inverse Model

The Linear Inverse Model (LIM) serves as a mathematical framework employed for estimating the Linear Operator from raw data which is then used to make analytical forecast and integration from a given starting point.

### 2.2.1 Green's Function Estimation & Logarithmic Matrix & Noise Covariance Matrix

The fitting procedure begins with the preparation of the raw data. We segment this data into two distinct components: $x$ and $x_\tau$, to ensure that their dimensions are aligned consistently. Specifically, $x$ encompasses the data from the initial segment, while $x_\tau$ represents data delayed by a time lag $\tau$.
The next step involves the computation of two covariance matrices: $C_0$ and $C_\tau$. These matrices are calculated using the following expressions:

$$\mathbf{C_0} = \frac{xx^T}{n_{\text{time}}}, \mathbf{C}_\tau = \frac{x_\tau x^T}{n_{\text{time}}}$$

Where $n_{\text{time}}$ refers to the number of time points. This parameter is determined as the difference between the total length of the data and the time lag $\tau$. The estimation of Green's function is achieved by applying the time-evolution operator, $C_\tau$, to the inverse of $C_0$:

$$\mathbf{G}_\tau = \mathbf{C}_\tau \cdot \mathbf{C_0^{-1}}$$

The Green's function encapsulates the dynamic information we aim to extract from the raw data. The following operation necessitates eigenvalue decomposition of the Green's function, resulting in eigenvalues $\mathbf{U}$ and eigenvectors $\mathbf{V}$.

Subsequently, we compute the logarithmic matrix as:

$$\mathbf{G} = \mathbf{U} \wedge \mathbf{U^{-1}}, \mathbf{L} = \log(\mathbf{G})/\tau$$

In order to estimate the noise covariance, we make use of the stationary assumption and employ the following equation:

$$0 = \mathbf{L} \cdot \mathbf{C_0} + \mathbf{C_0} \cdot \mathbf{L}^T + \mathbf{Q}$$

This equation allows us to compute the estimated noise covariance matrix $\mathbf{Q}$:

$$\mathbf{Q} = -\mathbf{L} \cdot \mathbf{C_0} - \mathbf{C_0} \cdot \mathbf{L}^T$$

### 2.2.2 Analytical Forecast

In this section, we describe the process of forecasting data using the time-evolution operator $\mathbf{L}$. The objective is to forecast the data at a future time point $t + \tau$ using the time-evolution operator $\mathbf{L}$: Here, $\mathbf{X}(t)$ represents the data at time $t$, and the operator $\exp(\mathbf{L}\tau)$ captures the system's dynamics over a time lag $\tau$.

### 2.2.3 Euler's Method Integration

In this section, we present an iterative numerical approach for solving a Stochastic Differential Equation (SDE) utilizing Euler's method. To achieve the numerical solution, we employ Euler's method. We discretize the time interval $[0, T]$ into a series of time steps.
At each time step $i$ we update using the following formula:

$$\mathbf{X}[i+1] = \mathbf{X}[i] + dt \cdot \mathbf{L} \cdot \mathbf{X}[i] + \sqrt{dt} \cdot S \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}), \tag{1}$$

Here, $\mathbf{L} \cdot \mathbf{X}[i]$ represents the deterministic part of the SDE, and $\sqrt{dt} \cdot \mathrm{N}(\mathbf{0}, \mathbf{Q})$ accounts for the stochastic component introduced by the noise covariance matrix $\mathbf{Q}$. The numerical solution method presented here is valuable for analyzing systems described by SDEs, providing insights into their temporal evolution and behavior in the presence of noise. We use it for generating synthetic data.

## 2.3 Feedforward Neural Network (FFN)

The FFN is one of the first neural network architecture, consisting of an input layer, one or more hidden layers, and an output layer [6]. The output of an FFN can be expressed as:

$$\hat{y} = f(W^{(L)} \cdot f(W^{(L-1)} \cdot \ldots \cdot f(W^{(1)} \cdot x + b^{(1)}) + b^{(L-1)}) + b^{(L)})$$

Where: $\hat{y}$ represents the predicted output. $x$ denotes the input data. $W^{(i)}$ and $b^{(i)}$ are the weight matrix and bias vector of the $i$-th layer, respectively. $f(\cdot)$ signifies the activation function.
During training an architecture with one hidden layer and 128 hidden states was used. The input consisted of a history of 6 concatenated datapoints and a batch size of 64 was used. For gradient optimization a learning rate of $0.0001$ was chosen and the Mean-Squared-Error (MSE) for loss calculation. As an activation function, Rectified Linear Unit (ReLu) was used.

## 2.4 Recurrent Neural Networks

In the subsequent study we use four different types of recurrent network architectures suited for sequence-to-sequence modeling: Long Short-Term Memory (LSTM) networks, Encoder-Decoder LSTM architecture, Encoder-Decoder LSTM with Teacher Forcing, and Gated Recurrent Units (GRU).
For the following architectures the same training parameters have been used: one hidden layer with 128 hidden states. A batch size of 128 and a learning rate of $0.0001$ with MSE as loss function for gradient optimization. The dataset was shuffled each epoch and the model was trained for $50$ epochs in total. Varying length for the input history and output horizon was used which will be specified in the results.

### 2.4.1 Long Short-Term Memory (LSTM) Networks

LSTM networks are designed for sequence modeling and memory retention. They include memory cells and gate mechanisms to control information flow [7]. The LSTM cell equations are as follows:

$$\begin{aligned}
\varphi_t &= W x_t + b \\
i_t &= \sigma(W_{xi}\varphi(x_t) + W_{hi} h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} x_t + W_{hf} h_{t-1} + b_f) \\
g_t &= \tanh(W_{xg} x_t + W_{hg} h_{t-1} + b_g) \\
o_t &= \sigma(W_{xo} x_t + W_{ho} h_{t-1} + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}$$

Where: $i_t$, $f_t$, $g_t$, $o_t$ represent input, forget, cell, and output gates. $c_t$ is the cell state.$h_t$ is the hidden state. $x_t$ is the input at time step $t$. $W$ and $b$ denote weight matrices and bias vectors. $\sigma$ is the sigmoid activation function, and $\odot$ denotes element-wise multiplication.

### 2.4.2 Encoder-Decoder LSTM Architecture

The Encoder-Decoder LSTM architecture is utilized for sequence-to-sequence tasks, such as machine translation [8]. It comprises an encoder network to process input sequences and a decoder network to generate output sequences. The equations for the Encoder-Decoder LSTM are as follows:

$$h_t^{\text{enc}} = \text{LSTM}_{\text{enc}}(x_t, h_{t-1}^{\text{enc}}) \qquad h_t^{\text{dec}} = \text{LSTM}_{\text{dec}}(h_{t-1}^{\text{dec}})$$
$$\hat{y}_t = f(W^{(L)} \cdot h_t^{\text{dec}} + b^{(L)})$$

Where: $h_t^{\text{enc}}$ and $h_t^{\text{dec}}$ are the hidden states of the encoder and decoder, respectively. At the beginning of each sequence the hidden states are randomly initialised for the encoder and the decoder recieves the ouput of the $\text{LSTM}_{\text{enc}}$ as input.$x_t$ and $y_t$ represent input and output sequences at time step $t$. $W$ and $b$ denote weight matrices and bias vectors. $f(\cdot)$ is the output activation function.

### 2.4.3 Encoder-Decoder LSTM with Input & Teacher Forcing

This variant extends the Encoder-Decoder LSTM by incorporating the start state from the sequence to predict in the input of the $\text{LSTM}_{\text{dec}}$ network as follows [9]:

$$h_t^{\text{dec}} = \text{LSTM}_{\text{dec}}(y_t, h_{t-1}^{\text{dec}})$$

Additionally, teacher-forcing can be incorporated during training, where the true target sequence sometimes replaces the prediction of the model and is used as input during a sequence prediction. The architecture aims at better sequence generation and training stability on the cost of introducing a bias.

### 2.4.4 Gated Recurrent Units (GRU)

GRU is another recurrent neural network variant, featuring update and reset gates [10]. The GRU equations are as follows:

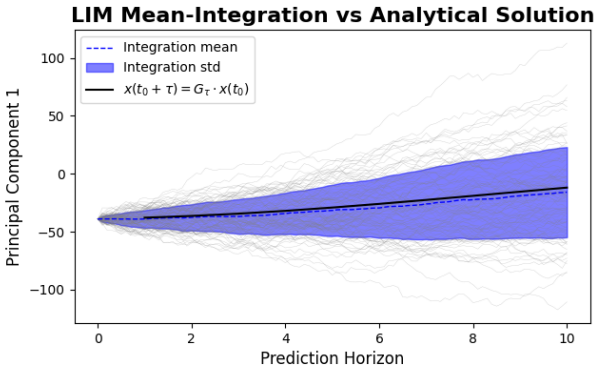
$$
\begin{aligned}
\varphi_t &= W x_t + b \\
z_t &= \sigma(W_{xz}\varphi(x_t) + W_{hz} h_{t-1} + b_z) \\
r_t &= \sigma(W_{xr} x_t + W_{hr} h_{t-1} + b_r) \\
\tilde{h}_t &= \tanh(W_{xh} x_t + r_t \odot (W_{hh} h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{t-1}
\end{aligned}
$$

Where: $z_t$ and $r_t$ are update and reset gates, respectively. $\tilde{h}_t$ is the candidate hidden state. $x_t$ is the input at time step $t$. $h_{t-1}$ is the previous hidden state. $W$ and $b$ represent weight matrices and bias vectors. $\sigma$ is the sigmoid activation function, and $\odot$ denotes element-wise multiplication.

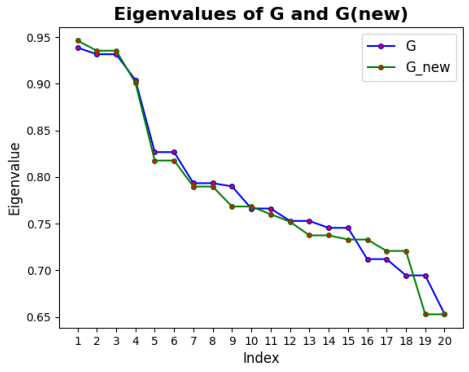
# 3 Results

## 3.1 LIM Integration

The following two plots aim to verify the integration of the LIM using the euler method described in section 2.2.3. In Figure 2 (a) the mean of 100 integration runs is compared against the analytical solution using the $\mathbf{G}$ operator and the respective target sequence. One needs to keep in mind that during integration noise is added to the forecast so it will always deviate more from the optimal solution but the mean of increasing number of integrations should manifest the same fundamental behavior as the analytical solution as illustrated below.




(a) Mean of LIM integration compared to analytical solution.




(b) Eigenvalues of Green Function of original LIM and to newly fitted LIM on synthetic data.

Figure 2: Evaluation of D4PG performance considering obtained reward and number of wins during evaluation

Figure 2 (b) shows the eigenvalues of $\mathbf{G}$ of the first 20 principal components when the LIM is fitted on the original data and the respective eigenvalues of $\mathbf{G}(\mathbf{new})$ which represents the time evolution operator of a new LIM being fitted on the synthetic data created by the first LIM. As shown above the eigenvalues nearly completely overlap which is evidence that the dynamics of the new synthetic data are nearly equivalent to the extracted linear dynamics of CESM2 data.

## 3.2 Neural Network Model Comparison

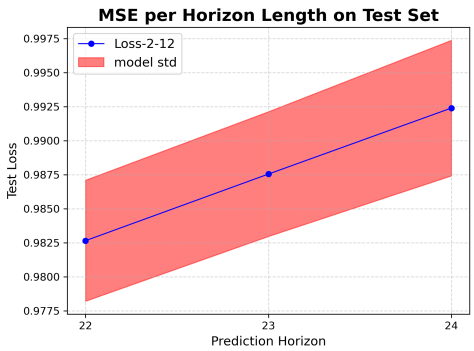
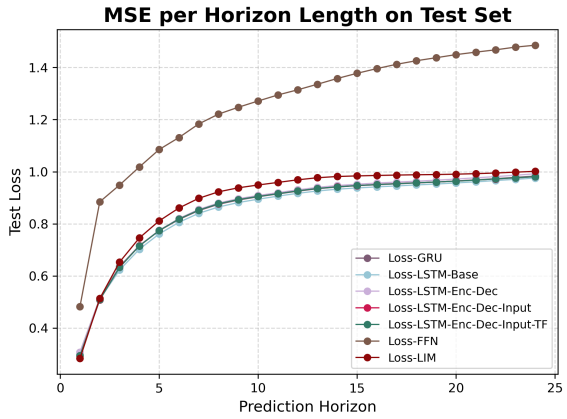
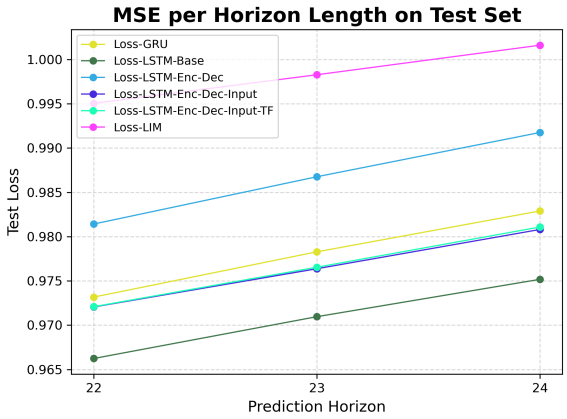



Figure 3: Spread around the mean of MSE on test set when training same model 15 times.

All of the following model results are trained on $100k$ synthetic datapoints generated through integration via euler method as described in section 2.2.3. The corresponding performance has subsequently been tested on the piControl CESM2 data. Figure 3 illustrates the resulting spread of the standard deviation when training an encoder-decoder LSTM 15 times with the same initial parameters. Due to its stochastic the results deviate each training when not manually fixing the seed of the script. This can be seen by the spread of around 0.005 around the mean.
Figure 4 (a) compares the performance of all the different models that were used to predict ENSO activity with varying horizon length. For all models an input-history of 2 and an output-horizon of 12 was chosen. However, during evaluation the horizon was iteratively increased from $1 - 24$ and the corresponding mean MSE was calculated for each horizon over the whole CESM2 test dataset. For the FFN an input-history of 6 was used. And for the LIM prediction the analytical forecast method was chosen.




(a) Comparison of models average MSE with varying horizon length on test set.



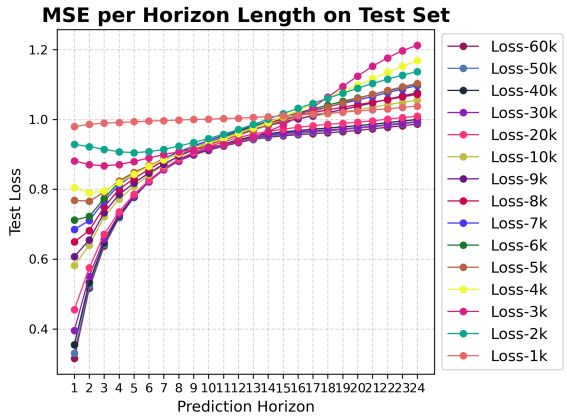

(b) Comparison of different recurrent network models on horizon-length [22, 23, 24]

Figure 4: Evaluation of D4PG performance considering obtained reward and number of wins during evaluation

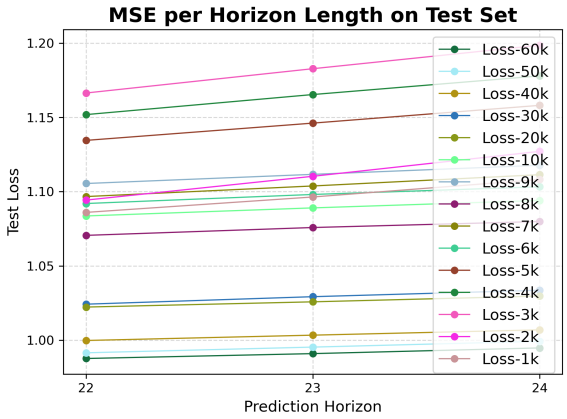
In the above plot it becomes apparent that the recurrent models perform better than the simple FFN model over all prediction horizons and outperform the LIM in later prediction horizons aswell. When comparing the different recurrent models in Figure 4 (b) over the the last horizons $[22, 23, 24]$ the standard LSTM model performs best but there is no significant difference in performance when considering the spread introduced through stochasiticity in each run.

### 3.3 Number of Datapoints

In order to see whether the prediction performance of our neural network architecture can be improved by increasing the number of datapoints we conducted an experiment, iteratively increasing the number of synthetic datapoints used for training and comparing the MSE loss over multiple prediction horizons as described above. In Figure 5 (a) illustrates that especially in the early and late prediction horizons the number of datapoints effects the performance. Overall, you can see that one can observe an improvement in performance until around 30000 datapoints. Afterwards the improvement is only very marginal, considering the stochasticity that is introduced for each run, which especially becomes apparent in Figure 5 (b) in the late prediction horizons.




(a) Comparison of models average MSE with varying horizon length on test set.




(b) Comparison of different recurrent network models on horizon-length [22, 23, 24]

Figure 5: Evaluation of D4PG performance considering obtained reward and number of wins during evaluation

### 3.4 Horizon Length Effect of LSTM

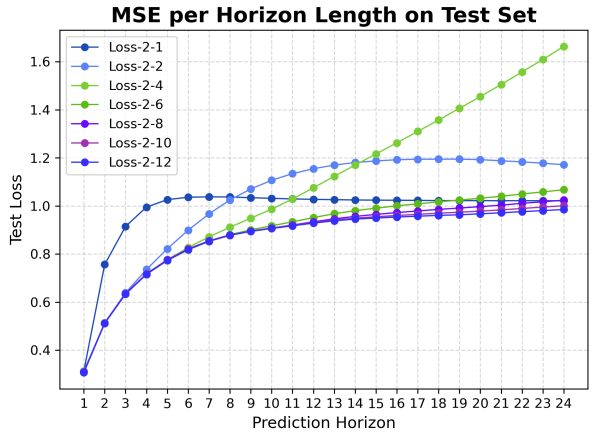



Figure 6: MSE-Loss over horizon between 1-24 on test set with different output-length

In the final experiment we were looking at the effect of different horizon lengths during training of the the encoder-decoder LSTM. Figure 6 illustrates that the chosen horizon-length during training plays a significant role in the prediction performance of the model. Especially models trained with smaller output lengths, such as $1, 2, 4$, really struggle with later prediction horizons. An output length of $12$ yields the best results, however the difference to an output length of $10$ is rather insignificant which suggests there is no benefit in even increasing the output length further considering the increasing computational costs.

# 4 Discussion

In the following section, I will discuss the outcomes of my experiments, where the primary objective was to assess the performance and behavior of neural network architectures for ENSO prediction, using synthetic data created by the Linear Inverse Model (LIM) as proposed by Penland et al. [2]. The LIM serves as our baseline, enabling us to compare the performance of neural networks effectively.
In the context of integrating the LIM with the Euler method, Figure 2 (a,b) demonstrates that the synthetic data generated by the model closely mirrors temporal behavior than the original CESM2 data the model was initially fitted on. Especially, Figure 2 (b) highlights that the eigenvalues of the temporal evolution operator $\mathbf{G}$ are nearly identical to the ones of the $\hat{\mathbf{G}}$ function fitted on the synthetic data. These observations affirm the correct implementation of the LIM, which is subsequently being used as a method for generating synthetic data.
When examining the performance of the different models in Figure 4 (a,b) it becomes apparent that the recurrent models outperform the feed-forward model significantly over all prediction horizons This outcome aligns seamlessly with our expectations, as recurrent models stand out in capturing temporal dependencies. Additionally, the recurrent models show better performance in later prediction horizons than the LIM which is a sign for better capturing temporal dependencies over longer historical sequences and being able to propagate the underlying dynamics forward more consistently than the LIM.
Concerning the impact of additional synthetic data on the prediction performance of the encoder-decoder LSTM, Figure 5 (a,b) shows that there is a substantial performance boost up to a training dataset size of $30k$. Beyond this point, further data augmentation does not seem to yield additional benefits in terms of the model's ability to learn the underlying dynamics. Consequently, the underlying dynamics of the principal components of SST and SSH of the CESM2 data do not appear excessively complex, suggesting that extremely large datasets may not be necessary for effective learning. However, the innate noise associated with the ENSO climate phenomenon presents a fundamental challenge for neural networks.
Lastly, Figure 6 demonstrates that opting for larger output-sequence lengths proves crucial when forecasting extended horizons. The reason for this might be that during the model's training, gradient updates tend to overly emphasize the initial one or two horizons when the output length is limited and consequently struggle to propagate the dynamics further into the future. By incorporating a longer output length during training, the model becomes better equipped to capture recurring dynamics that manifest over more extended horizons, distinct from those occurring within just one or two subsequent timesteps.

In summary, the results validate the LIM's reliability for synthetic data generation and its benefit for increasing prediction performance, as well as highlighting the superiority of recurrent models in capturing temporal dependencies over feed-forward models. Moreover, we show emphasize the importance of longer output-sequence lengths for improved forecasting, addressing challenges in modeling the ENSO climate phenomenon.

# Acknowledgements

Thank you for the supervision of the project by Jakob Schloer and Bedartha Goswami from the Machine Learning in Climate Science Research Group from University of Tübingen.

# References

[1] H. F. Diaz, M. P. Hoerling, and J. K. Eischeid, "Enso variability, teleconnections and climate change," *International Journal of Climatology: A Journal of the Royal Meteorological Society*,

vol. 21, no. 15, pp. 1845–1862, 2001.

[2] C. Penland and L. Matrosova, "A balance condition for stochastic numerical models with application to the el nino-southern oscillation," *Journal of climate*, vol. 7, no. 9, pp. 1352–1372, 1994.

[3] G. Danabasoglu, J.-F. Lamarque, J. Bacmeister, D. Bailey, A. DuVivier, J. Edwards, L. Emmons, J. Fasullo, R. Garcia, A. Gettelman, *et al.*, "The community earth system model version 2 (cesm2)," *Journal of Advances in Modeling Earth Systems*, vol. 12, no. 2, p. e2019MS001916, 2020.

[4] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[5] A. Hannachi, I. T. Jolliffe, and D. B. Stephenson, "Empirical orthogonal functions and related techniques in atmospheric science: A review," *International Journal of Climatology: A Journal of the Royal Meteorological Society*, vol. 27, no. 9, pp. 1119–1152, 2007.

[6] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," pp. 368–408, 1958.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

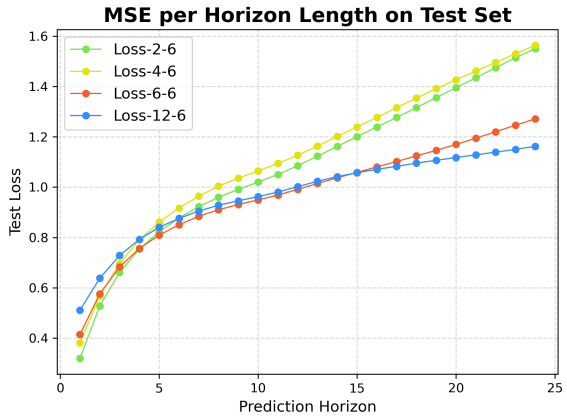
[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[9] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," 2015.

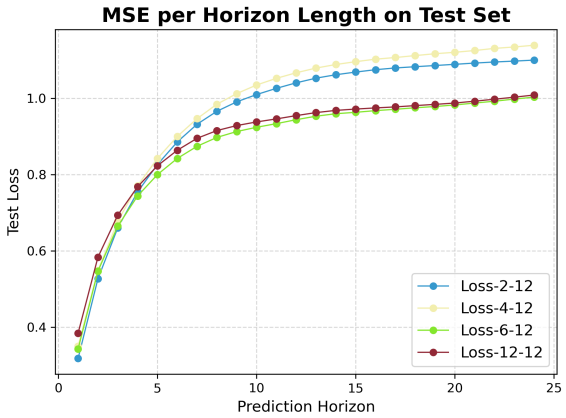
[10] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

# 5 Appendix

## 5.1 Examining Performance of different Ouput-Lengths during Training

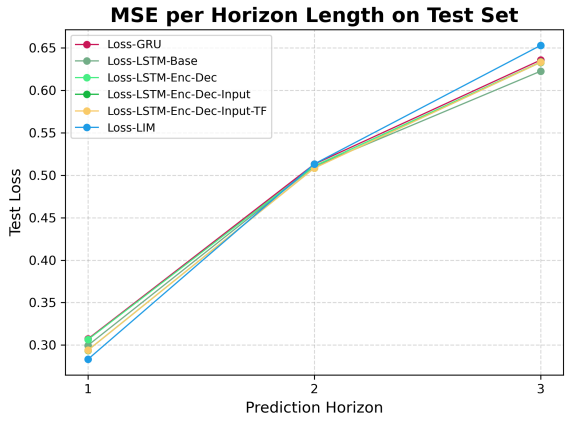



(a) Comparison of encoder-decoder LSTM average MSE with varying input-length and fixed output-length=6 on piControl test set.

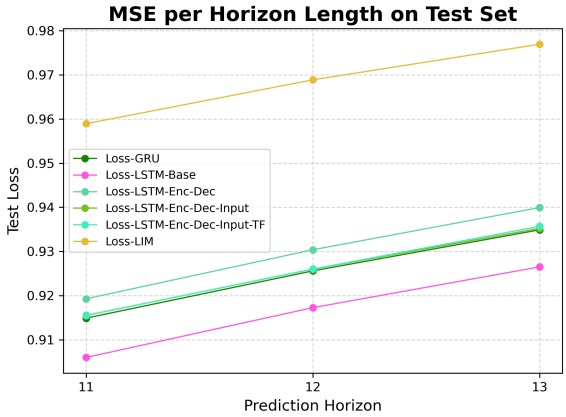



(b) Comparison of encoder-decoder LSTM average MSE with varying input-length and fixed output-length=12 on piControl test set.

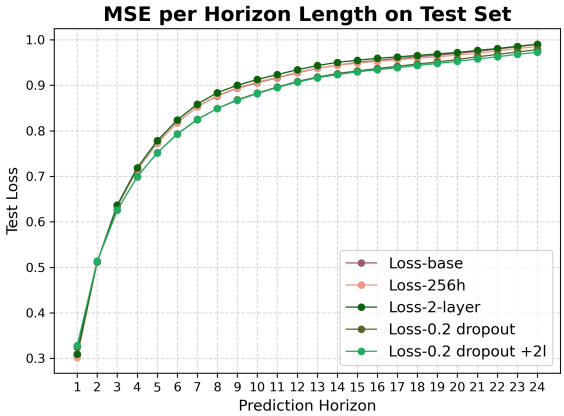
## 5.2 Examining specific Horizon Lengths similar to Figure 5b



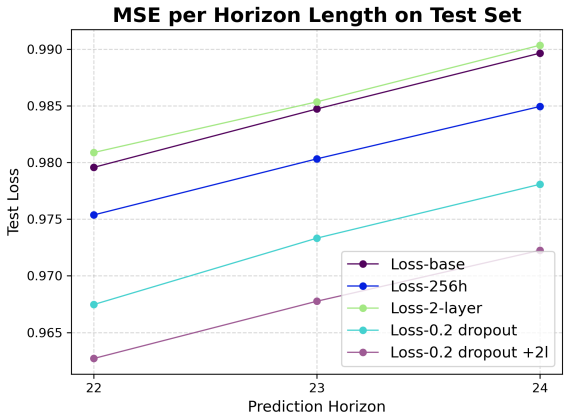

(a) Comparison of performance of different recurrent network models with horizon-length [1, 2, 3] on piControl test set.



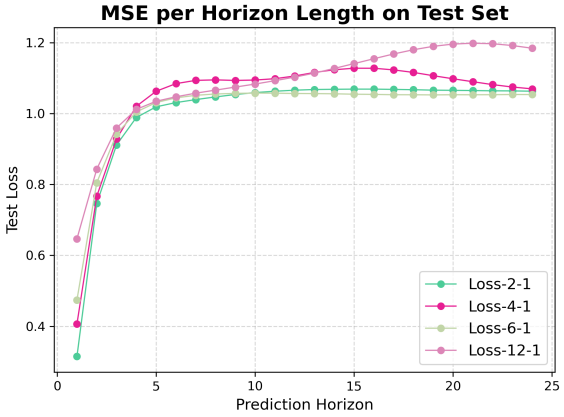

(b) Comparison of different recurrent network models with horizon-length [11, 12, 13] on piControl test set.



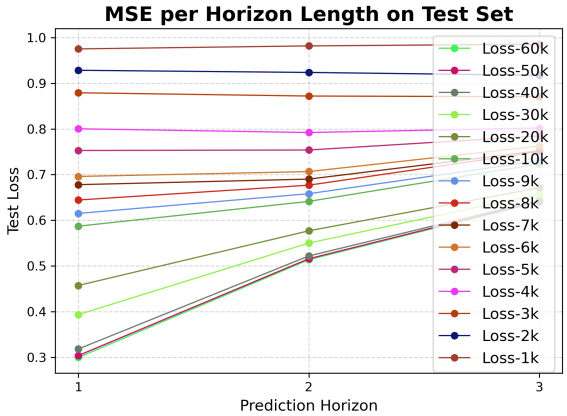

(a) Comparison of encoder-decoder LSTM average MSE with different hyperparameters and input-length=2, output-length=12 on piControl test set.




(b) Comparison of encoder-decoder LSTM average MSE with different hyperparameters on prediction horizon [22, 23, 24] with input-length=2 and output-length=12 on piControl test set.

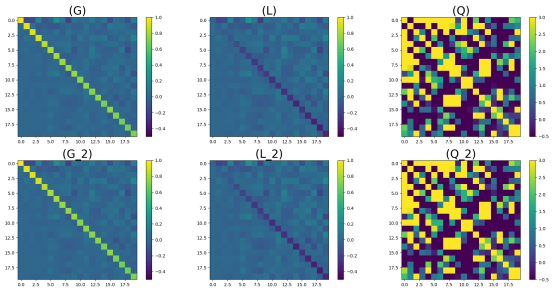



(a) Comparison of encoder-decoder LSTM average MSE with varying input-length and output-length=1 on piControl test set.

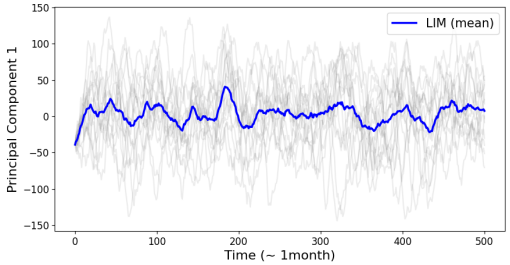



(b) Comparison of different data length for training on prediction horizon [1, 2, 3]

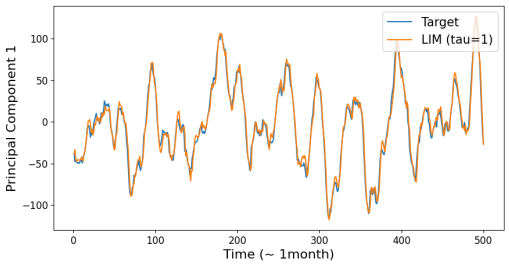
## 5.3 Verification of LIM implementation and shape of Train/Eval loss curve of Encoder-Decoder LSTM



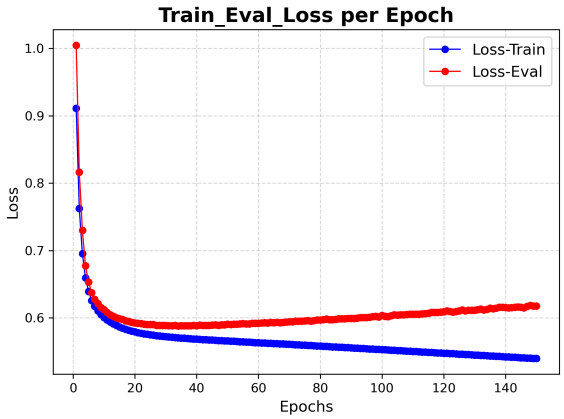

(a) G, Q and L function of LIM fitted to original data and on synthetic data generated through Euler Method to verify correctness.




(b) Plotted timeseries of LIM integration with Euler Method to verify its stationary.




(a) Analytical mean forecast with L function of LIM.




(b) Training and Evaluation Loss of encoder-decoder LSTM on $100k$ synthetic data generated by Euler Method to verify correct shape

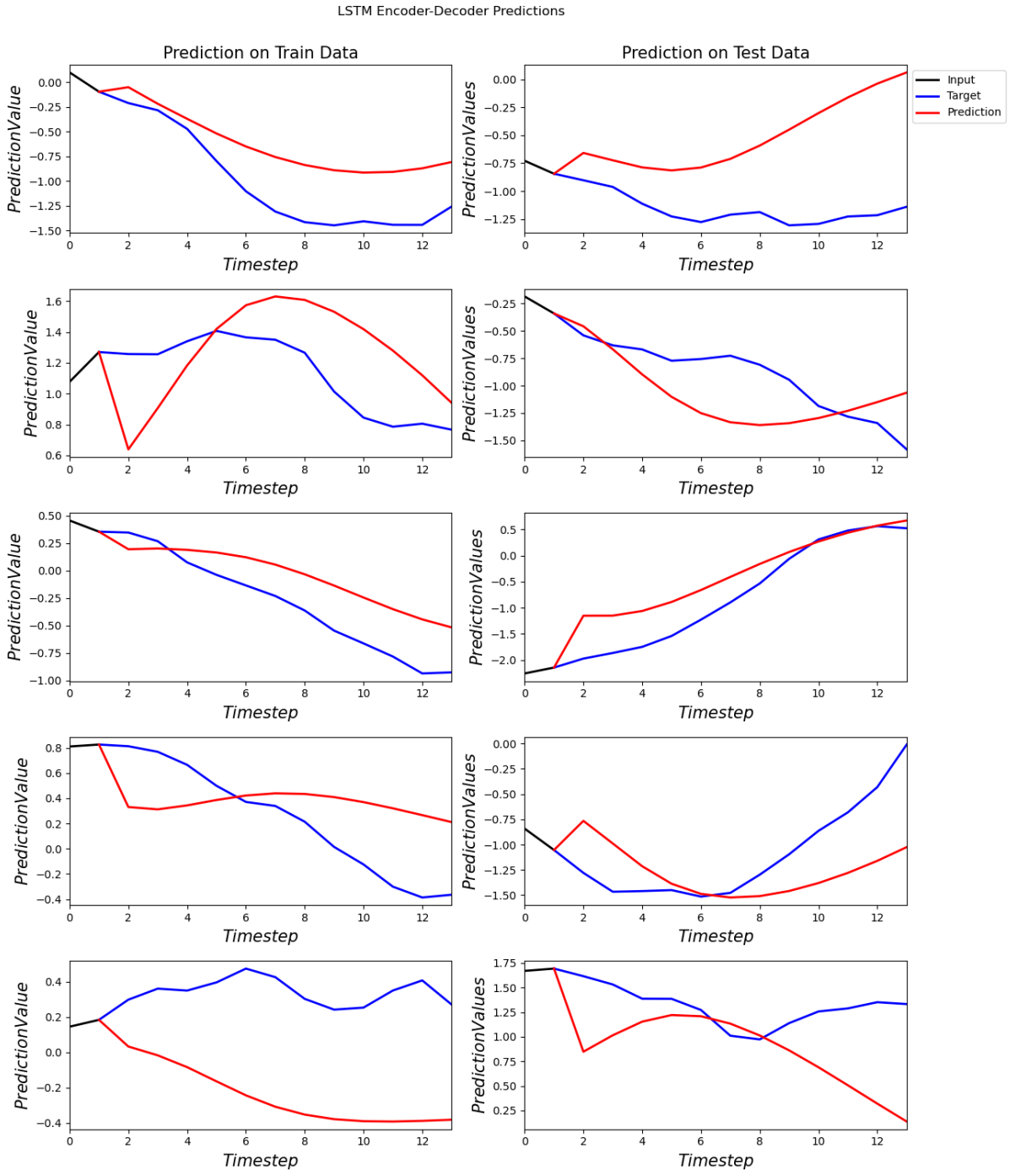



Figure 13: Prediction of timeseries of first Principal Component using encoder-decoder LSTM with input-length=2 and output-length=12 on piControl test set.